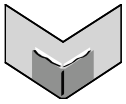
The Journal of Machine Learning Research
Volume 8
Print-Archive Edition

Pages 1393–2790



Microtome Publishing
Brookline, Massachusetts
www.mtome.com

The Journal of Machine Learning Research
Volume 8
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 2007.

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 Journal of Machine Learning Research Vol. 8 2007

Journal of Machine Learning Research 8 (2007) 1393-1429 Submitted 2/06; Revised 9/06; Published 7/07

Learning to Classify Ordinal Data: The Data Replication Method

Jaime S. Cardoso JAIME.CARDOSO@INESCPORTO.PT
INESC Porto, Faculdade de Engenharia, Universidade do Porto
Campus da FEUP, Rua Dr. Roberto Frias, n 378
4200-465 Porto, Portugal

Joaquim F. Pinto da Costa JPCOSTA@FC.UP.PT
Faculdade Ciências Universidade Porto
Rua do Campo Alegre, 687
4169-007 Porto, Portugal

Editor: Ralf Herbrich

Abstract

Classification of ordinal data is one of the most important tasks of relation learning. This paper introduces a new machine learning paradigm specifically intended for classification problems where the classes have a natural order. The technique reduces the problem of classifying ordered classes to the standard two-class problem. The introduced method is then mapped into support vector machines and neural networks. Generalization bounds of the proposed ordinal classifier are also provided. An experimental study with artificial and real data sets, including an application to gene expression analysis, verifies the usefulness of the proposed approach.

Keywords: classification, ordinal data, support vector machines, neural networks

1. Introduction

Predictive learning has traditionally been a standard inductive learning, where different sub-problem formulations have been identified. One of the most representative is *classification*, consisting on the estimation of a mapping from the feature space into a finite class space. Depending on the cardinality of the finite class space we are left with binary or multiclass classification problems. Finally, the presence or absence of a "natural" order among classes will separate nominal from ordinal problems.

Although two-class and nominal data classification problems have been thoroughly analysed in the literature, the ordinal sibling has not received nearly as much attention yet. Nonetheless, many real life problems require the classification of items into naturally ordered classes. The scenarios involved range from information retrieval (Herbrich et al., 1999a) and collaborative filtering (Shashua and Levin, 2002) to econometric modeling (Mathieson, 1995) and medical sciences (Cardoso et al., 2005). It is worth pointing out that distinct tasks of relational learning, where an example is no longer associated with a class or rank, which include preference learning and reranking (Shen and Joshi, 2005), are topics of research on their own.

Conventional methods for nominal classes or for regression problems could be employed to solve ordinal data problems. However, the use of techniques designed specifically for ordered classes yields simpler and with better performance classifiers. Although the ordinal formulation seems conceptually simpler than the nominal one, difficulties to incorporate in the algorithms this

piece of additional information—the order—may explain the widespread use of conventional methods to tackle the ordinal data problem.

This work addresses this void by introducing in Section 2 the data replication method, a nonparametric procedure for the classification of ordinal data. The underlying paradigm is the extension of the original data set with additional variables, reducing the classification task to the well known two-class problem. Starting with the simpler linear case, already established in Cardoso et al. (2005), the section develops the nonlinear case; from there the method is extended to incorporate the procedure of Frank and Hall (2001). Finally, the generic version of the data replication method is presented, allowing partial constraints on variables.

In section 3 the data replication method is instantiated in two important machine learning algorithms: support vector machines and neural networks. A comparison is made with a previous SVM approach introduced by Shashua and Levin (2002), the minimum margin principle, showing that the data replication method leads *essentially* to the same solution, but with some key advantages. The section is concluded with a reinterpretation of the neural network model as a generalization of the ordinal logistic regression model.

Section 4 describes the experimental methodology and the algorithms under comparison; results are reported and discussed in the succeeding sections. Finally, conclusions are drawn and future work is outlined in Section 8.

2. The Data Replication Method

Assume that examples in a classification problem come from one of K ordered classes, labeled from $\mathcal{C}_1$ to $\mathcal{C}_K$, corresponding to their natural order. Consider the training set $\{\mathbf{x}_i^{(k)}\}$, where $k = 1,\ldots,K$ denotes the class number, $i = 1,\ldots,\ell_k$ is the index within each class, and $\mathbf{x}_i^{(k)} \in \mathbb{R}^p$, with p the dimension of the feature space. Let $\ell = \sum_{k=1}^{K} \ell_k$ be the total number of training examples.

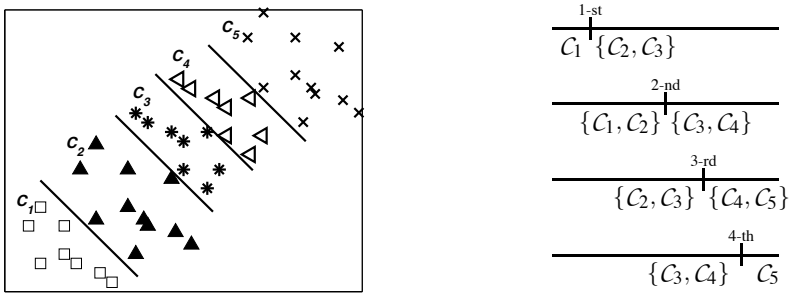
Suppose that a K-class classifier was forced, by design, to have $(K-1)$ *nonintersecting boundaries*, with boundary i discriminating classes $\mathcal{C}_1,\ldots,\mathcal{C}_i$ against classes $\mathcal{C}_{i+1},\ldots,\mathcal{C}_K$. As the intersection point of two boundaries would indicate an example with three or more classes equally probable—not plausible with ordinal classes—this strategy imposes a sensible restriction. With this constraint emerges a monotonic model, where a better value in an attribute does not lead to a lower decision class. For the linear case, this translates into choosing the same weighted sum for all decisions—the classifier would be just a set of weights, one for each feature, and a set of thresholds, the *scale* in the weighted sum. By avoiding the intersection of any two boundaries, this model tries to capture the essence of the ordinal data problem. Additionally, we foresee a better generalization performance due to the reduced number of parameters to be estimated.

This rationale leads to a straightforward generalization of the two-class separating hyperplane (Shashua and Levin, 2002). Define $(K-1)$ hyperplanes that separate the training data into K ordered classes by modeling the ranks as intervals on the real line. The geometric interpretation of this approach is to look for $(K-1)$ parallel hyperplanes, represented by vector $\mathbf{w} \in \mathbb{R}^p$ and scalars $b_1,\ldots,b_{K-1}$, such that the feature space is divided into K regions by the decision boundaries $\mathbf{w}^t\mathbf{x} + b_r = 0$, $r = 1,\ldots,K-1$.

Given that there are many good two-class learning algorithms, it is tempting to reduce this ordinal formulation to a two-class problem. The data replication method allows us to do precisely that.

2.1 Data Replication Method—the Linear Case[1]

Before moving to the formal presentation of the data replication method, it is instructive to motivate the method by considering a hypothetical, simplified scenario, with five classes in $\mathbb{R}^2$. The plot of the data set is presented in Figure 1(a); superimposed is also depicted a reasonable set of four parallel hyperplanes separating the five classes.



(a) Plot of the data points. Also shown are reasonable class boundaries.

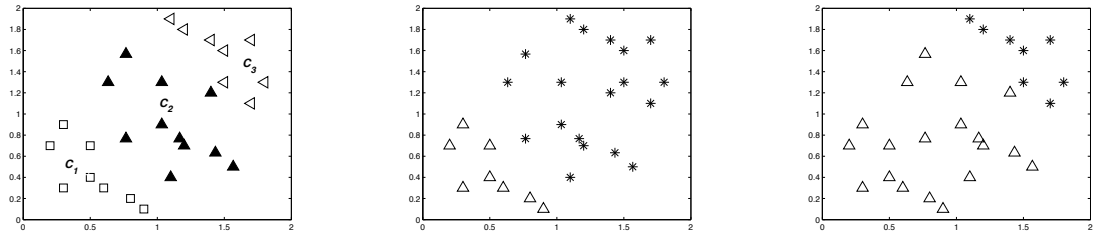
(b) Classes involved in the hyperplanes definition, for $K = 5$, $s = 2$.

Figure 1: Toy model with 5 classes in $\mathbb{R}^2$.

The i-th hyperplane discriminates classes $\mathcal{C}_1, \ldots, \mathcal{C}_i$ against classes $\mathcal{C}_{i+1}, \ldots, \mathcal{C}_K$. Due to the order among classes, the i-th hyperplane is essentially determined by classes $\mathcal{C}_i$ and $\mathcal{C}_{i+1}$. More generally, it can be said that the i-th hyperplane is determined by s classes to its 'left' and s classes to its 'right', with $1 \leq s \leq K-1$. Naturally, for some of the hyperplanes, there will be less than s classes to one (or both) of its sides. The i-th hyperplane has i classes on its left and $(K-i)$ on its right. Hence, for a chosen s value, we may say that classifier i depends on $\min(s,i)$ classes on its left—classes $\mathcal{C}_k, k = \max(i-s+1,1), \ldots, i$—and depends on $\min(s, K-i)$ classes on its right—classes $\mathcal{C}_k, k = i+1, \ldots, \min(i+1+s-1, i+1+K-i-1) = i+1, \ldots, \min(i+s, K)$. The classes involved in each hyperplane definition, for the five class data set with $s = 2$, are illustrated in Figure 1(b).

To start the presentation of the data replication method let us consider an even more simplified toy example with just three classes, as depicted in Figure 2(a). Here, the task is to find two parallel hyperplanes, the first one discriminating class $\mathcal{C}_1$ against classes $\{\mathcal{C}_2, \mathcal{C}_3\}$ (we are considering $s = K-1 = 2$ for the explanation) and the second discriminating classes $\{\mathcal{C}_1, \mathcal{C}_2\}$ against class $\mathcal{C}_3$. These hyperplanes will correspond to the solution of two binary classification problems but with the additional constraint of parallelism—see Figure 2. The data replication method suggests solving both problems simultaneously in an augmented feature space.

1. The linear version of the data replication method was already presented in Cardoso et al. (2005); here we provide, arguably, a cleaner presentation.



(a) Original data set in $\mathbb{R}^2$, $K = 3$.

(b) Binary problem $\mathcal{C}_1$ against classes $\{\mathcal{C}_2, \mathcal{C}_3\}$.

(c) Binary problem $\{\mathcal{C}_1, \mathcal{C}_2\}$ against class $\mathcal{C}_3$.

Figure 2: Binary problems to be solved simultaneously with the data replication method.

Using a transformation from the $\mathbb{R}^2$ initial feature-space to a $\mathbb{R}^3$ feature space, replicate each original point, according to the rule (see Figure 3(b)):

$$\mathbf{x} \in \mathbb{R}^2 \begin{cases} \nearrow \begin{bmatrix} \mathbf{x} \\ h \end{bmatrix} \in \mathbb{R}^3 \\ \searrow \begin{bmatrix} \mathbf{x} \\ 0 \end{bmatrix} \in \mathbb{R}^3 \end{cases}, \text{ where } h = \text{const} \in \mathbb{R}^+.$$

Observe that any two points created from the same original point differ only in the new variable. Define now a binary training set in the new (higher dimensional) space according to (see Figure 3(c)):

$$\begin{bmatrix} \mathbf{x}_i^{(1)} \\ 0 \end{bmatrix} \in \overline{\mathcal{C}}_1, \begin{bmatrix} \mathbf{x}_i^{(2)} \\ 0 \end{bmatrix}, \begin{bmatrix} \mathbf{x}_i^{(3)} \\ 0 \end{bmatrix} \in \overline{\mathcal{C}}_2 \qquad ; \begin{bmatrix} \mathbf{x}_i^{(1)} \\ h \end{bmatrix}, \begin{bmatrix} \mathbf{x}_i^{(2)} \\ h \end{bmatrix} \in \overline{\mathcal{C}}_1, \begin{bmatrix} \mathbf{x}_i^{(3)} \\ h \end{bmatrix} \in \overline{\mathcal{C}}_2 \,. \tag{1}$$

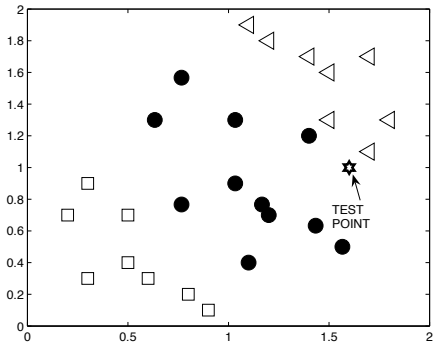
In this step we are defining the two binary problems as a single binary problem in the augmented feature space. A linear two-class classifier can now be applied on the extended data set, yielding a hyperplane separating the two classes, see Figure 3(d). The intersection of this hyperplane with each of the subspace replicas can be used to derive the boundaries in the original data set, as illustrated in Figure 3(e).

Although the foregoing analysis enables one to classify unseen examples in the original data set, classification can be done directly in the extended data set, using the binary classifier, without explicitly resorting to the original data set. For a given example $\in \mathbb{R}^2$, classify each of its two replicas $\in \mathbb{R}^3$, obtaining a sequence of two labels $\in \{\overline{\mathcal{C}}_1, \overline{\mathcal{C}}_2\}^2$. From this sequence infer the class according to the rule

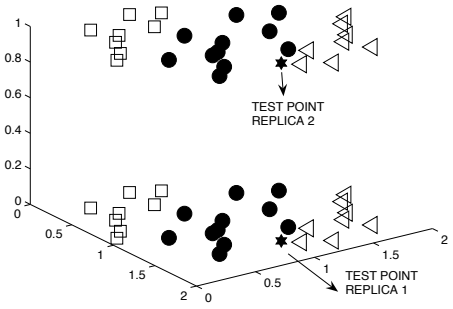
$$\overline{\mathcal{C}}_1\overline{\mathcal{C}}_1 \Longrightarrow \mathcal{C}_1, \qquad \overline{\mathcal{C}}_2\overline{\mathcal{C}}_1 \Longrightarrow \mathcal{C}_2, \qquad \overline{\mathcal{C}}_2\overline{\mathcal{C}}_2 \Longrightarrow \mathcal{C}_3.$$

To exemplify, from the test point in Figure 3(a), create two replicas and classify them in $\mathbb{R}^3$ (highlighted in Figure 3(d)). The first replica is classified as $\overline{\mathcal{C}}_2$ and the second as $\overline{\mathcal{C}}_1$. Hence, the class predicted for the test point is $\mathcal{C}_2$. This same class could have been obtained in the original feature space, as represented in Figure 3(e).

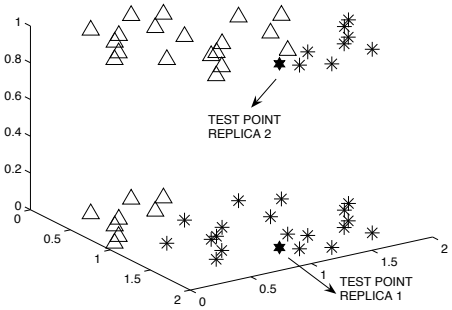
It is clear now that the principle behind the replication method is to have a replica of the original data set for each boundary. The replica i is binarized to discriminate classes $\mathcal{C}_1, \ldots, \mathcal{C}_i$ against classes



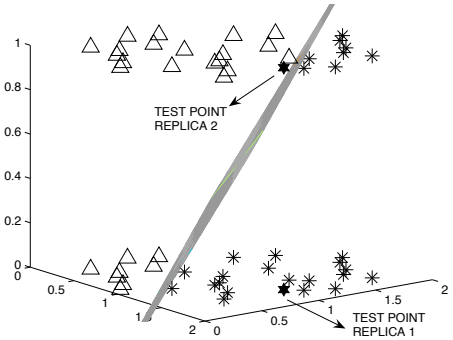
(a) Original data set in $\mathbb{R}^2$, $K = 3$.



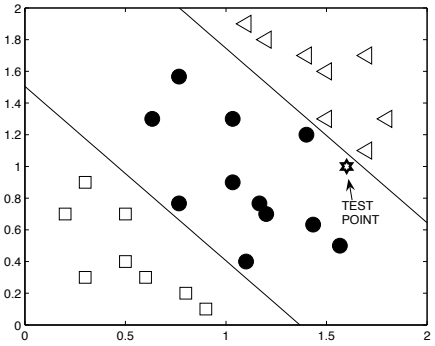
(b) Data set in $\mathbb{R}^3$, with samples replicated ($h = 1$).



(c) Transformation into a binary classification problem.



(d) Linear solution to the binary problem.



(e) Linear solution in the original data set.

Figure 3: Proposed data extension model in a toy example.

$\mathcal{C}_{i+1},\ldots,\mathcal{C}_K$ (more generally, when parameterized by s, discriminating classes $\mathcal{C}_k, k = \max(i-s+1,1),\ldots,i$ against classes $\mathcal{C}_k, k = i+1,\ldots,\min(i+s,K)$). The additional variables, in number of $(K-2)$, provide just the right amount of flexibility needed for having boundaries with the same direction but with different thresholds.

After the above exposition on a toy model, we will now formally describe a general K-class classifier for ordinal data classification. Define $\mathbf{e}_0$ as the sequence of $(K-2)$ zeros and $\mathbf{e}_q$ as the sequence of $(K-2)$ symbols $0,\ldots,0,h,0,\ldots,0$, with $h>0$ in the q-th position. Considering the problem of separating K ordered classes $\mathcal{C}_1,\ldots,\mathcal{C}_K$ with training set $\{\mathbf{x}_i^{(k)}\}$, define a new binary training data set in $\mathbb{R}^{p+K-2}$ as

$$\begin{aligned}
\begin{bmatrix}\mathbf{x}_i^{(k)}\\ \mathbf{e}_0\end{bmatrix} &\in \begin{cases}\overline{\mathcal{C}}_1 & k=1,\\ \overline{\mathcal{C}}_2 & k=2,\ldots,\min(K,1+s),\end{cases}\\
&\vdots\\
\begin{bmatrix}\mathbf{x}_i^{(k)}\\ \mathbf{e}_{q-1}\end{bmatrix} &\in \begin{cases}\overline{\mathcal{C}}_1 & k=\max(1,q-s+1),\ldots,q,\\ \overline{\mathcal{C}}_2 & k=q+1,\ldots,\min(K,q+s),\end{cases}\\
&\vdots\\
\begin{bmatrix}\mathbf{x}_i^{(k)}\\ \mathbf{e}_{K-2}\end{bmatrix} &\in \begin{cases}\overline{\mathcal{C}}_1 & k=\max(1,K-1-s+1),\ldots,K-1,\\ \overline{\mathcal{C}}_2 & k=K,\end{cases}
\end{aligned} \tag{2}$$

where parameter $s \in \{1,\ldots,K-1\}$ plays the role of bounding the number of classes defining each hyperplane. This setup allows controlling the increase of data points inherent to this method. The toy example in Figure 3(b) was illustrated with $s = K-1 = 2$; by setting $s = 1$ one would obtain the extended data set illustrated in Figure 4.

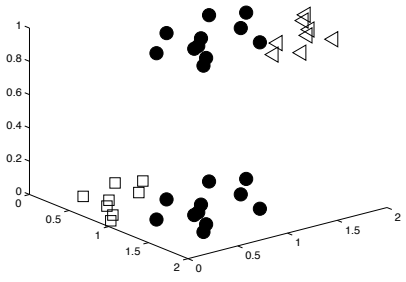


Figure 4: Toy data set replicated in $\mathbb{R}^3$, $h=1$, $s=1$.

Next, build a linear two-class classifier on this extended data set; to predict the class of an unseen example, obtain a sequence of $(K-1)$ labels $\in \{\overline{\mathcal{C}}_1,\overline{\mathcal{C}}_2\}^{(K-1)}$ by classifying each of the $(K-1)$ replicas in the extended data set with the binary classifier. Note that to make a prediction, all $(K-1)$ replicas of the test point are always classified, even if the classifier was trained with $s < K-1$. It is also worth noticing that the binary decision $\bar{\mathbf{w}}^t\bar{\mathbf{x}}+b=0$, with $\bar{\mathbf{w}},\bar{\mathbf{x}} \in \mathbb{R}^{p+K-2}$ when mapped into the

original space gives rise to the $(K-1)$ boundaries, in the form $\mathbf{w}^t\mathbf{x}+b_i$, with

$$b_i = \begin{cases} b & \text{if } i=1, \\ hw_{p+i-1}+b & \text{if } i>1. \end{cases}$$

Now, what possible sequences can one obtain?

Assume for now that the thresholds are correctly ordered as $-b_1 \leq -b_2 \leq \ldots \leq -b_{K-1}$. Or, equivalently, that $0 \geq hw_{p+1} \geq hw_{p+2} \ldots \geq hw_{p+K-2}$. If the replica $\left[\begin{smallmatrix}\mathbf{x}\\ \mathbf{e}_i\end{smallmatrix}\right]$ of a test point $\mathbf{x}$ is predicted as $\bar{\mathcal{C}}_1$, that is because $\bar{\mathbf{w}}^t\left[\begin{smallmatrix}\mathbf{x}\\ \mathbf{e}_i\end{smallmatrix}\right]+b<0$. But then also the replica $\left[\begin{smallmatrix}\mathbf{x}\\ \mathbf{e}_{i+1}\end{smallmatrix}\right]$ is predicted as $\bar{\mathcal{C}}_1$ because $0 \geq hw_{p+i} \geq hw_{p+i+1}$. One can conclude that the only K possible sequences and the corresponding predicted classes are

$$\begin{array}{ll} \overline{\mathcal{C}}_1, \overline{\mathcal{C}}_1, \ldots, \overline{\mathcal{C}}_1, \overline{\mathcal{C}}_1 & \Longrightarrow \mathcal{C}_1, \\ \overline{\mathcal{C}}_2, \overline{\mathcal{C}}_1, \ldots, \overline{\mathcal{C}}_1, \overline{\mathcal{C}}_1 & \Longrightarrow \mathcal{C}_2, \\ \qquad\vdots & \\ \overline{\mathcal{C}}_2, \overline{\mathcal{C}}_2, \ldots, \overline{\mathcal{C}}_2, \overline{\mathcal{C}}_1 & \Longrightarrow \mathcal{C}_{K-1}, \\ \overline{\mathcal{C}}_2, \overline{\mathcal{C}}_2, \ldots, \overline{\mathcal{C}}_2, \overline{\mathcal{C}}_2 & \Longrightarrow \mathcal{C}_K. \end{array}$$

Henceforth, the target class can be obtained by adding one to the number of $\overline{\mathcal{C}}_2$ labels in the sequence. To emphasize, the process is depicted in Figure 5 for a data set in $\mathbb{R}$ with four classes.

The reduction technique presented here uses a binary classifier to make multiclass ordinal predictions. Instead of resorting to multiple binary classifiers to make predictions in the ordinal problem (as is common in reduction techniques from multiclass to binary problems), the data replication method uses a single binary classifier to classify $(K-1)$ *dependent* replicas of a test point. A pertinent question is how the performance of the binary classifier translates into the performance on the ordinal problem. In Appendix A, a bound on the generalization error of the ordinal data classifier is expressed as a function of the error of the binary classifier.

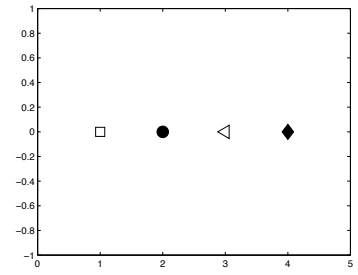
The thresholds b_i were assumed correctly ordered. It is not trivial to see how to keep them well ordered with this standard data replication method, for s general. We present next an alternative method of replicating the data, in which constraints on the thresholds are explicitly incorporated in the form of extra points added to the training set. This formulation enforces ordered boundaries for s values as low as 1, although compromising some of cleanliness in the interpretation as a binary classification problem in the extended space.

2.2 Homogeneous Data Replication Method

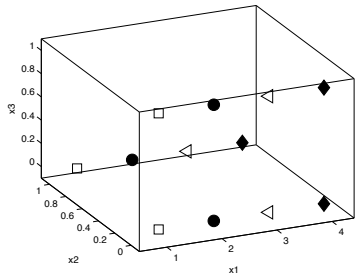
With the data replication method just presented, the boundary in the extended space $\bar{\mathbf{w}}^t\bar{\mathbf{x}}+b=0$ has correspondence in the original space to the $(K-1)$ boundaries $\mathbf{w}^t\mathbf{x}+b_i$, with $b_1=b$, $b_i = hw_{p+i-1}+b_1, i=2,\ldots,K-1$. It is notorious the asymmetry with respect to b_1.

One could attain a symmetric formulation by introducing the well-known homogenous coordinates. That would lead to the addition of a new variable and to the restriction of linear boundaries going through the origin. The same result can be obtained by starting with a slightly different extension of the data set.

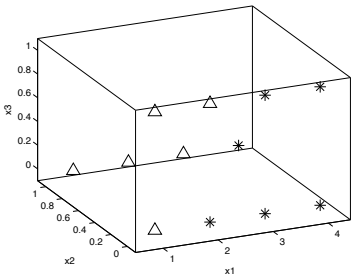
Define $\mathbf{u}_q$ as the sequence of $(K-1)$ symbols $0,\ldots,0,h,0,\ldots,0$, with $h>0$ in the q-th position. Considering the problem of separating K ordered classes $\mathcal{C}_1,\ldots,\mathcal{C}_K$ with training set $\{\mathbf{x}_i^{(k)}\}$, define a new binary training data set in $\mathbb{R}^{p+K-1}$ as



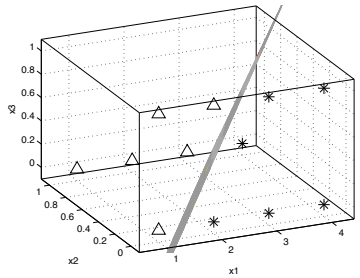
(a) Original data set in $\mathbb{R}$, $K = 4$.



(b) Data set in $\mathbb{R}^3$, with samples replicated ($h = 1$).



(c) Transformation into a binary classification problem.



(d) Linear solution to the binary problem.

Figure 5: Proposed data extension model for a data set in $\mathbb{R}$, with $K = 4$.

$$
\begin{aligned}
\begin{bmatrix} \mathbf{x}_i^{(k)} \\ \mathbf{u}_1 \end{bmatrix} &\in \begin{cases} \overline{C}_1 & k = 1, \\ \overline{C}_2 & k = 2, \ldots, \min(K, 1+s), \end{cases} \\
&\vdots \\
\begin{bmatrix} \mathbf{x}_i^{(k)} \\ \mathbf{u}_q \end{bmatrix} &\in \begin{cases} \overline{C}_1 & k = \max(1, q-s+1), \ldots, q, \\ \overline{C}_2 & k = q+1, \ldots, \min(K, q+s), \end{cases} \\
&\vdots \\
\begin{bmatrix} \mathbf{x}_i^{(k)} \\ \mathbf{u}_{K-1} \end{bmatrix} &\in \begin{cases} \overline{C}_1 & k = \max(1, K-1-s+1), \ldots, K-1, \\ \overline{C}_2 & k = K. \end{cases}
\end{aligned}
$$

Note that the homogeneous extended data set has dimension $p+K-1$, as opposed to $(p+K-2)$ in the standard formulation of the data replication method. It also comes that $b_i = h w_{p+i}$, $i = 1, \ldots, K-1$. Under the homogeneous approach, one has to look for a linear homogeneous boundary of the form $\bar{\mathbf{w}}^t \bar{\mathbf{x}} = 0$, as the bias of the boundary is incorporated as a new coordinate.

The main reason for preferring the standard over the homogeneous formulation of the data replication method is that most of the existing linear binary classifiers are formulated in terms of non-homogeneous boundaries having the form $\bar{\mathbf{w}}^t\bar{\mathbf{x}}+b=0$, instead of $\bar{\mathbf{w}}^t\bar{\mathbf{x}}=0$. Therefore, some adaptation is required before applying existing linear binary classifiers to the homogeneous data replication method.

Homogeneous Data Replication Method with Explicit Constrains on the Thresholds

Unless one sets $s=K-1$, the data replication method does not enforce ordered thresholds (we will return to this point later, when mapping to SVMs and neural networks). This is true for both the standard and the homogeneous formulations. Explicit constraints in the model's formulation can be introduced to enforce the correct order. With the homogeneous formulation, those explicit constraints can take the form of additional $(K-2)$ points in the training set.

Consider the relation $-b_i<-b_{i+1}$. This relation can be equivalently written as

$$-hw_{p+i}<-hw_{p+i+1}\Longleftrightarrow-\bar{\mathbf{w}}^t\begin{bmatrix}\mathbf{0}_p\\ \mathbf{u}_i\end{bmatrix}<-\bar{\mathbf{w}}^t\begin{bmatrix}\mathbf{0}_p\\ \mathbf{u}_{i+1}\end{bmatrix}\Longleftrightarrow\bar{\mathbf{w}}^t\begin{bmatrix}\mathbf{0}_p\\ \mathbf{u}_{i+1}-\mathbf{u}_i\end{bmatrix}<0.$$

As a result, constraining $-b_i$ to be less than $-b_{i+1}$ is equivalent to correctly classify the point $\begin{bmatrix}\mathbf{0}_p\\ \mathbf{u}_{i+1}-\mathbf{u}_i\end{bmatrix}$ in the $\mathcal{C}_1$ class. It is interesting to note that this point is not in the subspace of any of the data replicas. To introduce the $(K-2)$ explicit constraints on the thresholds, just enforce that the $(K-2)$ points $\begin{bmatrix}\mathbf{0}_p\\ \mathbf{u}_{i+1}-\mathbf{u}_i\end{bmatrix}, i=1,\ldots,K-2$ are correctly classified in $\mathcal{C}_1$. Note that violations of these constraints can not be allowed.

2.3 Data Replication Method—the Nonlinear Case

Previously, the data replication method was considered as a design methodology of a linear classifier for ordinal data. This section addresses scenarios where data is not linearly separable and for which the design of a linear classifier does not lead to satisfactory results. Therefore, the design of nonlinear classifiers emerges as a necessity. The only constraint to enforce during the design process is that boundaries should not intersect.

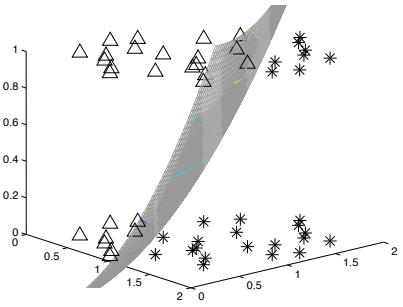
Inspired by the data replication method just presented, we now look for generic boundaries that are *level curves* of some nonlinear, real-valued function $G(\mathbf{x})$ defined in the feature space. The $(K-1)$ boundaries are defined as $G(\mathbf{x})=b_i, i=1,\ldots,K-1, b_i\in\mathbb{R}$. It is worth emphasizing that interpreting the decision boundaries as level curves of some (unknown) function does not result in loss of generality. For the linear version one take $G(\mathbf{x})=\mathbf{w}^t\mathbf{x}+b$.

Once again, the search for nonintersecting, nonlinear boundaries can be carried out in the extended space of the data replication method. First, extend and modify the feature space to a binary problem, as dictated by the data replication method. Next, search for a boundary $\overline{G}(\overline{\mathbf{x}})$ defined in the extended space that results on $(K-1)$ boundaries $G(\mathbf{x})=b_i$ when reverted to the original space. The simplest form for $\overline{G}(\overline{\mathbf{x}})$ is as a partially linear (nonlinear in the original variables but linear in the introduced variables) boundary $\overline{G}(\overline{\mathbf{x}})=G(\mathbf{x})+\underline{\mathbf{w}}^t\mathbf{e}_i=0$, with $\underline{\mathbf{w}}\in\mathbb{R}^{K-2}$, and $\overline{\mathbf{x}}=[\begin{smallmatrix}\mathbf{x}\\ \mathbf{e}_i\end{smallmatrix}]$. Notice that restricting the function $\overline{G}(\overline{\mathbf{x}})$ to be linear in the $(K-2)$ added variables imposes automatically nonintersecting boundaries in the original space: boundaries will have the form $G(\mathbf{x})+b_i=0$ with

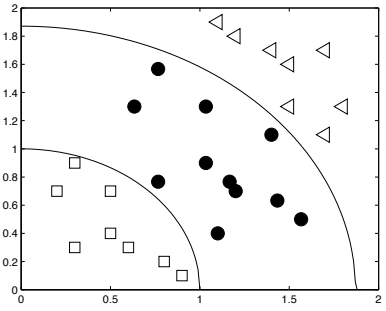
$$b_i=\begin{cases}0 & \text{if } i=1,\\ hw_{p+i-1} & \text{if } 2\le i\le K-1.\end{cases}$$

Although a partially linear function $\overline{G}(\overline{\mathbf{x}})$ is the simplest to provide nonintersecting boundaries in the original space (level curves of some function $G(\mathbf{x})$), it is by no means the only type of function to provide them.

The intersection of the constructed high-dimensional boundary with each of the subspace replicas provides the desired $(K-1)$ boundaries. This approach is plotted in Figure 6 for the toy example. The $\#\overline{C}_2+1$ rule can still be applied to predict the class of a test example directly in the extended feature space.



(a) Nonlinear solution to the binary problem. $\overline{G}(\overline{\mathbf{x}}) = 0.4(x_1^2+x_2^2-1)+x_3$



(b) Nonlinear solution in the original data set. $G(\mathbf{x}) = x_1^2+x_2^2-1$

Figure 6: Nonlinear data extension model in the toy example.

The nonlinear extension of the homogeneous data replication method follows the same rationale as the standard formulation. Now the $\overline{G}(\overline{\mathbf{x}}) = G(\mathbf{x}) + \underline{\mathbf{w}}^t\mathbf{u}_i = 0$ boundary, with $\underline{\mathbf{w}} \in \mathbb{R}^{K-1}$, must be constrained such that $G(\mathbf{0}) = 0 \Longleftrightarrow \overline{G}(\overline{\mathbf{0}}) = 0$. Finally, the enforcement of ordered thresholds with the introduction of additional training points is still valid in the nonlinear case, as

$$\overline{G}(\begin{bmatrix}\mathbf{0}_p \\ \mathbf{u}_{i+1}-\mathbf{u}_i\end{bmatrix}) = G(\mathbf{0}_p) + \underline{\mathbf{w}}^t(\mathbf{u}_{i+1}-\mathbf{u}_i) = hw_{p+i+1} - hw_{p+i} = b_{i+1} - b_i.$$

2.4 A General Framework

As presented so far, the data replication method allows only searching for parallel hyperplanes (*level curves* in the nonlinear case) boundaries. That is, a single direction is specified for all boundaries. In the quest for an extension allowing more loosely coupled boundaries, let us start by reviewing the method for ordinal data by Frank and Hall (2001).

2.4.1 THE METHOD OF FRANK AND HALL

Frank and Hall (2001) proposed to use $(K-1)$ standard binary classifiers to address the K-class ordinal data problem. Toward that end, the training of the i-th classifier is performed by converting the ordinal data set with classes $\mathcal{C}_1,\ldots,\mathcal{C}_K$ into a binary data set, discriminating $\mathcal{C}_1,\ldots,\mathcal{C}_i$ against $\mathcal{C}_{i+1},\ldots,\mathcal{C}_K$. To predict the class value of an unseen instance, the $(K-1)$ outputs are combined to produce a single estimation. If the i-th classifier predicts $\mathcal{C}_X > \mathcal{C}_i$ with probability p_i, Frank and

Hall (2001) suggest to estimate the probability values of each of the K classes as

$$\begin{aligned} p_{C_1} &= 1 - p_1, \\ p_{C_j} &= p_{j-1} - p_j \quad j = 2, \cdots, K-1, \\ p_{C_K} &= p_{K-1}. \end{aligned}$$

Note however that this approach may lead to negative estimates of probability values. A solution to that problem is to identify the output p_i of the i-th classifier with the conditional probability $p(\mathcal{C}_X > \mathcal{C}_i \,|\, \mathcal{C}_X > \mathcal{C}_{i-1})$. This meaning can be exploited to rank the classes according to the following formulas:

$$\begin{aligned} p(\mathcal{C}_X > \mathcal{C}_1) &= p_1, \\ p(\mathcal{C}_X > \mathcal{C}_j) &= p_j\, p(\mathcal{C}_X > \mathcal{C}_{j-1}), \end{aligned} \qquad \begin{aligned} p_{C_1} &= 1 - p_1, \\ p_{C_j} &= (1 - p_j)\, p(\mathcal{C}_X > \mathcal{C}_{j-1}) \quad j = 2, \cdots, K-1, \\ p_{C_K} &= p(\mathcal{C}_X > \mathcal{C}_{K-1}). \end{aligned}$$

Any binary classifier can be used as the building block of this scheme. Observe that, under our approach, the i-th boundary is also discriminating $\mathcal{C}_1, \ldots, \mathcal{C}_i$ against $\mathcal{C}_{i+1}, \ldots, \mathcal{C}_K$; the major difference lies in the *independence* of the boundaries found with Frank and Hall's method. This independence is likely to lead to intersecting boundaries.

2.4.2 A PARAMETERIZED FAMILY OF CLASSIFIERS

Thus far, nonintersecting boundaries have been motivated as the best way to capture ordinal relation among classes. That may be a too restrictive condition for problems where some features are not in relation with the ordinal property. Suppose then that the order of the classes is not totally reflected in a subset of the features. Without further information, it is unadvised to draw from them any ordinal information. It may be more advantageous to restrict the enforcing of the nonintersecting boundaries to the leftover features.

We suggest a generalization of the data replication method where the enforcement of nonintersecting boundaries is restricted only to the first j features, while the last $p - j$ features enjoy the independence as materialized in the Frank and Hall's method. Towards that end we start by showing how the independent boundaries approach of Frank and Hall can be subsumed in the data replication framework.

Instead of replicating the original train data set as expressed by Eq. (2), we indent to arrive at a strategy that still allows a single binary classifier to solve the $(K-1)$ classification problems simultaneously, but yielding independent boundaries. If the boundaries are expected to be independent, each of the $(K-1)$ data replicas of the original data should be made as 'independent' as possible.

Up to this point, when replicating the original data set, the original p variables were the first p variables of the $p+K-2$ variables of the augmented data set, for each subspace replica, as seen in Eq. (2). Each of the $(K-2)$ extra variables accounts for a different threshold term, while the fact that the original variables are 'shared' among the different replicas results in a common direction for all of the boundaries. The argument is that if the set of the original p features is mapped into a different set of p variables for each data replica, while keeping the $(K-2)$ extra variables to account for different thresholds, the binary classifier will return (almost) independent boundaries. It is worth noticing that this procedure increases the number of variables in the extended space to $(K-1) \times p + (K-2)$.

Returning to the toy example, assume that the replication was done *not* according to Eq. (1) but instead using the following rule:

$$\begin{bmatrix} \mathbf{x}_i^{(1)} \\ \mathbf{0}_2 \\ 0 \end{bmatrix} \in \overline{\mathcal{C}}_1, \begin{bmatrix} \mathbf{x}_i^{(2)} \\ \mathbf{0}_2 \\ 0 \end{bmatrix}, \begin{bmatrix} \mathbf{x}_i^{(3)} \\ \mathbf{0}_2 \\ 0 \end{bmatrix} \in \overline{\mathcal{C}}_2, \qquad \begin{bmatrix} \mathbf{0}_2 \\ \mathbf{x}_i^{(1)} \\ h \end{bmatrix}, \begin{bmatrix} \mathbf{0}_2 \\ \mathbf{x}_i^{(2)} \\ h \end{bmatrix} \in \overline{\mathcal{C}}_1, \begin{bmatrix} \mathbf{0}_2 \\ \mathbf{x}_i^{(3)} \\ h \end{bmatrix} \in \overline{\mathcal{C}}_2$$

where $\mathbf{0}_2$ is the sequence of 2 zeros. Intuitively, by misaligning variables involved in the determination of different boundaries (variables in different subspaces), we are decoupling those same boundaries.

Proceeding this way, boundaries can be designed almost independently (the mapping on SVMs will clarify this issue). In the linear case we have now four parameters to estimate, the same as for two independent lines in $\mathbb{R}^2$. Intuitively, this new rule to replicate the data allows the estimation of the direction of each boundary in *essentially* an independent way.

The general formulation in Eq. (2) becomes

$$\begin{array}{ll} \begin{bmatrix} \mathbf{x}_i^{(k)} \\ \mathbf{0}_{p(K-2)} \\ \mathbf{e}_0 \end{bmatrix} & \in \begin{cases} \overline{\mathcal{C}}_1 & k = 1, \\ \overline{\mathcal{C}}_2 & k = 2, \ldots, \min(K, 1+s), \end{cases} \\ \vdots & \\ \begin{bmatrix} \mathbf{0}_{p(q-1)} \\ \mathbf{x}_i^{(k)} \\ \mathbf{0}_{p(K-q-1)} \\ \mathbf{e}_{q-1} \end{bmatrix} & \in \begin{cases} \overline{\mathcal{C}}_1 & k = \max(1, q-s+1), \ldots, q, \\ \overline{\mathcal{C}}_2 & k = q+1, \ldots, \min(K, q+s), \end{cases} \\ \vdots & \\ \begin{bmatrix} \mathbf{0}_{p(K-2)} \\ \mathbf{x}_i^{(k)} \\ \mathbf{e}_{K-2} \end{bmatrix} & \in \begin{cases} \overline{\mathcal{C}}_1 & k = \max(1, K-1-s+1), \ldots, K-1, \\ \overline{\mathcal{C}}_2 & k = K, \end{cases} \end{array} \tag{3}$$

where $\mathbf{0}_*$ is the sequence of $*$ zeros.

While the basic linear data replication method requires the estimation of $(p-1)+(K-1)$ parameters, the new rule necessitates of $(p-1)(K-1)+(K-1) = p(K-1)$, the same as the Frank and Hall approach; this corresponds to the number of free parameters in $(K-1)$ independent p-dimensional hyperplanes. While this does not aim at being a practical alternative to Frank's method, it does pave the way for intermediate solutions, filling the gap between the totally coupled and totally independent boundaries.

To constrain only the first j variables of the p initial variables to have the same direction in all boundaries, while leaving the $(p-j)$ final variables unconstrained, we propose to extend the data according to

$$
\begin{bmatrix} \mathbf{x}_i^{(k)}(1{:}j) \\ \mathbf{x}_i^{(k)}(j{+}1{:}p) \\ \mathbf{0}_{(p-j)(K-2)} \\ \mathbf{e}_0 \end{bmatrix} \in \begin{cases} \overline{\mathcal{C}}_1 & k=1, \\ \overline{\mathcal{C}}_2 & k=2,\ldots,\min(K,1+s), \end{cases}
$$

$$
\vdots
$$

$$
\begin{bmatrix} \mathbf{x}_i^{(k)}(1{:}j) \\ \mathbf{0}_{(p-j)(q-1)} \\ \mathbf{x}_i^{(k)}(j{+}1{:}p) \\ \mathbf{0}_{(p-j)(K-q-1)} \\ \mathbf{e}_{q-1} \end{bmatrix} \in \begin{cases} \overline{\mathcal{C}}_1 & k=\max(1,q-s+1),\ldots,q, \\ \overline{\mathcal{C}}_2 & k=q+1,\ldots,\min(K,q+s), \end{cases}
$$

$$
\vdots
$$

$$
\begin{bmatrix} \mathbf{x}_i^{(k)}(1{:}j) \\ \mathbf{0}_{(p-j)(K-2)} \\ \mathbf{x}_i^{(k)}(j{+}1{:}p) \\ \mathbf{e}_{K-2} \end{bmatrix} \in \begin{cases} \overline{\mathcal{C}}_1 & k=\max(1,K-1-s+1),\ldots,K-1, \\ \overline{\mathcal{C}}_2 & k=K. \end{cases}
$$

With this rule $[p-1-(j-1)](K-1)+(K-1)+j-1$, $j \in \{1,\ldots,p\}$, parameters are to be estimated.

This general formulation of the data replication method allows the enforcement of only the amount of knowledge (constraints) that is effectively known *a priori*, building the right amount of parsimony into the model (see the pasture production experiment).

Now, in this general setting, we can no longer assume nonintersecting boundaries. Therefore, the space of features may be partitioned in more than K regions. To predict the class of an unseen instance we may estimate the probabilities of the K classes using the $(K-1)$ replicas, similarly to Frank and Hall (2001), or simply keep the $\#\overline{\mathcal{C}}_2+1$ rule.

3. Mapping the Data Replication Method to Learning Algorithms

In this section the data replication method just introduced is instantiated in two important machine learning algorithms: support vector machines and neural networks.

3.1 Mapping the Data Replication Method to SVMs

The learning task in a classification problem is to select a prediction function $f(\mathbf{x})$ from a family of possible functions that minimizes the expected *loss*.

In the absence of reliable information on relative costs, a natural approach for unordered classes is to treat every misclassification as equally likely. This translates into adopting the non-metric indicator function $l_{0-1}(f(\mathbf{x}),y)=0$ if $f(\mathbf{x})=y$ and $l_{0-1}(f(\mathbf{x}),y)=1$ if $f(\mathbf{x})\neq y$, where $f(\mathbf{x})$ and y are the predicted and true classes, respectively. Measuring the performance of a classifier using the l_{0-1} loss function is equivalent to simply considering the misclassification error rate. However, for ordered classes, losses that increase with the absolute difference between the class numbers are more natural choices in the absence of better information (Mathieson, 1995). This loss should be naturally incorporated during the training period of the learning algorithm.

A risk functional that takes into account the ordering of the classes can be defined as

$$
R(f)=\mathbf{E}\left[l^s\left(f(\mathbf{x}^{(k)}),k\right)\right] \tag{4}
$$

with

$$
l^s\left(f(\mathbf{x}^{(k)}),k\right)=\min\left(|f(\mathbf{x}^{(k)})-k|,s\right).
$$

The empirical risk is the average of the number of mistakes, where the magnitude of a mistake is related to the total ordering: $R^s_{emp}(f) = \frac{1}{\ell}\sum_{k=1}^{K}\sum_{i=1}^{\ell_k} l^s\left(f(\mathbf{x}_i^{(k)}),k\right)$.

Arguing as Herbrich et al. (1999a), we see that the role of parameter s (bounding the loss incurred in each example) is to allow for an incorporation of a priori knowledge about the probability of the classes, conditioned by $\mathbf{x}$, $P(\mathcal{C}_k|\mathbf{x})$. This can be treated as an assumption on the concentration of the probability around a "true" rank. Let us see how all this finds its place with the data replication method.

3.1.1 THE MINIMUM MARGIN PRINCIPLE

Let us formulate the problem of separating K ordered classes $\mathcal{C}_1,\ldots,\mathcal{C}_K$ in the spirit of SVMs. Starting from the generalization of the two-class separating hyperplane presented in the beginning of previous section, let us look for $(K-1)$ parallel hyperplanes represented by vector $\mathbf{w}\in\mathbb{R}^p$ and scalars $b_1,\ldots,b_{K-1}$, such that the feature space is divided into K regions by the decision boundaries $\mathbf{w}^t\mathbf{x}+b_r=0,\ r=1,\ldots,K-1$.

Going for a strategy to maximize the margin of the closest pair of classes, the goal becomes to maximize $\min|\mathbf{w}^t\mathbf{x}+b_i|/||\mathbf{w}||$. Recalling that an algebraic measure of the distance of a point to the hyperplane $\mathbf{w}^t\mathbf{x}+b$ is given by $(\mathbf{w}^t\mathbf{x}+b)/\|\mathbf{w}\|$, we can scale $\mathbf{w}$ and b_i so that the value of the minimum margin is $2/\|\mathbf{w}\|$.

The constraints to consider result from the $(K-1)$ binary classifications related to each hyperplane; the number of classes involved in each binary classification can be made dependent on a parameter s, as detailed in Section 2.1. For the hyperplane $q\in\{1,\ldots,K-1\}$, the constraints result as

$$\begin{array}{ll} -(\mathbf{w}^t\mathbf{x}_i^{(k)}+b_q) & \geq +1 \quad k=\max(1,q-s+1),\ldots,q,\\ +(\mathbf{w}^t\mathbf{x}_i^{(k)}+b_q) & \geq +1 \quad k=q+1,\ldots,\min(K,q+s). \end{array} \tag{5}$$

Reasoning as in the two-class SVM for the non-linearly separable data set, the inequalities can be relaxed using slack variables and the cost function modified to penalise any failure to meet the original (strict) inequalities. The model becomes (where $\text{sgn}\,(x)$ returns $+1$ if x is greater than zero; 0 if x equals zero; -1 if x is less than zero)

$$\begin{array}{lll} \min\limits_{\mathbf{w},b_i,\xi_i} & \frac{1}{2}\mathbf{w}^t\mathbf{w}+C\sum\limits_{q=1}^{K-1}\sum\limits_{k=\max(1,q-s+1)}^{\min(K,q+s)}\sum\limits_{i=1}^{\ell_k}\text{sgn}\,(\xi_{i,q}^{(k)}) & \\ & -(\mathbf{w}^t\mathbf{x}_i^{(k)}+b_1) \geq +1-\xi_{i,1}^{(k)} & k=1,\\ & +(\mathbf{w}^t\mathbf{x}_i^{(k)}+b_1) \geq +1-\xi_{i,1}^{(k)} & k=2,\ldots,\min(K,1+s),\\ & \vdots & \\ & -(\mathbf{w}^t\mathbf{x}_i^{(k)}+b_q) \geq +1-\xi_{i,q}^{(k)} & k=\max(1,q-s+1),\ldots,q,\\ s.t. & +(\mathbf{w}^t\mathbf{x}_i^{(k)}+b_q) \geq +1-\xi_{i,q}^{(k)} & k=q+1,\ldots,\min(K,q+s),\\ & \vdots & \\ & -(\mathbf{w}^t\mathbf{x}_i^{(k)}+b_{K-1}) \geq +1-\xi_{i,K-1}^{(k)} & k=\max(1,K-s),\ldots,K-1,\\ & +(\mathbf{w}^t\mathbf{x}_i^{(k)}+b_{K-1}) \geq +1-\xi_{i,K-1}^{(k)} & k=K,\\ & \xi_{i,q}^{(k)}\geq 0. & \end{array} \tag{6}$$

Since each point $\mathbf{x}_i^{(k)}$ is replicated $2s$ times, it is also involved in the definition of $2s$ boundaries; consequently, it can be shown to be misclassified $\min(|f(\mathbf{x}_i^{(k)}) - k|, s) = l^s(f(\mathbf{x}_i^{(k)}), k)$ times, where $f(\mathbf{x}_i^{(k)})$ is the class estimated by the model. As with the two-class example, $\sum_{q=1}^{K-1} \sum_{k=\max(1,q-s+1)}^{\min(K,q+s)} \sum_{i=1}^{\ell_k} \text{sgn}\,(\xi_{i,q}^{(k)})$ is an upper bound of $\sum_k \sum_i l^s(f(\mathbf{x}_i^{(k)}), k)$, proportional to the empirical risk.[2]

However, optimization of the above is difficult since it involves a discontinuous function sgn (). As it is common in such cases, we choose to optimize a closely related cost function, and the goal becomes

$$\min_{\mathbf{w},b_i,\xi_i} \quad \frac{1}{2}\mathbf{w}^t\mathbf{w} + C \sum_{q=1}^{K-1} \sum_{k=\max(1,q-s+1)}^{\min(K,q+s)} \sum_{i=1}^{\ell_k} \xi_{i,q}^{(k)}$$

subject to the same constraints as Eq. (6).

In order to account for different misclassification costs or sampling bias, the model can be extended to penalise the slack variables according to different weights in the objective function (Lin et al., 2002):

$$\min_{\mathbf{w},b_i,\xi_i} \quad \frac{1}{2}\mathbf{w}^t\mathbf{w} + \sum_{q=1}^{K-1} \sum_{k=\max(1,q-s+1)}^{\min(K,q+s)} \sum_{i=1}^{\ell_k} C_{i,q}^{(k)} \xi_{i,q}^{(k)}.$$

As easily seen, the proposed formulation resembles the fixed margin strategy in Shashua and Levin (2002). However, instead of using only the two closest classes in the constraints of an hyperplane, more appropriate for the loss function $l_{0-1}()$, we adopt a formulation that captures better the performance of a classifier for ordinal data.

Some problems were identified in the Shashua's approach. Firstly, it is an incompletely specified model. In fact, although the direction of the hyperplanes $\mathbf{w}$ is unique under the above formulation (proceeding as Vapnik (1998) for the binary case), the scalars $b_1, \ldots, b_{K-1}$ are not uniquely defined, as illustrated in Figure 7.

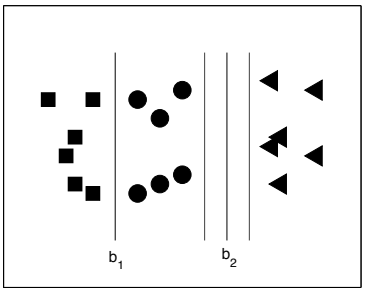


Figure 7: Scalar b_2 is undetermined over an interval under the fixed margin strategy.

Secondly, setting $s < (K-1)$ may produce awkward results on some unfortunate cases. In fact, as pointed out by Chu and Keerthi (2005), the ordinal inequalities on the thresholds $-b_1 \leq -b_2 \leq$

2. Two parameters named s have been introduced. In Section 2.1 the s parameter bounds the number of classes involved in the definition of each boundary, controlling the growth of the original data set. The parameter s introduced in Eq. (4) bounds the loss incurred in each example. Here we see that they are the same parameter.

$\ldots \leq -b_{K-1}$ are not guaranteed in this case. Only under the setting $s = K-1$ the ordinal inequalities on the thresholds are automatically satisfied (Chu and Keerthi, 2005).

Lastly, although the formulation was constructed from the two-class SVM, it is no longer solvable with the same algorithms. It would be interesting to accommodate this formulation under the two-class problem. That would allow the use of mature and optimized algorithms, developed for the training of support vector machines (Platt, 1998; Dong et al., 2005).

3.1.2 THE OSVM ALGORITHM

In order to get a better intuition of the general result, consider first the toy example previously presented. The binary SVM formulation for the extended and binarized training set can be described as $\left(\text{with } \overline{\mathbf{w}} = \left[\begin{smallmatrix}\mathbf{w}\\ w_3\end{smallmatrix}\right],\ \mathbf{w} \in \mathbb{R}^2\right)$

$$
\begin{aligned}
&\min_{\overline{\mathbf{w}},b} \quad \tfrac{1}{2}\overline{\mathbf{w}}^t\overline{\mathbf{w}} \\
&s.t. \quad
\begin{aligned}
&-(\overline{\mathbf{w}}^t \begin{bmatrix}\mathbf{x}_i^{(1)}\\ 0\end{bmatrix} + b) \geq +1,\\
&+(\overline{\mathbf{w}}^t \begin{bmatrix}\mathbf{x}_i^{(2)}\\ 0\end{bmatrix} + b) \geq +1,\\
&+(\overline{\mathbf{w}}^t \begin{bmatrix}\mathbf{x}_i^{(3)}\\ 0\end{bmatrix} + b) \geq +1,\\
&-(\overline{\mathbf{w}}^t \begin{bmatrix}\mathbf{x}_i^{(1)}\\ h\end{bmatrix} + b) \geq +1,\\
&-(\overline{\mathbf{w}}^t \begin{bmatrix}\mathbf{x}_i^{(2)}\\ h\end{bmatrix} + b) \geq +1,\\
&+(\overline{\mathbf{w}}^t \begin{bmatrix}\mathbf{x}_i^{(3)}\\ h\end{bmatrix} + b) \geq +1.
\end{aligned}
\end{aligned}
$$

But because

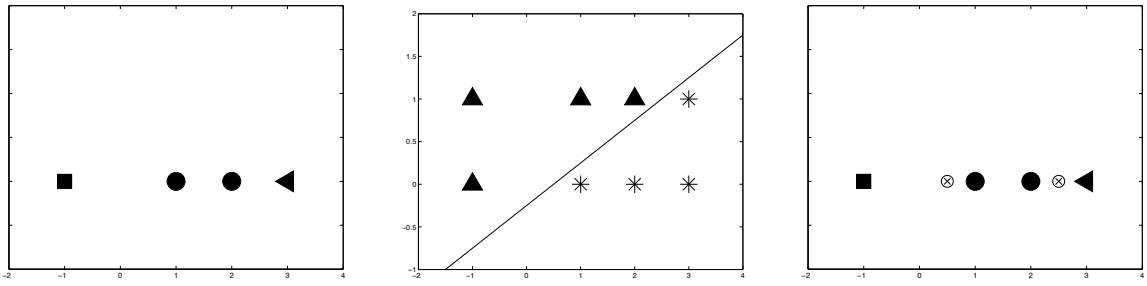
$$
\begin{cases}
\overline{\mathbf{w}}^t \left[\begin{smallmatrix}\mathbf{x}_i\\ 0\end{smallmatrix}\right] = \mathbf{w}^t\mathbf{x}_i \\
\overline{\mathbf{w}}^t \left[\begin{smallmatrix}\mathbf{x}_i\\ h\end{smallmatrix}\right] = \mathbf{w}^t\mathbf{x}_i + w_3 h
\end{cases}
$$

and renaming b to b_1 and $b + w_3 h$ to b_2 the formulation above simplifies to

$$
\begin{aligned}
&\min_{\mathbf{w},b_1,b_2} \quad \tfrac{1}{2}\mathbf{w}^t\mathbf{w} + \tfrac{1}{2}\tfrac{(b_2-b_1)^2}{h^2} \\
&s.t. \quad
\begin{aligned}
&-(\mathbf{w}^t\mathbf{x}_i^{(1)} + b_1) \geq +1,\\
&+(\mathbf{w}^t\mathbf{x}_i^{(2)} + b_1) \geq +1,\\
&+(\mathbf{w}^t\mathbf{x}_i^{(3)} + b_1) \geq +1,\\
&-(\mathbf{w}^t\mathbf{x}_i^{(1)} + b_2) \geq +1,\\
&-(\mathbf{w}^t\mathbf{x}_i^{(2)} + b_2) \geq +1,\\
&+(\mathbf{w}^t\mathbf{x}_i^{(3)} + b_2) \geq +1.
\end{aligned}
\end{aligned}
$$

Two points are worth mentioning: a) this formulation, being the result of a pure SVM method, has an unique solution (Vapnik, 1998); b) this formulation equals the formulation in Eq. (5) for ordinal data previously introduced, with $K = 3$, $s = K-1 = 2$, and a slightly modified objective function by the introduction of a regularization member, proportional to the distance between the hyperplanes. The oSVM solution is the one that simultaneously minimizes the distance between

boundaries and maximizes the minimum of the margins—see Figure 8. The h parameter controls the trade-off between the objectives of maximizing the margin of separation and minimizing the distance between the hyperplanes. To reiterate, the data replication method enabled us to formulate



(a) Original data set in $\mathbb{R}$.

(b) Data set in $\mathbb{R}^2$, with samples replicated ($s = 2$, $h = 1$) and oSVM solution to the binary problem.

(c) oSVM solution in the original feature space.

Figure 8: Effect of the regularization member in the oSVM solution.

the classification of ordinal data as a standard SVM problem and to remove the ambiguity in the solution by the introduction of a regularization term in the objective function.

The insight gained from studying the toy example paves the way for the formal presentation of the instantiation of the data replication method in SVMs. Consider a general extended data set, as defined in Eq. (2). After the simplifications and change of variables suggested for the toy example ($b = b_1$, $b + hw_{p+i} = b_{i+1}$, $i = 1, \ldots, K-2$), the binary SVM formulation for this extended data set yields

$$\min_{\mathbf{w}, b_i, \xi_i} \quad \frac{1}{2}\mathbf{w}^t\mathbf{w} + \frac{1}{h^2}\sum_{i=2}^{K-1}\frac{(b_i - b_1)^2}{2} + C\sum_{q=1}^{K-1}\sum_{k=\max(1,q-s+1)}^{\min(K,q+s)}\sum_{i=1}^{\ell_k}\xi_{i,q}^{(k)}$$

with the same set of constraints as in Eq. (6).

This formulation for the high-dimensional data set matches the proposed formulation for ordinal data up to an additional regularization member in the objective function. This additional member is responsible for the unique determination of the thresholds.[3]

From the equivalence of the instantiation of the data replication method in SVMs and the model in Shashua and Levin (2002) and Chu and Keerthi (2005), the proof on the order of the thresholds is automatically valid for the oSVM algorithm with $s = K - 1$ (see the footnote on page 5 of Chu and Keerthi, 2005). The model parameter s cannot be chosen arbitrarily without additional constraints on the scalars $b_1, \ldots, b_{K-1}$ or some additional information about classes' distribution.

To instantiate the homogeneous data replication method in support vector machines, some popular algorithm for binary SVMs, such as the SMO algorithm, must be adapted for outputting a so-

3. Different regulation members could be obtained by different extensions of the data set. For example, if $\mathbf{e}_q$ had been defined as the sequence $h, \ldots, h, 0, \ldots, 0$, with q h's and $(K-2-q)$ 0's, the regularization member would be $\frac{1}{2}\sum_{i=2}^{i=K-1}\frac{(b_i - b_{i-1})^2}{2}$.

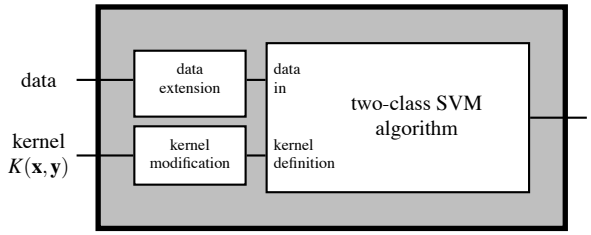


Figure 9: oSVM interpretation of an ordinal multiclass problem as a two-class problem.

lution without the bias term. Moreover, kernels have to be restricted to those satisfying $K(\mathbf{0},\mathbf{0}) = 0$, as for instance the linear kernel or the homogeneous polynomial kernel $K(x,y) = (\mathbf{x}^t\mathbf{y})^d$. To implement the enforcement on the thresholds with additional training points, one can use a different value for the C parameter in these points, sufficiently high to make certain a correct classification. Alternatively, these points must not be relaxed with a slack variable in the SVM formulation.

Nonlinear Boundaries As explained before, the search for nonlinear level curves can be pursued in the extended feature space by searching for a partially linear function $\overline{G}(\overline{\mathbf{x}}) = G(\mathbf{x}) + \underline{\mathbf{w}}^t\mathbf{e}_i$. Since nonlinear boundaries are handled in the SVM context making use of the well known kernel trick, a specified kernel $K(\mathbf{x}_i,\mathbf{x}_j)$ in the original feature space can be easily modified to $\overline{K}(\overline{\mathbf{x}}_i,\overline{\mathbf{x}}_j) = K(\mathbf{x}_i,\mathbf{x}_j) + \mathbf{e}^t_{\mathbf{x}_i}\mathbf{e}_{\mathbf{x}_j}$ in the extended space.

Summarizing, the nonlinear ordinal problem can be solved by extending the feature set and modifying the kernel function, as represented diagrammatically in Figure 9. Clearly, the extension to nonlinear decision boundaries follows the same reasoning as with the standard SVM (Vapnik, 1998).

It is true that the computational complexity of training a SVM model depends on the dimension of the input space, since the kernel functions contain the inner product of two input vectors for the linear or polynomial kernels or the distance of the two vectors for the Gaussian RBF kernel. However the matrix Q of inner products, with $(Q)_{ij} = K(x_i,x_j)$ can be computed once and kept in memory. Even on problems with many training examples, caching strategies can be developed to provide a trade-off between memory consumption and training time (Joachims, 1998). Therefore, most of the increase in the computational complexity of the problem is due to the duplication of the data; more generally, for a K-class problem, the data set is increased at most $(K-1)$ times, $O(\ell(K-1))$.

Independent Boundaries Considering now the setup for independent boundaries, as presented in Eq. (3), the linear, binary SVM formulation yields

$$
\begin{aligned}
\min_{\mathbf{w},b_i,\xi_i} \quad & \sum_{k=1}^{K-1} \frac{1}{2}\mathbf{w}^t(kp-p+1:kp)\mathbf{w}(kp-p+1:kp) + \frac{1}{h^2}\sum_{i=2}^{K-1}\frac{(b_i-b_1)^2}{2} + C\sum_{q=1}^{K-1}\sum_{k=\max(1,q-s+1)}^{\min(K,q+s)}\sum_{i=1}^{\ell_k}\xi_{i,q}^{(k)} \\
s.t. \quad & \begin{aligned}
-(\mathbf{w}^t(1:p)\mathbf{x}_i^{(k)} + b_1) &\geq +1-\xi_{i,1}^{(k)} && k=1, \\
+(\mathbf{w}^t(1:p)\mathbf{x}_i^{(k)} + b_1) &\geq +1-\xi_{i,1}^{(k)} && k=2,\ldots,\min(K,1+s), \\
&\vdots \\
-(\mathbf{w}^t(qp-p+1:qp)\mathbf{x}_i^{(k)} + b_q) &\geq +1-\xi_{i,q}^{(k)} && k=\max(1,q-s+1),\ldots,q, \\
+(\mathbf{w}^t(qp-p+1:qp)\mathbf{x}_i^{(k)} + b_q) &\geq +1-\xi_{i,q}^{(k)} && k=q+1,\ldots,\min(K,q+s), \\
&\vdots \\
-(\mathbf{w}^t((K-1)p-p+1:(K-1)p)\mathbf{x}_i^{(k)} + b_{K-1}) &\geq +1-\xi_{i,K-1}^{(k)} && k=\max(1,K-s),\ldots,K-1, \\
+(\mathbf{w}^t((K-1)p-p+1:(K-1)p)\mathbf{x}_i^{(k)} + b_{K-1}) &\geq +1-\xi_{i,K-1}^{(k)} && k=K, \\
\xi_{i,q}^{(k)} &\geq 0.
\end{aligned}
\end{aligned}
$$

If the regularization term $\frac{1}{h^2}\sum_{i=2}^{K-1}\frac{(b_i-b_1)^2}{2}$ is zero (in practice, small enough), the optimization problem could then be broken in $(K-1)$ *independent* optimization problems, reverting to the procedure of Frank and Hall (2001).

3.2 Mapping the Data Replication Method to NNs

When the nonlinear data replication method was formulated, the *real-valued* function $G(\mathbf{x})$ was defined arbitrarily. Nonintersecting boundaries were enforced by making use of a partially linear function $\overline{G}(\overline{\mathbf{x}}) = G(\mathbf{x}) + \underline{\mathbf{w}}^t\mathbf{e}_i$ defined in the extended space. Setting $G(\mathbf{x})$ as the output of a neural network, a flexible architecture for ordinal data can be devised, as represented diagrammatically in Figure 10. Because $G(\mathbf{x})$ an arbitrary real-valued function, it can be set as the output of a generic neural network with a single output. In Figure 10 $G(\mathbf{x})$ is represented as the output of a generic feedforward network. This value is then linearly combined with the added $(K-2)$ components to produce the desired $\overline{G}(\overline{\mathbf{x}})$ function.

For the simple case of searching for linear boundaries, the overall network simplifies to a single neuron with $p+K-2$ inputs. A less simplified model, also used in the conducted experiments, is to consider a single hidden layer, as depicted in Figure 11. Note that this architecture can be obtained from Figure 10 by collapsing layers 2 to $N-1$ into layer N, a valid operation when activation functions f_2 to f_{N-1} are all linear.

Similarly to the SVM mapping, it is possible to show that, if we allow the samples in all the classes to contribute errors for each threshold, by setting $s = K-1$, the order inequalities on the thresholds are satisfied automatically, in spite of the fact that such constraints on the thresholds are not explicitly included in the formulation. Refer to Appendix B for the detailed proof.

The mapping of the homogeneous data replication method to neural networks is easily realized. In order to obtain $G(\mathbf{0}) = 0$, just remove the biases inputs, represented in Figure 10, and restrict the activation functions $f_i()$ to those verifying $f_i(\mathbf{0}) = 0$. The constrains on the thresholds in form of additional training points can be realized in networks by adapting the performance function to penalise with a sufficiently high value any error in classifying these points.

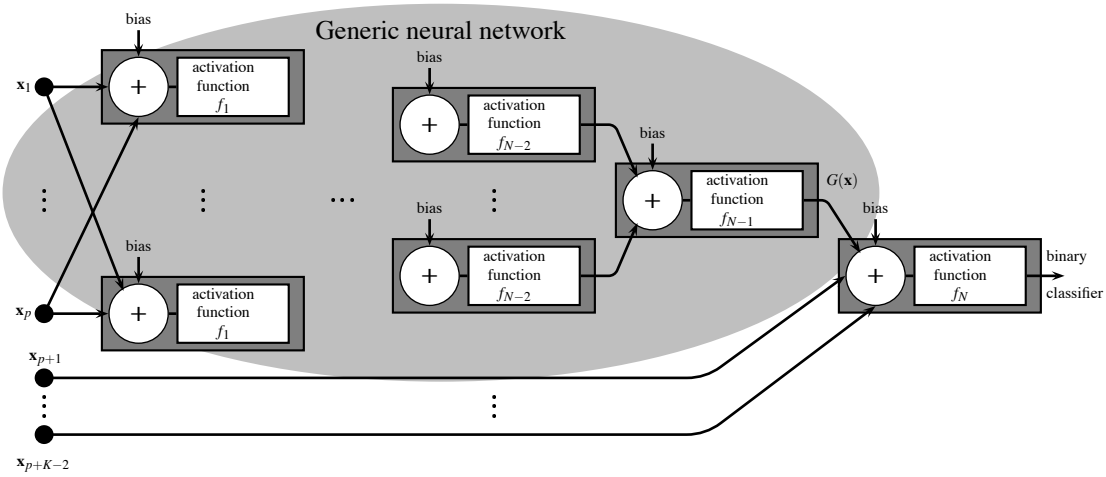


Figure 10: Data replication method for neural networks (oNN).

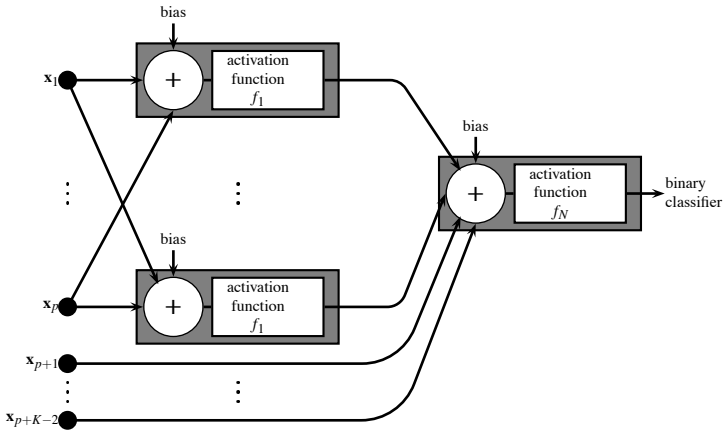


Figure 11: Simplified oNN model for neural networks.

3.2.1 ORDINAL LOGISTIC REGRESSION MODEL

Here we provide a probabilistic interpretation for the ordinal neural network model just introduced. The traditional statistical approach for ordinal classification models the cumulative class probability $P_k = p(C \leq k|\mathbf{x})$ by

$$\text{logit}(P_k) = \Phi_k - G(\mathbf{x}) \Leftrightarrow P_k = \text{logsig}(\Phi_k - G(\mathbf{x})), \quad k = 1, \ldots, K-1 \tag{7}$$

Remember that $\text{logit}(y) = \ln \frac{y}{1-y}$, $\text{logsig(y)} = \frac{1}{1+e^{-y}}$ and $\text{logsig}(\text{logit}(y)) = y$.

For the linear version (McCullagh, 1980; McCullagh and Nelder, 1989) we take $G(\mathbf{x}) = \mathbf{w}^t\mathbf{x}$. Mathieson (1995) presents a nonlinear version by letting $G(\mathbf{x})$ be the output of a neural network. However other setups can be devised. Start by observing that in Eq. (7) we can always assume $\Phi_1 = 0$ by incorporating an appropriate additive constant in $G(\mathbf{x})$. We are left with the estimation of

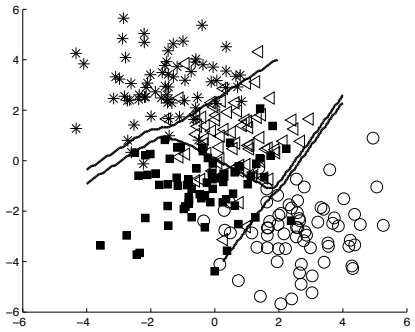


Figure 12: Decision boundaries for the oNN with 3 units in the hidden layer, for a synthetic data set from Mathieson (1995). $\mathcal{C}_1 = \circ$, $\mathcal{C}_2 = \blacksquare$, $\mathcal{C}_3 = \triangleleft$, $\mathcal{C}_4 = *$

$G(\mathbf{x})$ and $(K-2)$ cut points. By fixing $f_N() = \text{logsig}()$ as the activation function in the output layer of our oNN network, we can train the network to predict the values $P_k(\mathbf{x})$, when fed with $\overline{\mathbf{x}} = [\,{}^{\mathbf{x}}_{\mathbf{e}_{k-1}}]$, $k = 1,\ldots,K-1$. By setting $\overline{\mathcal{C}}_1 = 1$ and $\overline{\mathcal{C}}_2 = 0$ we see that the extended data set as defined in Eq. (2) can be used to train the oNN network. The predicted cut points are simply the weights of the connection of the added $K-2$ components, scaled by h.

Illustrating this model with the synthetic data set from Mathieson (1995), we attained the decision boundaries depicted in Figure 12.

3.3 Summation

The data replication method has some advantages over standard algorithms presented in the literature for the classification of ordinal data:

- It has an interesting and intuitive geometric interpretation. It provides a new conceptual framework integrating disparate algorithms for ordinal data classification: Chu and Keerthi (2005) algorithm, Frank and Hall (2001), ordinal logistic regression.

- While the algorithm presented in Shashua and Levin (2002); Chu and Keerthi (2005) is only formulated for SVMs, the data replication method is quite generic, with the possibility of being instantiated in different classes of learning algorithms, ranging from SVMs (or other kernel based approaches) to neural networks or something as simple as the Fisher method.

- Even the SVM instantiation of the data replication method possesses an advantage over the algorithm presented in Chu and Keerthi (2005): the latter misses the explicit inclusion of a regularization term in the objective function, leading to ambiguity in the solution. The data replication method incorporates naturally a regularization term; the unique regularization term allows interpreting the optimization problem as a single binary SVM in an extended space.

4. Experimental Methodology

In the following sections, experimental results are provided for several models based on SVMs and NNs, when applied to diverse data sets, ranging from synthetic to real ordinal data, and to a problem of feature selection. Here, the set of models under comparison is presented and different assessment criteria for ordinal data classifiers are examined.

4.1 Neural Network Based Algorithms

We compare the following algorithms:

- Conventional neural network (cNN). To test the hypothesis that methods specifically targeted for ordinal data improve the performance of a standard classifier, we tested a conventional feed forward network, fully connected, with a single hidden layer, trained with the special activation function *softmax*.

- Pairwise NN (pNN): Frank and Hall (2001) introduced a simple algorithm that enables standard classification algorithms to exploit the ordering information in ordinal prediction problems. First, the data is transformed from a K-class ordinal problem to $(K-1)$ binary problems. To predict the class value of an unseen instance the probabilities of the K original classes are estimated using the outputs from the $(K-1)$ binary classifiers.

- Costa (1996), following a probabilistic approach, proposes a neural network architecture (iNN) that exploits the ordinal nature of the data, by defining the classification task on a suitable space through a "partitive approach". It is proposed a feedforward neural network with $(K-1)$ outputs to solve a K-class ordinal problem. The probabilistic meaning assigned to the network outputs is exploited to rank the elements of the data set.

- Regression model (rNN): as stated in the introduction, regression models can be applied to solve the classification of ordinal data. A common technique for ordered classes is to estimate by regression any ordered *scores* $s_1 \leq \ldots \leq s_K$ by replacing the target class $\mathcal{C}_i$ by the score s_i. The simplest case would be setting $s_i = i, i = 1, \ldots, K$ (Mathieson, 1995; Moody and Utans, 1995; Agarwal et al., 2001). A neural network with a single output was trained to estimate the scores. The class variable $\mathcal{C}_{\mathbf{x}}$ was replaced by the score $s_{\mathbf{x}} = \frac{\mathcal{C}_{\mathbf{x}}-0.5}{K}$ before applying the regression algorithm. These scores correspond to take as target the midvalues of K equal-sized intervals in the range $[0,1)$. The adopted scores are suitable for a sigmoid output transfer function, which always outputs a value in $(0,1)$. In the test phase, if a test query obtains the answer $\hat{s}_{\mathbf{x}}$ the corresponding class is predicted as $\hat{\mathcal{C}}_{\mathbf{x}} = \lfloor K\hat{s}_{\mathbf{x}} \rfloor + 1$.

- Proposed ordinal method (oNN), based on the standard data extension technique, as previously introduced.

Experiments with neural networks were carried out in Matlab 7.0 (R14), making use of the Neural Network Toolbox. All models were configured with a single hidden layer and trained with Levenberg-Marquardt back propagation method, over at most 2000 epochs.

4.2 SVM Based Algorithms

We compare the following algorithms:

- A conventional multiclass SVM formulation (cSVM), as provided by the software implementation LIBSVM 2.8., based on the one-against-one decomposition. The one-against-one decomposition transforms the multiclass problem into a series of $K(K-1)/2$ binary subtasks that can be trained by a binary SVM. Classification is carried out by a voting scheme.

- Pairwise SVM (pSVM): mapping in support vector machines the strategy of Frank and Hall (2001) above mentioned for the pNN model.

- Regression SVM (rSVM): The considered class of support vector regression was that of ν-SVR Scholkopf et al. (2000), as provided by the software implementation LIBSVM 2.8. Note that the model was trained to estimate the scores $s_{\mathbf{x}}$ as defined before in respect to the rNN model. Because ν-SVR does not guarantee outputs $\in [0,1)$, the predicted class $\hat{\mathcal{C}}_{\mathbf{x}} = \lfloor K\hat{s}_{\mathbf{x}} \rfloor + 1$ was properly cropped.

- Proposed ordinal method (oSVM), based on the standard data extension technique, as previously introduced.

All support vector machine models were implemented in C++, using as core the software implementation provided by LIBSVM 2.8.

4.3 Measuring the Performance of Ordinal Data Classifiers

Having built a classifier, the obvious question is "how good is it?". This begs the question of what we mean by good. A common approach is to treat every misclassification as equally costly, adopting the misclassification error rate (MER) criterion to measure the performance of the classifier. However, as already expressed, losses that increase with the absolute difference between the class numbers capture better the fundamental structure of the ordinal problem. The mean absolute deviation (MAD) criterion takes into account the degree of misclassification and is thus a richer criterion than MER. The loss function corresponding to this criterion is $l(f(\mathbf{x}),y) = |f(\mathbf{x}) - y|$. A variant of the above MAD measure is the mean square error (MSE), where the absolute difference is replaced by the square of the difference, $l(f(\mathbf{x}),y) = (f(\mathbf{x}) - y)^2$.

Still, all these measures depend on the number assigned to each class, which is somewhat arbitrary. In order to try to avoid the influence of the numbers chosen to represent the classes on the performance assessment, we can look only at the order relation between true and predicted class numbers. The use of Spearman (r_s) and Kendall's tau-b (τ_b) coefficients, nonparametric rank-order correlation coefficients well established in the literature (Press et al., 1992) are a step forward in that direction.

To get r_s start by ranking the two vectors of true and predicted classes. Ranking is achieved by giving the ranking '1' to the biggest number in a vector, '2' to the second biggest value and so on. Obviously, there will be many examples in the class vector with common values; when ranking, those examples are replaced by average ranks. If $\mathbf{R}$ and $\mathbf{Q}$ represent two rank vectors, then

$$r_s = \frac{\sum(\mathbf{R}_i - \bar{\mathbf{R}})(\mathbf{Q}_i - \bar{\mathbf{Q}})}{\sqrt{\sum(\mathbf{R}_i - \bar{\mathbf{R}})^2 \sum(\mathbf{Q}_i - \bar{\mathbf{Q}})^2}}.$$

To define τ_b, start with the N data points $(\mathcal{C}_{\mathbf{x}_i}, \hat{\mathcal{C}}_{\mathbf{x}_i}), i = 1,\ldots,N$, associated with the true and predicted classes, and consider all $\frac{1}{2}N(N-1)$ pairs of data points. Following the notation in Press

et al. (1992), we call a pair (i, j) *concordant* if the relative ordering of the true classes $\mathcal{C}_{\mathbf{x}_i}$ and $\mathcal{C}_{\mathbf{x}_j}$ is the same as the relative ordering of the predicted classes $\hat{\mathcal{C}}_{\mathbf{x}_i}$ and $\hat{\mathcal{C}}_{\mathbf{x}_j}$. We call a pair *discordant* if the relative ordering of the true classes is opposite from the relative ordering of the predicted classes. If there is a tie in either the true or predicted classes, then we do not call the pair either concordant or discordant. If the tie is in the true classes, we will call the pair an "extra true pair", e_t. If the tie is in the predicted classes, we will call the pair an "extra predicted pair", e_p. If the tie is both on the true and the predicted classes, we ignore the pair. The τ_b coefficient can be computed as

$$\tau_b = \frac{concordant - discordant}{\sqrt{concordant + discordant + e_t}\sqrt{concordant + discordant + e_p}}.$$

Although insensitive to the number assigned to each class, both r_s and τ_b are in fact more appropriate for pairwise ranking rather than to ordinal regression, due to their failure to detect bias errors. In fact, if the predicted class is always a shift by a constant value of the true class, both indices will report perfect performance of the classifier.

Without a clear advantage of one criterion over the others, we decided on employing all the abovementioned assessment criteria in the conducted experiments.

5. Results for Synthetic Data

In a first comparative study we generated a synthetic data set in a similar way to Herbrich et al. (1999b). We generated 1000 example points $\mathbf{x} = [x_1 \; x_2]^t$ uniformly at random in the unit square $[0,1] \times [0,1] \subset \mathbb{R}^2$. Each point was assigned a rank y from the set $\{1,2,3,4,5\}$, according to

$$y = \min_{r \in \{1,2,3,4,5\}} \{r : b_{r-1} < 10(x_1 - 0.5)(x_2 - 0.5) + \varepsilon \; < b_r\},$$

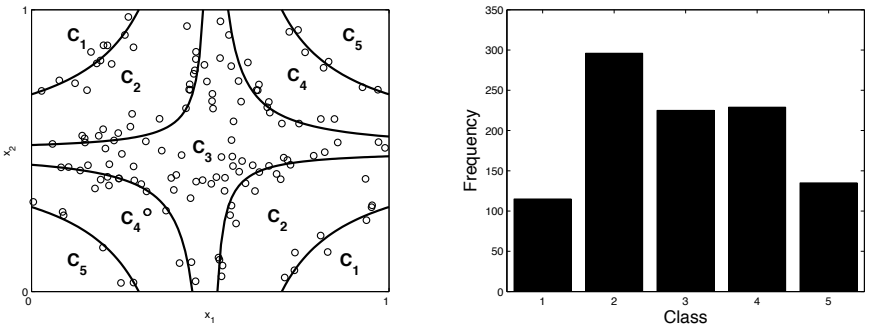
$$(b_0, b_1, b_2, b_3, b_4, b_5) = (-\infty, -1, -0.1, 0.25, 1, +\infty).$$

where $\varepsilon \sim N(0; 0.125^2)$ simulates the possible existence of error in the assignment of the true class to $\mathbf{x}$. Figure 13(a) depicts the 14.2% of examples which were assigned to a wrong class after the addition of ε. The unbalanced distribution of the random variable is shown is Figure 13(b).

In order to compare the different algorithms, we randomly split 100 times the data set into training, validation and test sets. Each model parameterization, namely the C parameter for SVMs (we tried values of the form $C = 1.25^i$, where $i \in \{-8, \ldots, 40\}$) and the number of neurons in the hidden layer for networks (varied from 0 to 10), was selected in accordance with the best mean performance over the 100 setups of the validation set. This was repeated taking $\ell \in \{20, 40, 80\}$ for size of the training set, ℓ for the validation set and $1000 - 2 \times \ell$ for the test set. The test results for SVMs are shown in Table 1, for the MAD criterion.

We also investigated the other introduced criteria to assess models' relative performance. Results depicted in Figure 14 for this synthetic data set are representative of the agreement observed throughout the experimental study. All indices portrayed essentially the same relative models' performance. For this reason, we shall restrict in the following to present only the results for the MAD criterion, possibly the most meaningful criterion for the ordinal regression problem.

The results attained with neural networks based models are presented in Table 2. Notice that the size of the training set, ℓ, was taken in $\{40, 80, 120\}$. The training time of both SVM and NN models represents the total time to search on the parameter range and over the 100 setups.



(a) Scatter plot of the 14.2% data points wrongly classified. Also shown are the class boundaries.

(b) Class distribution.

Figure 13: Synthetic data set with 5 classes in $\mathbb{R}^2$.

Model		Training sets size $\ell = 20$	$\ell = 40$	$\ell = 80$	Training time for $\ell = 80$ (sec)
		MAD			
cSVM		0.47 (0.11)	0.30 (0.05)	0.22 (0.03)	25
pSVM		0.40 (0.10)	0.27 (0.04)	0.22 (0.02)	26
rSVM		0.32 (0.06)	0.26 (0.04)	0.24 (0.03)	2 246
oSVM	$s = 1$	0.29 (0.07)	0.20 (0.03)	0.17 (0.02)	508
	$s = 2$	0.28 (0.07)	0.20 (0.03)	0.17 (0.02)	488
	$s = 4$	0.28 (0.07)	0.20 (0.03)	0.17 (0.02)	449

Table 1: Mean (standard deviation) of MAD over 100 setups of the test set. Configuration: K ($\mathbf{x}$,$\mathbf{y}$) $= (1+\mathbf{x}^t\mathbf{y})^2$, $h = 10$ for oSVM, $\nu = 0.5$ for rSVM.

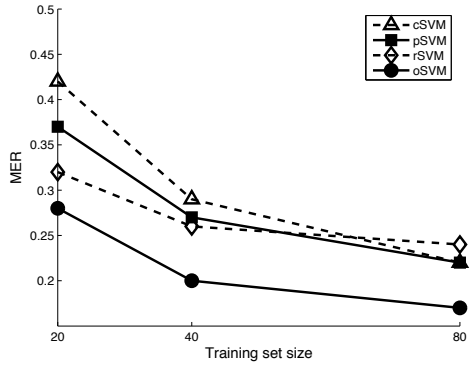
5.1 Accuracy Dependence on the Number of Classes and Data Dimension

To investigate the influence of the number of classes and data dimension on models' relative performance, the described experiment was repeated for a data set with 10 classes in $\mathbb{R}^4$. This time 2000 example points $\mathbf{x} = [x_1\ x_2\ x_3\ x_4]^t$ were generated uniformly at random in the unit square in $\mathbb{R}^4$. The rank of each example was assigned according to the rule

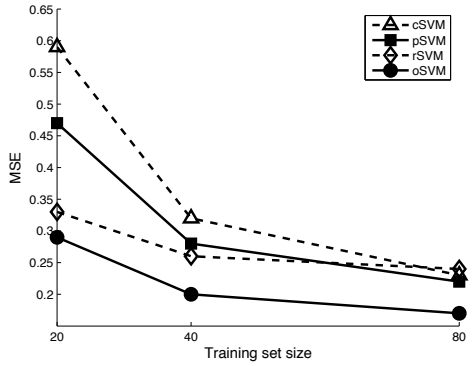
$$y = \min_{r\in\{1,2,3,4,5,6,7,8,9,10\}} \{r : b_{r-1} < 1000\prod_{i=1}^{4}(x_i - 0.5) + \varepsilon\ < b_r\},$$

$$(b_0, b_1, b_2, b_3, b_4, b_5, b_6, b_7, b_8, b_9, b_{10}) \quad = \quad (-\infty, -5, -2.5, -1, -0.4, 0.1, 0.5, 1.1, 3, 6, +\infty).$$

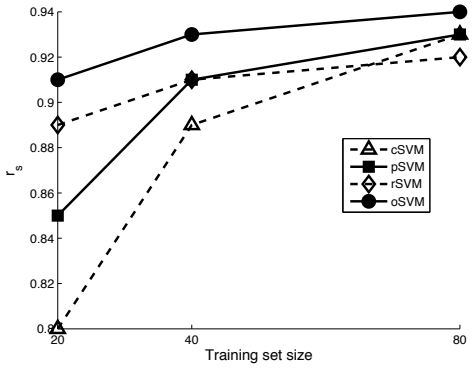
where $\varepsilon \sim N(0; 0.125^2)$. Class distributions are presented in Figure 15.



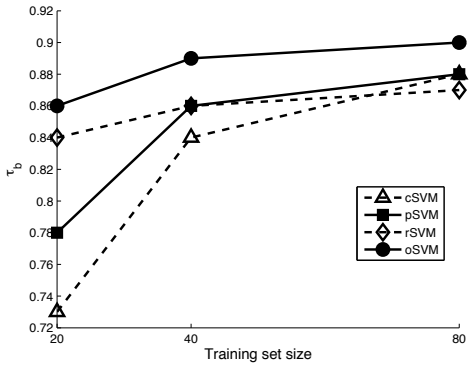
(a) MER coefficient.



(b) MSE coefficient.



(c) Spearman coefficient.



(d) Kendall's tau-b coefficient.

Figure 14: SVMs' results for 5 classes in $\mathbb{R}^2$. Configuration: K $(\mathbf{x},\mathbf{y}) = (1+\mathbf{x}^t\mathbf{y})^2$, $h = 10$ for oSVM, $\nu = 0.5$ for rSVM.

All models were trained following the same methodology as presented in the previous experiment, the only differences being those of a degree 4 for the polynomial kernel taken for the SVMs and the range of the hidden neurons for NNs, now from 0 to 20. Moreover, due to practical considerations, results were averaged only over twenty runs. Tables 3 and 4 show the test results.

5.2 Discussion

The main assertion concerns the superiority of all algorithms specific to ordinal data over conventional methods, both for support vector machines and neural networks. Additionally, neural networks exhibit a slower learning curve than support vector machine models. However, it is true that the nonlinearity selected for the data clearly favours the polynomial kernel of a support vector

Model		Training sets size $\ell = 40$	$\ell = 80$	$\ell = 120$	Training time for $\ell = 120$ (sec)
		MAD			
cNN		0.66 (0.21)	0.58 (0.18)	0.50 (0.18)	11 764
iNN		0.48 (0.13)	0.43 (0.18)	0.38 (0.16)	4 995
pNN		0.42 (0.10)	0.37 (0.09)	0.37 (0.11)	3 814
rNN		0.33 (0.15)	0.24 (0.04)	0.21 (0.03)	4 229
	$s = 1$	0.38 (0.14)	0.28 (0.05)	0.26 (0.08)	11 879
oNN	$s = 2$	0.37 (0.19)	0.30 (0.11)	0.25 (0.08)	11 079
	$s = 4$	0.38 (0.21)	0.30 (0.20)	0.26 (0.10)	11 477

Table 2: Mean (standard deviation) of MAD over 100 setups of the test set ($h = 10$ for oNN).

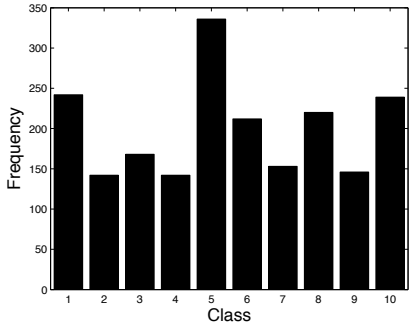


Figure 15: Class distribution for data set with 10 classes in $\mathbb{R}^4$.

Model		Training sets size $\ell = 80$	$\ell = 160$	$\ell = 240$	Training time for $\ell = 240$ (sec)
		MAD			
cSVM		2.26 (0.09)	2.06 (0.10)	1.88 (0.13)	536
pSVM		2.06 (0.13)	1.23 (0.11)	0.84 (0.07)	14 140
rSVM		1.73 (0.23)	1.29 (0.10)	1.20 (0.09)	382 867
	$s = 1$	1.36 (0.24)	0.51 (0.08)	0.33 (0.04)	97 691
oSVM	$s = 2$	1.35 (0.25)	0.50 (0.08)	0.31 (0.03)	113 383
	$s = 9$	1.35 (0.25)	0.50 (0.08)	0.31 (0.03)	115 917

Table 3: Mean (standard deviation) of MAD over twenty setups of the test set. Configuration: K $(\mathbf{x},\mathbf{y}) = (1+\mathbf{x}^t\mathbf{y})^4$, $h = 10$ for oSVM, $\nu = 0.5$ for rSVM.

machine algorithm over the possible models of a neural network with standard activation functions. The proposed data replication method, in spite of being the simplest model, exhibits the best performance among the support vector machine methods and the second best among networks. These

Model		Training sets size $\ell = 240$	$\ell = 320$	$\ell = 480$	Training time for $\ell = 480$ (sec)
		MAD			
cNN		2.28 (0.15)	2.31 (0.39)	2.01 (0.23)	89 347
iNN		1.23 (0.41)	0.87 (0.33)	0.57 (0.34)	89 775
pNN		1.26 (0.13)	1.12 (0.14)	0.93 (0.09)	15 786
rNN		0.59 (0.07)	0.42 (0.03)	0.44 (0.24)	12 216
oNN	$s = 1$	1.06 (0.32)	0.64 (0.15)	0.44 (0.26)	14 618
	$s = 2$	0.80 (0.39)	0.54 (0.07)	0.39 (0.17)	26 057
	$s = 9$	0.79 (0.28)	0.52 (0.27)	0.40 (0.15)	58 381

Table 4: Mean (standard deviation) of MAD over twenty setups of the test set ($h = 10$ for oNN).

conclusions were reinforced with the increase of the dimension and the number of classes of the data, where differences were exacerbated.

Finally, it is worth pointing out that in these experiments no difference is visible among the performance of the different instantiations of the proposed method with SVMs, when trained with $s = 1$, $s = 2$ or $s = K - 1$. Moreover, the difference is only visible for the neural network instantiation of the method when the number of classes is 10. This conclusion is likely to embody the fact that, while with neural networks all training points contribute to the boundary, with support vector machines only a small set of the same—the support vectors—have a say in the solution. Therefore, for well behaved data sets, the extra points intrinsic to $s > 1$ may not contribute to the set of support vectors, and the impact in the final solution be negligible.

6. Classifying Real Ordinal Data

In this section, we continue the experimental study by applying the algorithms considered to the classification of real ordinal data, namely to solving problems of prediction of pasture production and employee selection. The considered data sets are available at the WEKA website (`http://www.cs.waikato.ac.nz/ml/index.html`).

6.1 Pasture Production

The objective related to the pasture data set is to predict pasture production from a variety of biophysical factors. Vegetation and soil variables from areas of grazed North Island Hill Country with different management (fertilizer application/stocking rate) histories (1973-1994) were measured and subdivided into 36 paddocks. Nineteen vegetation (including herbage production); soil chemical, physical and biological; and soil water variables were selected as potentially useful biophysical indicators, totaling 22 attributes. The target feature, the pasture production, has been categorized in three classes (Low, Medium, High), evenly distributed in the data set of 36 instances. Before training, the data was scaled to fall always within the range $[0, 1]$, using the transformation $x' = \frac{x - x_{\min}}{x_{\max} - x_{\min}}$. The fertiliser attribute was represented using 4 variables: LL = $(1, 0, 0, 0)$, LN = $(0, 1, 0, 0)$, HL = $(0, 0, 1, 0)$ and HH = $(0, 0, 0, 1)$.

Continuing with the same experimental methodology, the C parameter was varied from 1.25^{-32} to 1.25^{32}; the number of neurons in the hidden layer for NNs was varied from 0 to 10. The results attained are summarized in Table 5.

Model		Training sets size			Training time for $\ell = 16$ (sec)
		$\ell = 3$	$\ell = 6$	$\ell = 16$	
		MAD			
cSVM		0.44 (0.12)	0.42 (0.13)	0.28 (0.26)	1
pSVM		0.50 (0.12)	0.42 (0.12)	0.26 (0.12)	2
rSVM		0.49 (0.12)	0.38 (0.12)	0.29 (0.21)	5
oSVM	$s = 1$	0.40 (0.10)	0.40 (0.12)	0.34 (0.23)	3
	$s = 2$	0.40 (0.10)	0.40 (0.12)	0.34 (0.23)	3
	$s = 1$, $j = 21$	0.39 (0.37)	0.36 (0.16)	0.21 (0.13)	3
cNN		0.51 (0.19)	0.44 (0.15)	0.29 (0.26)	708
pNN		0.49 (0.17)	0.43 (0.13)	0.26 (0.26)	278
iNN		0.52 (0.18)	0.43 (0.12)	0.34 (0.27)	282
rNN		0.74 (0.24)	0.36 (0.23)	0.26 (0.28)	748
oNN	$s = 1$	0.43 (0.13)	0.41 (0.15)	0.34 (0.26)	195
	$s = 2$	0.45 (0.14)	0.43 (0.15)	0.35 (0.15)	192

Table 5: Mean (standard deviation) of MAD over 100 setups of the test set. Configuration: K ($\mathbf{x}$,$\mathbf{y}$) $= (1 + \mathbf{x}^t\mathbf{y})^2$ for SVMs, $h = 1$ for oSVM and oNN, $\nu = 0.5$ for rSVM.

We start by observing that conventional methods performed as well as ordinal methods. We were led to the suggestion that some of the features may not properly reflect the ordinal relation among classes. A likely exemplar is the fertiliser attribute. The lack of motivation to impose an ordered relation in the fertiliser attribute, suggests a good scenario to apply the general version of the data replication method, where only 21 attributes ($j = 21$) are constrained to have the same direction, with the fertiliser attribute (coded with four binary variables) left free. Using a linear kernel emerges a classifier with expected MAD of 21%. This way, a very simple classifier was obtained at the best performance.

6.2 Employee Selection: the ESL Data Set

The ESL data set contains 488 profiles of applicants for certain industrial jobs. Expert psychologists of a recruiting company, based upon psychometric test results and interviews with the candidates, determined the values of the input attributes (4 attributes, with integer values from 0 to 9). The output is an overall score (1..9) corresponding to the degree of fitness of the candidate to this type of job, distributed according to Figure 16.

Continuing with the same methodology and range of values for the C and the number of hidden neurons parameters from Pasture, one obtained the results reported in Table 6. In this experiment, as well as in all the previous ones, the SVM instantiation of the data replication method exhibits a performance nearly independent of the s parameter. The reasons for this behaviour were already invoked. The neural network mapping, on the other hand, seems to perform better with the increase of the s value, mainly when the number of classes involved is significant. Nonetheless, in this

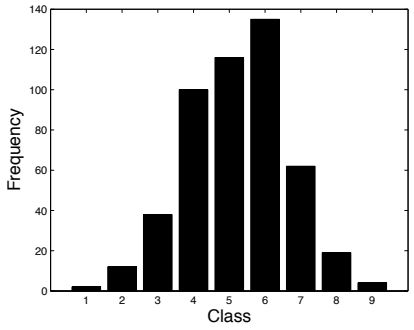


Figure 16: Class distribution for the 488 examples of the ESL data set.

Model		Training sets size $\ell = 25$	$\ell = 50$	$\ell = 100$	Training time for $\ell = 100$ (sec)
		MAD			
cSVM		0.52 (0.06)	0.47 (0.04)	0.39 (0.04)	3
pSVM		0.47 (0.05)	0.40 (0.03)	0.36 (0.03)	3
rSVM		0.36 (0.02)	0.34 (0.03)	0.33 (0.01)	32
oSVM	$s = 1$	0.46 (0.07)	0.42 (0.04)	0.35 (0.02)	47
	$s = 2$	0.45 (0.05)	0.39 (0.03)	0.34 (0.02)	64
	$s = 8$	0.45 (0.05)	0.39 (0.04)	0.34 (0.02)	85
cNN		0.88 (0.24)	0.80 (0.12)	0.49 (0.06)	8 611
pNN		0.56 (0.09)	0.52 (0.08)	0.39 (0.05)	3 498
iNN		0.55 (0.09)	0.46 (0.13)	0.39 (0.04)	7 398
rNN		0.43 (0.04)	0.38 (0.03)	0.34 (0.02)	1 622
oNN	$s = 1$	0.66 (0.28)	0.56 (0.24)	0.38 (0.16)	6 444
	$s = 2$	0.48 (0.08)	0.44 (0.08)	0.35 (0.03)	8 422
	$s = 8$	0.49 (0.14)	0.45 (0.13)	0.35 (0.05)	7 635

Table 6: Mean (standard deviation) of MAD over twenty setups of the test set. Configuration: K $(\mathbf{x},\mathbf{y}) = \mathbf{x}^t\mathbf{y}$ for SVMs, $h = 10$ for oSVM and oNN, $\nu = 0.5$ for rSVM.

experiment the instantiation with $s = 2$ already performed as well as the largest possible value of $s = 8$. Moreover, it is not clear a correlation between s and the training time. The overall results suggest that the data replication method learns faster than standard methods, although followed closely by the regression based models.

7. Gene Expression Analysis

Now we address the problem of selection of a small subset of genes from broad patterns of gene expression data, recorded on DNA micro-arrays. Singh et al. (2002) carried out microarray expression analysis on 12600 genes to identify genes that might anticipate the clinical behaviour of prostate cancer. Fifty-two samples of prostate tumour were investigated. For each sample, the degree of tumour cell differentiation or Gleason score (GS) was assessed by the pathologist; for tumour sam-

ples the GS ranged from 6 to 10. Predicting the Gleason score from the gene expression data is thus a typical ordinal classification problem, already addressed in Chu and Ghahramani (2005) using Gaussian processes. Following Chu and Ghahramani (2005), and since only 6 samples had a score greater than 7, we merged them as the top level, leading to three levels $\{=6,=7,\geq 8\}$, with 26, 20 and 6 samples respectively.

To evaluate the suitability of the oSVM algorithm for feature ranking and selection according to their relevance for the classification task, we applied the oSVM to the data set with the 12600 genes. A quality of SVMs is that the weights of the features in the borders $\mathbf{w}^t\mathbf{x}+b_r$ can be used for feature ranking and for the selection of subsets that are useful to build a good predictor (Guyon et al., 2002). Because the oSVM model computes the same set of weights for all borders, it provides an obvious feature ranking.

The application of the oSVM algorithm to the 12600 genes provided a first ranking of the genes given by the weights in the classifier. We then removed the irrelevant genes based on the rank list, assessing several subsets, as presented in Table 7(a). The best subset found had 500 genes. However, it is unlikely that the ranking of the 12600 genes have enough resolution to correctly rank genes in subsets much smaller than the original.

Therefore, we iterated the procedure starting now from the 500 genes selected by the first ranking: applying the oSVM algorithm to the 500 genes provided a second, refined ranking of these genes. Then, several subsets were assessed, Table 7(b). As visible, the ranking was in fact improved: the result for the first 100 genes was remarkably improved. Once again, starting with the best subset identified—100 genes—the oSVM algorithm was used to re-rank the genes, after which several subsets were again evaluated, Table 7(c). The procedure was successively repeated with 60, 40, 30, 26, 24, 19, 16 and 15 genes, when further reduction of the number of genes increased the misclassification error, Tables 7(d)-7(j).

During this process the leave one out method was used to estimate the MAD coefficient. The parameterization of the method was kept constant: $h=1$, $s=1$, $C=5$. Before training, the data was scaled to fall always within the range $[-1,1]$, using the transformation $x'=\frac{2x-x_{\max}-x_{\min}}{x_{\max}-x_{\min}}$. In this way the weight of each gene in the classifier conveys the relevance of the gene for the classification process.

Finally, the performance of the remaining algorithms was also determined in the considered subsets of genes.[4] Results are displayed in Table 8. We observe great and steady improvement of all classifiers using the subset of genes selected by the oSVM algorithm. The results attained agree with the results reported in Chu and Ghahramani (2005), although we were able to attain lower error rates. The best validation output was achieved with 15 genes, better than the 26 needed by the approach reported in Chu and Ghahramani (2005), revealing a better selection of genes for the classification task.

SVMs are particularly suited for the analysis of gene expression. They easily accommodate the high number of genes and perform well even with the small samples frequent in this area. Here the integration of the feature selection operation with the algorithm of classification in a consistent framework provided positive results.

4. Due to difficulties to handle such a high number of features with the Matlab `Neural Network Toolbox`, this experiment was only conducted with the SVM based algorithms.

n genes	MAD
12600	57.7
5000	67.3
500	**21.1**
100	38.5

(a) Ranking with 12600 genes.

n genes	MAD
500	21.1
300	13.5
100	**3.8**
50	13.5

(b) Ranking with 500 genes.

n genes	MAD
100	3.8
80	1.9
60	**0.0**
50	3.8

(c) Ranking with 100 genes.

n genes	MAD
60	**0.0**
50	**0.0**
40	**0.0**
30	7.7

(d) Ranking with 60 genes.

n genes	MAD
40	**0.0**
30	**0.0**
20	3.8
10	28.8

(e) Ranking with 40 genes.

n genes	MAD
30	**0.0**
28	**0.0**
26	**0.0**
25	1.9

(f) Ranking with 30 genes.

n genes	MAD
26	**0.0**
25	**0.0**
24	**0.0**
23	3.8

(g) Ranking with 26 genes.

n genes	MAD
24	**0.0**
21	**0.0**
19	**0.0**
17	5.8

(h) Ranking with 24 genes.

n genes	MAD
19	**0.0**
17	**0.0**
16	**0.0**
15	3.8

(i) Ranking with 19 genes.

n genes	MAD
16	**0.0**
15	**0.0**
14	1.9
13	7.7

(j) Ranking with 16 genes.

Table 7: MAD (%) for the oSVM algorithm, for the prostate cancer data set.

Model	MAD (%)										
	n genes										
	1	14	15	20	30	40	60	100	500	5000	12600
cSVM	81	6	10	6	2	2	6	8	23	58	48
pSVM	62	6	10	6	2	2	6	8	25	59	52
rSVM	62	15	11	2	10	19	4	6	21	69	60
oSVM	75	2	0	0	0	0	0	4	21	67	58

Table 8: Results using a linear kernel on the prostate cancer data of selected genes.

8. Conclusion

This study focuses on the application of machine learning methods, and in particular of neural networks and support vector machines, to the problem of classifying ordinal data. A novel approach to train learning algorithms for ordinal data was presented. The idea is to reduce the problem to the standard two-class setting, using the so called *data replication method*, a nonparametric procedure for the classification of ordinal categorical data. This method was mapped into neural networks and support vector machines. Two standard methods for the classification of ordinal categorical data were unified under this framework, the minimum margin principle (Shashua and Levin, 2002) and the generic approach by Frank and Hall (2001). Finally, a probabilistic interpretation for the neural network model was also presented.

The study compares the results of the proposed model with conventional learning algorithms for nominal classes and with models proposed in the literature specifically for ordinal data. Simple

misclassification, mean absolute error, Spearman and Kendall's tau-b coefficients are used as measures of performance for all models and used for model comparison. The new methods are likely to produce simpler and more robust classifiers, and compare favourably with state-of-the-art methods. In spite of being usually assumed that learning in a higher dimension becomes a harder problem, the performance of the data replication method does not seem to be affected, probably due to the dependence among the data replicas.

The data replication method is parameterised by h (and C); because it may be difficult and time consuming to choose the best value for h, it would be interesting to study possible ways to automatically set this parameter, probably as a function of the data and C. It would also be interesting to study if this algorithm can be successfully applied to nominal data. Although the data replication method was designed for ordinal classes, nothing impedes its application to nominal classes. It is expected that the classifier should be evaluated for each possible permutation of the classes, choosing the one conducting to the best performance (feasible only when the number of classes is small).

Acknowledgments

The authors are grateful to Luís F. Teixeira and Vitor Cardoso for their help on preparing the manuscript and to the Reviewers and the JMLR Editor for their many stimulating and thoughtful comments.

Appendix A.

In this appendix we derive a margin-based bound on the generalization error of the proposed ordinal classifier. The following theorem shows that the fat-shattering dimension gives generalization error bounds for large margin classifiers (Bartlett and Shawe-Taylor, 1999).

Theorem Consider a class $\mathcal{F}$ of real-valued functions. With probability at least $1-\delta$ over ℓ independently generated examples $\mathbf{x}$, if a classifier $\mathrm{sgn}(f) \in \mathrm{sgn}(\mathcal{F})$ has margin at least γ on $\mathbf{x}$, then the error of $\mathrm{sgn}(f)$ is bounded from above by

$$\frac{2}{\ell}\left(Z\log_2\frac{8e\ell}{Z}\log_2(32\ell)+\log_2\frac{8\ell}{\delta}\right),$$

where $Z=\mathrm{fat}_{\mathcal{F}}(\gamma/16)$, with $\mathrm{fat}_{\mathcal{F}}()$ the fat shattering dimension. Furthermore, with probability at least $1-\delta$, every classifier $\mathrm{sgn}(f)\in\mathrm{sgn}(\mathcal{F})$ has error no more than

$$R^{0-1}_{emp}(f)+\sqrt{\frac{2}{\ell}(Z\log_2(34e\ell/Z)\log_2(578\ell)+\log_2(4/\delta))},$$

where $R^{0-1}_{emp}(f)$ is the average of the number of training examples with margin less than γ.

It is not possible to apply the above equations directly to the extended data set because examples are not independently generated. We will proceed as follows. For each example $\mathbf{x}_i^{(k)}\in\mathbb{R}^p$ define $\mathbf{x'}_i^{(k)}$ as a single replica in $\mathbb{R}^{p+K-2}\times\{-1,1\}$ chosen uniformly at random from all the replicas from

$\mathbf{x}_i^{(k)}$. Since an error is made on example $\mathbf{x}_i^{(k)}$ if any of its $(K-1)$ replicas is wrongly classified, it is necessary to guarantee that no error is made. It follows that the generalization error for the ordinal problem is bounded from above by

$$\frac{2(K-1)}{\ell}\left(Z\log_2\frac{8e\ell}{Z}\log_2(32\ell)+\log_2\frac{8\ell}{\delta}\right)$$

or by

$$R_{emp}(f)+(K-1)\sqrt{\frac{2}{\ell}(Z\log_2(34e\ell/Z)\log_2(578\ell)+\log_2(4/\delta))}$$

where $R_{emp}(f)$ is the average of the magnitude of 'mistakes' of training examples:$R_{emp}(f) = \frac{1}{\ell}\sum_{k=1}^{K}\sum_{i=1}^{\ell_k} l\left(f(\mathbf{x}_i^{(k)}),k\right) = \frac{1}{\ell}\sum_{k=1}^{K}\sum_{i=1}^{\ell_k}|f(\mathbf{x}_i^{(k)})-k|$.

Appendix B.

In this appendix we prove the following claim from Section 3.2:

Claim For the mapping of the data replication method in NNs, if we allow the samples in all the classes to contribute errors for each threshold, by setting $s = K-1$, the order inequalities on the thresholds are satisfied automatically, in spite of the fact that such constraints on the thresholds are not explicitly included in the formulation.

Proof To prove the inequalities on the thresholds at the optimal solution, let us consider the situation where $\mathbf{w}$ is fixed (the first p components of $\overline{\mathbf{w}}$) and only the b_i's are optimized. Note that $w_{p+j}h + b_1 = b_{j+1}, j = 1..K-2$.

The error contributed at the data replica with $\overline{\mathbf{x}}_i^{(k)} = \begin{bmatrix}\mathbf{x}_i^{(k)}\\ \mathbf{e}_{q-1}\end{bmatrix}$, where $k = 1,\ldots,K$ denotes the class number, $i = 1,\ldots,\ell_k$ is the index within each class and $q = 2..K-1$ comes as

$$\sum_{k=1}^{q}\sum_{i=1}^{\ell_k} err\{f_N(\overline{\mathbf{w}}^t\overline{\mathbf{x}}_i^{(k)})-t_-\}+\sum_{k=q+1}^{K}\sum_{i=1}^{\ell_k} err\{f_N(\overline{\mathbf{w}}^t\overline{\mathbf{x}}_i^{(k)})-t_+\}$$
$$=\sum_{k=1}^{q}\sum_{i=1}^{\ell_k} err\{f_N(\mathbf{w}^t\mathbf{x}_i^{(k)}+w_{p+q-1}h)-t_-\}+\sum_{k=q+1}^{K}\sum_{i=1}^{\ell_k} err\{f_N(\mathbf{w}^t\mathbf{x}_i^{(k)}+w_{p+q-1}h)-t_+\}.$$

where err is the error performance function and t_- and t_+ are the target values for $\overline{C}_1$ and $\overline{C}_2$ respectively (f_N is the output transfer function).

Likewise, the error due to the data replica with $\overline{\mathbf{x}}_i^{(k)} = \begin{bmatrix}\mathbf{x}_i^{(k)}\\ \mathbf{e}_q\end{bmatrix}$ comes as

$$\sum_{k=1}^{q+1}\sum_{i=1}^{\ell_k} err\{f_N(\mathbf{w}^t\mathbf{x}_i^{(k)} + w_{p+q}h) - t_-\} + \sum_{k=q+2}^{K}\sum_{i=1}^{\ell_k} err\{f_N(\mathbf{w}^t\mathbf{x}_i^{(k)} + w_{p+q}h) - t_+\}.$$

The error contribution of the data replicas q and $q+1$ is just the sum of these two parcels:

$$\sum_{k=1}^{q}\sum_{i=1}^{\ell_k} err\{f_N(\mathbf{w}^t\mathbf{x}_i^{(k)}+w_{p+q-1}h)-t_-\}+\sum_{k=q+1}^{K}\sum_{i=1}^{\ell_k} err\{f_N(\mathbf{w}^t\mathbf{x}_i^{(k)}+w_{p+q-1}h)-t_+\}+$$
$$\sum_{k=1}^{q+1}\sum_{i=1}^{\ell_k} err\{f_N(\mathbf{w}^t\mathbf{x}_i^{(k)}+w_{p+q}h)-t_-\}+\sum_{k=q+2}^{K}\sum_{i=1}^{\ell_k} err\{f_N(\mathbf{w}^t\mathbf{x}_i^{(k)}+w_{p+q}h)-t_+\}. \quad (8)$$

Suppose that, at the optimal solution, we have $w_{p+q-1} < w_{p+q}$. Exchanging the value of w_{p+q-1} with the value of w_{p+q}, we obtain a solution where the error contribution of the data replicas $q-1$ and q is given by:

$$\sum_{k=1}^{q}\sum_{i=1}^{\ell_k} err\{f_N(\mathbf{w}^t\mathbf{x}_i^{(k)} + w_{p+q}h) - t_-\} + \sum_{k=q+1}^{K}\sum_{i=1}^{\ell_k} err\{f_N(\mathbf{w}^t\mathbf{x}_i^{(k)} + w_{p+q}h) - t_+\} +$$
$$\sum_{k=1}^{q+1}\sum_{i=1}^{\ell_k} err\{f_N(\mathbf{w}^t\mathbf{x}_i^{(k)} + w_{p+q-1}h) - t_-\} + \sum_{k=q+2}^{K}\sum_{i=1}^{\ell_k} err\{f_N(\mathbf{w}^t\mathbf{x}_i^{(k)} + w_{p+q-1}h) - t_+\}. \quad (9)$$

Observing that the error contribution of the other data replicas do not get affected by this swap of values, the total error difference between the new solution and the optimal solution simplifies to (given by subtracting the value of Eq. (8) to the value of Eq. (9))

$$\sum_{i=1}^{\ell_{q+1}} \{err\{f_N(\mathbf{w}^t\mathbf{x}_i^{(q+1)} + w_{p+q-1}h) - t_-\} - err\{f_N(\mathbf{w}^t\mathbf{x}_i^{(q+1)} + w_{p+q}h) - t_-\} +$$
$$err\{f_N(\mathbf{w}^t\mathbf{x}_i^{(q+1)} + w_{p+q}h) - t_+\} - err\{f_N(\mathbf{w}^t\mathbf{x}_i^{(q+1)} + w_{p+q-1}h) - t_+\}\}. \quad (10)$$

For clarity, assume now that f_N is the log-sigmoide transfer function logsig, $t_- = 0, t_+ = 1$, and err () is the absolute error performance function. Then, Eq. (10) simplifies to

$$\sum_{i=1}^{\ell_{q+1}} \{\text{logsig}(\mathbf{w}^t\mathbf{x}_i^{(q+1)} + w_{p+q-1}h) - \text{logsig}(\mathbf{w}^t\mathbf{x}_i^{(q+1)} + w_{p+q}h) +$$
$$- \text{logsig}(\mathbf{w}^t\mathbf{x}_i^{(q+1)} + w_{p+q}h) + \text{logsig}(\mathbf{w}^t\mathbf{x}_i^{(q+1)} + w_{p+q-1}h)\}.$$

Because this last value is clearly less than zero, we obtained a solution with lower error than the optimal solution. Then, at the optimal solution $w_{p+q-1} \geq w_{p+q} \Longleftrightarrow -b_q \leq -b_{q+1}, q = 2,\ldots,K-2$. Note that this reasoning does not get affected by using other transfer functions such as the tan-sigmoid function, together with $t_- = -1$ and $t_+ = +1$, or other error performance functions such as the squared error performance function. For the nonlinear case, we just need to replace $\mathbf{w}^t\mathbf{x}_i^{(k)}$ by $G(\mathbf{x}_i^{(k)})$ in the proof. The proof of the order relation between b_1 and b_2 in the optimal solution is left to the reader. ■

References

A. Agarwal, J. T. Davis, and T. Ward. Supporting ordinal four-state classification decisions using neural networks. In *Information Technology and Management*, pages 5–26, 2001.

P. L. Bartlett and J. Shawe-Taylor. Generalization performance of support vector machines and other pattern classifiers. In B. Scholkopf, C. J. C. Burges, and A. J. Smola, editors, *Advances in Kernel Methods - Support Vector Learning*, pages 43–54. MIT Press, Cambridge, MA, 1999.

J. S. Cardoso, J. F. Pinto da Costa, and M. J. Cardoso. Modelling ordinal relations with SVMs: an application to objective aesthetic evaluation of breast cancer conservative treatment. *Neural Networks*, 18:808–817, june-july 2005.

W. Chu and Z. Ghahramani. Gaussian processes for ordinal regression. *Journal of Machine Learning Research*, 6:1019–1041, 2005.

W. Chu and S. S. Keerthi. New approaches to support vector ordinal regression. In *Proceedings of International Conference on Machine Learning (ICML05)*, pages 145–152, 2005.

M. Costa. Probabilistic interpretation of feedforward network outputs, with relationships to statistical prediction of ordinal quantities. *International Journal Neural Systems*, 7(5):627–638, 1996.

J. Dong, A. Krzyzak, and C. Y. Suen. Fast SVM training algorithm with decomposition on very large data sets. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(4):603–618, 2005.

E. Frank and M. Hall. A simple approach to ordinal classification. In *Proceedings of the 12th European Conference on Machine Learning*, volume 1, pages 145–156, 2001.

I. Guyon, J. Weston, S. Barnhill, and V. Vapnik. Gene selection for cancer classification using support vector machines. *Machine Learning*, 46:389–422, 2002.

R. Herbrich, T. Graepel, and K. Obermayer. Regression models for ordinal data: a machine learning approach. Technical Report TR-99/03, TU Berlin, 1999a.

R. Herbrich, T. Graepel, and K. Obermayer. Support vector learning for ordinal regression. In *Ninth International Conference on Artificial Neural Networks ICANN*, volume 1, pages 97–102, 1999b.

T. Joachims. Making large-scale support vector machine learning practical. In A. Smola B. Schölkopf, C. Burges, editor, *Advances in Kernel Methods: Support Vector Machines*. MIT Press, Cambridge, MA, 1998.

Y. Lin, Y. Lee, and G. Wahba. Support vector machines for classification in nonstandard situations. *Machine Learning*, 46:191–202, 2002.

M. J. Mathieson. Ordinal models for neural networks. In A.-P.N Refenes, Y. Abu-Mostafa, and J. Moody, editors, *Neural Networks for Financial Engineering*. World Scientific, Singapore, 1995.

P. McCullagh. Regression models for ordinal data. *Journal of the Royal Statistical Society Series*, 42:109–142, 1980.

P. McCullagh and J. A. Nelder. *Generalized Linear Models*. Chapman and Hall, 1989.

J. Moody and J. Utans. Architecture selection strategies for neural networks: application to corporate bond rating prediction. In A.-P. Refenes, editor, *Neural Networks in the Capital Markets*, pages 277–300, Chichester, 1995. Wiley.

J. Platt. Fast training of support vector machines using sequential minimal optimization. In *Advances in Kernel Methods-Support Vector Learning*, pages 185–208, 1998.

W. Press, B. Flannery, S. Teukolsky, and W. Vetterling. *Numerical Recipes in C: the Art of Scientific Computing*. Cambridge University Press, 1992.

B. Scholkopf, A. J. Smola, R. C. Williamson, and P. L. Bartlett. New support vector algorithms. *Neural Computation*, 12:1207–1245, 2000.

A. Shashua and A. Levin. Ranking with large margin principle: Two approaches. In *Neural Information and Processing Systems (NIPS)*, 2002.

L. Shen and A. K. Joshi. Ranking and reranking with perceptron. *Machine Learning*, 60:73–96, September 2005.

D. Singh, P. G. Febbo, K. Ross, D. G. Jackson, J. Manola, C. Ladd, P. Tamayo, A. A. Renshaw, A. V. D'Amico, J. P. Richie, E. S. Lander, M. Loda, P. W. Kantoff, T. R. Golub, and W. R. Sellers. Gene expression correlates of clinical prostate cancer behavior. *Cancer Cell*, 1:1019–1041, 2002.

V. N. Vapnik. *Statistical Learning Theory*. John Wiley, 1998.

Journal of Machine Learning Research 8 (2007) 1431-1460 Submitted 5/06; Revised 10/06; Published 7/07

Attribute-Efficient and Non-adaptive Learning of Parities and DNF Expressions*

Vitaly Feldman† VITALY@EECS.HARVARD.EDU
School of Engineering and Applied Sciences
Harvard University
Cambridge, MA 02138

Editor: Peter Auer

Abstract

We consider the problems of attribute-efficient PAC learning of two well-studied concept classes: parity functions and DNF expressions over $\{0,1\}^n$. We show that attribute-efficient learning of parities with respect to the uniform distribution is equivalent to decoding high-rate random linear codes from low number of errors, a long-standing open problem in coding theory. This is the first evidence that attribute-efficient learning of a natural PAC learnable concept class can be computationally hard.

An algorithm is said to use membership queries (MQs) *non-adaptively* if the points at which the algorithm asks MQs do not depend on the target concept. Using a simple non-adaptive parity learning algorithm and a modification of Levin's algorithm for locating a weakly-correlated parity due to Bshouty et al. (1999), we give the first non-adaptive and attribute-efficient algorithm for learning DNF with respect to the uniform distribution. Our algorithm runs in time $\tilde{O}(ns^4/\varepsilon)$ and uses $\tilde{O}(s^4 \cdot \log^2 n/\varepsilon)$ non-adaptive MQs, where s is the number of terms in the shortest DNF representation of the target concept. The algorithm improves on the best previous algorithm for learning DNF (of Bshouty et al., 1999) and can also be easily modified to tolerate random persistent classification noise in MQs.

Keywords: attribute-efficient, non-adaptive, membership query, DNF, parity function, random linear code

1. Introduction

The problems of PAC learning parity functions and DNF expressions are among the most fundamental and well-studied problems in machine learning theory. Along with running time efficiency, an important consideration in the design of learning algorithms is their *attribute-efficiency*. A class $\mathcal{C}$ of Boolean functions is said to be *attribute-efficiently learnable* if there is an efficient algorithm which can learn any function $f \in \mathcal{C}$ using a number of examples which is polynomial in the "size" (description length) of the function f to be learned, rather than in n, the number of attributes in the domain over which learning takes place. Attribute-efficiency arises naturally from a ubiquitous practical scenario in which the total number of potentially influential attributes is much larger than the number of relevant attributes (i.e., the attributes on which the concept actually depends), whereas examples are either scarce or expensive to get.

*. Parts of this work are published in the Proceedings of 18th Annual Conference on Learning Theory, 2005.
†. Supported by grants from the National Science Foundation NSF-CCF-9877049, NSF-CCF-0432037, and NSF-CCF-0427129.

Learning of DNF expressions and attribute-efficient learning of parities from random examples with respect to the uniform distribution are both long-standing challenges in learning theory. The lack of substantial progress on these questions has resulted in attempts to solve them in stronger learning models. The most well-studied such model is one in which a *membership query oracle* is given to the learner in addition to the example oracle. The learning algorithm may query this oracle for a value of the target function at any point of its choice. Jackson (1997) gave the first algorithm that learns DNF from membership queries (MQs) under the uniform distribution and later Bshouty, Jackson, and Tamon (1999) gave a more efficient and attribute-efficient algorithm for learning DNF in the same setting. The first algorithm for attribute-efficient learning of parities using MQs is due to Blum et al. (1995), and their result was later refined by Uehara et al. (1997).

A restricted model of membership queries, which addresses some of the disadvantages of the MQ model, is the model in which MQs are asked non-adaptively. An algorithm is said to use MQs *non-adaptively* if the queries of the algorithm do not depend on the target concept (in our context we will often call it non-adaptive for brevity). In other words, the learning algorithm can be split into two stages. In the first stage, given the learning parameters, the algorithm generates a set S of queries for the membership oracle. In the second stage, given the answers to the queries in S, the algorithm produces a hypothesis (without further access to the oracle). An immediate advantage of this model (over the usual MQ model) is the fact that the queries to the membership oracle can be parallelized. This, for example, is crucial in DNA sequencing and other biological applications where tests are very time-consuming but can be parallelized (Farach et al., 1997; Damaschke, 1998, and references therein). Another advantage of a non-adaptive learner is that the same set of points can be used to learn numerous concepts. This is conjectured to happen in the human brain where a single example can be used to learn several different concepts and hence systems that aim to reproduce the learning abilities of the human brain need to possess this property (Valiant, 1994, 2000, 2006).

As it is detailed later, attribute-efficiency is easy to achieve using a simple technique that relies on adaptive MQs but there is no known general method to convert a learning algorithm to an attribute-efficient one using MQs non-adaptively. It is important to note that in the two practical applications mentioned above, attribute-efficiency is also a major concern. It is therefore natural to ask: which classes can be PAC learned attribute-efficiently by non-adaptive MQs? We refer to this model of learning as *ae.naMQ learning*. This question was first explicitly addressed by Damaschke (1998) who proved that any function of r variables is ae.naMQ learnable when it is represented by the truth table of the function (requiring $r \log n + 2^r$ bits). Later Hofmeister (1999) gave the first ae.naMQ algorithm for learning parities and Guijarro et al. (1999a) gave an algorithm for learning functions of at most $\log n$ variables in the decision tree representation. But the question remains open for numerous other representations used in learning theory.

1.1 Previous Results

Blum et al. (1995) were the first to ask whether parities are learnable attribute-efficiently (in the related *on-line mistake-bound* model). They also presented the first algorithm to learn parity functions attribute-efficiently using MQs. Their algorithm is based on the following approach: first all the relevant attributes are identified and then a simple (not attribute-efficient) algorithm restricted to the relevant variables is used to learn the concept. Since then other algorithms were proposed for attribute-efficient identification of relevant variables (Bshouty and Hellerstein, 1998; Guijarro et al., 1999b). All the algorithms are based on a binary search for a relevant variable given a positive

and a negative example. Binary search and the fact that queries in the second stage depend on the variables identified in the first stage only allows for the construction of adaptive algorithms via this approach. Uehara et al. (1997) gave several algorithms for attribute-efficient learning of parities that again used adaptiveness in an essential way.

Hofmeister gave the first ae.naMQ algorithm for learning parities based on BCH error-correcting codes. When learning the class of parities on at most k variables his algorithm has running time of $O(kn)$ and uses $O(k \log n)$ non-adaptive MQs. While the complexity of this algorithm is asymptotically optimal it is based on the relatively complex Berlekamp-Massey algorithm for creating and decoding BCH codes (Massey, 1969).

Little previous work has been published on attribute-efficient learning of parities from random examples only. Indeed, the first non-trivial result in this direction has only recently been given by Klivans and Servedio (2004). They prove that parity functions on at most k variables are learnable in polynomial time using $O(n^{1-\frac{1}{k}} \log n)$ examples.

1.1.1 LEARNING DNF

Efficient learning of unrestricted DNF formulae under the uniform distribution begins with a famous result by Jackson (1997). The algorithm, while polynomial-time, is somewhat impractical due to the $\tilde{O}(ns^{10}/\varepsilon^{12})$ bound on running time (where s is the number of terms in the target DNF). By substantially improving the key components of Jackson's algorithm, the works of Freund (1992), Bshouty et al. (1999), and Klivans and Servedio (2003) resulted in an algorithm that learns DNF in time $\tilde{O}(ns^6/\varepsilon^2)$ and uses $\tilde{O}(ns^4/\varepsilon^2)$ MQs.[1] This algorithm is non-adaptive, but is also not attribute-efficient. Using the algorithm for identification of relevant variables by Bshouty and Hellerstein mentioned above, Bshouty et al. (1999) gave an attribute-efficient version of their algorithm running in time $\tilde{O}(rs^6/\varepsilon^2 + n/\varepsilon)$ and using $\tilde{O}(rs^4 \log n/\varepsilon^2)$ adaptive MQs, where r is the number of relevant variables.

Bshouty et al. (2003) give an algorithm for learning DNF expressions from examples generated by a random walk on the Boolean hypercube. This model is more passive than non-adaptive MQs but their algorithm is not attribute-efficient as it is an adaptation of the non-attribute-efficient algorithm of Bshouty and Feldman (2002). In fact, it is information-theoretically impossible to learn anything non-trivial attribute-efficiently in this model.

1.2 Our Results

We give a simple and fast randomized algorithm for ae.naMQ learning of parities (Theorem 9) and provide a transformation that converts a non-adaptive parity learning algorithm into an algorithm for finding significant Fourier coefficients of a function while preserving attribute-efficiency and non-adaptiveness (Theorem 13). Using these components we give the first ae.naMQ algorithm for learning DNF expressions with respect to the uniform distribution (Theorem 24). It runs in time $\tilde{O}(ns^4/\varepsilon)$ and uses $\tilde{O}(s^4 \log^2 n/\varepsilon)$ MQs. The algorithm improves on the $\tilde{O}(ns^6/\varepsilon^2)$-time and $\tilde{O}(ns^4/\varepsilon^2)$-query algorithm of Bshouty et al. (1999). In Theorem 28 we also show a simple and general modification that allows the above algorithm to efficiently handle random persistent classification noise in MQs (see Section 2.1 for the formal definition of the noise model). Earlier algorithms for learning DNFs that handled persistent classification noise were based on Jackson's DNF learning algorithm and therefore are substantially less efficient (Jackson et al., 1997; Bshouty and Feldman, 2002).

1. Bshouty et al. claimed sample complexity $\tilde{O}(ns^2/\varepsilon^2)$ but this was in error as explained in Remark 19.

Alongside our ae.naMQ algorithm for learning of parities we establish the equivalence between attribute-efficient learning of parities from random uniform examples and decoding high-rate random linear codes from a low number of errors, a long-standing open problem in coding theory widely believed to be intractable (Theorems 6 and 8). Thus we may consider this equivalence as evidence of the hardness of attribute-efficient learning of parities from random examples only. Previously hardness of attribute-efficient learning results were only known for specially designed concept classes (Decatur et al., 1999; Servedio, 2000).

The connection between attribute-efficient learning of parities by membership queries and linear codes was earlier observed by Hofmeister (1999). His result allows to derive attribute-efficient parity learning algorithms from efficiently decodable linear codes with appropriate parameters. Our result can be seen as an adaptation of this connection to random and uniform examples. The restriction to the uniform distribution allows us to prove the connection in the other direction, giving the above-mentioned negative result for attribute-efficient learning of parities from random examples only.

1.3 Organization

In the next section we describe the models and tools that will be used in this work. In Section 3, we give the required background on binary linear codes and prove the equivalence between attribute-efficient learning of parities from random uniform examples and decoding high-rate random linear codes from a low number of errors. In Section 4, we show a simple algorithm for ae.naMQ learning of parities. Section 5 gives a way to convert a non-adaptive parity learning algorithm into an algorithm for finding significant Fourier coefficients of a function while preserving attribute-efficiency and non-adaptiveness, yielding an ae.naMQ algorithm for weakly learning DNF expressions. Then in Section 6 we describe our ae.naMQ algorithm for learning DNF expressions and in Section 7 we show how this algorithm can be modified to handle random persistent classification noise.

2. Preliminaries

For vectors $x, y \in \{0,1\}^n$ we denote by $x \oplus y$ the vector obtained by bitwise XOR of x and y; by $[k]$ the set $\{1,2,\ldots,k\}$; by e_i a vector with 1 in i-th position and zeros in the rest; by x_i the i-th element of vector x. Dot product $x \cdot y$ of vectors $x, y \in \{0,1\}^n$ denotes $\sum_i x_i y_i \pmod 2$ or simply vector product xy^T over $\mathbf{GF}(2)$ (with vectors being row vectors by default). By $\mathtt{wt}(x)$ we denote the Hamming weight of x and we define $\mathtt{dist}(x,y) = \mathtt{wt}(x \oplus y)$.

To analyze the accuracy and confidence of estimates produced by random sampling we will use the following standard inequalities.

Lemma 1 (Chernoff) *Let $X_1, \ldots, X_m$ be a sequence of m independent Bernoulli trials, each with probability of success $\mathbf{E}[X_i] = p$ and let $S = \sum_{i=1}^{m} X_i$. Then for $0 \leq \gamma \leq 1$,*

$$\mathbf{Pr}[S > (1+\gamma)pm] \leq e^{-mp\gamma^2/3}$$

and

$$\mathbf{Pr}[S < (1-\gamma)pm] \leq e^{-mp\gamma^2/2} .$$

Lemma 2 (Bienaymé-Chebyshev) *Let $X_1, \ldots, X_m$ be pairwise independent random variables all with mean μ and variance σ^2. Then for any $\lambda \geq 0$,*

$$\mathbf{Pr}\left[\left|\frac{1}{m}\sum_{i=1}^{m} X_i - \mu\right| \geq \lambda\right] \leq \frac{\sigma^2}{m\lambda^2}\ .$$

For a function $t(\cdots)$ we say a function $q(\cdots)$ (of the same parameters as t) is $\tilde{O}(t(\cdots))$ when there exist constants α and β such that $q(\cdots) \leq \alpha t(\cdots)\log^{\beta}(t(\cdots))$.

2.1 PAC Learning

We study learning of Boolean functions on the Boolean hypercube $\{0,1\}^n$. Our Boolean functions take values $+1$ (true) and -1 (false). Our main interest are the classes of parity functions and DNF expressions. A parity function $\chi_a(x)$ for a vector $a \in \{0,1\}^n$ is defined as $\chi_a(x) = (-1)^{a \cdot x}$. We refer to the vector associated with a parity function as its *index* and the Hamming weight of the vector as the *length* of the parity function. We denote the concept class of parity functions $\{\chi_a \mid a \in \{0,1\}^n\}$ by PAR and the class of all the parities of length at most k by PAR(k). We represent a parity function by listing all the variables on which it depends. This representation for a parity of length k requires $\theta(k \log n)$ bits.

For the standard DNF representation and any Boolean function f we denote by DNF-size(f) the number of terms in a DNF representation of f with the minimal number of terms. In context of learning DNF this parameter is always denoted s. The uniform distribution over $\{0,1\}^n$ is denoted $\mathcal{U}$.

Our learning model is Valiant's well-known PAC model (Valiant, 1984) for learning Boolean functions over $\{0,1\}^n$. In this model, for a concept c and distribution $\mathcal{D}$ over X, an *example oracle* $\mathrm{EX}_{\mathcal{D}}(c)$ is an oracle that upon request returns an example $\langle x, c(x)\rangle$ where x is chosen randomly with respect to $\mathcal{D}$, independently of any previous examples. For $\varepsilon \geq 0$ we say that function g ε-approximates a function f with respect to distribution $\mathcal{D}$ if $\mathbf{Pr}_{\mathcal{D}}[f(x) = g(x)] \geq 1 - \varepsilon$. For a concept class $\mathcal{C}$, we say that an algorithm $\mathcal{A}$ *efficiently* learns $\mathcal{C}$, if for every $\varepsilon > 0$, n, $c \in \mathcal{C}$, and distribution $\mathcal{D}$ over $\{0,1\}^n$, $\mathcal{A}(n, \varepsilon, s)$ (where s is the size of c in the representation associated with $\mathcal{C}$) outputs, with probability at least $1/2$, and in time polynomial in $n, 1/\varepsilon$, and s a hypothesis h that ε-approximates c. When a learning algorithm is guaranteed to learn only with respect to a specific distribution we specify the distribution explicitly. We say that an algorithm *weakly* learns $\mathcal{C}$ if it produces a hypothesis h that $(\frac{1}{2} - \frac{1}{p(n,s)})$-approximates (or *weakly approximates*) c for some polynomial p.

Note that in this definition of learning we do not use the confidence parameter δ that requires a learning algorithm to succeed with probability at least $1 - \delta$. Instead we assume that it equals $1/2$. In order to obtain an algorithm with success probability $1 - \delta$ one can always use a standard confidence boosting procedure (cf. the textbook by Kearns and Vazirani, 1994). The boosting procedure consists of repeating the original algorithm $k = \log(1/\delta) + 1$ times with slightly increased accuracy (e.g., $\varepsilon/2$), each time on new examples and independent coin flips. The hypotheses obtained from these runs are then tested on an independent sample of size $O(\varepsilon^{-1}\log(1/\delta))$ and the best one is chosen.

A membership query oracle MEM(c) is the oracle that, given any point $x \in \{0,1\}^n$, returns the value $c(x)$. When learning with respect to $\mathcal{U}$, $\mathrm{EX}_{\mathcal{U}}(c)$ can be trivially simulated using MEM(c) and therefore $\mathrm{EX}_{\mathcal{U}}(c)$ is not used at all.

An algorithm $\mathcal{A}$ is said to be *attribute-efficient* if the number of examples (both random and received from the MQ oracle) it uses is polynomial in the size of the representation of the concept and $1/\varepsilon$. We say that a variable x_i is *relevant* for a function f if there exists $y \in \{0,1\}^n$ such that $f(y) \neq f(y \oplus e_i)$. The number of relevant variables of the target concept is denoted by parameter r. Attribute-efficiency does not allow the number of examples to depend polynomially on n. Instead the number of examples used can depend polynomially on r and $\log n$ since for most representations (including the ones considered in this work) the size of the representation of f is lower bounded by both $\log n$ and r.

2.1.1 NOISE MODELS

We consider two standard models of noise in learning. The first one is the well-studied random classification noise model introduced by Angluin and Laird (1988). In this model for any $\eta \leq 1/2$ called the *noise rate* the regular example oracle $\mathrm{EX}_{\mathcal{D}}(c)$ is replaced with the faulty oracle $\mathrm{EX}^{\eta}_{\mathcal{D}}(c)$. On each call, $\mathrm{EX}^{\eta}_{\mathcal{D}}(c)$, draws x according to $\mathcal{D}$, and returns $\langle x, c(x)\rangle$ with probability η and $\langle x, \neg c(x)\rangle$ with probability $1-\eta$. When η approaches $1/2$ the result of the corrupted query approaches the result of the random coin flip, and therefore the running time of algorithms in this model is allowed to polynomially depend on $\frac{1}{1-2\eta}$.

This model of noise is not suitable for corrupting labels returned by $\mathrm{MEM}(c)$ since a learning algorithm can, with high probability, find the correct label at point x by asking the label of x polynomial (in $\frac{1}{1-2\eta}$) number of times and then returning the label that appeared in the majority of answers. An appropriate modification of the noise model is the introduction of *random persistent classification noise* by Goldman, Kearns, and Schapire (1993). In this model, as before, the answer to a query at each point x is flipped with probability $1-\eta$. However, if the membership oracle was already queried about the value of f at some specific point x or x was already generated as a random example, the returned label has the same value as in the first occurrence (i.e., in such a case the noise persists and is not purely random). If the learner does not ask for the label of a point more than once then this noise can be treated as the usual independent random classification noise.

2.1.2 FOURIER TRANSFORM

The Fourier transform is a technique for learning with respect to the uniform distribution (primarily) based on the fact that the set of all parity functions $\{\chi_a(x)\}_{a\in\{0,1\}^n}$ forms an orthonormal basis of the linear space of real-valued function over $\{0,1\}^n$. This fact implies that any real-valued function f over $\{0,1\}^n$ can be uniquely represented as a linear combination of parities, that is $f(x) = \sum_{a\in\{0,1\}^n} \hat{f}(a)\chi_a(x)$. The coefficient $\hat{f}(a)$ is called Fourier coefficient of f on a and equals $\mathbf{E}_{\mathcal{U}}[f(x)\chi_a(x)]$; a is called the *index* and $\mathtt{wt}(a)$ the *degree* of $\hat{f}(a)$. Given the values of f on all the points of the hypercube $\{0,1\}^n$ one can compute the values of all the Fourier coefficients $\{\hat{f}(a)\}_{a\in\{0,1\}^n}$ using the Fast Fourier Transform (FFT) algorithm in time $O(n2^n)$ (Cooley and Tukey, 1965). The same algorithm FFT also converts the set of all Fourier coefficients $\{\hat{f}(a)\}_{a\in\{0,1\}^n}$ into the values of the function f on all the points of the hypercube. This transformation is called inverse Fourier transform. For further details on the technique we refer the reader to the survey by Mansour (1994).

2.1.3 RANDOMIZED FUNCTIONS

Besides deterministic functions on $\{0,1\}^n$ we will also deal with functions whose value on a point x is a real-valued random variable $\Psi(x)$ independent of $\Psi(y)$ for any $y \neq x$ and of any previous evaluations of $\Psi(x)$. To extend learning and Fourier definitions to this case we include the probability over the random variable Ψ in estimations of probability, expectation and variance. For example, we say that a randomized function Ψ ε-approximates f with respect to $\mathcal{D}$ if $\mathbf{Pr}_{\mathcal{D},\Psi}[f(x) = \Psi(x)] \geq 1 - \varepsilon$. Similarly, $\widehat{\Psi}(a) = \mathbf{E}_{\mathcal{U},\Psi}[\Psi(x)\chi_a(x)]$.

2.2 Learning by Non-adaptive Membership Queries

We say that an algorithm $\mathcal{A}$ uses MQs *non-adaptively* if it can be split into two stages. The first stage, given all the parameters of learning, (n, ε and a bound on the size of the target concept) and access to points randomly sampled with respect to the target distribution, generates a set of points $S \subseteq \{0,1\}^n$. The second stage, given the labels of the random points and the answers from $\mathrm{MEM}(c)$ on points in S, that is, the set $\{(x, c(x)) \mid x \in S\}$, computes a hypothesis (or, in general, performs some computation). Neither of the stages has any other access to $\mathrm{MEM}(c)$.

We note that in the general definition of PAC learning we did not assume that size of the target concept (or a bound on it) is given to the learning algorithm. When learning with adaptive queries a good bound can be found via the "guess-and-double" technique, but for non-adaptive algorithms we will assume that this bound is always given. To emphasize this we specify the parameters that have to be given to a non-adaptive algorithm in the name of the algorithm. Clearly the same "guess-and-double" technique can be used to produce a sequence of independent and non-adaptive executions of the learning algorithm.

The immediate consequence of non-adaptiveness is that in order to parallelize a non-adaptive learning algorithm only the usual computation has to be parallelized since all the MQs can be made in parallel. Non-adaptiveness is also useful when learning ℓ concepts from the same concept class in parallel. The fact that queries are independent of the target concept implies that same set of points can be used for learning different concepts. To achieve probability of success $1/2$ in learning of all ℓ concepts we will have to learn with each concept with probability of success $1 - 1/(2\ell)$. This implies that the number of points needed for learning might grow by a factor of $\log \ell$ whereas in the general case ℓ times more examples might be required.

Results of Goldreich et al. (1986) imply that if one-way functions exist then the concept class of all polynomial circuits is not learnable even with respect to $\mathcal{U}$ and with access to a MQ oracle (Kearns and Valiant, 1994). By modifying the values of each circuit to encode the circuit itself in a polynomial number of fixed points one can make this class learnable by non-adaptive MQs but not learnable from random and uniform examples only (the modification is very unlikely to be detected by random examples yet MQs to the fixed points will reveal the circuit). Similarly, by placing the encoding of the circuit in some location that is encoded in a fixed location, one can create a function class learnable by adaptive membership queries but not learnable by the non-adaptive ones (if one-way functions exist). Further details of these simple separations are left to the reader.

3. Learning of Parities and Binary Linear Codes

In this section we show that attribute-efficient learning of parities with respect to the uniform distribution from random examples only is likely to be hard by proving that it is equivalent to an open

problem in coding theory. Unlike in the rest of the paper in this section and the following section parity functions will be functions to $\{0,1\}$. To emphasize this we use $\dot{\chi}$ instead of χ.

3.1 Background on Linear Codes

We say that a code C is an $[m,n]$ code if C is a binary linear code of block length m and message length n. Any such code can be described by its $n \times m$ *generator matrix* G as follows: $C = \{xG \mid x \in \{0,1\}^n\}$. Equivalently, a code can be described by its *parity-check* matrix H of size $m \times (m-n)$ by $C = \{y \mid yH = 0^{m-n}\}$. It is well-known that $G \cdot H = 0^{n \times (m-n)}$ and decoding given a corrupted message y is equivalent to decoding given the *syndrome* of the corrupted message. The syndrome equals to yH and the decoding consists of finding a vector e of Hamming weight at most w such that $y \oplus e = xG$, where $w = \lfloor (d-1)/2 \rfloor$ and d is the *distance* of the code (cf. the book by van Lint, 1998). For a linear code C the distance equals to the Hamming weight of a non-zero vector with the smallest Hamming weight.

By saying that C is a *random* $[m,n]$ *code* we mean that C is defined by choosing randomly, uniformly, and independently n vectors in $\{0,1\}^m$ that form the basis of C. Alternatively, we can say that the generator matrix G of C was chosen randomly with each entry equal to 1 with probability $1/2$ independently of others. We denote this distribution by $\mathcal{U}_{n \times m}$. Some authors restrict the random choice of G's to matrices of full rank n. As we will see, this definitions would only make our proofs simpler.

Binary linear codes generated randomly meet the Gilbert-Varshamov bound with high probability, that is, they achieve the best known *rate* (or n/m) versus distance trade-off (cf. the lecture notes by Sudan, 2002). However decoding a random linear code or even determining its distance is a notorious open problem in coding theory. For example the McEliece cryptosystem is based, among other assumptions, on the hardness of this problem (McEliece, 1978). Besides that, while the average-case hardness of this problem is unknown, a number of worst-case problems related to decoding linear codes are NP-hard (Barg, 1997; Vardy, 1997; Sudan, 2002).

A potentially simpler version of this problem in which the errors are assumed to be random and independent with some rate η (and not adversarial as in the usual definition) is equivalent to learning of parities with random classification noise of rate η, a long-standing open problem in learning theory. In fact, Feldman et al. (2006) have proved that when learning parities from random and uniform examples, random classification noise of rate η is as hard as adversarial noise of rate η (up to a polynomial blowup in the running time). The only known non-trivial algorithm for learning parities with noise is a slightly subexponential algorithm by Blum et al. (2000). In our discussion η is very low (e.g., $\frac{\log n}{n}$), yet even for this case no efficient noise-tolerant algorithms are known.

Correcting a random linear $[m,n]$ from up to w errors is defined as follows.

Definition 3 **Input:** *An $n \times m$ binary generator matrix G randomly chosen according to $\mathcal{U}_{n \times m}$ and $y \in \{0,1\}^m$.*
Output: *$x \in \{0,1\}^n$ such that $\mathtt{dist}(xG, y) \leq w$ if there exists one.*

A successful algorithm for this problem is an algorithm that would allow to correct up to w errors in a "good" fraction of randomly created linear codes. That is, with non-negligible probability over the choice of G, and for every y, the algorithm should produce the desired output. Note that the algorithm can only be successful when the code generated by G has distance at least $2w+1$.

For simplicity, we will usually assume a constant probability of success but all the results can be translated to algorithms having the success probability lower-bounded by a polynomial (in m) fraction.

3.2 The Reduction

The equivalence of attribute-efficient learning of parities with respect to the uniform distribution and decoding of random linear codes relies on two simple lemmas. The first one, due to Hofmeister (1999), is that the syndrome decoding of a linear code implies attribute-efficient learning of parities. We include it with a proof for completeness.

Lemma 4 (Hofmeister) *Let H be a parity-check matrix of some $[m,n]$ w-error correcting code C. Let $\mathcal{A}$ be an algorithm that for any $y \in \{0,1\}^m$ such that $y = c \oplus e$ where $c \in C$ and $\mathtt{wt}(e) \leq w$, given the syndrome yH, finds e. Then $\mathcal{A}$ learns PAR(w) over $\{0,1\}^m$ given the values of an unknown parity on the columns of H.*

Proof The condition $y = c \oplus e$ for $c \in C$ implies that $yH = eH$. Therefore the syndrome yH is equal to the vector $eH = \dot{\chi}_e(H_1), \dot{\chi}_e(H_2), \ldots, \dot{\chi}_e(H_{m-n})$ where H_i is the i-th column of H. Therefore finding an error vector e of weight at most w using the syndrome yH is the same as finding a parity of length at most w given the values of the unknown parity on the columns of H. ■

This observation has lead Hofmeister to a simple ae.naMQ algorithm for learning parities that uses the columns of the parity check matrix of BCH code as MQs. We note that the converse of this lemma is only true if the learning algorithm is *proper*, that is, produces a parity function in PAR(w) as a hypothesis.

To obtain the claimed equivalence for the uniform distribution we first need to prove that generating a linear code by choosing a random and uniform parity check matrix (that is, from $\mathcal{U}_{n\times m-n}$) is equivalent to (or indistinguishable from) generating a linear code by choosing a random and uniform generator matrix (that is, from $\mathcal{U}_{n\times m}$).

Let $p(i,j)$ denote the probability that i vectors chosen randomly and uniformly from $\{0,1\}^j$ are linearly independent. Each $i \geq 1$ linearly independent vectors span subspace of size 2^i and therefore there are $2^j - 2^i$ vectors that are linearly independent of them. This implies that, $p(i+1,j) = p(i,j)(1-2^{-j+i})$. All vectors except for 0^j form a linearly independent set of size 1. Therefore $p(1,j) = (1-2^{-j})$. Hence

$$p(i,j) = (1-2^{-j}) \cdot (1-2^{-j+1}) \cdots (1-2^{-j+i-1}) \,.$$

Note that

$$p(i,j) \geq 1 - 2^{-j} - 2^{-j+1} - \cdots - 2^{-j+i-1} > 1 - 2^{-j+i} \tag{1}$$

and for $i = j$, $p(j,j) = \frac{1}{2}p(j,j-1) > \frac{1}{2}(1-\frac{1}{2}) = \frac{1}{4}$. This means that for any $i \leq j$, $p(i,j) > 1/4$.

Let $\mathcal{V}_{n\times m}$ denote the distribution on matrices of size $n \times m$ resulting from the following process. Choose randomly and uniformly a $m \times (m-n)$ matrix H of rank $m-n$ and then choose randomly and uniformly a matrix G of size $n \times m$ of rank n such that $GH = 0^{n\times(m-n)}$. To generate G's like this we find a basis $b_1, \ldots, b_n$ for the subspace of $\{0,1\}^m$ that is "orthogonal" to H in the standard (and efficient) way. Let G_0 denote the matrix whose rows are the vectors $b_1, \ldots, b_n$. It is easy to see that any matrix G of rank n such that $GH = 0^{n\times(m-n)}$, can be represented uniquely as $F \cdot G_0$ where F is a matrix of size $n \times n$ and full rank $(*)$. Therefore we can generate G's as above by choosing

randomly and uniformly a matrix F of rank n. If we choose a random matrix F according $\mathcal{U}_{n\times n}$, with probability at least $p(n,n) > 1/4$, it will have the full rank. We can repeatedly sample from $\mathcal{U}_{n\times n}$ to get a full-rank F with any desired probability. This implies that we can generate a matrix according to $\mathcal{V}_{n\times m}$ with probability $1-\delta$ in time $O(m^3 \log(1/\delta))$ (or less if a non-trivial matrix multiplication algorithm is used).

All we need to prove now is that $\mathcal{V}_{n\times m}$ is "close" to $\mathcal{U}_{n\times m}$. More specifically, the *statistical distance* between two distributions $\mathcal{D}_1$ and $\mathcal{D}_2$ over X is defined to be $\Delta(\mathcal{D}_1, D_2) = \frac{1}{2}\sum_{x\in X} |\mathcal{D}_1(x) - \mathcal{D}_2(x)|$. It is well known and easy to see that for any event $E \subseteq X$, $|\mathbf{Pr}_{\mathcal{D}_1}[x \in E] - \mathbf{Pr}_{\mathcal{D}_2}[x \in E]| \leq \Delta(\mathcal{D}_1, D_2)$.

Lemma 5 *The distribution $\mathcal{V}_{n\times m}$ is uniform over matrices of size $n\times m$ and rank n. In particular, $\Delta(\mathcal{V}_{n\times m}, \mathcal{U}_{n\times m}) \leq 2^{-m+n}$.*

Proof Let G be any matrix of size $n\times m$ with linearly independent rows. Its probability under $\mathcal{U}_{n\times m}$ is $\mathcal{U}_{n\times m}(G) = 2^{-mn}$. When sampling with respect to $\mathcal{V}_{n\times m}$, G can be obtained only if all the columns of H are "orthogonal" to rows of G, that is belong to a linear subspace of $\{0,1\}^m$ of dimension $m-n$. The total number of H's like these of rank $m-n$ is $2^{(m-n)^2} p(m-n, m-n)$ (as follows from $(*)$) and the total number of matrices size $m\times(m-n)$ of rank $m-n$ is $2^{m(m-n)} p(m-n,m)$. Therefore the probability of getting each H like this is $2^{-n(m-n)} \frac{p(m-n,m-n)}{p(m-n,m)}$. Given H the total number of matrices of size $n\times m$ and rank n that are "orthogonal" to H is $p(n,n)2^{n^2}$ (as follows from $(*)$) and therefore G will be generated with probability $2^{-n^2}/p(n,n)$. Hence the total probability of G under $\mathcal{V}_{n\times m}$ is $\mathcal{V}_{n\times m}(G) = 2^{-mn}\frac{p(m-n,m-n)}{p(m-n,m)p(n,n)}$. For every $i<j$, $p(j-i,j)p(i,i) = p(j,j)$. Therefore $\mathcal{V}_{n\times m}(G) = 2^{-mn}/p(n,m)$. This implies that $\mathcal{V}_{n\times m}$ is uniform over matrices of size $n\times m$ and rank n. The statistical distance between $\mathcal{V}_{n\times m}$ and $\mathcal{U}_{n\times m}$ equals to

$$\frac{1}{2}\sum_{G\in\{0,1\}^{n\times m}} |\mathcal{V}_{n\times m}(G) - \mathcal{U}_{n\times m}(G)| \quad =$$

$$\frac{1}{2}\left[\sum_{\mathtt{rank}(G)<n} 2^{-mn} + \sum_{\mathtt{rank}(G)=n} 2^{-mn}\left(\frac{1}{p(n,m)} - 1\right)\right] \quad = \quad 1 - p(n,m).$$

According to Equation (1), $1-p(n,m) < 1-(1-2^{-m+n}) = 2^{-m+n}$. ■

We can now prove that decoding of random linear codes implies attribute-efficient learning of parities from random examples only.

Theorem 6 *Assume that there exists an algorithm* `RandDec` *that corrects a random linear $[m,n]$ code from up to w errors with probability at least $1/2+\gamma$ for any constant γ. Then PAR(w) over $\{0,1\}^m$ is efficiently learnable from $m-n$ random examples.*

Proof Let $\chi_e \in \mathrm{PAR}(w)$ be the unknown parity function and $z_1, z_2, \ldots, z_{m-n}$ be random and uniform examples given by the example oracle. Let H be the $m\times(m-n)$ matrix whose column i is equal to z_i for each $i \leq m-n$. If H does not have rank $m-n$ we return χ_{0^m}. Otherwise let G be a random matrix such that $GH = 0^{n\times(m-n)}$ generated as in the description of $\mathcal{V}_{n\times m}$ for $\delta = \gamma/2$. The values of χ_e on z_i's give us the vector eH. Let y be any solution to the linear equation $yH = eH$. Clearly $(y\oplus e)H = 0^{m-n}$ and therefore $y\oplus e$ equals to xG for some $x\in\{0,1\}^n$. This means that `RandDec` (if

successful) will output x on input G and y. By the definition of x, $e = xG \oplus y$, giving us the desired parity function.

To analyze the success probability of the algorithm we observe that the procedure above generates G according to $\mathcal{V}_{n\times m}$ with probability at least $p(m-n,m)(1-\gamma/2) \geq 1-2^{-n}-\gamma/2$. According to Lemma 5, the statistical distance between the G generated as above and $\mathcal{U}_{n\times m}$ is at most 2^{-m+n}. `RandDec` is successful with probability $1/2+\gamma$ and therefore our algorithm will succeed with probability at least $1/2+\gamma-(\gamma/2+2^{-n}+2^{-m+n}) \geq 1/2$. ■

The transformation above produces an attribute-efficient algorithm only if $m-n$ is polynomial in w and $\log m$. According to the Gilbert-Varshamov bound, a random linear code will, with high probability, have distance $d = \Omega(\frac{m-n}{\log m})$. Therefore if the number of errors that `RandDec` can correct is at least $w = d^{\alpha}$ errors for some constant $\alpha > 0$ then the sample complexity of learning a parity of length at most w over m variables would equal $O(w^{1/\alpha}\log m)$. Therefore such an algorithm could be used to obtain an attribute-efficient algorithm for learning parities.

We have noted previously that using a parity learning algorithm to obtain a syndrome decoding algorithm requires the parity learning algorithm to be proper. When a distribution over examples is not restricted it is unknown whether proper learning of parities is harder than non-proper. Fortunately, when learning with respect to the uniform distribution any learning algorithm for parities can be converted to a proper and exact one (that is, with a hypothesis equal to the target function). We include a proof of this folklore fact for completeness.

Fact 7 *Let $\mathcal{A}$ be an algorithm that learns PAR(k) in time $t(n,k,\varepsilon)$ and with sample complexity $s(n,k,\varepsilon)$. Then there exists a probabilistic algorithm $\mathcal{A}'$ that learns PAR(k) properly and exactly in time $t(n,k,1/5)+\tilde{O}(nk)$ and using $s(n,k,1/5)$ samples.*

Proof We assume for simplicity that if $\mathcal{A}$ is probabilistic then it succeeds with probability at least $3/4$. Let h be the output of $\mathcal{A}$ when running on an unknown parity $\dot{\chi}_e \in \mathrm{PAR}(k)$ with $\varepsilon = 1/5$. Given h that is correct on $4/5$ of all the points we can use it simulate membership queries to $\dot{\chi}_e(x)$ as follows. Let $y \in \{0,1\}^n$ be any point and let x be a randomly and uniformly chosen point. Then $h(x) = \dot{\chi}_e(x)$ with probability at least $4/5$ and $h(x\oplus y) = \dot{\chi}_e(x\oplus y)$ with probability at least $4/5$. Therefore with probability at least $3/5$, $h(x)\oplus h(x\oplus y) = \dot{\chi}_e(x)\oplus\dot{\chi}_e(x\oplus y) = \dot{\chi}_e(y)$. We can increase the confidence in the label to $1-\delta$ by repeating this procedure for $O(\log(1/\delta))$ independent x's. Given these membership queries we can use a proper and exact MQ algorithm for learning PAR(k). A number of such algorithms are known running in time $\tilde{O}(nk)$ and using $O(k\log n)$ MQs (including `AEParityStat`(k) given in Theorem 9). In order to get correct answers to all the membership queries with probability at least $3/4$ we need each of the MQs to be correct with probability $1-\delta$ for $\delta = \Omega(\frac{1}{k\log n})$. This means that making $O(k\log n)$ MQs will take $O(nk\log n\log(k\log n)) = \tilde{O}(nk)$ steps. Altogether we get algorithm $\mathcal{A}'$ that succeeds with probability at least $1/2$ and has the claimed complexity bounds. ■

We can now assume that algorithms for learning parity with respect to the uniform distribution are proper and exact (and in particular do not require parameter ε) and use this to obtain the other direction of the equivalence.

Theorem 8 *Assume that there exists an algorithm `AELearnPar`$_{\mathcal{U}}$(`k`) that efficiently learns PAR(k) over $\{0,1\}^m$ using at most $q(m,k)$ random examples. Then there exists an algorithm `RandDec` that corrects a random linear $[m,m-q(m,k)]$ code from up to k errors with probability at least $1/2-\gamma$ for any constant $\gamma > 0$.*

Proof Let G and y be the input of $\texttt{RandDec}$, $n = m - q(m,k)$, x be the vector for which $y = xG \oplus e$ where $\texttt{wt}(e) \leq k$. If G is not of rank n we just return the vector 0^n. Otherwise let H be a random matrix such that $GH = 0^{n\times(m-n)}$ generated as rank $m-n$ we return χ_{0^m}. Otherwise let G be a random matrix such that $GH = 0^{n\times(m-n)}$ generated as in the description of $\mathcal{V}_{n\times m}$ for $\delta = \gamma/2$ (with the roles of G and H reversed).

The syndrome yH is equal to eH and gives the values of $\dot{\chi}_e$ on $q(m,k)$ columns of H. We feed these columns as random examples to $\texttt{AELearnPar}_{\mathcal{U}}(\texttt{k})$ and obtain $\dot{\chi}_e$ from it (if $\texttt{AELearnPar}_{\mathcal{U}}(\texttt{k})$ is successful). Given e we obtain x by solving the system of linear equations $xG = y \oplus e$. To analyze the success probability of the algorithm we observe that the procedure above generates H according to $\mathcal{V}_{m\times(m-n)}$ with probability at least $p(n,m)(1-\gamma/2) \geq 1 - 2^{-q(m,k)} - \gamma/2$. According to Lemma 5, the statistical distance between H's generated as above and $\mathcal{U}_{m\times(m-n)}$ is at most $2^{-m+(m-n)} = 2^{-n}$. Therefore $\texttt{AELearnPar}_{\mathcal{U}}(\texttt{k})$ will succeed with probability at least $1/2 - 2^{-n}$. This implies that $\texttt{RandDec}$ will return the correct x with probability at least $1/2 - (2^{-m+q(m,k)} + 2^{-q(m,k)} + \gamma/2) \geq 1/2 - \gamma$. ■

4. A Fast Randomized Algorithm for ae.naMQ Learning of Parities

We next present a simple randomized algorithm for ae.naMQ learning of parities. The only previously known ae.naMQ algorithm for learning parities is due to Hofmeister (1999) and is a deterministic algorithm based on constructing and decoding of BCH binary linear codes (see also Section 3.2). The algorithm we present is substantially simpler and has essentially the same asymptotic complexity as Hofmeister's.

The basic idea of our algorithm is to use a distribution over $\{0,1\}^n$ for which each attribute is correlated with the parity function if and only if it is present in the parity.

Theorem 9 *For each $k \leq n$ there exists an algorithm $\texttt{AEParityStat}(k)$ that ae.naMQ learns the class PAR(k) in time $O(nk\log n)$ and asks $O(k\log n)$ MQs.*

Proof Let $\dot{\chi}_c$ be the target concept (such that $\texttt{wt}(c) \leq k$). We define $\mathcal{D}_{\frac{1}{t}}$ to be the product distribution such that for each i, $\mathbf{Pr}[x_i = 1] = \frac{1}{t}$. Let us draw a point x randomly according to distribution $\mathcal{D}_{\frac{1}{4k}}$. Then for each $i \leq n$

$$\begin{aligned}\mathbf{Pr}_{\mathcal{D}_{\frac{1}{4k}}}[x_i = 1 \text{ and } \dot{\chi}_c(x) = 1] &= \mathbf{Pr}_{\mathcal{D}_{\frac{1}{4k}}}[\dot{\chi}_c(x) = 1 \mid x_i = 1]\,\mathbf{Pr}_{\mathcal{D}_{\frac{1}{4k}}}[x_i = 1] \\ &= \frac{1}{4k}\mathbf{Pr}_{\mathcal{D}_{\frac{1}{4k}}}[\dot{\chi}_c(x) = 1 \mid x_i = 1]\,.\end{aligned}$$

Our second observation is that for any set of indices $B \subseteq [n]$ and the corresponding parity function $\dot{\chi}_b$,

$$\mathbf{Pr}_{\mathcal{D}_{\frac{1}{4k}}}[\dot{\chi}_b(x) = 1] \leq 1 - \mathbf{Pr}_{\mathcal{D}_{\frac{1}{4k}}}[\forall i \in B,\ x_i = 0] = 1 - (1 - \frac{1}{4k})^{|B|} \leq \frac{|B|}{4k}\,.$$

First examine the case that $c_i \neq 1$ and therefore does not influence $\dot{\chi}_c$. Then by the second observation,

$$\mathbf{Pr}_{\mathcal{D}_{\frac{1}{4k}}}[\dot{\chi}_c(x) = 1 \mid x_i = 1] = \mathbf{Pr}_{\mathcal{D}_{\frac{1}{4k}}}[\dot{\chi}_c(x) = 1] \leq \frac{k}{4k} \leq 1/4\,.$$

Now assume that $c_i = 1$ and let $c' = c \oplus e_i$. Then $\dot{\chi}_{c'}(x)$ is independent of x_i and $\dot{\chi}_c(x) = 1$ if and only if $\dot{\chi}_{c'}(x) = 0$. Therefore

$$\begin{aligned}\mathbf{Pr}_{\mathcal{D}_{\frac{1}{4k}}}[\dot{\chi}_c(x) = 1 \mid x_i = 1] &= \mathbf{Pr}_{\mathcal{D}_{\frac{1}{4k}}}[\dot{\chi}_{c'}(x) = 0 \mid x_i = 1] \\ &= 1 - \mathbf{Pr}_{\mathcal{D}_{\frac{1}{4k}}}[\dot{\chi}_{c'}(x) = 1] \geq 1 - \frac{k-1}{4k} > 3/4\,.\end{aligned}$$

Hence estimation of $\mathbf{Pr}_{\mathcal{D}_{\frac{1}{4k}}}[x_i = 1 \text{ and } \dot{\chi}_c(x) = 1]$ within the half of the expectation can be used to find out whether $c_i = 1$. Lemma 1 for $\gamma = 1/2$ implies that by taking $O(k \log n)$ independent samples with respect to $\mathcal{D}_{\frac{1}{4k}}$ we will get that each estimate is correct with probability at least $1 - 1/(2n)$ and therefore we will discover c with probability at least $1 - n/(2n) = 1/2$. The running time of $\texttt{AEParityStat}(k)$ is clearly $O(nk \log n)$. ■

5. Finding Fourier Coefficients and Weak DNF Learning

The original Jackson's algorithm for learning DNF expressions with respect to the uniform distribution is based on a procedure that weakly learns DNF with respect to the uniform distribution (Jackson, 1997). The procedure for weak learning is essentially an algorithm that, given a Boolean function f finds a significant Fourier coefficient of f, if one exist. Jackson's algorithm is based on a technique by Goldreich and Levin (1989) for finding a significant Fourier coefficient (also called the KM algorithm (Kushilevitz and Mansour, 1991)). Bshouty, Jackson, and Tamon (1999) used a later algorithm by Levin (1993) to give a significantly faster weak learning algorithm. In this section we will briefly describe Levin's algorithm with improvements by Bshouty et al.. Building on their ideas we then present an attribute-efficient and non-adaptive version of the improved Levin's algorithm. This algorithm will give us an ae.naMQ algorithm for weak learning of DNF expressions that will serve as the basis of our ae.naMQ algorithm for DNF learning.

A Fourier coefficient $\hat{\phi}(a)$ of a real-valued function ϕ over $\{0,1\}^n$ is said to be θ-*heavy* if $|\hat{\phi}(a)| \geq \theta$. For a Boolean f, $\mathbf{E}[f\chi_a] \geq \theta$ if and only if $\mathbf{Pr}[f = \chi_a] \geq 1/2 + \theta/2$. This means that $|\hat{f}(a)| \geq \theta$ is equivalent to either χ_a or $-\chi_a$ being a $(1/2 - \theta/2)$-approximator of f. Therefore finding a significant Fourier coefficient of f is sometimes called weak parity learning (Jackson, 1997). It can also be interpreted as a learning algorithm for parities in the agnostic learning framework of Haussler (1992) and Kearns et al. (1994) Feldman et al. (see the work of 2006, for details).

Definition 10 (Weak Parity Learning) *Let f be a Boolean function with at least one θ-heavy Fourier coefficient. Given $\theta > 0$ and access to MEM(f), the weak parity learning problem consists of finding a vector z such that $\hat{f}(z)$ is $\theta/2$-heavy.*

We will only consider algorithms for weak parity learning that are efficient, that is, produce the result in time polynomial in n, and θ^{-1}. In addition we are interested in weak parity learning algorithms that are attribute-efficient.

Definition 11 (Attribute-Efficient Weak Parity Algorithm) Attribute-efficient weak parity algorithm *is an algorithm that given k, θ, and MEM(f) for f that has a θ-heavy Fourier coefficient of degree at most k efficiently solves weak parity learning problem and asks polynomial in $k, \log n$, and θ^{-1} number of MQs.*

We follow the presentation of Levin's weak parity algorithm given by Bshouty et al. and refer the reader to their paper for detailed proofs of all the statements and smaller remarks (we use the same definitions and notation to simplify the reference). Levin's algorithm is based on estimating a Fourier coefficient $\hat{f}(a)$ by sampling f on randomly-chosen pairwise independent points. More specifically, the following pairwise independent distribution is generated. For a fixed m, a random m-by-n 0-1 matrix R is chosen and the set $Y = \{pR \mid p \in \{0,1\}^m \setminus \{0^m\}\}$ is formed. For different vectors p_1 and p_2 in $\{0,1\}^m \setminus \{0^m\}$, p_1R and p_2R are pairwise independent. The variance σ^2 of a Boolean function is upper-bounded by 1 and thus Bienaymé-Chebyshev's inequality (Lemma 2) implies that

$$\mathbf{Pr}_R\left[\left|\frac{\sum_{x\in Y} f(x)\chi_a(x)}{2^m-1} - \hat{f}(a)\right| \geq \gamma\right] \leq \frac{1}{(2^m-1)\gamma^2} \tag{2}$$

Therefore using a sample for $m = \log(16\rho^{-1}\theta^{-2}+1)$, $\sum_{x\in Y} f(x)\chi_a(x)$ will, with probability at least $1-\rho$, approximate $\hat{f}(a)$ within $\theta/4$.

On the other hand, $\sum_{x\in Y} f(x)\chi_a(x)$ is a summation over all (but one[2]) elements of a linear subspace of $\{0,1\}^n$ and therefore can be seen as a Fourier coefficient of f restricted to subspace Y. That is, if we define $f_R(p) = f(pR)$ then, by definition of Fourier transform, for every $z \in \{0,1\}^m$

$$\widehat{f_R}(z) = 2^{-m} \sum_{p\in\{0,1\}^m} f_R(p)\chi_z(p) \ .$$

This together with equality $\chi_a(pR) = \chi_{aR^T}(p)$ implies that $\hat{f}(a)$ is approximated by $\widehat{f_R}(aR^T)$ (with probability at least $1-\rho$).

All the coefficients $\widehat{f_R}(z)$ can be computed exactly in time $O(m2^m)$ via the FFT algorithm giving estimations to all the Fourier coefficients of f.

Another key element of the weak parity algorithm is the following equation (Bshouty et al., 1999).

Lemma 12 *For $c \in \{0,1\}^n$ let $f_c(x) = f(x \oplus c)$. Then $\widehat{f_c}(a) = \hat{f}(a)\chi_a(c)$.*

Proof

$$\widehat{f_c}(a) = 2^{-n} \sum_{x\in\{0,1\}^n} f(x\oplus c)\chi_a(x) = 2^{-n} \sum_{x\in\{0,1\}^n} f(x)\chi_a(x\oplus c) = \hat{f}(a)\chi_a(c) \ .$$

■

Assuming that $\hat{f}(a) \geq \theta$ estimation of $\hat{f}(a)$ within $\theta/4$ (when successful) has the same sign as $\hat{f}(a)$. Similarly we can obtain the sign of $\widehat{f_c}(a)$. By Lemma 12, the sign of the product $\hat{f}(a)\widehat{f_c}(a)$ is equal to $\chi_a(c)$. This gives a way to make MQs for χ_a using the values $\widehat{f_{c,R}}(aR^T)$ for a random R. Levin and Bshouty et al. implicitly used this technique with a basic membership query algorithm for learning parities. The speed-up in Levin's algorithm is achieved by making each MQ to many χ_a's in parallel. Therefore only a non-adaptive membership query algorithm for learning parities can be used. In our next theorem we give an interpretation of improved Levin's algorithm that makes the use of a non-adaptive membership query algorithm explicit.

2. The value at 0^m does not influence the estimation substantially and therefore can be offset by slightly increasing the size of sample space Y (Bshouty et al., 1999).

Theorem 13 *Let $\mathcal{B}(k)$ be an ae.naMQ algorithm for learning parities that runs in time $t(n,k)$ and uses $q(n,k)$ MQs. There exists an attribute-efficient and non-adaptive algorithm* `AEBoundedSieve-`$\mathcal{B}(\theta,k)$ *that, with probability at least $1-\delta$, solves the weak parity learning problem.* `AEBoundedSieve-`$\mathcal{B}(\theta,k)$ *runs in time $\tilde{O}\left(\theta^{-2}t(n,k)\cdot q(n,k)\log(1/\delta)\right)$ and asks $\tilde{O}\left(\theta^{-2}q^2(n,k)\log(1/\delta)\right)$ MQs.*

Proof We assume for simplicity that $\mathcal{B}(k)$ succeeds with probability at least $3/4$. Besides that according to Fact 7, we can assume that $\mathcal{B}(k)$ is a proper algorithm.

Let S be the set of MQs for an execution of $\mathcal{B}(k)$. Choose randomly an m-by-n matrix R for $m=\log(16\theta^{-2}\cdot 4\cdot(q(n,k)+1)+1)$ and compute the Fourier transforms of $f_R=f_{0^n,R}$ and $f_{y,R}$ for each $y\in S$ via the FFT algorithm. Then, for each $z\in\{0,1\}^m$, we run $\mathcal{B}(k)$ with the answer to MQ $y\in S$ equal to $\mathtt{sign}(\widehat{f_R}(z)\widehat{f_{y,R}}(z))$. If the output of $\mathcal{B}(k)$ is a parity function χ_a of length at most k then we test that $(i):|\widehat{f_R}(z)|\geq 3\theta/4$ and $(ii):aR^T=z$. If both conditions are satisfied we add a to the set of hypotheses H.

By Equation (2), for a such that $|\hat{f}(a)|\geq\theta$ and $\mathtt{wt}(a)\leq k$, with probability at least $1-\frac{1}{4(q(n,k)+1)}$, each of the estimations $\widehat{f_{y,R}}(aR^T)$ for $y\in S\cup\{0^n\}$ will be within $\theta/4$ of $\widehat{f_y}(a)$. In particular, with probability at least $3/4$, for all $y\in S\cup\{0^n\}$, $\mathtt{sign}(\widehat{f_y}(a))=\mathtt{sign}(\widehat{f_{y,R}}(aR^T))$. If all the signs are correct then by Lemma 12, $\mathtt{sign}(\widehat{f_R}(z)\widehat{f_{y,R}}(z))=\chi_a(y)$ and as a result $\mathcal{B}(k)$ will succeed with probability at least $3/4$. Therefore a will satisfy both conditions (i) and (ii) and will be added as a possible hypothesis with probability at least $1/2$. Note that $\mathcal{B}(k)$ is executed on up to 2^m possible hypotheses while using the same set of queries S. This is only possible for a non-adaptive algorithm $\mathcal{B}(k)$.

On the other hand, for any fixed b such that $|\hat{f}(b)|<\theta/2$, if $bR^T=z$ (condition (ii)) then with probability at least $1-\frac{1}{4(q(n,k)+1)}\geq 7/8$, $\widehat{f_R}(z)$ approximates $\hat{f}(b)$ within $\theta/4$. This implies that $|\widehat{f_R}(z)|<3\theta/4$ and therefore condition (i) will be failed with probability at least $7/8$. This implies that b can be added to the set of hypotheses with probability at most $1/8$.

Now we use a simple method of Bshouty et al. (1999) to remove all "bad" (not $\theta/2$-heavy) hypotheses from the set of hypotheses without removing the "good" ones (θ-heavy). We repeat the described algorithm ℓ times for independent choices of R and S generating ℓ sets of hypotheses (each of size at most 2^m). This procedure generates at most $\ell 2^m$ hypotheses. According to Chernoff's bound (Lemma 1) each "good" hypothesis appears in at least $1/3$ of all the sets with probability at least $1-2^{-\alpha\ell}$ and each fixed "bad" hypothesis appears in at least $1/3$ of all the sets with probability at most $2^{-\alpha\ell}$, for a fixed constant α (since $1/8<1/3<1/2$). Note that we need to fix a "bad" hypothesis to apply this argument. A hypothesis can be fixed as soon as it has appeared in a set of hypotheses. We then exclude the first set in which a hypothesis has appeared when counting the fraction of sets in which the hypothesis has appeared (Chernoff bound is now on $\ell-1$ trials but this is insubstantial). By setting $\ell=(m+\log m+2\log(1/\delta)+3)/\alpha$ we will get that $\ell 2^m 2^{-\alpha\ell-1}\leq\delta/2$. Therefore the probability that a "bad" hypothesis will appear in $1/3$ of the sets is at most $\delta/2$. Similarly all "good" hypotheses will appear in $1/3$ of the sets with probability at least $1-\delta/2$. Thus by picking any a that appears in at least $1/3$ of all the sets we will find a $\theta/2$-heavy coefficient with probability at least $1-\delta$.

Computing each of the Fourier transforms takes $O(m2^m)=\tilde{O}(\theta^{-2}\cdot q(n,k))$ time. They are performed for each of $q(n,k)$ MQs of $\mathcal{B}$ and this is repeated $\ell=O(m+\log(1/\delta))$ times giving the total bound of $\tilde{O}(\theta^{-2}q^2(n,k)\log(1/\delta))$. For each of the 2^m values of z we run $\mathcal{B}(k)$ and tests (i) and (ii).

This takes $O(2^m(t(n,k)+mn)) = \tilde{O}(\theta^{-2}t(n,k)\cdot q(n,k))$ time and is repeated $\ell = O(m+\log(1/\delta))$ times. Therefore the total running time is $\tilde{O}(\theta^{-2}\cdot t(n,k)\cdot q(n,k)\log(1/\delta))$. Similarly we observe that each of the estimations via FFT uses 2^m examples and $\ell\cdot(q(n,k)+1)$ such estimations are done. This implies that the sample complexity of the algorithm is $\tilde{O}\left(\theta^{-2}q^2(n,k)\log(1/\delta)\right)$. It can also be easily seen that all MQs are non-adaptive. ■

Another way to see Theorem 13 is as a way to convert an ae.naMQ algorithm for learning of parities to an ae.naMQ algorithm for agnostic learning of parities.

By plugging $\texttt{AEParityStat}(k)$ algorithm (Theorem 9) into Theorem 13 we obtain our weak parity learning algorithm.

Corollary 14 *There exists an attribute-efficient and non-adaptive weak parity learning algorithm* $\texttt{AEBoundedSieve}(\theta,\texttt{k})$ *that succeeds with probability at least* $1-\delta$*, runs in time* $\tilde{O}\left(nk^2\theta^{-2}\log(1/\delta)\right)$*, and asks* $\tilde{O}\left(k^2\log^2 n\cdot\theta^{-2}\log(1/\delta)\right)$ *MQs.*

Jackson (1997) has proved that for every distribution $\mathcal{D}$, every DNF formula f has a parity function that weakly approximates f with respect to $\mathcal{D}$. A refined version of this claim by Bshouty and Feldman (2002) shows that f has a short parity that weakly approximates f if the distribution is not too far from the uniform. More formally, for a real-valued function ϕ we define $L_\infty(\phi) = \max_x\{|\phi(x)|\}$ and we view a distribution $\mathcal{D}$ as a function over $\{0,1\}^n$ that for a point x gives its probability weight under $\mathcal{D}$.

Lemma 15 *For any Boolean function f of DNF-size s and a distribution $\mathcal{D}$ over $\{0,1\}^n$ there exists a parity function χ_a such that*

$$|\mathbf{E}_{\mathcal{D}}[f\chi_a]| \geq \frac{1}{2s+1} \text{ and } \texttt{wt}(a) \leq \log\left((2s+1)L_\infty(2^n\mathcal{D})\right).$$

By combining this fact with Corollary 14 we get an algorithm for weakly learning DNF.

Theorem 16 *There exist an algorithm* $\texttt{WeakDNF}_{\mathcal{U}}(s)$ *that for a Boolean function f of DNF-size s given n,s, and access to MEM(f), with probability at least $1/2$, finds a $(\frac{1}{2}-\Omega(\frac{1}{s}))$-approximator to f with respect to $\mathcal{U}$. Furthermore,* $\texttt{WeakDNF}_{\mathcal{U}}(s)$ *runs in time* $\tilde{O}\left(ns^2\right)$ *and asks* $\tilde{O}\left(s^2\log^2 n\right)$ *non-adaptive MQs.*

Proof Lemma 15 implies that there exists a parity χ_a on at most $\log(2s+1)$ variables such that $|\mathbf{E}_{\mathcal{U}}[f\chi_a]| = |\hat{f}(a)| \geq \frac{1}{2s+1}$. This means that f has a $\frac{1}{2s+1}$-heavy Fourier coefficient of degree at most $\log(2s+1)$. Using Corollary 14 for $\delta = 1/2$, we can find a $\frac{1}{2(2s+1)}$-heavy Fourier coefficient $\hat{f}(a')$ in time $\tilde{O}\left(ns^2\right)$ and using $\tilde{O}\left(s^2\log^2 n\right)$ non-adaptive MQs. The parity $\chi_{a'}$ or its negation $(\frac{1}{2}-\frac{1}{4(2s+1)})$-approximates f. ■

The algorithm for weakly learning DNFs by Bshouty et al. (1999) requires $\tilde{O}\left(ns^2\right)$ MQs and runs in time[3] $\tilde{O}\left(ns^2\right)$.

3. The running time bound is based on use of a membership query oracle, that given any two vectors $x,y\in\{0,1\}^n$, passed to it "by reference", returns $f(x\oplus y)$ in $O(1)$ time.

6. Learning DNF Expressions

In this section we show an ae.naMQ algorithm for learning DNF expressions. Following Jackson's approach we first show how to generalize our weak DNF learning algorithm to other distributions (Jackson, 1997). We then use Freund's boosting algorithm to obtain a strong DNF learning algorithm (Freund, 1992). Besides achieving attribute-efficiency and non-adaptiveness we show a way to speed up the boosting process by exploiting several properties of our `WeakDNF` algorithm.

6.1 Weak DNF Learning with Respect to Any Distribution

The first step in Jackson's approach is to generalize a weak parity algorithm to work for any real-valued function. We follow this approach and give a generalization of our `AEBoundedSieve`$(\theta,\mathtt{k})$ algorithm (Corollary 14) to any real-valued and also randomized functions.

Lemma 17 *There exists an algorithm* `AEBoundedSieveRV`$(\theta,\mathtt{k},\mathtt{V})$ *that for any real-valued randomized function* Ψ *with a* θ*-heavy Fourier coefficient of degree at most* k*, given* k*,* θ*,* $V \geq \mathbf{Var}_{\mathcal{U},\Psi}(\Psi(x))$*, and an oracle access to* Ψ*, finds, with probability at least* $1-\delta$*, a* $\theta/2$*-heavy Fourier coefficient of* Ψ *of degree at most* k*. The algorithm runs in time* $\tilde{O}\left(nk^2\theta^{-2}V\log(1/\delta)\right)$ *and asks* $\tilde{O}\left(k^2\log^2 n\cdot\theta^{-2}V\log(1/\delta)\right)$ *non-adaptive MQs.*

Proof By revisiting the proof of Theorem 13, we can see that the only place where we used the fact that f is Boolean and deterministic is when relying on Equation (2) in which the variance of the random variable $f(x)\in\{-1,+1\}$ was upper-bounded by 1. In this bound $f(x)$ is already treated as a random variable on pairwise independent x's. For any point x, $\Psi(x)$ is independent of any other evaluations of Ψ and therefore evaluations of Ψ on pairwise independent points are pairwise independent. This implies that in order to estimate $\widehat{\Psi}(a)$ within $\theta/4$ we only need to account for the fact that the variance of $\Psi(x)$ is not necessarily bounded by 1. This can be done by using $\mathbf{Var}(\Psi)\leq V$ times more samples, that is, we set $m=\log(16V\theta^{-2}\cdot 4\cdot(q(n,k)+1)+1)$. It is now straightforward to verify that the rest of the proof of Theorem 13 is unchanged. The increase in the required sample size increases the running time and the sample complexity of the algorithm by a factor $\tilde{O}(V)$ giving us the claimed bounds. ■

As in Jackson's work we use the generalized weak parity algorithm to obtain an algorithm that weakly learns DNF expressions with respect to any distribution. The algorithm is efficient only when the distribution function is "close" to the uniform and requires access to the value of the distribution function at any point x.

Theorem 18 *There exist an algorithm* `WeakDNF`$(\mathtt{s},\mathtt{B})$ *that for a Boolean function* f *of DNF-size* s *and any distribution* $\mathcal{D}$*, given* $n,s,B\geq L_\infty(2^n\mathcal{D}(x))$*, access to MEM*$(f)$*, and an oracle access to* $\mathcal{D}$*, with probability at least* $1-\delta$*, finds a* $(\frac{1}{2}-\Omega(\frac{1}{s}))$*-approximator to* f *with respect to* $\mathcal{D}$*. Furthermore,* `WeakDNF`$(\mathtt{s},\mathtt{B})$

- *runs in time* $\tilde{O}(ns^2B\log(1/\delta))$*;*
- *asks* $\tilde{O}(s^2\log^2 n\cdot B\log(1/\delta))$ *non-adaptive MQs;*
- *returns a parity function of length at most* $O(\log(sB))$ *or its negation.*

Proof Lemma 15 states that there exists a vector a of Hamming weight bounded by $O(\log(sL_\infty(2^n\mathcal{D})))$ such that $|\mathbf{E}_{\mathcal{D}}[f(x)\chi_a(x)]| = \Omega(1/s)$. But

$$\mathbf{E}_{\mathcal{D}}[f(x)\chi_a(x)] = \sum_x [f(x)\mathcal{D}(x)\chi_a(x)] = \mathbf{E}[f(x)2^n\mathcal{D}(x)\chi_a(x)] = \widehat{\psi}(a)\ , \tag{3}$$

where $\psi(x) = f(x)2^n\mathcal{D}(x)$. This means that $\psi(x)$ has a $\Omega(1/s)$-heavy Fourier coefficient of degree bounded by $O(\log(sL_\infty(2^n\mathcal{D}))) = O(\log(sB))$. We can apply `AEBoundedSieveRV` on $\psi(x)$ to find its $\Omega(1/s)$-heavy Fourier coefficient of degree $O(\log(sB))$. All we need to do this is to provide a bound V on the variance of $f(x)2^n\mathcal{D}(x)$.

$$\begin{aligned}
\mathbf{Var}(f(x)2^n\mathcal{D}(x)) &= \mathbf{E}[(f(x)2^n\mathcal{D}(x))^2] - \mathbf{E}^2[f(x)2^n\mathcal{D}(x)] \\
&\leq L_\infty(2^n\mathcal{D}(x))\mathbf{E}[2^n\mathcal{D}(x)] - \mathbf{E}^2[f(x)2^n\mathcal{D}(x)] \leq L_\infty(2^n\mathcal{D}(x))\mathbf{E}[2^n\mathcal{D}(x)] \\
&= L_\infty(2^n\mathcal{D}(x)) \leq B
\end{aligned} \tag{4}$$

This bound on variance relies essentially on the fact that $\mathcal{D}(x)$ is a distribution function [4] and therefore $\mathbf{E}[2^n\mathcal{D}(x)] = \mathbf{E}_{\mathcal{D}}[1] = 1$. This improves on $L_\infty^2(2^n\mathcal{D}(x))$ bound for an unrestricted function $\mathcal{D}(x)$ that was used in analysis of previous weak DNF learning algorithms (Jackson, 1997; Bshouty et al., 1999).

We can now run `AEBoundedSieveRV`$(\theta, \mathtt{k}, \mathtt{V})$ for $\theta = \Omega(1/s)$, $k = O(\log(sB))$, $V = B$, and a simulated oracle access to $\psi = f2^n\mathcal{D}$ to obtain a' such that $|\widehat{\psi}(a')| = \Omega(1/s)$ and $\mathtt{wt}(a') = O(\log(sB))$. By equation (3), we get that $|\mathbf{E}_{\mathcal{D}}[f(x)\chi_{a'}(x)]| = \Omega(1/s)$ and therefore $\chi_{a'}(x)$ or its negation $(\frac{1}{2} - \Omega(\frac{1}{s}))$-approximates f with respect to $\mathcal{D}$. The claimed complexity bounds can be obtained by using Lemma 17 for θ, k and V as above. ■

6.2 Background on Boosting a Weak DNF Learner

Jackson (1997) obtained his DNF learning algorithm by converting a weak DNF learning algorithm to a strong one via a *boosting* algorithm. *Boosting* is a general technique for improving the accuracy of a learning algorithm. It was introduced by Schapire (1990) who gave the first efficient boosting algorithm. Let $\mathcal{C}$ be a concept class and let $\mathtt{WL}_\gamma$ be a weak learning algorithm for $\mathcal{C}$ that for any distribution $\mathcal{D}$, produces a $(1/2-\gamma)$-approximating hypothesis. Known boosting algorithms have the following structure.

- At stage zero $\mathtt{WL}_\gamma$ is run on $\mathcal{D}_0 = \mathcal{D}$ to obtain h_0.
- At stage i a distribution $\mathcal{D}_i$ is constructed using $\mathcal{D}$ and previous weak hypotheses $h_0, \ldots, h_{i-1}$. The distribution $\mathcal{D}_i$ usually favors the points on which the previous weak hypotheses do poorly. Then random examples from $\mathcal{D}_i$ are simulated to run $\mathtt{WL}_\gamma$ with respect to $\mathcal{D}_i$ and obtain h_i.
- After repeating this for a number of times an ε-approximating hypothesis h is created using all the generated weak hypotheses.

4. Actual $\mathcal{D}(x)$ given to a weak learner will be equal to $c\mathcal{D}'(x)$ where $\mathcal{D}'(x)$ is a distribution and c is a constant in $[2/3, 4/3]$ (Bshouty et al., 1999). This modifies the bound above by a small constant factor.

Jackson's use of Freund's boosting algorithm slightly deviates from this scheme as it provides the weak learner with the oracle that returns the density of the distribution function $\mathcal{D}_i$ at any desired point instead of simulating random examples with respect to $\mathcal{D}_i$. The `WeakDNF` algorithm also requires oracle access to $\mathcal{D}_i(x)$ and therefore we will use a boosting algorithm in the same way. The running time of Jackson's (and our) algorithm for weak learning of DNF expression depends polynomially on $L_\infty(2^n\mathcal{D})$ and therefore it can only be boosted by a boosting algorithm that produces distributions that are *polynomially-close* to the uniform distribution; that is, the distribution function is bounded by $p2^{-n}$ where p is a polynomial in learning parameters (such boosting algorithms are called *p-smooth*). In Jackson's result Freund's (1990) boost-by-majority algorithm is used to produce distribution functions bounded by $O(\varepsilon^{-2})$. More recently, Klivans and Servedio (2003) have observed that a later boosting algorithm of Freund (1992) produces distribution functions bounded by $\tilde{O}(1/\varepsilon)$, thereby improving the dependence of running time and sample complexity on ε. This improvement together with improved weak DNF learning algorithm of Bshouty et al. (1999) gives DNF learning algorithm that runs in $\tilde{O}(ns^6/\varepsilon^2)$ time and has sample complexity of $\tilde{O}(ns^4/\varepsilon^2)$.

Remark 19 *Bshouty et al. claimed sample complexity of $\tilde{O}(ns^2/\varepsilon^2)$ based on erroneous assumption that sample points for weak DNF learning can be reused across boosting stages. A distribution function $\mathcal{D}_i$ in i-th stage depends on hypotheses produced in previous stages. The hypotheses depend on random sample points and therefore in i-th stage the same set of sample points cannot be considered as chosen randomly and independently of $\mathcal{D}_i$ (Jackson, 2004). This implies that new and independent points have to be sampled for each boosting stage and increases the sample complexity of the algorithm by Bshouty et al. by a factor of $O(s^2)$.*

As in the work of Klivans and Servedio (2003), we use Freund's (1992) `B-Comb` boosting algorithm to boost the accuracy of our weak DNF learning algorithm. We will now briefly describe the `B-Comb` boosting algorithm (see also the work of Klivans and Servedio (2003) for a detailed discussion on application of `B-Comb` to learning DNF expressions).

6.2.1 FREUND'S `B-Comb` BOOSTING ALGORITHM

`B-Comb` boosting algorithm is based on a combination of two other boosting algorithms. The first one in an earlier `F1` algorithm due to Freund (1990) and is used to boost from accuracy $\frac{1}{2}-\gamma$ to accuracy $1/4$. Its output is the function equal to the majority vote of the weak hypotheses that it received. This algorithm is used as a weak learner by the second boosting algorithm `B-Filt`. At stage k `B-Filt` sets h_ℓ to be either the output of a weak learner or a random coin flip (that is a randomized function equal to either 1 or -1, each with probability $1/2$). Accordingly the distribution function generated at stage i depends on random coin flips and the final hypothesis is a majority vote over hypotheses from the weak learner and random coin flips. As it is done by Freund (1992), we analyze the algorithm for a fixed setting of these coin flip hypotheses. Freund's analysis shows that with overwhelming probability over the coin flips the randomized hypothesis produced by the boosting algorithm ε-approximates the target function.

Each of the executions of `F1` has $O(\gamma^{-2})$ stages and `B-Filt` has $O(\log(1/\varepsilon))$ stages. We denote the distribution function generated at stage i of `F1` during stage ℓ of `B-Filt` as $\mathcal{D}^{\mathtt{Comb}}_{\ell,i}$. In both boosting algorithms $\mathcal{D}_i(x) = \beta(i,N(x))\mathcal{D}/\alpha$, where $N(x)$ is the number of previous hypotheses that are correct on x, β is a fixed function from a pair of integers to the interval $[0,1]$ computable in polynomial (in the length of its input) time, and α is the normalization factor equal to $\mathbf{E}_{\mathcal{D}}[\beta(i,N(x))]$.

We can therefore say that

$$\mathcal{D}^{\texttt{Comb}}_{\ell,i}(x) = \beta(\ell, N_{\texttt{Filt}}(x)) \cdot \beta(i, N_{\texttt{F1}}(x)) \mathcal{D}(x)/(\alpha_\ell \alpha_{\ell,i}) \ , \tag{5}$$

where $N_{\texttt{Filt}}(x)$ and $N_{\texttt{F1}}(x)$ count the correct hypotheses so far for `B-Filt` and `F1` respectively. The normalization factor α_ℓ equals $\mathbf{E}_{\mathcal{D}}[\beta(\ell, N_{\texttt{Filt}}(x))]$ and

$$\alpha_{\ell,i} = \mathbf{E}_{\mathcal{D}}[\beta(\ell, N_{\texttt{Filt}}(x)) \cdot \beta(i, N_{\texttt{F1}}(x)) \mathcal{D}(x)/\alpha_\ell] \ .$$

The analysis by Freund implies that for every ℓ and i,

$$L_\infty(2^n \mathcal{D}^{\texttt{Comb}}_{\ell,i}) \leq 1/(\alpha_\ell \alpha_{\ell,i}) = \tilde{O}(1/\varepsilon) \ .$$

In Figure 6.2.1 we include the pseudocode of `B-Comb` algorithm simplified and adapted to our setting.

6.3 Optimized Boosting

We now use Freund's (1992) `B-Comb` boosting algorithm to boost the accuracy of our weak DNF learning algorithm. Unlike in the previous work, we will exploit several properties of `WeakDNF` to achieve faster execution of each boosting stage. Specifically, we note that evaluation of the distribution function $\mathcal{D}_i(x)$ at boosting stage i involves evaluation of $i-1$ previous hypotheses on x and therefore, in a general case, for a sample of size q will require $\Omega(i \cdot q)$ steps, making the last stages of boosting noticeably slower. Our goal is to show that for our `WeakDNF` algorithm and the `B-Comb` boosting algorithm the evaluation of $\mathcal{D}_i(x)$ for the whole sample needed by `WeakDNF` can be made more efficiently.

The idea of the speed-up is to use Equation (5) together with the facts that weak hypotheses are parities and MQs of `WeakDNF` come from a "small" number of low-dimension linear subspaces. Let g be a function that is equal to a linear combination of short parity functions. We start by showing a very efficient way to compute the values of g on a linear subspace of $\{0,1\}^n$. We will assume that vectors of Hamming weight at most w are represented by the list of indices where the vector is equal to 1 (as we did for parities). One can easily see that adding such vectors or multiplying them by any vector takes $O(w \log n)$ time.

Lemma 20 *Let $\{c_1, c_2, \ldots, c_i\}$ be a set of vectors in $\{0,1\}^n$ of Hamming weight at most w; $\bar{\alpha} \in \mathbb{R}^i$ be a real-valued vector, and R be a m-by-n 0-1 matrix. Then the set of pairs*

$$S = \{\langle p, \sum_{j \leq i} \alpha_j \chi_{c_j}(pR) \rangle \mid p \in \{0,1\}^m\}$$

can be computed in time $\tilde{O}(i \cdot w \log n + 2^m)$.

Proof We define $g(x) = \sum_{j \leq i} \alpha_j \chi_{c_j}(x)$ and for $p \in \{0,1\}^m$ we define $g_R(p) = g(pR)$ (as in Sect. 5). Our goal is to find the values of function g_R on all the points of $\{0,1\}^m$. The function g is given as a linear combination of parities, or in other words, we are given its Fourier transform. Given the Fourier transform of g we can derive the Fourier transform of g_R from the following equation:

$$g_R(p) = \sum_{j \leq i} \alpha_j \chi_{c_j}(pR) = \sum_{j \leq i} \alpha_j \chi_{c_j R^T}(p) = \sum_{z \in \{0,1\}^m} \left[\left(\sum_{j \leq i;\ c_j R^T = z} \alpha_j \right) \chi_z(p) \right] \ .$$

B-Comb$(\varepsilon, \delta, \mathcal{D}, \mathtt{WL}_\gamma)$

1. $k \leftarrow c_0 \log(1/\varepsilon)$
2. $\Theta \leftarrow c_1 \varepsilon / \log(1/\varepsilon)$
3. $h_0 \leftarrow \mathtt{F1}(1/4, \delta/(2k+1), \mathcal{D}, \mathtt{WL}_\gamma)$
4. **for** $\ell \leftarrow 1$ **to** k
5. $\quad N(x) \equiv |\{h_j \mid 0 \le j \le \ell - 1 \text{ and } h_j(x) = f(x)\}|$
6. $\quad \alpha'_\ell \leftarrow \mathtt{EstExpRel}(\beta(\ell, N(x)), \mathcal{D}, 1/3, \delta/(2k+1))$
7. $\quad$ **if** $\alpha'_\ell \ge \Theta$ **then**
8. $\quad\quad \mathcal{D}'_\ell \equiv \beta(\ell, N(x))/\alpha'_\ell$
9. $\quad\quad h_\ell \leftarrow \mathtt{F1}(1/4, \delta/(2k+1), \mathcal{D}'_\ell, \mathtt{WL}_\gamma)$
10. $\quad$ **else**
11. $\quad\quad h_\ell \leftarrow \mathtt{Random}(1/2)$
12. **end for**
13. **return** $\mathtt{Majority}(h_0, h_1, \ldots, h_k)$

F1$(\varepsilon, \delta, \mathcal{D}, \mathtt{WL}_\gamma)$

1. $k \leftarrow c_2/\gamma^2$
2. $\Theta \leftarrow c_3 \varepsilon^2$
3. $h_0 \leftarrow \mathtt{WL}_\gamma(\mathcal{D}, \delta/(2k+1))$
4. **for** $i \leftarrow 1$ **to** k
5. $\quad N(x) \equiv |\{h_j \mid 0 \le j \le i - 1 \text{ and } h_j(x) = f(x)\}|$
6. $\quad \alpha'_i \leftarrow \mathtt{EstExpRel}(\beta(i, N(x)), \mathcal{D}, 1/3, \delta/(2k+1))$
7. $\quad$ **if** $\alpha'_i \ge \Theta$ **then**
8. $\quad\quad \mathcal{D}'_i \equiv \beta(i, N(x))/\alpha'_i$
9. $\quad\quad h_i \leftarrow \mathtt{WL}_\gamma(\mathcal{D}'_i, \delta/(2k+1))$
10. $\quad$ **else**
11. $\quad\quad k \leftarrow i - 1$
12. $\quad\quad$ **break for**
13. **end for**
14. **return** $\mathtt{Majority}(h_0, h_1, \ldots, h_k)$

Figure 1: Pseudocode of B-Comb boosting algorithm. The first part is B-Filt with F1 used as a weak learner. $\mathtt{WL}_\gamma$ is a weak learning algorithm that has accuracy $\frac{1}{2} - \gamma$ and takes an oracle for a distribution $\mathcal{D}$ and confidence δ as parameters. $\mathtt{EstExpRel}(R, \mathcal{D}, \lambda, \delta)$ produces estimates of the expectation of a random variable R with respect to a distribution $\mathcal{D}$ within relative accuracy λ and confidence δ (that is the estimate $v' \in [(1-\lambda)v, (1+\lambda)v]$, where v is the true expectation). Various unspecified constants are denoted by $c_0, c_1, \ldots$ The membership query oracle for the target function f is available to all procedures.

Hence $\widehat{g_R}(z) = \sum_{j \le i;\, c_j R^T = z} \alpha_j$. Given the Fourier transform of g_R we can use the FFT algorithm to perform the inverse Fourier transform of g_R giving us the desired values of $g_R(p)$ on all the points of $\{0,1\}^m$. This task can be performed in $O(m2^m)$ steps. To compute the Fourier transform of g_R we need to compute $c_j R^T$ for each $j \le i$ and sum the ones that correspond to the same z. Given that each c_j is of Hamming weight w, $c_j R^T$ can be computed in $O(wm \log n)$ steps (note that we do not read the entire matrix R). Therefore the computation of the Fourier transform and the inversion using the FFT algorithm will take $O(m(iw \log n + 2^m)) = \tilde{O}(i \cdot w \log n + 2^m)$ steps. ■

Note that a straightforward computation would take $\Omega(iw2^m \log n)$ steps. We apply Lemma 20 to speed up the evaluation of $\mathcal{D}^{\mathtt{Comb}}_{\ell,i}(x)$ on points at which `WeakDNF` asks non-adaptive MQs (here again we will rely on the non-adaptiveness of the weak learning algorithm). The speed-up is based on the following observations.

1. `WeakDNF` is based on estimating Fourier coefficients on a "small" number of linear subspaces of $\{0,1\}^n$ (as in Equation 2).

2. `WeakDNF` produces a short parity function (or its negation) as the hypothesis.

3. In computation of $\mathcal{D}^{\mathtt{Comb}}_{\ell,i}(x)$ the only information that is needed about the previous hypotheses is $N_{\mathtt{Filt}}(x)$ and $N_{\mathtt{F1}}(x)$, that is the number of hypotheses so far that are correct on the given point. The number of correct hypotheses is determined by $f(x)$ and the sum (in particular, a linear combination) of the values of the hypotheses on x.

Now we prove these observations formally and show a more efficient way to compute $\mathcal{D}^{\mathtt{Comb}}_{\ell,i}(x)$ given oracle access to $N_{\mathtt{Filt}}(x)$, in other words, we show a more efficient way to compute $N_{\mathtt{F1}}(x)$.

Lemma 21 *Let $\{b_1\chi_{c_1}, b_2\chi_{c_2}, \ldots, b_i\chi_{c_i}\}$ be the set of hypotheses returned by $\mathtt{WeakDNF}(s,B)$ in i first stages of `F1` boosting algorithm during stage ℓ of `B-Filt`, where $b_j \in \{-1,+1\}$ is the sign of χ_{c_j} (indicating whether or not it is negated). Let W be the set of queries for the $(i+1)$-th execution of $\mathtt{WeakDNF}(s,B)$ with confidence parameter δ and $B \ge L_\infty(2^n \mathcal{D}^{\mathtt{Comb}}_{\ell,i})$. Then, given MEM$(f)$ and an oracle access to $N_{\mathtt{Filt}}(x)$, the set of pairs $S = \{\langle x, \lambda \mathcal{D}^{\mathtt{Comb}}_{\ell,i}(x)\rangle \mid x \in W\}$ for some constant $\lambda \in [2/3, 4/3]$, can be computed, with probability at least $1-\delta$, in time $\tilde{O}((i+s^2B)\log^2 n \log(1/\delta))$.*

Proof We start by proving our first observation. By revisiting the proof of Theorem 13 we can see that our weak parity algorithm asks queries on $Y = \{pR \mid p \in \{0,1\}^m\}$ for a randomly chosen R and then for each query z of a ae.naMQ parity algorithm it asks queries on points of the set $Y_z = \{z \oplus y \mid y \in Y\}$. The set Y_z is a subset of the linear subspace of dimension $m+1$ spanned by the rows of R and vector y. These queries are then repeated $O(m + \log(1/\delta))$ times to single out "good" Fourier coefficients. Therefore by substituting the parameters of $\mathtt{WeakDNF}(s,B)$ into the proofs of Lemma 17 and Theorem 13, we can see that W can be decomposed into $\tilde{O}(\log^2(sB) \log n \log(1/\delta))$ linear subspaces of dimension $m = \log T$ for $T = \tilde{O}(s^2 B \log n)$.

Our second observation is given by Theorem 18 and states that for each $j \le i$, χ_{c_j} is a parity on at most $\log sB$ variables.

Our next observation is that the number of hypotheses from $\{b_1\chi_{c_1}, b_2\chi_{c_2}, \ldots, b_i\chi_{c_i}\}$ that agree with f on x equals to

$$N_{\mathtt{F1}}(x) = \frac{f(x)\left(\sum_{j \le i} b_j \chi_{c_j}(x)\right)}{2} + \frac{i}{2},$$

that is, given $\sum_{j\le i} b_j \chi_{c_j}(x)$ and $f(x)$, $N_{\mathtt{F1}}(x)$ can be computed in $O(1)$ steps. According to Lemma 20, we can compute $\sum_{j\le i} b_j \chi_{c_j}(x)$ on a linear subspace of dimension m in time $\tilde{O}(iw\log n + 2^m)$. Together with the first observation this implies that computing $N_{\mathtt{F1}}(x)$ for all points in W can be done in time

$$\tilde{O}(\log^2(sB)\log n\log(1/\delta))\tilde{O}(i\cdot\log(sB)\log n + s^2B\log n) = \tilde{O}((i+s^2B)\log^2 n\log(1/\delta))\ .$$

Equation (5) implies that for every point x, given $N_{\mathtt{F1}}(x)$, oracle access to $N_{\mathtt{Filt}}(x)$ and $\alpha_\ell\alpha_{\ell,i}$ we obtain $\mathcal{D}^{\mathtt{Comb}}_{\ell,i}(x)$. The normalization factor $\alpha_{\ell,i}$ is estimated with relative accuracy $1/3$ and therefore instead of the true $\mathcal{D}^{\mathtt{Comb}}_{\ell,i}(x)$ we will obtain $\lambda\mathcal{D}^{\mathtt{Comb}}_{\ell,i}(x)$ for some constant $\lambda\in[2/3,4/3]$. ■

Lemma 21 assumes oracle access to $N_{\mathtt{Filt}}(x)$. In the next lemma we show that this oracle can be simulated efficiently.

Lemma 22 *Let $\{h_0,h_1,\ldots,h_{\ell-1}\}$ be the set of hypotheses obtained by* `B-Comb` *in ℓ first stages of boosting. Let W be the set of queries for the $(i+1)$-th execution of* `WeakDNF`(s,B) *with confidence parameter δ and $B\ge L_\infty(2^n\mathcal{D}^{\mathtt{Comb}}_{\ell,i})$. Then, given MEM$(f)$, the set of pairs $S=\{\langle x,N_{\mathtt{Filt}}(x)\rangle \mid x\in W\}$ can be computed, with probability at least $1-\delta$, in time $\tilde{O}(\ell s^2B\cdot\log^2 n\log(1/\delta))$.*

Proof For each $j\le\ell-1$, h_j is an output of `F1` or a random coin flip hypothesis. `WeakDNF`(s,B) returns $(\frac{1}{2}-\Omega(\frac{1}{s}))$-approximate hypotheses and therefore each hypothesis generated by `F1` is a majority vote of $O(\gamma^{-}2)=O(s^2)$ short parities (or their negations). A majority vote of these parities and their negations is simply the sign of their sum, and in particular is determined by a linear combination of parity functions. Hence, as in Lemma 21, $h_j(x)$ for all points in W can be computed $\tilde{O}((s^2+s^2B)\log^2 n\log(1/\delta))$ time. Therefore for any stage ℓ, $h_0,h_1,\ldots,h_{\ell-1}$ can be computed on points in W in $\tilde{O}(\ell s^2B\log^2 n\log(1/\delta))$ steps giving the required oracle $N_{\mathtt{Filt}}(x)$. ■

Remark 23 *In this simulation of* `B-Comb` *we ignored the complexity of procedure* `EstExpRel` *that is used to evaluate the normalization factors. The factor $\alpha_\ell=\mathbf{E}[\beta(\ell,N_{\mathtt{Filt}}(x))]$ needs to be estimated within relative accuracy $1/3$ and its value is only used when the estimate $\alpha'_\ell\ge\Theta=c_1\varepsilon/\log(1/\varepsilon)$ for some constant c_1 since otherwise* `B-Comb` *uses a random coin flip hypothesis (see line 7 of the pseudocode). This implies that the estimate is only used when $\alpha_\ell\ge 3\Theta/4$. The Chernoff bound (Lemma 1) implies that if $\alpha_\ell\ge 3\Theta/4$ then using $M=O(\frac{log(1/\varepsilon)\log(1/\delta)}{\varepsilon})$ random uniform samples will be sufficient to estimate α_ℓ within relative accuracy $1/3$ with confidence $1-\delta$. If $\alpha_\ell<3\Theta/4$ then with probability $1-\delta$ the obtained estimate α'_ℓ will be less than Θ and therefore will not be used. Evaluating $N_{\mathtt{Filt}}(x)$ on each of these points will take $O(\ell ns^2)$ steps and therefore each of these estimation will run in time $\tilde{O}(\ell ns^2\log(1/\delta)/\varepsilon)$.*

At each stage of the `F1` *boosting algorithm we need to estimate*

$$\alpha_{\ell,i}=\mathbf{E}[\beta(\ell,N_{\mathtt{Filt}}(x))\cdot\beta(i,N_{\mathtt{F1}}(x))/\alpha'_\ell]$$

to within relative accuracy $1/3$ and its value is only used when the estimate $\alpha'_{\ell,i}\ge c$ for some constant c. Therefore it is sufficient to estimate $\alpha_{\ell,i}$ to within constant additive accuracy. With probability at least $1-\delta$ this can be achieved by using a sample of $O(\log(1/\delta))$ random uniform points. Estimating both $N_{\mathtt{Filt}}(x)$ and $N_{\mathtt{F1}}(x)$ on each point takes $O(\ell ns^2)$ steps and therefore each of these estimations runs in time $O(\ell ns^2\log(1/\delta))$.

We are now ready to describe the resulting ae.naMQ algorithm for learning DNF expressions.

Theorem 24 *There exists an algorithm* $\texttt{AENALearnDNF}(s)$ *that for any Boolean function* f *of DNF-size* s*, given* n, s, ε*, and access to MEM*(f)*, with probability at least* $1/2$*, finds an* ε*-approximator to* f *with respect to* $\mathcal{U}$*. Furthermore,* $\texttt{AENALearnDNF}(s)$ *runs in time* $\tilde{O}\left(ns^4/\varepsilon\right)$ *and asks* $\tilde{O}\left(s^4 \log^2 n/\varepsilon\right)$ *non-adaptive MQs.*

Proof As we know from the description of `B-Filt`, it has $O(\log(1/\varepsilon))$ stages and for each ℓ and i, $L_\infty(2^n \mathcal{D}^{\texttt{Comb}}_{\ell,i}) = \tilde{O}(1/\varepsilon)$. Therefore the running time of each execution of $\texttt{WeakDNF}(s, B)$ is $\tilde{O}(ns^2/\varepsilon)$. In particular, for every boosting stage of `F1`, it dominates the running time of computing the distribution function $\mathcal{D}^{\texttt{Comb}}_{\ell,i}$ (Lemmas 21 and 22) and estimations of α_ℓ and $\alpha_{\ell,i}$ (Remark 23). There are total $O(s^2 \log(1/\varepsilon))$ executions of `WeakDNF` and therefore the total running time of $\texttt{AENALearnDNF}(s)$ is $\tilde{O}\left(ns^4/\varepsilon\right)$ and the total number of non-adaptive MQs used is $\tilde{O}\left(s^4 \log^2 n/\varepsilon\right)$. ■

The improvements to the algorithm by Bshouty et al. (1999) are summarized below.

- The use of attribute-efficient weak learning improves the total sample complexity from $\tilde{O}\left(ns^4/\varepsilon^2\right)$ to $\tilde{O}\left(s^4 \log^2 n/\varepsilon^2\right)$ and the same running time is achieved without assumptions on the MQ oracle (see Theorem 16).
- Faster computation of distribution functions used in boosting improves the total running time from $\tilde{O}\left(ns^6/\varepsilon^2\right)$ to $\tilde{O}\left(ns^4/\varepsilon^2\right)$ (see Lemmas 20, 21 and 22).
- Tighter estimation of variance improves the dependence of running time and sample complexity on ε from $1/\varepsilon^2$ to $1/\varepsilon$ (Equation 4).

Remark 25 *While the analysis of the speedup was done for Freund's* `B-Comb` *booster the same idea works for any other booster in which estimation of new weight function is based on a linear combination of previous hypotheses. In particular, for the other known boosting algorithms that produce smooth distributions:* `SmoothBoost` *by Servedio (2003) and* `AdaFlat` *by Gavinsky (2003).*

7. Handling Noise

Now we would like to show that our DNF learning algorithm can be modified to tolerate random persistent classification noise in MQs. To simplify the proof we first show that we can assume that we are dealing with random and independent classification noise.

Lemma 26 *The probability that* $\texttt{AENALearnDNF}(s)$ *asks an MQ for the same point more than once is upper bounded by* $P \cdot 2^{-n/\log Q}$ *where* P *and* Q *are polynomial in* n, s *and* $1/\varepsilon$.

Proof We start by observing that in the algorithm $\texttt{AENALearnDNF}(s)$ all the points that are given to the MQ oracle are chosen uniformly and the points that are used in different executions of `WeakDNF` are independent. As can be seen from the proof of Theorem 13, the generated points are of the form $pR \oplus y$, where R is a randomly and uniformly chosen matrix, y is chosen randomly according to $\mathcal{D}_{\frac{1}{4k}}$ (defined in Theorem 9) or equal to 0^n, and $p \in \{0,1\}^m$. Points generated for two randomly chosen R_1 and R_2 are independent of each other and uniformly distributed. Let $y_0 = 0^n$, q be the number of samples taken from $\mathcal{D}_{\frac{1}{4k}}$, and $y_1, y_2, \ldots, y_q$ denote the samples.

For some randomly chosen R, let $x_1 = p_1 R \oplus y_i$ and $x_2 = p_2 R \oplus y_j$ be two different sample points. For two different sample points either $i \neq j$ or $p_1 \neq p_2$. If $i \neq j$ then either $i \neq 0$ or $j \neq 0$. Without loss of generality we assume that $i \neq 0$. Then

$$\mathbf{Pr}_{y_i \sim \mathcal{D}_{\frac{1}{4k}}}[p_1 R \oplus y_i = p_2 R \oplus y_j] = \mathbf{Pr}_{y_i \sim \mathcal{D}_{\frac{1}{4k}}}[y_i = p_1 R \oplus p_2 R \oplus y_j] \leq (1 - \frac{1}{4k})^n \leq e^{-n/(4k)} .$$

If $p_1 \neq p_2$ then $\mathbf{Pr}_{R \sim \mathcal{U}_{m \times n}}[(p_1 \oplus p_2)R = y_i \oplus y_j] = 2^{-n}$. This implies that for any two MQs made by $\texttt{AENALearnDNF}(s)$, probability that they are equal is at most $e^{-n/(4k)}$. As it can be seen from the analysis of $\texttt{AENALearnDNF}(s)$, $k = O(\log(s/\varepsilon))$ and the total number of MQs used is polynomial in n, s and $1/\varepsilon$. ■

If an algorithm does not ask a MQ for the same point again then persistent classification noise can be treated as random and independent.

7.1 Boosting Weak Parity Learning Algorithm in the Presence of Noise

The main part of the modification is to show an algorithm that can locate heavy Fourier coefficients of any randomized function can be used to learn DNFs in the presence of noise. Our method can be applied in more general setting. In particular, it could be used to prove that Jackson's original algorithm is resistant to persistent noise in MQs and was recently used to produce a noise tolerant DNF learning algorithm by Feldman et al. (2006). Previous methods to produce noise-tolerant DNF learning algorithms gave statistical query analogues of Jackson's algorithm and then simulated statistical queries[5] in the presence of noise (Jackson et al., 1997; Bshouty and Feldman, 2002). Our approach is more direct and the resulting algorithm is substantially more efficient than the previous ones.

The goal of a weak DNF learning algorithm at stage i of boosting is to find a parity correlated with the function $2^n \mathcal{D}_i(x) f(x)$ given an oracle access to values of $\mathcal{D}_i(x)$ and the oracle for f with noise of rate $\eta < 1/2$ instead of $\mathrm{MEM}(f)$. Handling the noisy case is further complicated by the fact that the computation of $\mathcal{D}_i(x)$ by the boosting algorithm uses the value $f(x)$ (in particular, `B-Comb` and `B-Filt` need the value of $f(x)$ to compute $N(x)$) which is not available in the noisy case. To make this dependence explicit we define $\mathcal{D}_i(x,b)$ (for $b \in \{-1,+1\}$) to be the value of $\mathcal{D}_i$ on x when the boosting algorithm is supplied with the value b in place of $f(x)$ to compute $D_i(x)$ (in particular, $\mathcal{D}_i(x) = \mathcal{D}_i(x, f(x))$). We will now show a general method to compute a Fourier coefficient of a function that depends on $f(x)$ given a noisy oracle for f.

Lemma 27 *Let $g(x,b)$ be any real-valued function over $\{0,1\}^n \times \{-1,+1\}$ and let Φ^η denote a randomized function such that for every x, $\Phi^\eta(x) = f(x)$ with probability $1-\eta$ and $\Phi^\eta(x) = -f(x)$ with probability η. Then for each $a \in \{0,1\}^n$, $[\widehat{g(x, f(x))}](a) = \widehat{\Psi_{g,\eta}}(a)$, where $\Psi_{g,\eta}$ is a randomized function defined as*

$$\Psi_{g,\eta}(x) = \frac{1}{2}\left(\frac{1}{1-2\eta}(g(x,1) - g(x,-1)) \cdot \Phi^\eta(x) + g(x,1) + g(x,-1)\right) .$$

5. They used stronger versions of statistical queries than those introduced by Kearns (1998).

Proof We use the following observation due to Bshouty and Feldman (2002). For any real-valued function $\psi(x,b)$

$$\psi(x,f(x)) = \psi(x,-1)\frac{1-f(x)}{2} + \psi(x,1)\frac{1+f(x)}{2} =$$
$$\frac{1}{2}((\psi(x,1)-\psi(x,-1))f(x) + \psi(x,1) + \psi(x,-1)) .$$

Then

$$\mathbf{E}_{x,\Phi^\eta(x)}[\frac{1}{2}(\psi(x,1)-\psi(x,-1))\cdot\Phi^\eta(x)] = (1-2\eta)\mathbf{E}_x[\frac{1}{2}(\psi(x,1)-\psi(x,-1))f(x)] ,$$

and therefore we can offset the effect of noise in $g(x,f(x))$ as follows.

$$\widehat{[g(x,f(x))]}(a) = \mathbf{E}[g(x,f(x))\chi_a(x)]$$
$$= \frac{1}{2}\left(\mathbf{E}_x[(g(x,1)-g(x,-1))\chi_a(x)f(x)] + \mathbf{E}_x[(g(x,1)+g(x,-1))\chi_a(x)]\right)$$
$$= \frac{1}{2}\left(\frac{1}{1-2\eta}\mathbf{E}_{x,\Phi^\eta(x)}[(g(x,1)-g(x,-1))\chi_a(x)\cdot\Phi^\eta(x)] + \mathbf{E}_x[(g(x,1)+g(x,-1))\chi_a(x)]\right)$$
$$= \mathbf{E}_{x,\Phi^\eta(x)}\left[\frac{1}{2}\left(\frac{1}{1-2\eta}(g(x,1)-g(x,-1))\cdot\Phi^\eta(x) + g(x,1) + g(x,-1)\right)\chi_a(x)\right] = \widehat{\Psi}(a)$$

■

An oracle for $\Phi^\eta(x)$ is exactly the membership query oracle for $f(x)$ with noise of rate η that is given to us (by Lemma 26 we can ignore the persistency of noise). Therefore Lemma 27 gives a way to find heavy Fourier coefficients using an oracle for $\Phi^\eta(x)$ instead of the membership query oracle for $f(x)$. We apply it to `WeakDNF` and obtain our noise-tolerant ae.naMQ DNF learning algorithm.

Theorem 28 *There exists an algorithm* $\texttt{AENALearnDNF}(s,\eta)$ *that for any Boolean function f of DNF-size s, given n,s,η,ε, and access to MEM(f) corrupted by random persistent classification noise of rate η, with probability at least $1/2$, finds an ε-approximator to f with respect to $\mathcal{U}$. Furthermore,* $\texttt{AENALearnDNF}(s,\eta)$ *runs in time* $\tilde{O}\left(ns^4/(\varepsilon(1-2\eta)^2)\right)$ *and asks* $\tilde{O}\left(s^4\log^2 n/(\varepsilon(1-2\eta)^2)\right)$ *non-adaptive MQs.*

Proof Section 6.3 gives a way to efficiently compute $\mathcal{D}^{\texttt{Comb}}_{\ell,i}(x)$ given the label $f(x)$. This computation defines the oracle for $\mathcal{D}^{\texttt{Comb}}_{\ell,i}(x,b)$ where b is the supposed label of $f(x)$. Let $g(x,b) = b\cdot 2^n\mathcal{D}^{\texttt{Comb}}_{\ell,i}(x,b)$ and let $\Psi_{g,\eta}(x)$ be defined as in Lemma 27. Given the oracle for $\mathcal{D}^{\texttt{Comb}}_{\ell,i}(x,b)$ and oracle access to $\Phi^\eta(x)$ we use $\texttt{AEBoundedSieveRV}(\theta,\mathtt{k},\mathtt{V})$ on $\Psi_{g,\eta}(x)$ in the same way it was used on $\psi(x)$ by $\texttt{WeakDNF}(s,B)$ (see the proof of Theorem 18). By Lemma 27, $\Psi_{g,\eta}(x)$ has the same Fourier coefficients as $f(x)2^n\mathcal{D}^{\texttt{Comb}}_{\ell,i}(x,f(x))$. Therefore this modified weak learning algorithm will produce an equivalent hypothesis. We can deal with the noise while estimating the normalization factor $\alpha_{\ell,i}$ in exactly the same way.

Furthermore, the definition of $\Psi_{g,\eta}$ and Equation (4) imply that

$$L_\infty(\Psi_{g,\eta}) \le \frac{2}{1-2\eta}L_\infty(2^n\mathcal{D}^{\texttt{Comb}}_{\ell,i}) \text{ and } \mathbf{Var}(\Psi_{g,\eta}) \le \frac{4}{(1-2\eta)^2}L_\infty(2^n\mathcal{D}^{\texttt{Comb}}_{\ell,i}) .$$

By substituting these bounds into Theorem 18 we obtain that the running time and the sample complexity of each execution of the modified weak learner will grow by $(1-2\eta)^2$. They also imply that $\texttt{WeakDNF}(s,B)$ will produce parities on $\log(s^2/(\varepsilon(1-2\eta)))$ variables (this change is absorbed by $\tilde{O}$ notation). ■

8. Conclusions and Open Problems

In this work we have demonstrated equivalence of attribute-efficient learning of parities from random and uniform examples and decoding of random linear binary codes. This result appears to be the only known evidence of hardness of attribute-efficient learning for a natural concept class. Many other problems remain open in this area. For example it is unknown whether decision lists or linear thresholds are learnable attribute-efficiently.

Our results show that some of the most important concepts classes that are learnable attribute efficiently with respect to the unform distribution using membership queries are also learnable by significantly weaker non-adaptive MQs. We believe that it is interesting to understand if similar results can be obtained in the distribution-independent setting. In particular whether monotone DNF formulae and decision trees can be learned attribute-efficiently using non-adaptive MQs in the distribution-independent PAC model.

We have also shown an improved algorithm for learning DNF expressions with respect to the uniform distribution. In addition to being the most efficient known algorithm for learning DNF, it is attribute-efficient, noise tolerant, and uses membership queries non-adaptively. All known efficient algorithms for learning DNF are based on Jackson's (1997) approach to learning DNF expressions. It would be interesting to find other approaches to learning DNF, possibly avoiding some of the overheads of the current approach (such as boosting a weak DNF learning algorithm).

Acknowledgments

We thank Leslie Valiant for his advice and encouragement of this research. We are grateful to Jeffrey Jackson for discussions and clarifications on the DNF learning algorithm of Bshouty et al.. We also thank Alex Healy, Dmitry Gavinsky, and anonymous COLT and JMLR reviewers for valuable and insightful comments.

References

A. Aho, J. Hopcroft, and J. Ullman. *The Design and Analysis of Computer Algorithms*. Addison-Wesley Series in Computer Science and Information Processing. Addison-Wesley, 1974.

D. Angluin and P. Laird. Learning from noisy examples. *Machine Learning*, 2:343–370, 1988.

A. Barg. Complexity issues in coding theory. *Electronic Colloquium on Computational Complexity (ECCC)*, 4(046), 1997.

A. Blum, L. Hellerstein, and N. Littlestone. Learning in the presence of finitely or infinitely many irrelevant attributes. *Journal of Computer and System Sciences*, 50:32–40, 1995.

A. Blum, A. Kalai, and H. Wasserman. Noise-tolerant learning, the parity problem, and the statistical query model. In *Proceedings of STOC*, pages 435–440, 2000.

N. Bshouty and V. Feldman. On using extended statistical queries to avoid membership queries. *Journal of Machince Learning Research*, 2:359–395, 2002.

N. Bshouty and L. Hellerstein. Attribute efficient learning with queries. *Journal of Computer and System Sciences*, 56:310–319, 1998.

N. Bshouty, J. Jackson, and C. Tamon. More efficient PAC learning of DNF with membership queries under the uniform distribution. In *Proceedings of COLT*, pages 286–295, 1999.

N. Bshouty, E. Mossel, R. O'Donnell, and R. Servedio. Learning DNF from random walks. In *Proceedings of FOCS*, pages 189–199, 2003.

H. Chernoff. A measure of asymptotic efficiency for tests of a hypothesis based on the sum of observations. *Ann. Math. Statist.*, 23:493–507, 1952.

J. Cooley and J. Tukey. An Algorithm for the Machine Calculation of Complex Fourier Series. *Math. Computat.*, 19:297–301, 1965.

P. Damaschke. Adaptive versus nonadaptive attribute-efficient learning. In *Proceedings of STOC*, pages 590–596, 1998.

S. Decatur, O. Goldreich, and D. Ron. Computational sample complexity. *SIAM Journal on Computing*, 29(3):854–879, 1999.

M. Farach, S. Kannan, E. Knill, and S. Muthukrishnan. Group testing problems in experimental molecular biology. In *Proceedings of Sequences '97*, 1997.

V. Feldman, P. Gopalan, S. Khot, and A. Ponuswami. New Results for Learning Noisy Parities and Halfspaces. In *Proceedings of FOCS*, pages 563–574, 2006.

Y. Freund. Boosting a weak learning algorithm by majority. In *Proceedings of the Third Annual Workshop on Computational Learning Theory*, pages 202–216, 1990.

Y. Freund. An improved boosting algorithm and its implications on learning complexity. In *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, pages 391–398, 1992.

D. Gavinsky. Optimally-smooth adaptive boosting and application to agnostic learning. *Journal of Machine Learning Research*, 4:101–117, 2003.

S. Goldman, M. Kearns, and R. Schapire. Exact identification of read-once formulas using fixed points of amplification functions. *SIAM Journal on Computing*, 22(4):705–726, 1993.

O. Goldreich and L. Levin. A hard-core predicate for all one-way functions. In *Proceedings of STOC*, pages 25–32, 1989.

O. Goldreich, S. Goldwasser, and S. Micali. How to construct random functions. *Journal of the ACM*, 33(4):792–807, 1986.

D. Guijarro, V. Lavin, and V. Raghavan. Exact learning when irrelevant variables abound. In *Proceedings of EuroCOLT '99*, pages 91–100, 1999a.

D. Guijarro, J. Tarui, and T. Tsukiji. Finding relevant variables in PAC model with membership queries. *Lecture Notes in Artificial Intelligence*, 1720:313 – 322, 1999b.

D. Haussler. Decision theoretic generalizations of the PAC model for neural net and other learning applications. *Information and Computation*, 100(1):78–150, 1992.

T. Hofmeister. An application of codes to attribute-efficient learning. In *Proceedings of EuroCOLT*, pages 101–110, 1999.

J. Jackson. Personal communication, 2004.

J. Jackson. An efficient membership-query algorithm for learning DNF with respect to the uniform distribution. *Journal of Computer and System Sciences*, 55:414–440, 1997.

J. Jackson, E. Shamir, and C. Shwartzman. Learning with queries corrupted by classification noise. In *Proceedings of the Fifth Israel Symposium on the Theory of Computing Systems*, pages 45–53, 1997.

M. Kearns. Efficient noise-tolerant learning from statistical queries. *Journal of the ACM*, 45(6): 983–1006, 1998.

M. Kearns and L. Valiant. Cryptographic limitations on learning boolean formulae and finite automata. *Journal of the ACM*, 41(1):67–95, 1994.

M. Kearns and U. Vazirani. *An introduction to computational learning theory*. MIT Press, Cambridge, MA, 1994.

M. Kearns, R. Schapire, and L. Sellie. Toward efficient agnostic learning. *Machine Learning*, 17 (2-3):115–141, 1994.

A. Klivans and R. Servedio. Boosting and hard-core set construction. *Machine Learning*, 51(3): 217–238, 2003.

A. Klivans and R. Servedio. Toward attribute efficient learning of decision lists and parities. In *Proceedings of COLT*, pages 234–248, 2004.

E. Kushilevitz and Y. Mansour. Learning decision trees using the Fourier spectrum. In *Proceedings of STOC*, pages 455–464, 1991.

L. Levin. Randomness and non-determinism. *Journal of Symbolic Logic*, 58(3):1102–1103, 1993.

Y. Mansour. Learning boolean functions via the fourier transform. In V. P. Roychodhury, K. Y. Siu, and A. Orlitsky, editors, *Theoretical Advances in Neural Computation and Learning*, pages 391–424. Kluwer, 1994.

J. Massey. Shift-register synthesis and BCH decoding. *IEEE Trans. Inform. Theory*, 15:122–127, 1969.

R. J. McEliece. A public-key cryptosystem based on algebraic coding theory. *DSN progress report*, 42-44, 1978.

R. Motwani and P. Raghavan. *Randomized Algorithms*. Cambridge University Press, 1995.

R. Schapire. The strength of weak learnability. *Machine Learning*, 5(2):197–227, 1990.

R. Servedio. Computational sample complexity and attribute-efficient learning. *Journal of Computer and System Sciences*, 60(1):161–178, 2000.

R. Servedio. Smooth boosting and learning with malicious noise. *Journal of Machine Learning Research*, 4:633–648, 2003.

M. Sudan. Essential coding theory (lecture notes). Available at `http://theory.lcs.mit.edu/~madhu/FT02/`, 2002.

R. Uehara, K. Tsuchida, and I. Wegener. Optimal attribute-efficient learning of disjunction, parity, and threshold functions. In *Proceedings of EuroCOLT '97*, pages 171–184, 1997.

L. Valiant. A neuroidal architecture for cognitive computation. *Journal of the ACM*, 47(5):854–882, 2000.

L. Valiant. A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142, 1984.

L. Valiant. *Circuits of the Mind*. Oxford University Press, 1994.

L. G. Valiant. Knowledge infusion. In *Proceedings of AAAI*, 2006.

J.H. van Lint. *Introduction to Coding Theory*. Springer, Berlin, 1998.

A. Vardy. Algorithmic complexity in coding theory and the minimum distance problem. In *Proceedings of STOC*, pages 92–109, 1997.

Journal of Machine Learning Research 8 (2007) 1461-1487 Submitted 7/06; Revised 4/07; Published 7/07

PAC-Bayes Risk Bounds for Stochastic Averages and Majority Votes of Sample-Compressed Classifiers

François Laviolette FRANCOIS.LAVIOLETTE@IFT.ULAVAL.CA
Mario Marchand MARIO.MARCHAND@IFT.ULAVAL.CA
Département IFT-GLO
Université Laval
Québec (QC)
Canada, G1K 7P4

Editor: Manfred K. Warmuth

Abstract

We propose a PAC-Bayes theorem for the sample-compression setting where each classifier is described by a compression subset of the training data and a message string of additional information. This setting, which is the appropriate one to describe many learning algorithms, strictly generalizes the usual data-independent setting where classifiers are represented only by data-independent message strings (or parameters taken from a continuous set). The proposed PAC-Bayes theorem for the sample-compression setting reduces to the PAC-Bayes theorem of Seeger (2002) and Langford (2005) when the compression subset of each classifier vanishes. For posteriors having all their weights on a single sample-compressed classifier, the general risk bound reduces to a bound similar to the tight sample-compression bound proposed in Laviolette et al. (2005). Finally, we extend our results to the case where each sample-compressed classifier of a data-dependent ensemble may abstain of predicting a class label.

Keywords: PAC-Bayes, risk bounds, sample-compression, set covering machines, decision list machines

1. Introduction

The PAC-Bayes approach, initiated by McAllester (1999), aims at providing PAC guarantees to "Bayesian-like" learning algorithms. These algorithms are specified in terms of a *prior distribution* P over a space of classifiers that characterizes our prior belief about good classifiers (before the observation of the data) and a *posterior distribution* Q (over the same space of classifiers) that takes into account the additional information provided by the training data. A remarkable result that came out from this line of research, known as the "PAC-Bayes theorem", provides a tight upper bound on the risk of a stochastic classifier (defined on the posterior Q) called the *Gibbs classifier*.

This PAC-Bayes bound (see Theorem 1) depends both on the empirical risk (i.e., training errors) of the Gibbs classifier and on "how far" is the data-dependent posterior Q from the data-independent prior P. Consequently, a Gibbs classifier with a posterior Q having all its weight on a single classifier will have a larger risk bound than another Gibbs classifier, making the same amount of training errors, using a "broader" posterior Q that gives weight to many classifiers. Hence, the PAC-Bayes theorem quantifies the additional predictive power that stochastic classifier selection might have over deterministic classifier selection.

A constraint normally imposed by the PAC-Bayes theorem is that the prior P must be defined without reference to the training data. Consequently, we cannot directly use the PAC-Bayes theorem to bound the risk of sample-compression learning algorithms (Littlestone and Warmuth, 1986, Floyd and Warmuth, 1995) because the set of classifiers considered by these algorithms are those that can be reconstructed from various subsets of the training data. However, this is an important class of learning algorithms since many well known learning algorithms, such as the support vector machine (SVM) and the perceptron learning rule, can be considered as sample-compression learning algorithms (Graepel et al., 2005). Moreover, some sample-compression algorithms (Marchand and Shawe-Taylor, 2002, Marchand and Sokolova, 2005) have achieved very good performance in practice by deterministically choosing a sparse classifier making few training errors. It is therefore worthwhile to investigate how the stochastic selection of sample-compressed classifiers provides an additional predictive power over the deterministic selection of a single sample-compressed classifier.

In this paper, we extend the PAC-Bayes theorem in such a way that it applies now to both the usual data-independent setting and the more general sample-compression setting. In the sample-compression setting, each classifier is represented by two independent sources of information: a *compression set* which consists of a small subset of the training data, and a *message string* of the additional information needed to obtain a classifier. In the limit where the compression set vanishes, each classifier is identified only by a message string and the new PAC-Bayes theorem reduces to the "usual" PAC-Bayes theorem of Seeger (2002) and Langford (2005). However, new quantities appear in the risk bound when classifiers are also described by their compression sets. As in the case for the usual data-independent setting, the PAC-Bayes theorem for the sample-compression setting states that a stochastic Gibbs classifier defined on a posterior over several sample-compressed classifiers generally has a smaller risk bound than any such single (deterministic) sample-compressed classifier. Nevertheless, in the limit where the posterior Q puts all its weight on a single sample-compressed classifier, the new PAC-Bayes risk bound reduces to a bound similar to the tight sample-compression bound of Laviolette et al. (2005) (which applies only to single sample-compressed classifiers).

Several "PAC-Bayesian sample-compression bounds" have recently been proposed by Graepel et al. (2005). However, all these bounds, except one (that concerns consistent SVM classifiers with fixed sparsity), deals with classifiers that use a fixed subset of the training examples. In contrast, we provide bounds that applies to a stochastic average (and a majority vote) of classifiers using different subsets (of different sizes) of the training examples. Finally, we extend our results to the important case where we have an ensemble of sample-compressed classifiers that can abstain of predicting a class label.

The paper is organized as follows. After providing a few definitions in Section 2, we review, in Section 3, the PAC-Bayes theorem for the data-independent setting. Section 4 is the "core" section of this paper. In that section, we present the sample-compression setting and show how it generalizes the usual data-independent setting. We then provide the main theorem of this paper, Theorem 3, which is a PAC-Bayes theorem for the sample-compression setting. In Section 5, we provide examples of learning algorithms that produce classifiers that are well-described within this sample-compression setting. We then show, in Section 6, that Theorem 3 reduces to a bound similar to the tight sample-compression bound of Laviolette et al. (2005) in the limit where the posterior Q puts all its weight on a single sample-compressed classifier. In that section, we also present a bound for the "intermediate" case where the posterior has all its weight on a single compression sequence

and non-zero weight on several messages. We then show that the risk bound reduces to the one recently proposed in Laviolette et al. (2006) for the PAC-Bayes SCM. In Section 7, we provide an alternative formulation of Theorem 3 by including the training errors into the compression sequence. We then generalize, in Section 8, Theorem 3 to the case were the individual sample-compressed classifiers may abstain of predicting a class label. Finally, we conclude in Section 9.

This paper extends the preliminary work of Laviolette and Marchand (2005) and Laviolette et al. (2006).

2. Basic Definitions

We consider binary classification problems where the input space $\mathcal{X}$ consists of an arbitrary subset of $\mathbb{R}^n$ and the output space $\mathcal{Y} = \{-1, +1\}$. An example $(\mathbf{x}, y)$ is an input-output pair where $\mathbf{x} \in \mathcal{X}$ and $y \in \mathcal{Y}$.

Throughout the paper, we adopt the PAC setting where each example $(\mathbf{x}, y)$ is drawn according to a fixed, but unknown, probability distribution D on $\mathcal{X} \times \mathcal{Y}$. The risk $R(f)$ of any classifier f is defined as the probability that it misclassifies an example drawn according to D. Hence,

$$R(f) \stackrel{\text{def}}{=} \Pr_{(\mathbf{x},y)\sim D}\Big(f(\mathbf{x}) \neq y\Big) = \mathop{\mathbf{E}}_{(\mathbf{x},y)\sim D} I(f(\mathbf{x}) \neq y)\,,$$

where $I(a) = 1$ if predicate a is true and 0 otherwise.

Given a training sequence $S = \langle(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)\rangle$ of m examples, the *empirical risk* $R_S(f)$ on S, of any classifier f, is defined according to

$$R_S(f) \stackrel{\text{def}}{=} \frac{1}{m}\sum_{i=1}^{m} I(f(\mathbf{x}_i) \neq y_i) \stackrel{\text{def}}{=} \mathop{\mathbf{E}}_{(\mathbf{x},y)\sim S} I(f(\mathbf{x}) \neq y)\,.$$

In this paper, we will distinguish the usual data-independent setting from the (more general) sample-compression setting. By the *data-independent setting*, we mean the "usual" setting where a space $\mathcal{H}$ of classifiers is defined without making any reference to the training data S. Examples of such a space $\mathcal{H}$ include the set of linear classifiers on $\mathbb{R}^n$, the set of radial-basis functions on $\mathbb{R}^n$, the set of k-CNF Boolean formulae (Valiant, 1984) on $\{0,1\}^n$, the set of decision lists (Rivest, 1987) on $\{0,1\}^n$. In contrast, the set of data-dependent balls (Marchand and Shawe-Taylor, 2002)—where each ball of this set is centered on a training example—is an example of a set of classifiers which is defined only after observing the training data S. Such a set of classifiers is qualified as being *data-dependent*. Moreover, since each data-dependent ball is constructed from a small subset of the training data S, it is an example of what we call a *sample-compressed classifier*. We define more formally the sample-compression setting in section 4 in such a way that it extends the usual data-independent setting. The next section presents the PAC-Bayes theorem within the (restricted) data-independent setting.

3. The PAC-Bayes Theorem in the Data-Independent Setting

The PAC-Bayes theorem provides tight upper and lower bounds on the risk of a stochastic classifier called the *Gibbs classifier*. Given an input example $\mathbf{x}$, the label assigned to $\mathbf{x}$ by the Gibbs classifier G_Q is defined by the following process. We first choose randomly a classifier h according to the

posterior distribution Q and then use h to assign the label to $\mathbf{x}$. The risk of G_Q is defined as the expected risk of classifiers drawn according to Q. Hence,

$$R(G_Q) \stackrel{\text{def}}{=} \mathop{\mathbf{E}}_{h\sim Q} R(h) = \mathop{\mathbf{E}}_{h\sim Q} \mathop{\mathbf{E}}_{(\mathbf{x},y)\sim D} I(h(\mathbf{x}) \neq y)\,.$$

Similarly, the empirical risk $R_S(G_Q)$ of G_Q, on a training sequence S of m examples, is given by

$$R_S(G_Q) \stackrel{\text{def}}{=} \mathop{\mathbf{E}}_{h\sim Q} R_S(h) = \mathop{\mathbf{E}}_{h\sim Q} \frac{1}{m}\sum_{i=1}^{m} I(h(\mathbf{x}_i) \neq y_i)\,.$$

The PAC-Bayes theorem was first proposed by McAllester (1999, 2003a). The version presented here is due to Seeger (2002) and Langford (2005).

Theorem 1 *Given any space $\mathcal{H}$ of classifiers. For any data-independent prior distribution P over $\mathcal{H}$ and any $\delta \in (0,1]$, we have*

$$\Pr_{S\sim D^m}\left(\forall Q \text{ on } \mathcal{H}:\ \mathrm{kl}(R_S(G_Q)\|R(G_Q)) \ \leq\ \frac{1}{m}\left[\mathrm{KL}(Q\|P) + \ln\frac{m+1}{\delta}\right]\right) \ \geq\ 1-\delta\,,$$

where $\mathrm{KL}(Q\|P)$ *is the Kullback-Leibler divergence between distributions Q and P:*

$$\mathrm{KL}(Q\|P) \stackrel{\text{def}}{=} \mathop{\mathbf{E}}_{h\sim Q} \ln\frac{Q(h)}{P(h)}\,,$$

and where $\mathrm{kl}(q\|p)$ *is the Kullback-Leibler divergence between the Bernoulli distributions with probability of success q and probability of success p:*

$$\mathrm{kl}(q\|p) \stackrel{\text{def}}{=} q\ln\frac{q}{p} + (1-q)\ln\frac{1-q}{1-p}\,.$$

It is rarely mentioned that this theorem provides both an upper bound and a lower bound on the true risk $R(G_Q)$ based on its empirical risk $R_S(G_Q)$. With probability at least $1-\delta$ over the random draws of S, $R(G_Q)$ is upper-bounded by

$$\sup\left(R\colon \mathrm{kl}(R_S(G_Q)\|R) \ \leq\ \frac{1}{m}\left[\mathrm{KL}(Q\,\|P) + \ln\frac{m+1}{\delta}\right]\right)$$

and lower-bounded by

$$\inf\left(R\colon \mathrm{kl}(R_S(G_Q)\|R) \ \leq\ \frac{1}{m}\left[\mathrm{KL}(Q\,\|P) + \ln\frac{m+1}{\delta}\right]\right).$$

The bounds provided by Theorem 1 hold for any *fixed* prior P on $\mathcal{H}$ and also hold *uniformly* for all posteriors Q on $\mathcal{H}$; this includes any Q chosen by the learner *after* observing S. This is specified in the theorem by the fact that the quantifier $\forall Q$ occurs inside the probability over the random draws of S whereas the quantifier $\forall P$ occurs (textually) outside that probability.

The upper bound given by the PAC-Bayes theorem for the risk of Gibbs classifiers can be turned into an upper bound for the risk of majority-vote classifiers (often called Bayes classifiers) in the following way. Given a posterior distribution Q, the Bayes classifier B_Q performs a majority vote

(under measure Q) of binary classifiers in $\mathcal{H}$. Then B_Q misclassifies an example $\mathbf{x}$ iff at least half of the binary classifiers (under measure Q) misclassifies $\mathbf{x}$. It follows that the error rate of G_Q is at least half of the error rate of B_Q. Hence $R(B_Q) \leq 2R(G_Q)$. It has been shown (Langford and Shawe-Taylor, 2003, McAllester, 2003b, Germain et al., 2007, Lacasse et al., 2007) that there exists circumstances where this "factor-of-two" rule can be improved. However, for many [1] posteriors Q, one can often find a data-generating distribution where we have $R(B_Q) = 2R(G_Q) - \varepsilon$ for arbitrary small $\varepsilon > 0$.

Finally, for certain distributions Q, a bound for $R(B_Q)$ can be turned into a bound for the risk of a single classifier whenever there exists $h^* \in \mathcal{H}$ such that $h^*(\mathbf{x}) = B_Q(\mathbf{x})\ \forall \mathbf{x} \in \mathcal{X}$. Such a classifier h^* is equivalent to B_Q since it performs the same classification $\forall \mathbf{x} \in \mathcal{X}$. For example, a linear classifier with weight vector $\mathbf{w}$ is equivalent to a Bayes classifier B_Q over linear classifiers with any distribution Q rotationally invariant around $\mathbf{w}$. By choosing a Gaussian (or a rectified Gaussian tail) centered on $\mathbf{w}$ for Q and Gaussian centered at the origin for P, Langford (2005), Langford and Shawe-Taylor (2003), and McAllester (2003b) have been able to derived tight risk bounds for the SVM from the PAC-Bayes theorem in terms of the "margin errors" achieved on the training data.

4. A PAC-Bayes Theorem for the Sample-Compression Setting

In the sample-compression setting, learning algorithms have access to a data-dependent set of classifiers defined as follows. Given a training sequence $S = \langle \mathbf{z}_1, \ldots, \mathbf{z}_m \rangle$ of m examples, each classifier is described entirely by two *complementary sources of information*: a subsequence $S_{\mathbf{i}}$ of S, called the *compression sequence*, and a *message* σ which represents the additional information needed to obtain a classifier from the compression sequence.

Given a training sequence S of m examples, the compression subsequence $S_{\mathbf{i}}$ of S is defined by the following vector $\mathbf{i}$ of indices

$$\begin{array}{rcl} \mathbf{i} & \stackrel{\text{def}}{=} & (i_1, i_2, \ldots, i_{|\mathbf{i}|}) \\ \text{with} & : & i_j \in \{1, \ldots, m\}\ \forall j \\ \text{and} & : & i_1 < i_2 < \ldots < i_{|\mathbf{i}|}\,, \end{array}$$

where $|\mathbf{i}|$ denotes the number of indices present in $\mathbf{i}$. Hence, $S_{\mathbf{i}}$ denotes the $|\mathbf{i}|$-tuple of examples of S that are pointed by the vector $\mathbf{i}$ of indices defined above. We will also use $\bar{\mathbf{i}}$ to denote the vector of indices not present in $\mathbf{i}$. Hence, the union of all the examples of $S_{\mathbf{i}}$ and $S_{\bar{\mathbf{i}}}$ gives all the m examples of S. Finally, we will denote by I the set of the 2^m possible realizations of $\mathbf{i}$.

The fact that each classifier is described by a compression sequence and a message implies that there exists a *reconstruction function* $\mathcal{R}$ that outputs a classifier $\mathcal{R}(\sigma, S_{\mathbf{i}})$ when given an arbitrary compression sequence $S_{\mathbf{i}}$ and a message σ chosen from the set $\mathcal{M}(S_{\mathbf{i}})$ of all distinct messages that can be supplied to $\mathcal{R}$ with the compression sequence $S_{\mathbf{i}}$. This set $\mathcal{M}(S_{\mathbf{i}})$ must be defined *a priori* (before observing S) for all possible sequences $S_{\mathbf{i}}$ of examples. For any sequence S of m examples, we will also use

$$\mathcal{M}_S \stackrel{\text{def}}{=} \bigcup_{\mathbf{i} \in I} \mathcal{M}(S_{\mathbf{i}})\,.$$

1. For example, if there exists $(\mathbf{x}, y)$ such that $B_Q(\mathbf{x}) \neq y$ and $\Pr_{h \sim Q}(h(\mathbf{x}) \neq y) = \frac{1}{2} + \varepsilon$, then $R(B_Q) = 2R(G_Q) - 2\varepsilon$ for the data-generating distribution that has all its weight on $(\mathbf{x}, y)$.

The perceptron learning rule and the SVM are examples of learning algorithms where the final classifier can be reconstructed solely from a compression sequence (Graepel et al., 2005). In contrast, the reconstruction functions for the set covering machine (Marchand and Shawe-Taylor, 2002) and the decision list machine (Marchand and Sokolova, 2005) need both a compression sequence and a message string. Furthermore, Marchand and Sokolova (2005) provide numerous examples where it is advantageous to have a set $\mathcal{M}(S_{\mathbf{i}})$ of possible messages that depend on the compression sequence $S_{\mathbf{i}}$. In these circumstances, the set of messages can be substantially reduced by using the information contained in $S_{\mathbf{i}}$. We will provide detailed examples below of data-dependent distributions of messages $\mathcal{M}(S_{\mathbf{i}})$.

It is important to realize that the sample-compression setting is strictly more general than the usual data-independent setting where the space $\mathcal{H}$ of possible classifiers (considered by learning algorithms) is defined without reference to the training data. Indeed, we recover this usual setting when each classifier is identified only by a message σ taken from a set $\mathcal{M} \stackrel{\text{def}}{=} \mathcal{M}(\emptyset)$. In that case, for each $\sigma \in \mathcal{M}$, we have a classifier $\mathcal{R}(\sigma)$. Hence, in this limit, we have a data-independent set $\mathcal{H}$ of classifiers given by $\mathcal{R}$ and $\mathcal{M}$ such that

$$\mathcal{H} = \{\mathcal{R}(\sigma) \mid \sigma \in \mathcal{M}\}.$$

However, the validity of Theorem 1 has been established only in the usual data-independent setting where the priors are defined without reference to the training data S. More recently, Catoni (2004) has introduced priors where some data-dependence is allowed. Here, we derive here a new PAC-Bayes theorem for priors that are more natural for sample-compression algorithms. These are priors defined over $I \times \mathcal{M}_S$ for any possible $S \in (\mathcal{X} \times \mathcal{Y})^m$. More precisely, for each $S \in (\mathcal{X} \times \mathcal{Y})^m$, we will only consider priors P_S on $I \times \mathcal{M}_S$ that can be be written as the product

$$P_S(\mathbf{i}, \sigma) \;=\; P_I(\mathbf{i}) P_{\mathcal{M}(S_{\mathbf{i}})}(\sigma), \tag{1}$$

where $P_I(\mathbf{i})$ is the prior probability of using the vector $\mathbf{i}$ of indices (defined above) and where $P_{\mathcal{M}(S_{\mathbf{i}})}(\sigma)$ is the prior probability of using the message string σ given that we use the compression sequence $S_{\mathbf{i}}$ (i.e., a vector $\mathbf{i}$ with a sequence S). The message string σ could also be a *parameter* chosen from a continuous set $\mathcal{M}(S_{\mathbf{i}})$. In this case, $P_{\mathcal{M}(S_{\mathbf{i}})}(\sigma)$ specifies a probability density function. Throughout the paper, a distribution on $I \times \mathcal{M}_S$, prior or posterior, will always mean a distribution that factorizes as Equation 1.

We consider learning algorithms that output a posterior distribution Q on $I \times \mathcal{M}_S$ after observing some training sequence S. The posterior Q has the same form $Q_I(\mathbf{i}) Q_{\mathcal{M}(S_{\mathbf{i}})}(\sigma)$ as the one given for the prior P_S but both $Q_I(\mathbf{i})$ and $Q_{\mathcal{M}(S_{\mathbf{i}})}(\sigma)$ can be chosen *after* observing the training data S, that is, they can both depend on S in any way. In contrast, $P_I(\mathbf{i})$ cannot depend on S at all and $P_{\mathcal{M}(S_{\mathbf{i}})}$ can only depend on S through $\mathcal{M}(S_{\mathbf{i}})$. This implies that $P_I(\mathbf{i})$ must be defined *before* observing S and $P_{\mathcal{M}(S_{\mathbf{i}})}$ defined[2] for all possible values of S. Consequently, the set of messages $\mathcal{M}(S_{\mathbf{i}})$ must be defined *a priori* for any compression sequence $S_{\mathbf{i}}$ (we will provide examples in the next section).

Since we do not allow any dependence on S for $P_I(\mathbf{i})$, we cannot discriminate a priori between two vectors of indicies $\mathbf{i}, \mathbf{i}' \in I$ that have same size. Hence, we propose to assign the same prior probability to every vector $\mathbf{i}$ having the same size, that is, we choose

$$P_I(\mathbf{i}) = \zeta(|\mathbf{i}|) \cdot \binom{m}{|\mathbf{i}|}^{-1}, \tag{2}$$

2. As we will precisely see later, the allowed dependence on $S_{\mathbf{i}}$ of the prior comes from the fact that the empirical risk of the classifiers will be computed only on the examples of S that are not in the compression sequence $S_{\mathbf{i}}$.

where ζ can be any function satisfying $\sum_{d=0}^{m}\zeta(d)=1$. However, since the risk upper bound will deteriorate as we put more weight on classifiers with large compression sizes $|\mathbf{i}|$, it will be preferable to choose a function $\zeta(d)$ that puts more weight on small values of d.

To shorten the notation, we will denote the true risk $R(\mathcal{R}(\sigma,S_{\mathbf{i}}))$ of classifier $\mathcal{R}(\sigma,S_{\mathbf{i}})$ simply by $R(\sigma,S_{\mathbf{i}})$. Similarly, we will denote the empirical risk $R_{S_{\bar{\mathbf{i}}}}(\mathcal{R}(\sigma,S_{\mathbf{i}}))$ of classifier $\mathcal{R}(\sigma,S_{\mathbf{i}})$ simply by $R_{S_{\bar{\mathbf{i}}}}(\sigma,S_{\mathbf{i}})$. Recall that $S_{\bar{\mathbf{i}}}$ is the set of training examples which are *not* in the compression set $S_{\mathbf{i}}$. Indeed, it will become obvious that the bound on the risk of classifier $\mathcal{R}(\sigma,S_{\mathbf{i}})$ depends only on its empirical risk on $S_{\bar{\mathbf{i}}}$.

Given a training sequence S and a distribution Q, and given a new (testing) input example $\mathbf{x}$, a *sample-compressed Gibbs classifier* G_Q chooses randomly $\mathbf{i}$ according to Q_I and then chooses σ according to $Q_{\mathcal{M}(S_{\mathbf{i}})}$ to obtain classifier $\mathcal{R}(\sigma,S_{\mathbf{i}})$ which is then used to determine the class label of $\mathbf{x}$. Therefore, given a training sequence S and a distribution Q, the true risk $R(G_Q)$ of the sample-compressed Gibbs classifier G_Q is given by

$$R(G_Q)=\mathop{\mathbf{E}}_{\mathbf{i}\sim Q_I}\ \mathop{\mathbf{E}}_{\sigma\sim Q_{\mathcal{M}(S_{\mathbf{i}})}} R(\sigma,S_{\mathbf{i}})\,.$$

Furthermore, its empirical risk $R_S(G_Q)$ is given by

$$R_S(G_Q)=\mathop{\mathbf{E}}_{\mathbf{i}\sim Q_I}\ \mathop{\mathbf{E}}_{\sigma\sim Q_{\mathcal{M}(S_{\mathbf{i}})}} R_{S_{\bar{\mathbf{i}}}}(\sigma,S_{\mathbf{i}})\,.$$

Note that these expectations are defined only within the context of a training sequence S.

Given a posterior Q, some expectations below will be performed on a re-scaled distribution defined by the following.

Definition 2 *Given a distribution Q on $I\times\mathcal{M}_S$, we will denote by $\overline{Q}_I$ the distribution defined as*

$$\overline{Q}_I(\mathbf{i})\stackrel{\text{def}}{=}\frac{Q_I(\mathbf{i})}{|\bar{\mathbf{i}}|\mathop{\mathbf{E}}_{\mathbf{i}\sim Q_I}\frac{1}{|\bar{\mathbf{i}}|}}\quad\forall\mathbf{i}\in I\,,\tag{3}$$

where $|\bar{\mathbf{i}}|\stackrel{\text{def}}{=}m-|\mathbf{i}|$. We will also denote by $\overline{Q}$, the distribution on $I\times\mathcal{M}_S$ given by the product

$$\overline{Q}_I(\mathbf{i})Q_{\mathcal{M}_{S_{\mathbf{i}}}}(\sigma)\,.$$

Furthermore, let

$$d_{\overline{Q}}\stackrel{\text{def}}{=}\mathop{\mathbf{E}}_{\mathbf{i}\sim\overline{Q}_I}|\mathbf{i}|\,.\tag{4}$$

It follows directly from these definitions that

$$\mathop{\mathbf{E}}_{\mathbf{i}\sim Q_I}\frac{1}{|\bar{\mathbf{i}}|}=\frac{1}{\mathop{\mathbf{E}}_{\mathbf{i}\sim\overline{Q}_I}|\bar{\mathbf{i}}|}=\frac{1}{m-d_{\overline{Q}}}\,.\tag{5}$$

Let $\mathbf{i}_f\stackrel{\text{def}}{=}(1,2,\ldots,m)$ be the (full) vector $\mathbf{i}$ that contains all the m indicies. Since $|\bar{\mathbf{i}}_f|=0$, we might think that $\overline{Q}_I$ is undefined whenever $Q_I(\mathbf{i}_f)>0$. However, we can simply show that

definition 2 implies that we must have $\overline{Q}_I(\mathbf{i}_f) = 1$ (and $\overline{Q}_I(\mathbf{i}) = 0 \ \forall \mathbf{i} \neq \mathbf{i}_f$) whenever $Q_I(\mathbf{i}_f) > 0$. This claim simply follows from the fact that for all $\mathbf{i}$ we can write

$$|\bar{\mathbf{i}}| \mathop{\mathbf{E}}_{\mathbf{j}\sim Q_I} \frac{1}{|\bar{\mathbf{j}}|} \ = \ Q_I(\mathbf{i}) + |\bar{\mathbf{i}}| \sum_{\mathbf{j}\neq\mathbf{i}} Q_I(\mathbf{j}) \frac{1}{|\bar{\mathbf{j}}|} \, .$$

Consequently, we have

$$\overline{Q}_I(\mathbf{i}_f) \ = \ \frac{Q_I(\mathbf{i}_f)}{|\bar{\mathbf{i}}_f| \mathop{\mathbf{E}}_{\mathbf{j}\sim Q_I} \frac{1}{|\bar{\mathbf{j}}|}} \ = \ \frac{Q_I(\mathbf{i}_f)}{Q_I(\mathbf{i}_f) + |\bar{\mathbf{i}}_f| \sum_{\mathbf{j}\neq\mathbf{i}_f} Q_I(\mathbf{j}) \frac{1}{|\bar{\mathbf{j}}|}} \ = \ 1 \, .$$

And for all $\mathbf{i} \neq \mathbf{i}_f$, we have

$$\begin{aligned} \overline{Q}_I(\mathbf{i}) \ &= \ \frac{Q_I(\mathbf{i})}{|\bar{\mathbf{i}}| \mathop{\mathbf{E}}_{\mathbf{j}\sim Q_I} \frac{1}{|\bar{\mathbf{j}}|}} \ = \ \frac{Q_I(\mathbf{i})}{Q_I(\mathbf{i}) + |\bar{\mathbf{i}}| \sum_{\mathbf{j}\neq\mathbf{i}} Q_I(\mathbf{j}) \frac{1}{|\bar{\mathbf{j}}|}} \\ &\leq \ \frac{Q_I(\mathbf{i})}{Q_I(\mathbf{i}) + |\bar{\mathbf{i}}| Q_I(\mathbf{i}_f) \frac{1}{|\bar{\mathbf{i}}_f|}} \ = \ 0 \, , \end{aligned}$$

which proves the claim.

The next theorem constitutes our main result.

Theorem 3 *For any $\delta \in (0,1]$, for any reconstruction function mapping compression sequences and messages to classifiers, for any $T \in (\mathcal{X}\times\mathcal{Y})^m$ and for any prior P_T on $I \times \mathcal{M}_T$, we have*

$$\Pr_{S\sim D^m} \left(\forall Q \text{ on } I\times\mathcal{M}_S \colon \mathrm{kl}(R_S(G_Q) \| R(G_Q)) \right. \\ \left. \leq \ \frac{1}{m - d_{\overline{Q}}} \left[\mathrm{KL}(\overline{Q} \| P_S) + \ln \frac{m+1}{\delta} \right] \right) \geq 1 - \delta \, .$$

Similarly as Theorem 1, Theorem 3 provides both an upper bound and a lower bound on the true risk $R(G_Q)$ based on the empirical risk $R_S(G_Q)$.

Note that

$$\begin{aligned} \mathrm{KL}(\overline{Q} \| P_S) \ &= \ \mathop{\mathbf{E}}_{\mathbf{i}\sim\overline{Q}_I} \mathop{\mathbf{E}}_{\sigma\sim Q_{\mathcal{M}(S_\mathbf{i})}} \ln \frac{\overline{Q}_I(\mathbf{i}) Q_{\mathcal{M}(S_\mathbf{i})}(\sigma)}{P_I(\mathbf{i}) P_{\mathcal{M}(S_\mathbf{i})}(\sigma)} \\ &= \ \mathop{\mathbf{E}}_{\mathbf{i}\sim\overline{Q}_I} \ln \frac{\overline{Q}_I(\mathbf{i})}{P_I(\mathbf{i})} + \mathop{\mathbf{E}}_{\mathbf{i}\sim\overline{Q}_I} \mathop{\mathbf{E}}_{\sigma\sim Q_{\mathcal{M}(S_\mathbf{i})}} \ln \frac{Q_{\mathcal{M}(S_\mathbf{i})}(\sigma)}{P_{\mathcal{M}(S_\mathbf{i})}} \\ &= \ \mathrm{KL}(\overline{Q}_I \| P_I) \ + \ \mathop{\mathbf{E}}_{\mathbf{i}\sim\overline{Q}_I} \mathrm{KL}(Q_{\mathcal{M}(S_\mathbf{i})} \| P_{\mathcal{M}(S_\mathbf{i})}) \, . \end{aligned}$$

Although we must define *a priori* a continuous family of priors (one prior P_T on $I \times \mathcal{M}_T$ per possible sequence $T \in (\mathcal{X}\times\mathcal{Y})^m$), only the prior on the observed training sequence S will contribute to the bounds.

Theorem 3 is a generalization of Theorem 1 because the latter corresponds to the case where the probability distribution Q has non-zero weight only for $|\mathbf{i}| = 0$. Indeed, in this case we have $\frac{1}{m-d_{\overline{Q}}} = \frac{1}{m}$ and $\overline{Q} = Q$.

Note also that, when Q_I is non-zero only for one compression size $|\mathbf{i}| = d$, we have $\overline{Q}_I = Q_I$ and $d_{\overline{Q}} = d$. Hence, for a stochastic average of sample-compressed classifiers of fixed compression size d, the risk bounds depend only on the "original" posterior Q_I.

More generally, note that $\overline{Q}_I(\mathbf{i})$ is smaller than $Q_I(\mathbf{i})$ for classifiers having a compression size $|\mathbf{i}|$ smaller than the Q-average. This, combined with the fact that $\mathrm{KL}(\overline{Q}\|P_S)$ favors $\overline{Q}$'s close to P_S, implies that there will be a specialization performed by Q on classifiers having small compression sizes. As an example, in the case where $\overline{Q} = P_S$, it is easy to see that Q will put more weight than P_S on "small" classifiers. The specialization suggested by Theorem 3 is therefore stronger than what it would have been if $\mathrm{KL}(Q\|P_S)$ would have been in the risk bound instead of $\mathrm{KL}(\overline{Q}\|P_S)$. Thus, Theorem 3 reinforces Occam's principle of parsimony.

Note also that, since $R(B_Q) \leq 2R(G_Q)$, Theorem 3 provides an upper bound for the true risk of the (deterministic) majority vote B_Q. Consider, for example, a majority vote of m classifiers, each having a compression size $|\mathbf{i}| = 1$. In that case, this majority vote uses all the m training examples of S. However, the upper bound given by Theorem 3 will be small (whenever $\mathrm{KL}(\overline{Q}\|P_S)$ and $R_S(G_Q)$ are both small) since $d_{\overline{Q}} = 1$.

The rest of this section is devoted to the proof of Theorem 3. We first provide a lemma about the following quantity.

Definition 4 *Let $S \in (\mathcal{X}\times\mathcal{Y})^m$ and D be a distribution on $\mathcal{X}\times\mathcal{Y}$. We will denote by $B_S(\mathbf{i},\sigma)$, the probability that the classifier $\mathcal{R}(\sigma,S_\mathbf{i})$ of (true) risk $R(\sigma,S_\mathbf{i})$ makes exactly $|\bar{\mathbf{i}}|R_{S_{\bar{\mathbf{i}}}}(\sigma,S_\mathbf{i})$ errors on $S'_{\bar{\mathbf{i}}} \sim D^{|\bar{\mathbf{i}}|}$. Hence, equivalently, we have*

$$B_S(\mathbf{i},\sigma) \stackrel{\text{def}}{=} \binom{|\bar{\mathbf{i}}|}{|\bar{\mathbf{i}}|R_{S_{\bar{\mathbf{i}}}}(\sigma,S_\mathbf{i})} (R(\sigma,S_\mathbf{i}))^{|\bar{\mathbf{i}}|R_{S_{\bar{\mathbf{i}}}}(\sigma,S_\mathbf{i})} (1-R(\sigma,S_\mathbf{i}))^{|\bar{\mathbf{i}}|-|\bar{\mathbf{i}}|R_{S_{\bar{\mathbf{i}}}}(\sigma,S_\mathbf{i})} .$$

Lemma 5 *For any $\delta \in (0,1]$, for any reconstruction function mapping compression sequences and messages to classifiers, for any $T \in (\mathcal{X}\times\mathcal{Y})^m$ and for any prior P_T on $I \times \mathcal{M}_T$, we have*

$$\Pr_{S\sim D^m}\left(\mathop{\mathbf{E}}_{\mathbf{i}\sim P_I} \mathop{\mathbf{E}}_{\sigma\sim P_{\mathcal{M}(S_\mathbf{i})}} \frac{1}{B_S(\mathbf{i},\sigma)} \leq \frac{m+1}{\delta} \right) \geq 1-\delta .$$

Proof First observe that (for any $\mathbf{i} \in I$, $S_\mathbf{i} \in (\mathcal{X}\times\mathcal{Y})^{|\mathbf{i}|}$, and $\sigma \in \mathcal{M}(S_\mathbf{i})$)

$$\begin{aligned}
\mathop{\mathbf{E}}_{S_{\bar{\mathbf{i}}}\sim D^{|\bar{\mathbf{i}}|}} \frac{1}{B_S(\mathbf{i},\sigma)} &= \sum_{k=0}^{|\bar{\mathbf{i}}|} \Pr_{S_{\bar{\mathbf{i}}}\sim D^{|\bar{\mathbf{i}}|}} \left(R_{S_{\bar{\mathbf{i}}}}(\sigma,S_\mathbf{i}) = \frac{k}{|\bar{\mathbf{i}}|} \right) \left[\mathop{\mathbf{E}}_{S_{\bar{\mathbf{i}}}\sim D^{|\bar{\mathbf{i}}|}\left| R_{S_{\bar{\mathbf{i}}}}(\sigma,S_\mathbf{i})=\frac{k}{|\bar{\mathbf{i}}|}\right.} \left(\frac{1}{B_S(\mathbf{i},\sigma)} \right) \right] \\
&= \sum_{k=0}^{|\bar{\mathbf{i}}|} \frac{\Pr_{S_{\bar{\mathbf{i}}}\sim D^{|\bar{\mathbf{i}}|}} \left(|\bar{\mathbf{i}}|R_{S_{\bar{\mathbf{i}}}}(\sigma,S_\mathbf{i}) = k \right)}{\binom{|\bar{\mathbf{i}}|}{k} (R(\sigma,S_\mathbf{i}))^k (1-R(\sigma,S_\mathbf{i}))^{|\bar{\mathbf{i}}|-k}} = \sum_{k=0}^{m-|\mathbf{i}|} 1 = m-|\mathbf{i}|+1 .
\end{aligned}$$

Since the expectation over $S_{\bar{\mathbf{i}}}$ is independent of $S_\mathbf{i}$, for any P_I and $P_{\mathcal{M}(S_\mathbf{i})}$ we have

$$\begin{aligned}
\mathop{\mathbf{E}}_{S\sim D^m} \mathop{\mathbf{E}}_{\mathbf{i}\sim P_I} \mathop{\mathbf{E}}_{\sigma\sim P_{\mathcal{M}(S_\mathbf{i})}} \frac{1}{B_S(\mathbf{i},\sigma)} &= \mathop{\mathbf{E}}_{\mathbf{i}\sim P_I} \mathop{\mathbf{E}}_{S_\mathbf{i}\sim D^{|\mathbf{i}|}} \mathop{\mathbf{E}}_{\sigma\sim P_{\mathcal{M}(S_\mathbf{i})}} \mathop{\mathbf{E}}_{S_{\bar{\mathbf{i}}}\sim D^{|\bar{\mathbf{i}}|}} \frac{1}{B_S(\mathbf{i},\sigma)} \\
&= \mathop{\mathbf{E}}_{\mathbf{i}\sim P_I} \mathop{\mathbf{E}}_{S_\mathbf{i}\sim D^{|\mathbf{i}|}} \mathop{\mathbf{E}}_{\sigma\sim P_{\mathcal{M}(S_\mathbf{i})}} m-|\mathbf{i}|+1 \\
&= m-|\mathbf{i}|+1 \leq m+1 .
\end{aligned}$$

In the first equation above, note that $\mathcal{M}(S_{\mathbf{i}})$ must be defined for all possible values of $S_{\mathbf{i}}$ since the expectation on the left-hand side is performed for all possible values of S. Note also that the dependence of the prior P on S comes only through $S_{\mathbf{i}}$. Finally, since $\mathop{\mathbf{E}}_{\mathbf{i}\sim P_I} \mathop{\mathbf{E}}_{\sigma\sim P_{\mathcal{M}(S_{\mathbf{i}})}} \frac{1}{B_S(\mathbf{i},\sigma)}$ is a non-negative random variable (function of S) having an expectation of at most $m+1$, we can use Markov's inequality to obtain the lemma. ■

The next step is to transform the expectation over P_S into an expectation over Q to obtain the following lemma.

Lemma 6 *For any $\delta \in (0,1]$, for any reconstruction function mapping compression sequences and messages to classifiers, for any $T \in (\mathcal{X}\times\mathcal{Y})^m$ and for any prior P_T on $I \times \mathcal{M}_T$, we have*

$$\Pr_{S\sim D^m}\left(\forall Q \text{ on } I\times\mathcal{M}_S\colon \mathop{\mathbf{E}}_{\mathbf{i}\sim Q_I}\mathop{\mathbf{E}}_{\sigma\sim Q_{\mathcal{M}(S_{\mathbf{i}})}} \frac{1}{|\bar{\mathbf{i}}|}\ln\frac{1}{B_S(\mathbf{i},\sigma)} \right.$$
$$\left. \leq \frac{1}{m-d_{\overline{Q}}}\left[\mathrm{KL}(\overline{Q}\|P_S)+\ln\frac{m+1}{\delta}\right]\right) \geq 1-\delta\,.$$

Proof Lemma 5 gives us

$$\Pr_{S\sim D^m}\left(\ln\left[\mathop{\mathbf{E}}_{\mathbf{i}\sim P_I}\mathop{\mathbf{E}}_{\sigma\sim P_{\mathcal{M}(S_{\mathbf{i}})}}\frac{1}{B_S(\mathbf{i},\sigma)}\right] \leq \ln\frac{m+1}{\delta}\right)\geq 1-\delta\,.$$

Now, for any distribution Q (possibly dependent on S), we have

$$\mathop{\mathbf{E}}_{\mathbf{i}\sim P_I}\mathop{\mathbf{E}}_{\sigma\sim P_{\mathcal{M}(S_{\mathbf{i}})}}\frac{1}{B_S(\mathbf{i},\sigma)} = \mathop{\mathbf{E}}_{\mathbf{i}\sim \overline{Q}_I}\mathop{\mathbf{E}}_{\sigma\sim Q_{\mathcal{M}(S_{\mathbf{i}})}}\frac{P_I(\mathbf{i})P_{\mathcal{M}(S_{\mathbf{i}})}(\sigma)}{\overline{Q}_I(\mathbf{i})Q_{\mathcal{M}(S_{\mathbf{i}})}(\sigma)}\frac{1}{B_S(\mathbf{i},\sigma)}\,.$$

Since $\ln x$ is concave, we can use Jensen's inequality to obtain

$$\begin{aligned}
\ln\left(\mathop{\mathbf{E}}_{\mathbf{i}\sim P_I}\mathop{\mathbf{E}}_{\sigma\sim P_{\mathcal{M}(S_{\mathbf{i}})}}\frac{1}{B_S(\mathbf{i},\sigma)}\right) &\geq \mathop{\mathbf{E}}_{\mathbf{i}\sim \overline{Q}_I}\mathop{\mathbf{E}}_{\sigma\sim Q_{\mathcal{M}(S_{\mathbf{i}})}}\ln\left(\frac{P_I(\mathbf{i})P_{\mathcal{M}(S_{\mathbf{i}})}(\sigma)}{\overline{Q}_I(\mathbf{i})Q_{\mathcal{M}(S_{\mathbf{i}})}(\sigma)}\frac{1}{B_S(\mathbf{i},\sigma)}\right)\\
&= \mathop{\mathbf{E}}_{\mathbf{i}\sim \overline{Q}_I}\mathop{\mathbf{E}}_{\sigma\sim Q_{\mathcal{M}(S_{\mathbf{i}})}}\ln\left(\frac{P_I(\mathbf{i})P_{\mathcal{M}(S_{\mathbf{i}})}(\sigma)}{\overline{Q}_I(\mathbf{i})Q_{\mathcal{M}(S_{\mathbf{i}})}(\sigma)}\right)\\
&\quad + \mathop{\mathbf{E}}_{\mathbf{i}\sim \overline{Q}_I}\mathop{\mathbf{E}}_{\sigma\sim Q_{\mathcal{M}(S_{\mathbf{i}})}}\ln\left(\frac{1}{B_S(\mathbf{i},\sigma)}\right)\\
&= -\mathrm{KL}(\overline{Q}\|P_S)+\mathop{\mathbf{E}}_{\mathbf{i}\sim \overline{Q}_I}\mathop{\mathbf{E}}_{\sigma\sim Q_{\mathcal{M}(S_{\mathbf{i}})}}\ln\left(\frac{1}{B_S(\mathbf{i},\sigma)}\right).
\end{aligned}$$

Consequently, we have

$$\Pr_{S\sim D^m}\left(\forall Q \text{ on } I\times\mathcal{M}_S\colon \mathop{\mathbf{E}}_{\mathbf{i}\sim \overline{Q}_I}\mathop{\mathbf{E}}_{\sigma\sim Q_{\mathcal{M}(S_{\mathbf{i}})}}\ln\left[\frac{1}{B_S(\mathbf{i},\sigma)}\right]\leq \mathrm{KL}(\overline{Q}\|P_S)+\ln\frac{m+1}{\delta}\right)\geq 1-\delta\,.$$

The lemma is obtained from this last equation by using Equations 3, 4, and 5 to transform the expectation with respect to $\overline{Q}_I$ into an expectation with respect to Q_I. ■

We can now prove that Theorem 3 is a direct consequence of Lemma 6, of the convexity of $\mathrm{kl}(q\|p)$, and of a trivial upper-bound on the Binomial.

Proof of Theorem 3

For all non-negative integers n and k such that $k \leq n$ and $n \geq 1$, we have

$$\binom{n}{k}\left(\frac{k}{n}\right)^k\left(1-\frac{k}{n}\right)^{n-k} \;\leq\; 1\,.$$

From Definition 4, we then have (for any $\mathbf{i}$, σ, and S)

$$B_S(\mathbf{i},\sigma) \;\leq\; \left(\frac{R(\sigma,S_{\mathbf{i}})}{R_{S_{\bar{\mathbf{i}}}}(\sigma,S_{\mathbf{i}})}\right)^{|\bar{\mathbf{i}}|R_{S_{\bar{\mathbf{i}}}}(\sigma,S_{\mathbf{i}})}\left(\frac{1-R(\sigma,S_{\mathbf{i}})}{1-R_{S_{\bar{\mathbf{i}}}}(\sigma,S_{\mathbf{i}})}\right)^{|\bar{\mathbf{i}}|-|\bar{\mathbf{i}}|R_{S_{\bar{\mathbf{i}}}}(\sigma,S_{\mathbf{i}})}\,.$$

Consequently, for any $\mathbf{i}$, σ, and S, we have

$$\begin{aligned}\frac{1}{|\bar{\mathbf{i}}|}\ln\frac{1}{B_S(\mathbf{i},\sigma)} \;&\geq\; R_{S_{\bar{\mathbf{i}}}}(\sigma,S_{\mathbf{i}})\ln\frac{R_{S_{\bar{\mathbf{i}}}}(\sigma,S_{\mathbf{i}})}{R(\sigma,S_{\mathbf{i}})}+(1-R_{S_{\bar{\mathbf{i}}}}(\sigma,S_{\mathbf{i}}))\ln\frac{1-R_{S_{\bar{\mathbf{i}}}}(\sigma,S_{\mathbf{i}})}{1-R(\sigma,S_{\mathbf{i}})}\\ &\stackrel{\text{def}}{=}\; \mathrm{kl}\left(R_{S_{\bar{\mathbf{i}}}}(\sigma,S_{\mathbf{i}})\|R(\sigma,S_{\mathbf{i}})\right)\,. \qquad (6)\end{aligned}$$

We now exploit the fact that $\mathrm{kl}(q\|p)$ is a convex function of the pair (q,p) of variables. Indeed, from the log-sum inequality (Cover and Thomas, 1991), we can show that for any $(q,p)\in[0,1]\times[0,1]$, any $(r,s)\in[0,1]\times[0,1]$, and any $\alpha\in[0,1]$, we have

$$\mathrm{kl}\big(\alpha q+(1-\alpha)r\|\alpha p+(1-\alpha)s\big) \;\leq\; \alpha\mathrm{kl}(q\|p)+(1-\alpha)\mathrm{kl}(r\|s)\,.$$

Hence, from Equation 6 and Jensen's inequality applied to $\mathrm{kl}(q\|p)$, we have

$$\begin{aligned}\mathop{\mathbf{E}}_{\mathbf{i}\sim Q_I}\mathop{\mathbf{E}}_{\sigma\sim Q_{\mathcal{M}(S_{\mathbf{i}})}}\frac{1}{|\bar{\mathbf{i}}|}\ln\frac{1}{B_S(\mathbf{i},\sigma)} \;&\geq\; \mathop{\mathbf{E}}_{\mathbf{i}\sim Q_I}\mathop{\mathbf{E}}_{\sigma\sim Q_{\mathcal{M}(S_{\mathbf{i}})}}\mathrm{kl}\left(R_{S_{\bar{\mathbf{i}}}}(\sigma,S_{\mathbf{i}})\|R(\sigma,S_{\mathbf{i}})\right)\\ &\geq\; \mathrm{kl}\left(R_S(G_Q)\|R(G_Q)\right)\,.\end{aligned}$$

Theorem 3 then directly follows from this equation and Lemma 6. ■

5. Learning Algorithms for Stochastic Averages and Majority-Votes of Sample-Compressed Classifiers

There exists numerous learning algorithms for producing a single sample-compressed classifier. The perceptron learning rule, for example, produces a linear classifier which can be reconstructed from the subsequence of training examples that have been used to update the weight vector and the bias of the linear separator (Graepel et al., 2005). This subsequence of examples then constitutes the compression sequence of the sample-compressed classifier and the reconstruction function just

consists of the perceptron learning rule executed on the compression sequence. Another example, also studied by Graepel et al. (2005), is the support vector machine (SVM). Here, the compression sequence consists of the set of support vectors and the reconstruction function, again, just consists of running the original learning algorithm on the compression sequence.

Theorem 3 bounds the risk a stochastic average (and the associated majority-vote) of sample-compressed classifiers. Hence, to apply Theorem 3 to algorithms producing a single classifier (as the ones described above), we need to use a posterior Q that has all its weight on a single classifier. In that case, Theorem 3 reduces to Theorem 7 of the next section. However, due to the presence of $\mathrm{KL}(\overline{Q}\|P_S)$ in Theorem 3, and because the prior P_I in Equation 2 gives an equal *a priori* weight to every vector $\mathbf{i}$ having the same number $|\mathbf{i}|$ of indices, Theorem 3 can provide a smaller risk upper-bound for posteriors Q having non-zero weight on several sample-compressed classifiers than for posteriors having all their weight on a single sample-compressed classifier. In short, the guarantee provided by Theorem 3 might be better for a stochastic average of sample-compressed classifiers than for any single sample-compressed classifier. This observation motivates the consideration of learning algorithms for producing posteriors having non-zero weight over several classifiers.

One way to produce a (hopefully) good posterior over several sample-compressed classifiers is to exploit some inherent randomness, or variability, present in the base learning algorithm for single classifiers. The perceptron learning rule is a good example of a learning algorithm that naturally presents some variability that can be exploited. Indeed, the linear classifier produced by the perceptron learning rule is generally very sensitive to the order of the training sequence of examples. Different permutations of the training sequence are likely to produce different linear classifiers. This variability has been exploited by Herbrich et al. (2001) to produce a large-scaled Bayes point machine. If we are not concerned by the space occupied by a large population of classifiers, it is clear from the work of Herbrich et al. (2001) that we could equally well produce a uniformly-weighted majority vote of perceptrons obtained from a large number of permutations of the training sequence.[3] In that case, the stochastic average of these classifiers would represent the typical perceptron that we would obtain by choosing at random a permutation of the training sequence S. We therefore expect that this stochastic Gibbs classifier would be less sensitive to S than any single perceptron obtained from S. Theorem 3 confirms this intuition with an upper-bound on the risk that increases with the amount of the KL-divergence between the posterior and the prior.

Often, the base learning algorithm has no obvious randomness or variability that can be exploited. The SVM provides an obvious example of this type of algorithm since, given any training sequence S, the maximum soft-margin classifier is unique (and the same for any permutation of S). In these cases, a population of distinct classifiers can be obtained by training the base learning algorithm on several training sequences sampled from the bootstrap distribution defined on the original training sequence S; as done in bagging (Breiman, 1996). Another possibility, is to boost (Freund and Schapire, 1997) the base learning algorithm by adaptively re-weighting a distribution defined on the training sequence S.

5.1 Stochastic Averages and Majority-Votes of Set Covering Machines

The perceptron learning rule and the SVM are examples of learning algorithms that produce sample-compressed classifiers $\mathcal{R}(S_{\mathbf{i}})$ that can be reconstructed *solely* from a compression sequence $S_{\mathbf{i}}$.

3. For the case where the training sequence is not linearly-separable, we could simply add a correction to the diagonal of the kernel matrix as was done by Herbrich et al. (2001).

Hence, they are not examples illustrating all the potential of the machinery of data-dependent messages and priors that was developed in Section 4. We therefore present here an example of a learning algorithm, called the set covering machine (SCM) (Marchand and Shawe-Taylor, 2002), that does make use of data-dependent messages to represent sample-compressed classifiers.

As described in Marchand and Shawe-Taylor (2002) and Marchand and Sokolova (2005), a SCM classifier is a conjunction of data-dependent Boolean-valued features.[4] For simplicity, let us limit ourselves to the case where this set of features consists of data-dependent balls and holes; as described in Marchand and Sokolova (2005).

Each such feature g is identified by a *center* $\mathbf{x}_c$ and a radius ρ. Let $d : \mathcal{X} \times \mathcal{X} \to \mathbb{R}^+$ be any metric. In the case where feature g is a *ball*, its output $g(\mathbf{x})$, for any $\mathbf{x} \in \mathcal{X}$, is given by

$$g(\mathbf{x}) \;=\; \begin{cases} 1 & \text{if} \quad d(\mathbf{x},\mathbf{x}_c) \leq \rho \\ 0 & \text{if} \quad d(\mathbf{x},\mathbf{x}_c) > \rho \end{cases} \qquad \text{(for balls)} \;.$$

When feature g is a *hole*, its output $g(\mathbf{x})$, for any $\mathbf{x} \in \mathcal{X}$, is given by

$$g(\mathbf{x}) \;=\; \begin{cases} 0 & \text{if} \quad d(\mathbf{x},\mathbf{x}_c) \leq \rho \\ 1 & \text{if} \quad d(\mathbf{x},\mathbf{x}_c) > \rho \end{cases} \qquad \text{(for holes)} \;.$$

Each possible pair $(\mathbf{z}_c, \mathbf{z}_b)$ of training examples taken from S defines a feature (ball or hole) that could potentially be included in the (final) conjunction classifier. The first example, $\mathbf{z}_c = (\mathbf{x}_c, y_c)$, defines the center of the feature and the second example, $\mathbf{z}_b = (\mathbf{x}_b, y_b)$, called the *border*, determines its radius $\rho = d(\mathbf{x}_c, \mathbf{x}_b) \pm \varepsilon$; where ε is some very small positive constant chosen *a priori*. To obtain a conjunction consistent with each center of its features, we limit ourselves to the case where a feature centered on a positive example must be a ball and a feature centered on a negative example must be a hole. Moreover, because we want to obtain a conjunction that makes few training errors and that contains a small number of features, it is a good strategy to try to use features that, individually, assigns 0 to a large number of negative examples and to a small number of positive examples.[5] In order to achieve this goal reasonably rapidly and to reduce as far as possible the expected size of the message strings, we will here limit ourselves to conjunctions of features satisfying the following conditions:

1. no two concentric balls or holes can belong to the same conjunction;
2. border points are only chosen among the positive examples of S;
3. the conjunction makes no error on its compression sequence $S_{\mathbf{i}}$.

The algorithm proposed by Marchand and Shawe-Taylor (2002) greedily constructs conjunctions of balls and holes that respect these conditions. Condition 1 is motivated by the fact that a conjunction of two concentric balls gives the same classifier as the single inner ball (and, symmetrically, a conjunction of two concentric holes can be replaced by the outer hole). Condition 2 follows from our strategy of looking for features that, individually, assign 0 to a large number of negative examples

4. Here we use the usual convention that the truth values "false" and "true" of Boolean-valued classifiers are mapped, respectively, to the output values of -1 and $+1$ of binary-valued classifiers.
5. We exploit here some properties of a conjunction. Namely, a conjunction that makes no training errors consists of features that, individually, classify correctly all the positive examples in S. In addition, each negative example of S must be correctly classified by at least one feature in the conjunction.

and to a small number of positive examples. Indeed, a hole, that can assign 0 to one more negative example by increasing its radius, without assigning 0 to an extra positive example, will be a "better" hole to include in the conjunction (a similar observation applies to balls). For Condition 3, we will see, in the next paragraph, how it helps in reducing the expected size of the messages strings. For now, note that, to make each feature consistent with its border point, we have to choose $\rho = d(\mathbf{x}_c, \mathbf{x}_b) + \varepsilon$ for a ball and $\rho = d(\mathbf{x}_c, \mathbf{x}_b) - \varepsilon$ for a hole.

Even under these three conditions, the compression sequence $S_{\mathbf{i}}$ alone, does not give enough information to reconstruct the conjunction of features. From the previous paragraph, we do know that each negative example in $S_{\mathbf{i}}$ must be the center of a hole. However, each positive example in $S_{\mathbf{i}}$ could either be a center (of a ball) or a border point (or both). Hence, we will use messages to identify centers from among the positive examples in $S_{\mathbf{i}}$. For this task, let $P(S_{\mathbf{i}})$ denote the set of positive examples among $S_{\mathbf{i}}$. It follows from Condition 1 that, once we use a message that specifies which are the examples among $P(S_{\mathbf{i}})$ that are used for centers, we automatically know how many balls to reconstruct from $S_{\mathbf{i}}$. Moreover, since, by Condition 2, each negative example of $S_{\mathbf{i}}$ must be the center of one hole, the number of holes to reconstruct from $S_{\mathbf{i}}$ is equal to the number of negative examples in $S_{\mathbf{i}}$. What remains to be determined, to specify the (final) conjunction classifier, is the border point in $P(S_{\mathbf{i}})$ for each center in $S_{\mathbf{i}}$. Let us now recall that a conjunction that makes no errors on $S_{\mathbf{i}}$ consists of features that, individually, classify correctly each example in $P(S_{\mathbf{i}})$. Thus, Condition 3 implies that the border point of each ball center $\mathbf{x}_c$ must be the example in $P(S_{\mathbf{i}})$ which is located furthest from $\mathbf{x}_c$ and the border point of each hole center $\mathbf{x}'_c$ must be the example in $P(S_{\mathbf{i}})$ which is located closest from $\mathbf{x}'_c$. Hence, the additional information message $\sigma \in \mathcal{M}(S_{\mathbf{i}})$ only needs to specify which are the examples in $P(S_{\mathbf{i}})$ that are centers. For this task, we can simply use a single bit for each example in $P(S_{\mathbf{i}})$ to indicate if that example is a center. The set $\mathcal{M}(S_{\mathbf{i}})$ then consists of the set of all such possible bit sequences that we can use, given $S_{\mathbf{i}}$. The total number $|\mathcal{M}(S_{\mathbf{i}})|$ of possible messages is then equal to $2^{p(S_{\mathbf{i}})}$, where $p(S_{\mathbf{i}}) = |P(S_{\mathbf{i}})|$.

Now, to use Theorem 3 as a risk bound for a stochastic average (i.e., Gibbs) of SCMs, we need to specify the prior $P_I(\mathbf{i})P_{\mathcal{M}(S_{\mathbf{i}})}(\sigma)$ over sets of indices and messages. For $P_I(\mathbf{i})$, we use the form proposed in Equation 2. For $P_{\mathcal{M}(S_{\mathbf{i}})}(\sigma)$, we can simply[6] assign the same probability to each message $\sigma \in \mathcal{M}(S_{\mathbf{i}})$. In that case we have

$$P_{\mathcal{M}(S_{\mathbf{i}})}(\sigma) \;=\; 2^{-p(S_{\mathbf{i}})} \quad \forall \sigma \in \mathcal{M}(S_{\mathbf{i}}) \; .$$

The task of the learner will then be to produce a posterior $Q_I(\mathbf{i})Q_{\mathcal{M}(S_{\mathbf{i}})}(\sigma)$ such that, hopefully, $R(G_Q)$ (and thus, indirectly, $R(B_Q)$) will be minimal. For this task we could either bag (Breiman, 1996) or boost (Freund and Schapire, 1997) the SCM learning algorithm proposed by Marchand and Shawe-Taylor (2002). In these cases, the message part of the posterior, $Q_{\mathcal{M}(S_{\mathbf{i}})}(\sigma)$, will be non-zero only for the particular message that is actually used for each compression sequence $S_{\mathbf{i}}$ that occurs in the majority-vote of SCMs. Hence, for each such $S_{\mathbf{i}}$, we have

$$\mathrm{KL}(Q_{\mathcal{M}(S_{\mathbf{i}})} \| P_{\mathcal{M}(S_{\mathbf{i}})}) = p(S_{\mathbf{i}}) \ln 2 \, .$$

Another version of the SCM, called PAC-Bayes SCM, has been proposed recently by Laviolette et al. (2006). In this version each ball radius is specified by a real-valued "message" instead of

6. Marchand and Sokolova (2005) proposed a different data-dependent set of messages and prior which can give a (slightly) tighter risk bound than the solution we present here. For pedagogical reasons, we have chosen to present here a simpler (and easier to understand) example of $\mathcal{M}(S_{\mathbf{i}})$ and $P_{\mathcal{M}(S_{\mathbf{i}})}(\sigma)$.

a border point. Hence, each compression sequence $S_{\mathbf{i}}$ consists only of the centers. Given $S_{\mathbf{i}}$, the message $\boldsymbol{\sigma}$ consists of a $|\mathbf{i}|$-tuple of radius values. Given some large distance R defined *a priori*, the prior is given by

$$P_{\mathcal{M}(S_{\mathbf{i}})}(\boldsymbol{\sigma}) \;=\; \prod_{i\in\mathbf{i}} \frac{1}{R} \quad \forall \sigma_i \in [0,R]\,, \tag{7}$$

and the posterior is given by

$$Q_{\mathcal{M}(S_{\mathbf{i}})}(\boldsymbol{\sigma}) \;=\; \prod_{i\in\mathbf{i}} \frac{1}{b_i - a_i} \quad \forall \sigma_i \in [a_i,b_i] \subseteq [0,R]\,, \tag{8}$$

where each a_i and b_i values are selected by the learner. This gives

$$\mathrm{KL}(Q_{\mathcal{M}(S_{\mathbf{i}})} \| P_{\mathcal{M}(S_{\mathbf{i}})}) = \sum_{i\in\mathbf{i}} \ln\left(\frac{R}{b_i - a_i}\right).$$

A posterior over several SCMs of this type could be constructed by bagging (Breiman, 1996) or boosting (Freund and Schapire, 1997) the soft-greedy algorithm proposed by Laviolette et al. (2006). Theorem 3 then provides an upper bound for the risk of these stochastic averages (and associated majority-votes) of PAC-Bayes SCMs.

6. Single Sample-Compressed Classifiers

In this section, we examine the case when the posterior has all its weight on a single sample-compressed classifier and show that the risk upper-bound for this case is competitive with the currently-known tightest sample-compression risk bounds (Langford, 2005, Laviolette et al., 2005).

Let us examine the case when, given a training sequence S, the (stochastic) sample-compressed Gibbs classifier becomes a deterministic classifier with a posterior having all its weight on a single sample-compressed classifier $\mathcal{R}(S_{\mathbf{i}},\sigma)$. In that case, $\overline{Q}_I = Q_I$, $d_{\overline{Q}} = |\mathbf{i}|$, and $\mathrm{KL}(\overline{Q}\|P_S) = -\ln(P_I(\mathbf{i})P_{\mathcal{M}(S_{\mathbf{i}})}(\sigma))$. Consequently, Lemma 6 gives the following inequality for any prior P_S and any reconstruction function.

$$\Pr_{S\sim D^m}\left(\forall \mathbf{i}\in I, \forall \sigma\in\mathcal{M}(S_{\mathbf{i}}):\; \mathop{\mathbf{E}}_{\mathbf{i}\sim Q_I}\; \mathop{\mathbf{E}}_{\sigma\sim Q_{\mathcal{M}(S_{\mathbf{i}})}}\; \ln\frac{1}{B_S(\mathbf{i},\sigma)} \right.$$
$$\left. \leq\; \ln\frac{1}{P_I(\mathbf{i})P_{\mathcal{M}(S_{\mathbf{i}})}(\sigma)} + \ln\frac{m+1}{\delta}\right) \geq 1-\delta\,. \tag{9}$$

Now, let us use the binomial distribution

$$\mathrm{Bin}\,(m,k,r) \;\stackrel{\text{def}}{=}\; \binom{m}{k} r^k (1-r)^{m-k},$$

to express $B_S(\mathbf{i},\sigma)$ as

$$B_S(\mathbf{i},\sigma) \;=\; \mathrm{Bin}\left(m, mR_{S_{\mathbf{i}}}(\sigma,S_{\mathbf{i}}), R(\sigma,S_{\mathbf{i}})\right).$$

Let us now define the *binomial inversion* as

$$\overline{\mathrm{Bin}}\,(m,k,\delta) \;\stackrel{\text{def}}{=}\; \sup\{r\colon \mathrm{Bin}\,(m,k,r) \geq \delta\}\,.$$

Equation 9 then gives the following theorem.

Theorem 7 *For any $\delta \in (0,1]$, for any reconstruction function mapping compression sequences and messages to classifiers, for any $T \in (\mathcal{X} \times \mathcal{Y})^m$ and for any prior P_T on $I \times \mathcal{M}_T$, we have*

$$\Pr_{S \sim D^m} \left(\forall \mathbf{i} \in I, \forall \sigma \in \mathcal{M}(S_{\mathbf{i}}) : \; R(\sigma, S_{\mathbf{i}}) \leq \overline{\mathrm{Bin}} \left(m, m R_{S_{\bar{\mathbf{i}}}}(\sigma, S_{\mathbf{i}}), \frac{P_I(\mathbf{i}) P_{\mathcal{M}(S_{\mathbf{i}})}(\sigma) \delta}{(m+1)} \right) \right) \geq 1 - \delta \, .$$

Let us now compare this risk bound with the tightest currently known sample-compression risk bounds. The bound proposed in Laviolette et al. (2005) generalizes the the bound proposed by Langford (2005) to the case where message strings are also used to identify classifiers. With the current notation, the bound proposed by Laviolette et al. (2005) can be written as

$$\Pr_{S \sim D^m} \left(\forall \mathbf{i} \in I, \forall \sigma \in \mathcal{M}(S_{\mathbf{i}}) : \; R(\sigma, S_{\mathbf{i}}) \leq \overline{\mathrm{BinT}} \left(m, m R_{S_{\bar{\mathbf{i}}}}(\sigma, S_{\mathbf{i}}), P_I(\mathbf{i}) P_{\mathcal{M}(S_{\mathbf{i}})}(\sigma) \delta \right) \right) \geq 1 - \delta \, ,$$

where, instead of the binomial inversion, we use the binomial *tail* inversion defined as

$$\overline{\mathrm{BinT}}(m, k, \delta) \stackrel{\text{def}}{=} \sup \left\{ r : \; \sum_{i=0}^{k} \mathrm{Bin}(m, i, r) \geq \delta \right\} .$$

Consequently, for all values of m, k, δ, we have

$$\overline{\mathrm{Bin}}(m, k, \delta) \; \leq \; \overline{\mathrm{BinT}}(m, k, \delta)$$

When both m and δ are non zero, the equality is realized only for $k = 0$. Hence, the bound of Theorem 7 would be tighter than the bound of Laviolette et al. (2005) if Theorem 7 would hold for δ instead of $\delta/(m+1)$. The bound of Theorem 7 is therefore "competitive" with the currently-known tightest sample-compression risk bound.

Let us now examine the "intermediate" case where Q_I has all its weight on a single vector of indices $\mathbf{i}$ and $Q_{\mathcal{M}(S_{\mathbf{i}})}$ has non-zero weight on several messages $\sigma \in \mathcal{M}(S_{\mathbf{i}})$. In this case we have $\overline{Q} = Q$ and $\mathrm{KL}(Q \| P_S) = -\ln(P_I(\mathbf{i})) + \mathrm{KL}(Q_{\mathcal{M}(S_{\mathbf{i}})} \| P_{\mathcal{M}(S_{\mathbf{i}})})$. Moreover, given a training sequence S of m examples and a vector $\mathbf{i}$ selected by the learner, the Gibbs classifier $G_{Q_{\mathcal{M}(S_{\mathbf{i}})}}$ just chooses randomly according to $Q_{\mathcal{M}(S_{\mathbf{i}})}$ a message σ to classify any new example $\mathbf{x}$ with classifier $\mathcal{R}(\sigma, S_{\mathbf{i}})$. Consequently, Theorem 3 reduces to the following corollary.

Corollary 8 *For any $\delta \in (0,1]$, for any reconstruction function mapping compression sequences and messages to classifiers, for any $T \in (\mathcal{X} \times \mathcal{Y})^m$ and for any prior P_T on $I \times \mathcal{M}_T$, we have*

$$\Pr_{S \sim D^m} \left(\forall \mathbf{i} \in I, \forall Q_{\mathcal{M}(S_{\mathbf{i}})} : \mathrm{kl}(R_S(G_{Q_{\mathcal{M}(S_{\mathbf{i}})}}) \| R(G_{Q_{\mathcal{M}(S_{\mathbf{i}})}}) \right.$$
$$\left. \leq \; \frac{1}{m - |\mathbf{i}|} \left[\ln \frac{1}{P_I(\mathbf{i})} + \mathrm{KL}(Q_{\mathcal{M}(S_{\mathbf{i}})} \| P_{\mathcal{M}(S_{\mathbf{i}})}) + \ln \frac{m+1}{\delta} \right] \right) \geq 1 - \delta \, .$$

This corollary gives the risk bound proposed in Laviolette et al. (2006) for the PAC-Bayes SCM when the prior is given by Equation 7 and the posterior is given by Equation 8.

7. Compression Sequences that Include Training Errors

In their pioneering work on sample compression, Littlestone and Warmuth (1986) have handled the case of non-zero training errors (also called the "lossy" compression case) by including the training error points in the sample compression sequence $S_{\mathbf{i}}$. Within this methodology, a part of the message string σ is used to indicate which indices in $\mathbf{i}$ are pointing to training error examples. The other indices in $\mathbf{i}$ are then automatically pointing to the training examples actually used for constructing the classifier. We can also use this methodology for deriving another upper-bound on the risk of a stochastic average of sample-compressed classifiers. The resulting upper-bound is generally slightly looser than the one given by Theorem 3 but it has the advantage of being simpler (and easier to interpret). In addition, it becomes slightly *tighter* than the bound of Theorem 3 in the limit of a consistent Gibbs classifier (i.e., when G_Q makes no training errors). We will thus present (and prove) this other upper-bound.

Since each sample-compressed classifier $\mathcal{R}(\sigma, S_{\mathbf{i}})$ is still given by $\mathbf{i}$ and σ (once we have a training sequence S), we can still use all the definitions up to Theorem 3. However, $\mathbf{i}$ points also to training errors in S and σ specifies also the indices of $\mathbf{i}$ pointing to training errors. In particular, this implies that we still use posteriors of the form $Q_I(\mathbf{i})Q_{\mathcal{M}(S_{\mathbf{i}})}(\sigma)$ but now $R_S(G_Q)$ is always zero. With this important difference in mind, we have the following theorem.

Theorem 9 *For any $\delta \in (0,1]$, for any reconstruction function mapping compression sequences and messages to classifiers, for any $T \in (\mathcal{X}\times\mathcal{Y})^m$ and for any prior P_T on $I \times \mathcal{M}_T$, we have*

$$\Pr_{S\sim D^m}\left(\forall Q \text{ on } I\times\mathcal{M}_S \text{ such that } R_S(G_Q)=0\colon \right.$$
$$\left. R(G_Q) \;\leq\; 1-\exp\left[\frac{-1}{m-d_{\overline{Q}}}\left(\mathrm{KL}(\overline{Q}\|P_S)+\ln\frac{1}{\delta}\right)\right]\right) \geq 1-\delta\,.$$

Before we prove this theorem, let us compare it with Theorem 3 in the consistent case (when G_Q makes no training errors). For Theorem 9, this means that no part of the message σ is needed to indicate the indices in $\mathbf{i}$ that point to training errors. Hence, the messages used for the reconstruction function are identical for both theorems in the consistent case. Theorem 3, however, applies for $R_S(G_Q)=0$. Since

$$\mathrm{kl}(0\|R(G_Q)) = \ln\left(\frac{1}{1-R(G_Q)}\right),$$

the upper bound of Theorem 3 becomes identical to the bound of Theorem 9 except for the presence of a $\ln(m+1)$ term in the argument of the exponential in Theorem 3. Consequently, the bound of Theorem 9 is slightly tighter in the consistent case.

Proof Since the index vector $\mathbf{i}$, used by classifier $\mathcal{R}(\sigma, S_{\mathbf{i}})$, now contains pointers to error points, all the classifiers having non-zero posterior weight will have $R_{S_{\bar{\mathbf{i}}}}(\sigma, S_{\mathbf{i}}) = 0$. Consequently, instead of using $B_S(\mathbf{i},\sigma)$ (see Definition 4), we will now use $C_S(\mathbf{i},\sigma)$ defined as

$$C_S(\mathbf{i},\sigma) \stackrel{\text{def}}{=} \frac{1}{(1-R(\sigma,S_{\mathbf{i}}))^{m-|\mathbf{i}|}} I\left(R_{S_{\bar{\mathbf{i}}}}(\sigma,S_{\mathbf{i}})=0\right).$$

For any $\mathbf{i} \in I$, $S_{\mathbf{i}} \in (\mathcal{X} \times \mathcal{Y})^{|\mathbf{i}|}$, and $\sigma \in \mathcal{M}(S_{\mathbf{i}})$, we have

$$\mathop{\mathbf{E}}_{S_{\overline{\mathbf{i}}} \sim D^{|\overline{\mathbf{i}}|}} C_S(\mathbf{i}, \sigma) \;=\; \Pr_{S_{\overline{\mathbf{i}}} \sim D^{|\overline{\mathbf{i}}|}} \left(R_{S_{\overline{\mathbf{i}}}}(\sigma, S_{\mathbf{i}}) = 0 \right) \left[\mathop{\mathbf{E}}_{S_{\overline{\mathbf{i}}} \sim D^{|\overline{\mathbf{i}}|} | R_{S_{\overline{\mathbf{i}}}}(\sigma, S_{\mathbf{i}}) = 0} C_S(\mathbf{i}, \sigma) \right] \;=\; 1 \,.$$

Then for any P_I and $P_{\mathcal{M}(S_{\mathbf{i}})}$, we have

$$\mathop{\mathbf{E}}_{S \sim D^m} \mathop{\mathbf{E}}_{\mathbf{i} \sim P_I} \mathop{\mathbf{E}}_{\sigma \sim P_{\mathcal{M}(S_{\mathbf{i}})}} C_S(\mathbf{i}, \sigma) \;=\; \mathop{\mathbf{E}}_{\mathbf{i} \sim P_I} \mathop{\mathbf{E}}_{S_{\mathbf{i}} \sim D^{|\mathbf{i}|}} \mathop{\mathbf{E}}_{\sigma \sim P_{\mathcal{M}(S_{\mathbf{i}})}} \mathop{\mathbf{E}}_{S_{\overline{\mathbf{i}}} \sim D^{|\overline{\mathbf{i}}|}} C_S(\mathbf{i}, \sigma) \;=\; 1 \,.$$

Since $\mathop{\mathbf{E}}_{\mathbf{i} \sim P_I} \mathop{\mathbf{E}}_{\sigma \sim P_{\mathcal{M}(S_{\mathbf{i}})}} C_S(\mathbf{i}, \sigma)$ is a non-negative random variable (function of S) having an expectation of 1, we can use Markov's inequality to obtain

$$\Pr_{S \sim D^m} \left(\mathop{\mathbf{E}}_{\mathbf{i} \sim P_I} \mathop{\mathbf{E}}_{\sigma \sim P_{\mathcal{M}(S_{\mathbf{i}})}} C_S(\mathbf{i}, \sigma) \;\leq\; \frac{1}{\delta} \right) \geq 1 - \delta \,.$$

Thus

$$\Pr_{S \sim D^m} \left(\ln \left[\mathop{\mathbf{E}}_{\mathbf{i} \sim P_I} \mathop{\mathbf{E}}_{\sigma \sim P_{\mathcal{M}(S_{\mathbf{i}})}} C_S(\mathbf{i}, \sigma) \right] \;\leq\; \ln \frac{1}{\delta} \right) \geq 1 - \delta \,.$$

Given this last result, we can use the same technique as in the proof of Lemma 6 to convert the expectation over P_S into an expectation over Q. With this technique, we find that, for any prior P_S, we have

$$\Pr_{S \sim D^m} \Bigg(\forall Q \text{ on } I \times \mathcal{M}_S \text{ such that } R_S(G_Q) = 0 : \\ \mathop{\mathbf{E}}_{\mathbf{i} \sim Q_I} \mathop{\mathbf{E}}_{\sigma \sim Q_{\mathcal{M}(S_{\mathbf{i}})}} \frac{1}{|\overline{\mathbf{i}}|} \ln C_S(\mathbf{i}, \sigma) \;\leq\; \frac{1}{m - d_{\overline{Q}}} \left[\mathrm{KL}(\overline{Q} \| P_S) + \ln \frac{1}{\delta} \right] \Bigg) \geq 1 - \delta \,. \quad (10)$$

Since the posterior $Q_I(\mathbf{i}) Q_{\mathcal{M}(S_{\mathbf{i}})}(\sigma)$ is non-zero only when $R_{S_{\overline{\mathbf{i}}}}(\sigma, S_{\mathbf{i}}) = 0$, we have

$$\mathop{\mathbf{E}}_{\mathbf{i} \sim Q_I} \mathop{\mathbf{E}}_{\sigma \sim Q_{\mathcal{M}(S_{\mathbf{i}})}} \frac{1}{|\overline{\mathbf{i}}|} \ln C_S(\mathbf{i}, \sigma) \;=\; \mathop{\mathbf{E}}_{\mathbf{i} \sim Q_I} \mathop{\mathbf{E}}_{\sigma \sim Q_{\mathcal{M}(S_{\mathbf{i}})}} \ln \left(\frac{1}{1 - R(\sigma, S_{\mathbf{i}})} \right) \;\geq\; \ln \left(\frac{1}{1 - R(G_Q)} \right) ,$$

where the last inequality results from Jensen's inequality applied to the convex function $\ln(1/(1-x))$.

The theorem is obtained by including this last result into Equation 10. ■

8. A PAC-Bayes Theorem for Classifiers that Can Abstain

Many commercial learning systems are producing a "meta-classifier" that consists of an ensemble of rules. In these cases, each rule is basically a classifier that abstains of predicting the class label of an example $\mathbf{x}$ whenever its premiss (which often consists of a conjunction of Boolean-valued features) is false on $\mathbf{x}$. If several rules predict a class label for $\mathbf{x}$, the assigned class label will be the one which is predicted by the largest number of rules in the ensemble. When there is a tie, the whole

ensemble may abstain or predict the most frequently encountered class in the training sequence S. Hence, an ensemble of rules is just a majority-vote of classifiers that may abstain.

To bound the risk of majority-votes and stochastic averages of classifiers that can abstain, it is "natural" to consider loss functions that may take values $\notin \{0,1\}$. If we limit ourselves to loss functions taking values in the $[0,1]$ interval (including the loss value for abstaining), we can use the risk bound of Theorem 1 of McAllester (2003a). However, that bound can be considerably higher than the trivial upper-bound of 1. A better approach would be to use the bound of Theorem 3.2 of Seeger (2003)—which is valid for classifiers predicting a class among k possible values (for any integer $k > 1$). In this section, we propose to generalize this latter approach to the sample-compression setting. [7] Consequently, we will generalize Theorem 3 for a stochastic average of sample-compressed classifiers that may abstain. As before, the theorem will also apply to the usual data-independent setting in the limit where each classifier is only described by a data-independent message (and an empty compression sequence).

Each classifier h has now three possible outcomes $h(\mathbf{x}) \in \{-1,0,+1\}$ on any $\mathbf{x} \in \mathcal{X}$. Classifier h abstains on $\mathbf{x}$ whenever $h(\mathbf{x}) = 0$. Therefore, each classifier h is now characterized by two different probabilities with respect to the random draws of an example $(\mathbf{x}, y) \in \mathcal{X} \times \mathcal{Y}$ (where $\mathcal{Y}$ is still equal to $\{-1,+1\}$). First, we are concerned with the probability $a(h)$ that classifier h abstains on a new example, where

$$a(h) \stackrel{\text{def}}{=} \Pr_{(\mathbf{x},y)\sim D} \left(h(\mathbf{x}) = 0\right).$$

And we are also concerned with the probability $e(h)$ that classifier h predicts the wrong class label of a new example, where

$$e(h) \stackrel{\text{def}}{=} \Pr_{(\mathbf{x},y)\sim D} \left(h(\mathbf{x}) \neq y \wedge h(\mathbf{x}) \neq 0\right).$$

The probability that classifier h predicts the correct class label of a new example is then equal to $1 - e(h) - a(h)$. In contrast with the case where classifiers cannot abstain, each classifier is now characterized by a trivalent random variable (instead of a Bernoulli random variable).

The empirical estimates (of these probabilities) on a training sequence S of m examples are defined as

$$\begin{aligned} e_S(h) &\stackrel{\text{def}}{=} \frac{1}{m}\sum_{i=1}^{m} I(h(\mathbf{x}_i) \neq 0) \cdot I(h(\mathbf{x}_i) \neq y_i), \\ a_S(h) &\stackrel{\text{def}}{=} \frac{1}{m}\sum_{i=1}^{m} I(h(\mathbf{x}_i) = 0). \end{aligned}$$

Similarly as before, each classifier $\mathcal{R}(\sigma, S_{\mathbf{i}})$ is described by a compression sequence $S_{\mathbf{i}}$ and a message σ taken from a data-dependent set $\mathcal{M}(S_{\mathbf{i}})$. Given a training sequence S, the prior and the posterior have the same form as before. Moreover, $a(\sigma, S_{\mathbf{i}})$ and $e(\sigma, S_{\mathbf{i}})$ will denote, respectively, the probability of abstaining and the probability of incorrectly predicting a label for classifier $\mathcal{R}(\sigma, S_{\mathbf{i}})$. We will also denote by $a_{S_{\bar{\mathbf{i}}}}(\sigma, S_{\mathbf{i}})$ and $e_{S_{\bar{\mathbf{i}}}}(\sigma, S_{\mathbf{i}})$ the empirical estimates (of the corresponding probabilities) on the subsequence $S_{\bar{\mathbf{i}}}$ of examples in S that are *not* used for constructing classifier $\mathcal{R}(\sigma, S_{\mathbf{i}})$.

7. We consider here the particular case where each classifier either predicts -1 or +1 or abstains of predicting. The generalization to k classes is straightforward.

The stochastic sample-compressed Gibbs classifier is the same as before. Namely, given a training sequence S and a distribution Q, and given a new (testing) input example $\mathbf{x}$, a sample-compressed Gibbs classifier G_Q chooses randomly $\mathbf{i}$ according to Q_I and then chooses σ according to $Q_{\mathcal{M}(S_\mathbf{i})}$ to obtain classifier $\mathcal{R}(\sigma, S_\mathbf{i})$ which is then used to determine the class label of $\mathbf{x}$. Therefore, given a training sequence S and a distribution Q, $e(G_Q)$ and $a(G_Q)$ are given by

$$\begin{aligned} e(G_Q) &= \mathop{\mathbf{E}}_{\mathbf{i}\sim Q_I} \mathop{\mathbf{E}}_{\sigma\sim Q_{\mathcal{M}(S_\mathbf{i})}} e(\sigma, S_\mathbf{i}), \\ a(G_Q) &= \mathop{\mathbf{E}}_{\mathbf{i}\sim Q_I} \mathop{\mathbf{E}}_{\sigma\sim Q_{\mathcal{M}(S_\mathbf{i})}} a(\sigma, S_\mathbf{i}). \end{aligned}$$

Furthermore, their empirical estimates (on a training sequence S of m examples) are given by

$$\begin{aligned} e_S(G_Q) &= \mathop{\mathbf{E}}_{\mathbf{i}\sim Q_I} \mathop{\mathbf{E}}_{\sigma\sim Q_{\mathcal{M}(S_\mathbf{i})}} e_{S_{\bar{\mathbf{i}}}}(\sigma, S_\mathbf{i}), \\ a_S(G_Q) &= \mathop{\mathbf{E}}_{\mathbf{i}\sim Q_I} \mathop{\mathbf{E}}_{\sigma\sim Q_{\mathcal{M}(S_\mathbf{i})}} a_{S_{\bar{\mathbf{i}}}}(\sigma, S_\mathbf{i}). \end{aligned}$$

Note that these expectations are defined only within the context of a training sequence S.

Given a posterior Q, the re-scaled posterior $\overline{Q}$ is still given by Definition 2.

Theorem 10 *For any $\delta \in (0,1]$, for any reconstruction function mapping compression sequences and messages to classifiers that may abstain, for any $T \in (\mathcal{X}\times\mathcal{Y})^m$ and for any prior P_T on $I \times \mathcal{M}_T$, we have*

$$\Pr_{S\sim D^m}\left(\forall Q \text{ on } I\times\mathcal{M}_S\colon \mathrm{kl}(a_S(G_Q), e_S(G_Q)\|a(G_Q), e(G_Q)) \right. \\ \left. \leq \frac{1}{m-d_{\overline{Q}}}\left[\mathrm{KL}(\overline{Q}\,\|P_S) + \ln\frac{(m+1)(m+2)}{2\delta}\right]\right) \geq 1-\delta,$$

where $\mathrm{kl}(q_1,q_2\|p_1,p_2)$ is the Kullback-Leibler divergence between the distributions of two trivalent random variables Y_q and Y_p with probabilities (q_1,q_2) and (p_1,p_2) respectively. Hence,

$$\mathrm{kl}(q_1,q_2\|p_1,p_2) = q_1\ln\frac{q_1}{p_1} + q_2\ln\frac{q_2}{p_2} + (1-q_1-q_2)\ln\frac{1-q_1-q_2}{1-p_1-p_2}.$$

Proof The proof essentially parallels the one given for Theorem 3 but with the important difference that the empirical estimates $e_{S_{\bar{\mathbf{i}}}}(\sigma, S_\mathbf{i})$ and $a_{S_{\bar{\mathbf{i}}}}(\sigma, S_\mathbf{i})$ are distributed according to a trinomial (instead of a binomial) with respect to the random draws of S. Consequently, we now define $B_S(\mathbf{i},\sigma)$ as

$$B_S(\mathbf{i},\sigma) \stackrel{\text{def}}{=} \binom{|\bar{\mathbf{i}}|}{|\bar{\mathbf{i}}|a_{S_{\bar{\mathbf{i}}}}(\sigma,S_\mathbf{i})} \binom{|\bar{\mathbf{i}}|(1-a_{S_{\bar{\mathbf{i}}}}(\sigma,S_\mathbf{i}))}{|\bar{\mathbf{i}}|e_{S_{\bar{\mathbf{i}}}}(\sigma,S_\mathbf{i})} (a(\sigma,S_\mathbf{i}))^{|\bar{\mathbf{i}}|a_{S_{\bar{\mathbf{i}}}}(\sigma,S_\mathbf{i})} \\ (e(\sigma,S_\mathbf{i}))^{|\bar{\mathbf{i}}|e_{S_{\bar{\mathbf{i}}}}(\sigma,S_\mathbf{i})} (1-a(\sigma,S_\mathbf{i})-e(\sigma,S_\mathbf{i}))^{|\bar{\mathbf{i}}|(1-a_{S_{\bar{\mathbf{i}}}}(\sigma,S_\mathbf{i})-e_{S_{\bar{\mathbf{i}}}}(\sigma,S_\mathbf{i}))}.$$

Then, for any $\mathbf{i} \in I$, $S_{\mathbf{i}} \in (\mathcal{X} \times \mathcal{Y})^{|\mathbf{i}|}$ and $\sigma \in \mathcal{M}(S_{\mathbf{i}})$, we have

$$\begin{aligned}
\mathop{\mathbf{E}}_{S_{\bar{\mathbf{i}}} \sim D^{|\bar{\mathbf{i}}|}} \frac{1}{B_S(\mathbf{i},\sigma)} &= \sum_{j=0}^{|\bar{\mathbf{i}}|} \sum_{k=0}^{|\bar{\mathbf{i}}|-j} \Pr_{S_{\bar{\mathbf{i}}} \sim D^{|\bar{\mathbf{i}}|}} \left(a_{S_{\bar{\mathbf{i}}}}(\sigma, S_{\mathbf{i}}) = \frac{j}{|\bar{\mathbf{i}}|} \wedge e_{S_{\bar{\mathbf{i}}}}(\sigma, S_{\mathbf{i}}) = \frac{k}{|\bar{\mathbf{i}}|} \right) \\
&\qquad \mathop{\mathbf{E}}_{\left[S_{\bar{\mathbf{i}}} \sim D^{|\bar{\mathbf{i}}|} \,\middle|\, \begin{smallmatrix} a_{S_{\bar{\mathbf{i}}}}(\sigma, S_{\mathbf{i}}) = \frac{j}{|\bar{\mathbf{i}}|}, \\ e_{S_{\bar{\mathbf{i}}}}(\sigma, S_{\mathbf{i}}) = \frac{k}{|\bar{\mathbf{i}}|} \end{smallmatrix} \right]} \left(\frac{1}{B_S(\mathbf{i},\sigma)} \right) \\
&= \sum_{j=0}^{|\bar{\mathbf{i}}|} \sum_{k=0}^{|\bar{\mathbf{i}}|-j} \frac{\Pr_{S_{\bar{\mathbf{i}}} \sim D^{|\bar{\mathbf{i}}|}} \left(a_{S_{\bar{\mathbf{i}}}}(\sigma, S_{\mathbf{i}}) = \frac{j}{|\bar{\mathbf{i}}|} \wedge e_{S_{\bar{\mathbf{i}}}}(\sigma, S_{\mathbf{i}}) = \frac{k}{|\bar{\mathbf{i}}|} \right)}{\binom{|\bar{\mathbf{i}}|}{j} \binom{|\bar{\mathbf{i}}|-j}{k} (a(\sigma, S_{\mathbf{i}}))^j (e(\sigma, S_{\mathbf{i}}))^k (1 - a(\sigma, S_{\mathbf{i}}) - e(\sigma, S_{\mathbf{i}}))^{|\bar{\mathbf{i}}|-j-k}} \\
&= \frac{(|\bar{\mathbf{i}}|+1)(|\bar{\mathbf{i}}|+2)}{2}.
\end{aligned}$$

Since the expectation over $S_{\bar{\mathbf{i}}}$ is independent of $S_{\mathbf{i}}$, for any P_I and $P_{\mathcal{M}(S_{\mathbf{i}})}$ we have

$$\begin{aligned}
\mathop{\mathbf{E}}_{S \sim D^m} \mathop{\mathbf{E}}_{\mathbf{i} \sim P_I} \mathop{\mathbf{E}}_{\sigma \sim P_{\mathcal{M}(S_{\mathbf{i}})}} \frac{1}{B_S(\mathbf{i},\sigma)} &= \mathop{\mathbf{E}}_{\mathbf{i} \sim P_I} \mathop{\mathbf{E}}_{S_{\mathbf{i}} \sim D^{|\mathbf{i}|}} \mathop{\mathbf{E}}_{\sigma \sim P_{\mathcal{M}(S_{\mathbf{i}})}} \mathop{\mathbf{E}}_{S_{\bar{\mathbf{i}}} \sim D^{|\bar{\mathbf{i}}|}} \frac{1}{B_S(\mathbf{i},\sigma)} \\
&= \mathop{\mathbf{E}}_{\mathbf{i} \sim P_I} \mathop{\mathbf{E}}_{S_{\mathbf{i}} \sim D^{|\mathbf{i}|}} \mathop{\mathbf{E}}_{\sigma \sim P_{\mathcal{M}(S_{\mathbf{i}})}} \frac{(|\bar{\mathbf{i}}|+1)(|\bar{\mathbf{i}}|+2)}{2} \\
&= \frac{(|\bar{\mathbf{i}}|+1)(|\bar{\mathbf{i}}|+2)}{2} \leq \frac{(m+1)(m+2)}{2}.
\end{aligned}$$

Since $\mathop{\mathbf{E}}_{\mathbf{i} \sim P_I} \mathop{\mathbf{E}}_{\sigma \sim P_{\mathcal{M}(S_{\mathbf{i}})}} \frac{1}{B_S(\mathbf{i},\sigma)}$ is a non-negative random variable (function of S) having an expectation of at most $(m+1)(m+2)/2$, we can use Markov's inequality to obtain

$$\Pr_{S \sim D^m} \left(\mathop{\mathbf{E}}_{\mathbf{i} \sim P_I} \mathop{\mathbf{E}}_{\sigma \sim P_{\mathcal{M}(S_{\mathbf{i}})}} \frac{1}{B_S(\mathbf{i},\sigma)} \leq \frac{(m+1)(m+2)}{2\delta} \right) \geq 1 - \delta.$$

Hence,

$$\Pr_{S \sim D^m} \left(\ln \left[\mathop{\mathbf{E}}_{\mathbf{i} \sim P_I} \mathop{\mathbf{E}}_{\sigma \sim P_{\mathcal{M}(S_{\mathbf{i}})}} \frac{1}{B_S(\mathbf{i},\sigma)} \right] \leq \ln \frac{(m+1)(m+2)}{2\delta} \right) \geq 1 - \delta.$$

Similarly as in the proof of Lemma 6, we can transform the expectation over P_S into an expectation over Q to obtain

$$\begin{aligned}
\Pr_{S \sim D^m} \Bigg(\forall Q \text{ on } I \times \mathcal{M}_S \colon & \mathop{\mathbf{E}}_{\mathbf{i} \sim Q_I} \mathop{\mathbf{E}}_{\sigma \sim Q_{\mathcal{M}(S_{\mathbf{i}})}} \frac{1}{|\bar{\mathbf{i}}|} \ln \frac{1}{B_S(\mathbf{i},\sigma)} \\
& \leq \frac{1}{m - d_{\overline{Q}}} \left[\mathrm{KL}(\overline{Q} \| P_S) + \ln \frac{(m+1)(m+2)}{2\delta} \right] \Bigg) \geq 1 - \delta. \quad (11)
\end{aligned}$$

For all non-negative integers n, j, k such that $j + k \leq n$ and $n \geq 1$, we have

$$\binom{n}{j} \binom{n-j}{k} \left(\frac{j}{n} \right)^j \left(\frac{k}{n} \right)^k \left(1 - \frac{j}{n} - \frac{k}{n} \right)^{n-j-k} \leq 1.$$

Then, for any $\mathbf{i}$, σ, and S, we have

$$B_S(\mathbf{i},\sigma) \;\leq\; \left(\frac{a(\sigma,S_{\mathbf{i}})}{a_{S_{\bar{\mathbf{i}}}}(\sigma,S_{\mathbf{i}})}\right)^{|\bar{\mathbf{i}}|a_{S_{\bar{\mathbf{i}}}}(\sigma,S_{\mathbf{i}})} \cdot \left(\frac{e(\sigma,S_{\mathbf{i}})}{e_{S_{\bar{\mathbf{i}}}}(\sigma,S_{\mathbf{i}})}\right)^{|\bar{\mathbf{i}}|e_{S_{\bar{\mathbf{i}}}}(\sigma,S_{\mathbf{i}})} \cdot \left(\frac{1-a(\sigma,S_{\mathbf{i}})-e(\sigma,S_{\mathbf{i}})}{1-a_{S_{\bar{\mathbf{i}}}}(\sigma,S_{\mathbf{i}})-e_{S_{\bar{\mathbf{i}}}}(\sigma,S_{\mathbf{i}})}\right)^{|\bar{\mathbf{i}}|(1-a_{S_{\bar{\mathbf{i}}}}(\sigma,S_{\mathbf{i}})-e_{S_{\bar{\mathbf{i}}}}(\sigma,S_{\mathbf{i}}))}.$$

Consequently, for any $\mathbf{i}$, σ, and S, we have

$$\begin{aligned}\frac{1}{|\bar{\mathbf{i}}|}\ln\frac{1}{B_S(\mathbf{i},\sigma)} \;&\geq\; a_{S_{\bar{\mathbf{i}}}}(\sigma,S_{\mathbf{i}})\ln\frac{a_{S_{\bar{\mathbf{i}}}}(\sigma,S_{\mathbf{i}})}{a(\sigma,S_{\mathbf{i}})} + e_{S_{\bar{\mathbf{i}}}}(\sigma,S_{\mathbf{i}})\ln\frac{e_{S_{\bar{\mathbf{i}}}}(\sigma,S_{\mathbf{i}})}{e(\sigma,S_{\mathbf{i}})} \\ &\quad +(1-a_{S_{\bar{\mathbf{i}}}}(\sigma,S_{\mathbf{i}})-e_{S_{\bar{\mathbf{i}}}}(\sigma,S_{\mathbf{i}}))\ln\frac{1-a_{S_{\bar{\mathbf{i}}}}(\sigma,S_{\mathbf{i}})-e_{S_{\bar{\mathbf{i}}}}(\sigma,S_{\mathbf{i}})}{1-a(\sigma,S_{\mathbf{i}})-e(\sigma,S_{\mathbf{i}})} \\ &\stackrel{\text{def}}{=}\; \mathrm{kl}\left(a_{S_{\bar{\mathbf{i}}}}(\sigma,S_{\mathbf{i}}),e_{S_{\bar{\mathbf{i}}}}(\sigma,S_{\mathbf{i}})\|a(\sigma,S_{\mathbf{i}}),e(\sigma,S_{\mathbf{i}})\right).\end{aligned}$$

Since $\mathrm{kl}(q_1,q_2\|p_1,p_2)$ is a convex function of (q_1,q_2,p_1,p_2), we can use Jensen's inequality to obtain

$$\begin{aligned}\mathop{\mathbf{E}}_{\mathbf{i}\sim Q_I}\mathop{\mathbf{E}}_{\sigma\sim Q_{\mathcal{M}(S_{\mathbf{i}})}}\frac{1}{|\bar{\mathbf{i}}|}\ln\frac{1}{B_S(\mathbf{i},\sigma)} \;&\geq\; \mathop{\mathbf{E}}_{\mathbf{i}\sim Q_I}\mathop{\mathbf{E}}_{\sigma\sim Q_{\mathcal{M}(S_{\mathbf{i}})}}\mathrm{kl}\left(a_{S_{\bar{\mathbf{i}}}}(\sigma,S_{\mathbf{i}}),e_{S_{\bar{\mathbf{i}}}}(\sigma,S_{\mathbf{i}})\|a(\sigma,S_{\mathbf{i}}),e(\sigma,S_{\mathbf{i}})\right) \\ &\geq\; \mathrm{kl}\left(a_S(G_Q),e_S(G_Q)\|a(G_Q),e(G_Q)\right).\end{aligned}$$

The theorem directly follows from this equation and Equation 11. ■

All PAC-Bayes theorems, including the above, are statements about probabilities and their empirical estimates on a sample—no loss functions are involved. Here, let us consider that the loss $\ell(h(\mathbf{x}),y)$ suffered by classifier h on an example $(\mathbf{x},y)$ is given by

$$\ell(h(\mathbf{x}),y) \;=\; \begin{cases} 1 & \text{if} \quad h(\mathbf{x})\neq y \;\wedge\; h(\mathbf{x})\neq 0 \\ 0 & \text{if} \quad h(\mathbf{x})=y \\ c & \text{if} \quad h(\mathbf{x})=0\,, \end{cases}$$

for some constant $c\in[0,1]$. Since the *risk* $R(h)$ of classifier h is its expected loss, we have

$$R(h) \;=\; e(h)+c\cdot a(h)\,.$$

Hence, for a sample-compressed Gibbs classifier G_Q, we have $R(G_Q)=e(G_Q)+c\cdot a(G_Q)$ (with a similar relation for the empirical estimates on a training sequence S). Therefore, to upper-bound $R(G_Q)$, we simply need to find the largest value of $e(G_Q)+c\cdot a(G_Q)$ permitted by Theorem 10 given that we know $e_S(G_Q)$ and $a_S(G_Q)$. Consequently, Theorem 10 has the following corollary.

Corollary 11 *For any $\delta\in(0,1]$, for any reconstruction function mapping compression sequences and messages to classifiers that may abstain, for any $T\in(\mathcal{X}\times\mathcal{Y})^m$ and for any prior P_T on $I\times\mathcal{M}_T$, we have*

$$\Pr_{S\sim D^m}\left(\forall Q \text{ on } I\times\mathcal{M}_S\colon R(G_Q) \;\leq\; \sup\left\{e+ca \;\middle|\; \mathrm{kl}(a_S(G_Q),e_S(G_Q)\|a,e) \;\leq\; \frac{1}{m-d_{\overline{Q}}}\left[\mathrm{KL}(\overline{Q}\,\|P_S)+\ln\frac{(m+1)(m+2)}{2\delta}\right]\right\}\right)\geq 1-\delta\,.$$

To upper-bound the risk of the majority-vote B_Q with Theorem 10, we need to redefine the risk $R(B_Q)$ (in terms of the loss function ℓ defined above) and related it to $e(G_Q)$ and $a(G_Q)$. For this task, let us adopt the convention that $B_Q(\mathbf{x})$ abstains of predicting the label of $\mathbf{x}$ whenever the Q-weight of classifiers predicting $+1$ is equal to the Q-weight of classifiers predicting -1 (this includes the case when all the classifiers having non-zero posterior weight abstain).

Similarly as our definition of $e(G_Q)$, let $e(B_Q)$ denote the probability that B_Q predicts incorrectly the label of $\mathbf{x}$ on a random draw of $(\mathbf{x},y)$. Furthermore, let $e_{(\mathbf{x},y)}(G_Q)$ denote the probability that G_Q predicts incorrectly the label of $(\mathbf{x},y)$ and let $a_{(\mathbf{x},y)}(G_Q)$ denote the probability that G_Q abstains on $(\mathbf{x},y)$, that is,

$$\begin{aligned} e_{(\mathbf{x},y)}(G_Q) &\stackrel{\text{def}}{=} \mathop{\mathbf{E}}_{h\sim Q} I(h(\mathbf{x}) \neq y \wedge h(\mathbf{x}) \neq 0) \\ a_{(\mathbf{x},y)}(G_Q) &\stackrel{\text{def}}{=} \mathop{\mathbf{E}}_{h\sim Q} I(h(\mathbf{x}) = 0)\,. \end{aligned}$$

Similarly, let $e_{(\mathbf{x},y)}(B_Q) = 1$ iff B_Q predicts incorrectly the label of $(\mathbf{x},y)$. Therefore,

$$e_{(\mathbf{x},y)}(B_Q) = 1 \iff e_{(\mathbf{x},y)}(G_Q) > \frac{1 - a_{(\mathbf{x},y)}(G_Q)}{2}\,.$$

Hence,

$$\begin{aligned} e(B_Q) &\stackrel{\text{def}}{=} \mathop{\mathbf{E}}_{(\mathbf{x},y)\sim D} e_{(\mathbf{x},y)}(B_Q) \\ &= \mathop{\mathbf{E}}_{(\mathbf{x},y)\sim D} I\left(e_{(\mathbf{x},y)}(G_Q) > \frac{1 - a_{(\mathbf{x},y)}(G_Q)}{2}\right) \\ &= \Pr_{(\mathbf{x},y)\sim D}\left(2e_{(\mathbf{x},y)}(G_Q) + a_{(\mathbf{x},y)}(G_Q) > 1\right) \\ &< 2e(G_Q) + a(G_Q)\,, \end{aligned}$$

where, for the last line, we have used Markov's inequality for the non-negative random variable $2e_{(\mathbf{x},y)}(G_Q) + a_{(\mathbf{x},y)}(G_Q)$ with expectation $2e(G_Q) + a(G_Q)$.

Since $R(B_Q) = e(B_Q) + c \cdot a(B_Q)$, we can obtain an upper bound on $R(B_Q)$ by upper-bounding $a(B_Q)$. However,

$$a(B_Q) = \Pr_{(\mathbf{x},y)\sim D}\left(e_{(\mathbf{x},y)}(G_Q) = \frac{1 - a_{(\mathbf{x},y)}(G_Q)}{2}\right).$$

Since Theorem 10 gives non control on the domain of $(e(G_Q), a(G_Q))$ that is bounded with high probability, we cannot use it to find a tight upper-bound for $a(B_Q)$. Therefore, since the loss c of abstaining is at most 1, we will simply use

$$\begin{aligned} R(B_Q) &\leq e(B_Q) + a(B_Q) = \mathop{\mathbf{E}}_{(\mathbf{x},y)\sim D} I\left(e_{(\mathbf{x},y)}(G_Q) \geq \frac{1 - a_{(\mathbf{x},y)}(G_Q)}{2}\right) \\ &= \Pr_{(\mathbf{x},y)\sim D}\left(2e_{(\mathbf{x},y)}(G_Q) + a_{(\mathbf{x},y)}(G_Q) \geq 1\right) \leq 2e(G_Q) + a(G_Q)\,, \end{aligned}$$

where we have, once again, used Markov's inequality. [8] Consequently, we have the following corollary to bound $R(B_Q)$.

8. We might think that this upper bound for $R(B_Q)$ is worse than the the upper bound of $2R(G_Q)$ for classifiers that cannot abstain. However, the two upper bounds coincides whenever the cost c of abstaining is $1/2$ or, equivalently, if we force the abstaining classifiers to predict and if their predictions are correct with probability $1/2$.

Corollary 12 *For any $\delta \in (0,1]$, for any reconstruction function mapping compression sequences and messages to classifiers that may abstain, for any $T \in (\mathcal{X} \times \mathcal{Y})^m$ and for any prior P_T on $I \times \mathcal{M}_T$, we have*

$$\Pr_{S\sim D^m}\left(\forall Q \text{ on } I\times\mathcal{M}_S\colon R(B_Q) \;\le\; \sup\left\{2e+a \;\middle|\; \mathrm{kl}(a_S(G_Q),e_S(G_Q)\|a,e) \right.\right.$$
$$\left.\left. \le\; \frac{1}{m-d_{\overline{Q}}}\left[\mathrm{KL}(\overline{Q}\|P_S)+\ln\frac{(m+1)(m+2)}{2\delta}\right]\right\}\right) \ge 1-\delta\,.$$

Note that the values of e and a for which the supremum in Corollary 11 and 12 are attained are generally not upper bounds of both $e(G_Q)$ and $a(G_Q)$. Consequently, the risk bound given by Corollary 11 and 12 are tighter than those we would have obtained by bounding $e(G_Q)$ and $a(G_Q)$ separately.

8.1 Reduced Coulomb Energy Networks

Corollaries 8 and 11 can be used to bound the risk of stochastic averages and majority-votes of sample-compressed classifiers that can abstain. The reduced Coulomb energy (RCE) network (see Reilly et al. 1982 and Duda et al. 2000) provides a simple example of such a majority-vote. Indeed, a RCE network is basically a majority-vote of single balls. As for the SCM case, each ball is described by a training example called a *center* $(\mathbf{x}_c, y_c)$, and another training example called a *border* $(\mathbf{x}_b, y_b)$. Given any metric d, the output $h(\mathbf{x})$ on any input example $\mathbf{x}$ of a ball is given by y_c if $d(\mathbf{x},\mathbf{x}_c) \le d(\mathbf{x},\mathbf{x}_b)$, otherwise (if $d(\mathbf{x},\mathbf{x}_c) > d(\mathbf{x},\mathbf{x}_b)$) it *abstains* of predicting a class label. Hence, each sample-compressed classifier has here a compression sequence $S_{\mathbf{i}}$ of only two examples. Given $S_{\mathbf{i}}$, the message string (which consists here of a single bit) just specifies which of the two examples of $S_{\mathbf{i}}$ is the center.

Consequently, the prior $P_I(\mathbf{i})$ will be non-zero only for $|\mathbf{i}| = 2$ and distributed uniformly over all pairs of (distinct) indices. The posterior $Q_I(\mathbf{i})$ will also be non-zero only for $|\mathbf{i}| = 2$ but only balls selected by the RCE network learning algorithm, described in Reilly et al. (1982) and Duda et al. (2000), will give pairs of indices of non-zero posterior weight. The message-part of the prior, $P_{\mathcal{M}(S_{\mathbf{i}})}(\sigma)$, assigns equal probability to the two possible single-bit messages and the message-part of the posterior, $Q_{\mathcal{M}(S_{\mathbf{i}})}(\sigma)$, assigns a weight of one to the single-bit message that is actually used with the two-example compression sequence $S_{\mathbf{i}}$. With this form for the prior and the posterior, Corollary 8 provides a tight bound for the risk of the stochastic average G_Q. However, the empirical error rate $e_S(G_Q)$ may be large on some S for simple classifiers that are constructed from only two examples in the RCE network. Hence, since $e_S(G_Q)$ may be large, Corollary 11 may only provide a loose bound for the majority-vote B_Q due to the looseness involved in upper-bounding $e(B_Q)$ by $2e(G_Q)+a(G_Q)$.

9. Conclusion

We have derived a PAC-Bayes theorem for the sample-compression setting where each classifier is described by a compression subset of the training data and a message string of additional information. We have emphasized that many learning algorithms are producing classifiers that are well-described within this setting. We have seen that the PAC-Bayes theorem for the sample-compression setting reduces to the PAC-Bayes theorem of Seeger (2002) and Langford (2005) in the

usual data-independent setting when classifiers are represented only by data-independent message strings (or parameters taken from a continuous set). For posteriors having all their weights on a single sample-compressed classifier, the general risk bound reduces to a bound similar to the tight sample-compression bound of Laviolette et al. (2005). The PAC-Bayes risk bound of Theorem 3 is, however, valid for sample-compressed Gibbs classifiers with arbitrary posteriors. Moreover, we have seen that the risk bound supports the strategy of randomizing the predictions over several sample-compressed classifiers instead of predicting with a single sample-compressed classifier. Indeed, a stochastic Gibbs classifier defined on a posterior over several sample-compressed classifiers can have a smaller risk bound than any such single (deterministic) sample-compressed classifier. Finally, to obtain a performance guarantee for many "rule-based systems" and RCE networks, we have generalized the PAC-Bayes theorem to the case where each sample-compressed classifier in the ensemble can abstain of predicting a class label.

Since the risk bounds derived in this paper are tight for stochastic averages of classifiers, it is hoped that they will be effective at guiding learning algorithms for choosing the optimal tradeoff between the empirical risk, the sample compression set size, and the "distance" between the prior and the posterior. However, given an ensemble of classifiers, we usually prefer to predict with the majority-vote B_Q instead of the stochastic average G_Q. In these cases, the PAC-Bayesian guarantee for B_Q only comes indirectly through the inequality $R(B_Q) \leq 2R(G_Q)$ (for ensemble of classifiers that cannot abstain). This is clearly inappropriate for many extensively-used learning algorithms, such as Adaboost (Freund and Schapire, 1997), that produce majority-votes having a large underlying $R(G_Q)$ and a very small $R(B_Q)$. Finding better guarantees in these circumstances, along the lines proposed by Lacasse et al. (2007) and Germain et al. (2007), is therefore an important open problem in machine learning.

Acknowledgments

Work supported by NSERC Discovery grants 262067 and 122405.

References

Leo Breiman. Bagging predictors. *Machine Learning*, 24:123–140, 1996.

Olivier Catoni. A PAC-Bayesian approach to adaptive classification. Thecnical report, Université Paris 6, 2004.

Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory*, chapter 12. Wiley, 1991.

Richard O. Duda, Peter E. Hart, and David G. Stork. *Pattern Classification*. Wiley, 2000.

Sally Floyd and Manfred Warmuth. Sample compression, learnability, and the Vapnik-Chervonenkis dimension. *Machine Learning*, 21(3):269–304, 1995.

Yoav Freund and Robert E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55:119–139, 1997.

Pascal Germain, Alexandre Lacasse, Francois Laviolette, and Mario Marchand. A pac-bayes risk bound for general loss functions. In B. Schölkopf, J. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems 19*. MIT Press, Cambridge, MA, 2007.

Thore Graepel, Ralf Herbrich, and John Shawe-Taylor. PAC-Bayesian compression bounds on the prediction error of learning algorithms for classification. *Machine Learning*, 59:55–76, 2005.

Ralf Herbrich, Thore Graepel, and Colin Campbell. Bayes point machines. *Journal of Machine Learning Research*, 1:245–279, 2001.

Alexandre Lacasse, Francois Laviolette, Mario Marchand, Pascal Germain, and Nicolas Usunier. PAC-Bayes bounds for the risk of the majority vote and the variance of the Gibbs classifier. In B. Schölkopf, J. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems 19*. MIT Press, Cambridge, MA, 2007.

John Langford. Tutorial on practical prediction theory for classification. *Journal of Machine Learning Research*, 6:273–306, 2005.

John Langford and John Shawe-Taylor. PAC-Bayes & margins. In S. Thrun S. Becker and K. Obermayer, editors, *Advances in Neural Information Processing Systems 15*, pages 423–430. MIT Press, Cambridge, MA, 2003.

François Laviolette and Mario Marchand. PAC-Bayes risk bounds for sample-compressed Gibbs classifiers. *Proceedings of the 22nth International Conference on Machine Learning (ICML 2005)*, pages 481–488, 2005.

François Laviolette, Mario Marchand, and Mohak Shah. Margin-sparsity trade-off for the set covering machine. *Proceedings of the* 16^{th} *European Conference on Machine Learning (ECML 2005); Lecture Notes in Artificial Intelligence*, 3720:206–217, 2005.

François Laviolette, Mario Marchand, and Mohak Shah. A PAC-Bayes approach to the set covering machine. *Proceedings of the 2005 conference on Neural Information Processing Systems (NIPS 2005)*, 2006.

Nicholas Littlestone and Manfred K. Warmuth. Relating data compression and learnability. Technical report, University of California Santa Cruz, Santa Cruz, CA, 1986.

Mario Marchand and John Shawe-Taylor. The set covering machine. *Journal of Machine Learning Reasearch*, 3:723–746, 2002.

Mario Marchand and Marina Sokolova. Learning with decision lists of data-dependent features. *Journal of Machine Learning Reasearch*, 6:427–451, 2005.

David McAllester. Some PAC-Bayesian theorems. *Machine Learning*, 37:355–363, 1999.

David McAllester. PAC-Bayesian stochastic model selection. *Machine Learning*, 51:5–21, 2003a.

David McAllester. Simplified PAC-Bayesian margin bounds. *Proceedings of the 16th Annual Conference on Learning Theory, Lecture Notes in Artificial Intelligence*, 2777:203–215, 2003b.

Douglas L. Reilly, Leon N. Cooper, and Charles Elbaum. A neural model for category learning. *Biological Cybernetics*, 45:35–41, 1982.

Ronald L. Rivest. Learning decision lists. *Machine Learning*, 2:229–246, 1987.

Matthias Seeger. PAC-Bayesian generalization bounds for gaussian processes. *Journal of Machine Learning Research*, 3:233–269, 2002.

Matthias Seeger. Bayesian gaussian process models: PAC-Bayesian generalisation error bounds and sparse approximations. *PhD Thesis, University of Edinburgh*, 2003.

Leslie G. Valiant. A theory of the learnable. *Communications of the Association of Computing Machinery*, 27(11):1134–1142, November 1984.

Journal of Machine Learning Research 8 (2007) 1489-1517 Submitted 7/06; Revised 3/07; Published 7/07

On the Effectiveness of Laplacian Normalization for Graph Semi-supervised Learning

Rie Johnson RIE1@US.IBM.COM
IBM T.J. Watson Research Center
Hawthorne, NY 10532, USA

Tong Zhang TZHANG@YAHOO-INC.COM
Yahoo! Inc.
New York City, NY 10011, USA

Editor: Charles Elkan

Abstract

This paper investigates the effect of Laplacian normalization in graph-based semi-supervised learning. To this end, we consider multi-class transductive learning on graphs with Laplacian regularization. Generalization bounds are derived using geometric properties of the graph. Specifically, by introducing a definition of graph cut from learning theory, we obtain generalization bounds that depend on the Laplacian regularizer. We then use this analysis to better understand the role of graph Laplacian matrix normalization. Under assumptions that the cut is small, we derive near-optimal normalization factors by approximately minimizing the generalization bounds. The analysis reveals the limitations of the standard degree-based normalization method in that the resulting normalization factors can vary significantly within each connected component with the same class label, which may cause inferior generalization performance. Our theory also suggests a remedy that does not suffer from this problem. Experiments confirm the superiority of the normalization scheme motivated by learning theory on artificial and real-world data sets.

Keywords: transductive learning, graph learning, Laplacian regularization, normalization of graph Laplacian

1. Introduction

Graph-based methods, such as spectral embedding, spectral clustering, and semi-supervised learning, have drawn much attention in the machine learning community. While various ideas have been proposed based on different intuitions, only recently have there been theoretical studies trying to understand why these methods work.

In spectral clustering, a traditional starting point is to find a partition of a graph that minimizes a certain definition of "graph cut" that quantifies the quality of the partition. The cut is the objective one attempts to minimize. Spectral methods can then be derived as a certain continuous relaxation that approximately solves the "graph cut" problem. Based on various intuitions and heuristics, various definitions of cuts have been proposed in the literature (for example, Shi and Malik, 2000, Ding et al., 2001, among others). In order to understand such methods, we need to ask the following two questions. First what is the quality of the relaxation approach as an approximation method to solve the original "graph cut" problem. Second, and more importantly, we need to understand why one should optimize one definition of "cut" instead of other alternatives. In the literature, different arguments and intuitions have been proposed to justify different choices. However, without a more

universally acceptable criterion, it is difficult to argue that one cut definition is better than another just based on heuristics. If a universally agreeable standard does exist, then one should focus on that criterion instead of an artificially defined cut problem.

For example, in the context of spectral clustering, there are two well-known types of graph cut, the ratio cut (Hagen and Kahng, 1992) and the normalized cut (Shi and Malik, 2000). Approximate optimization of the ratio cut leads to eigenvector computation of the *unnormalized* graph Laplacian matrix (which we will define later), and that of the normalized cut involves the *normalized* graph Laplacian matrix (normalized using node degrees). Although a number of empirical studies indicate that the normalized cut often leads to better clustering results, there isn't any direct theoretical proof except for some implicit evidence. As another example, the definition of graph Laplacian in the spectral graph theory in Chung (1998) is normalized, but that is for graph theoretical reasons instead of statistical reasons. Specifically, the normalized Laplacian allows easier translation of results from differential geometry, and it also allows consistent relations with conductance on Markov chains. The compatibility of continuous Laplacian on manifold and normalized graph Laplacian was also noted by von Luxburg et al. (2005) from a different perspective. Similarly, some analysis of spectral clustering employs normalized cut (Meilă et al., 2005, for example) as that makes derivation easier. These can be regarded as implicit evidence for preferring the normalized Laplacian over the unnormalized Laplacian. However, it has not been directly proved that such degree-based normalization (corresponding to normalized cut) should improve performance.

In order to understand this issue better, we take a different approach in this paper. Observe that for spectral clustering applications, there are often pre-defined (but unknown) clusters (classes). In this setting, the goal is to find such classes either using unsupervised or semi-supervised methods. Therefore for such problems, a universally agreeable standard is to find clusters that overlap significantly with the underlying class labels. That is, instead of using any artificially defined cut, we should design an algorithm to minimize the classification error. This is the criterion we focus on in this paper. We will see that normalization comes naturally into the generalization analysis we develop. By optimizing the corresponding generalization bounds, we seek to obtain a better understanding of the effect of Laplacian normalization.

In spectral clustering or graph based semi-supervised learning, one starts with *similarity graphs* that link similar data points. For example, one may connect data points that are close in the feature space to form a k-nearest neighbor graph. If the graph is fully connected within each class and disconnected between the classes, then appropriate cut minimization leads to perfect classification. It was proposed in Ng et al. (2002) that one may first project these data points into the eigenspace corresponding to the largest eigenvalues of a normalized adjacency matrix of the graph and then use the standard k-means method to perform clustering. The basic motivation is quite similar to that of Shi and Malik (2000). It can be shown that in the ideal case (each class forms a connected subgraph, and there is no inter-class edge), points in the same cluster will be mapped into a single point in the reduced eigenspace, while points in different clusters will be mapped to different points. This implies that for clustering the distance in the reduced space is better than the original distance. A natural question in this setting is how to design a distance function that leads to better clustering. While the argument in Ng et al. (2002) gives a satisfactory answer in the idealized case, it is far less clear what happens in general. One approach to address this problem is to learn a distance metric that can lead to more desirable clustering results from a set of labeled examples (for example, as in Xing et al., 2003). The inner product associated with a distance metric can be viewed as a kernel,

and the kernel fully determines the outcome of the k-means algorithm. Therefore this approach can also be viewed as designing a kernel optimal for clustering.

Closely related to clustering, one may also consider kernel design methods in semi-supervised learning using a discriminative method such as SVM (e.g., Lanckriet et al., 2004). In this setting, the change of the distance metric becomes a change of the underlying kernel. If the kernel is induced from a graph, then one may formulate semi-supervised learning directly on the graph; for example, see Belkin and Niyogi (2004), Szummer and Jaakkola (2002), Zhou et al. (2004) and Zhu et al. (2003). In these studies, the kernel is induced from the adjacency matrix $\mathbf{W}$ whose (i, j)-entry is the weight of edge (i, j). $\mathbf{W}$ is often normalized by $\mathbf{D}^{-1/2}\mathbf{W}\mathbf{D}^{-1/2}$ as in Chung (1998), Shi and Malik (2000), Ng et al. (2002), Zhou et al. (2004), where $\mathbf{D}$ is a diagonal matrix whose (j, j)-entry is the degree of the j-th node, but sometimes not (Belkin and Niyogi, 2004, Zhu et al., 2003). Although such normalization can significantly affect the performance, the issue has not been carefully studied from the learning theory perspective. The relationship of kernel design and graph learning was investigated in Zhang and Ando (2006), where it was argued that quadratic regularization-based graph learning can be regarded as kernel design in the spectral domain. That is, one keeps the kernel eigenvectors and modifies the corresponding eigenvalues. Moreover if input data are corrupted with noise, then such spectral kernel design can help to improve classification performance. However, the analysis does not explain why normalization of the adjacency matrix $\mathbf{W}$ is useful for practical purposes.

Our goals here are twofold. First we present a model for transductive learning on graphs and develop a margin analysis for multi-class graph learning. We then analyze graph learning using graph properties such as graph-cut and a concept we call *pure subgraph*. The analysis naturally employs quantities formalizing the standard graph-learning assumption that well connected nodes are likely to have the same label. Second, we use this analysis to obtain a better understanding of normalizing the Laplacian matrix ($\mathbf{D} - \mathbf{W}$) in graph semi-supervised learning. As mentioned above, normalization has been commonly practiced and appears to be useful, but there hasn't been any solid theoretical justification on why it should be useful. Our analysis addresses this issue from a learning theoretical point of view, and reveals a limitation of the standard degree-based normalization scheme. We then propose a remedy based on the learning theory results and use experiments to demonstrate that the remedy leads to improved classification performance.

This paper expands on our preliminary results reported in Ando and Zhang (2007).

2. Transductive Learning Model

We consider the following multi-category transductive learning model defined on a graph. Let $V = \{v_1, \ldots, v_m\}$ be a set of m nodes, and let $\mathcal{Y}$ be a set of K possible output values. Assume that each node v_j is associated with an output value $y_j \in \mathcal{Y}$, which we are interested in predicting. In order to do so, we randomly draw a set of n indices $Z_n = \{j_i : 1 \leq i \leq n\}$ from $\{1, \ldots, m\}$ uniformly and without replacement. We manually label the n nodes v_{j_i} with labels $y_{j_i} \in \mathcal{Y}$, and then automatically label the remaining $m - n$ nodes. The goal is to estimate the labels on the remaining $m - n$ nodes as accurately as possible.

In this paper, we shall assume that the labels $y = [y_1, \ldots, y_m]$ are deterministic. However, the analysis can also be applied if we have random labels. In the transductive learning setting considered in this paper, we may assume that we are given a single random draw $y = [y_1, \ldots, y_m]$, which we fix. With this fixed y vector, we are interested in the performance of reconstructing it from a

subset of labels. This formulation is more appropriate for problems such as classification on graphs considered here.

In modern machine learning, instead of estimating the labels y_j directly, y_j is often encoded into a vector in R^K, so that the problem becomes that of generating an estimation vector $\mathbf{f}_j = [f_{j,1},\ldots,f_{j,K}] \in R^K$, which can then be used to recover the label y_j. In multi-category classification with K classes $\mathcal{Y} = \{1,\ldots,K\}$, we encode each $y_j = k \in \mathcal{Y}$ as $e_k \in R^K$, where e_k is a vector of zero entries except for the k-th entry being one. Given a function $\mathbf{f}_j = [f_{j,1},\ldots,f_{j,K}] \in R^K$ (which is intended to approximate e_{y_j}), we decode the corresponding label estimation $\hat{y}_j$ as:

$$\hat{y}_j = \hat{y}(\mathbf{f}_j) = \arg\max_k \left\{ f_{j,k} : k = 1,\ldots,K \right\}.$$

If the true label is y_j, then the corresponding classification error is:

$$\mathbf{err}(\mathbf{f}_j, y_j) = I(\hat{y}(\mathbf{f}_j) \neq y_j),$$

where we use $I(\cdot)$ to denote the set indicator function.

In order to estimate the concatenated vector $\mathbf{f} = [\mathbf{f}_j] = [f_{j,k}] \in R^{mK}$ from only a subset of labeled nodes, we have to impose restrictions on possible values of $\mathbf{f}$. In this paper, we consider restrictions defined through a quadratic regularizer of the following form:

$$\mathbf{f}^T \mathbf{Q}_\mathbf{K} \mathbf{f} = \sum_{k=1}^{K} \mathbf{f}_{\cdot,k}^T \mathbf{K}^{-1} \mathbf{f}_{\cdot,k},$$

where $\mathbf{K} \in R^{m\times m}$ is a positive definite kernel matrix and $\mathbf{f}_{\cdot,k} = [f_{1,k},\ldots,f_{m,k}] \in R^m$. That is, the predictive vector for each class k is regularized separately. We assume that the kernel matrix $\mathbf{K}$ is full-rank. We will consider the kernel matrix induced by the graph Laplacian, which we shall define later in the paper. Note that we use the bold symbol $\mathbf{K}$ to denote the kernel matrix and the regular capitalized K to denote the number of classes.

Given a vector $\mathbf{f} \in R^{mK}$, the accuracy of its component $\mathbf{f}_j = [f_{j,1},\ldots,f_{j,K}] \in R^K$ is measured by a loss function $\phi(\mathbf{f}_j, y_j)$. Our learning method attempts to minimize the empirical risk on the set Z_n of n labeled training nodes, subject to $\mathbf{f}^T \mathbf{Q}_\mathbf{K} \mathbf{f}$ being small:

$$\hat{\mathbf{f}}(Z_n) = \arg\min_{\mathbf{f}\in R^{mK}} \left[\frac{1}{n} \sum_{j\in Z_n} \phi(\mathbf{f}_j, y_j) + \lambda \mathbf{f}^T \mathbf{Q}_\mathbf{K} \mathbf{f} \right]. \tag{1}$$

where $\lambda > 0$ is an appropriately chosen regularization parameter.

In this paper, we focus on a special class of loss functions of the form $\phi(\mathbf{f}_j, y_j) = \sum_{k=1}^K \phi_0(f_{j,k}, \delta_{k,y_j})$, where $\delta_{a,b}$ is the delta function defined as: $\delta_{a,b} = 1$ when $a = b$ and $\delta_{a,b} = 0$ otherwise. In addition, we introduce the following assumption for convenience.

Assumption 1 *Let $\phi(\mathbf{f}_j, y_j) = \sum_{k=1}^K \phi_0(f_{j,k}, \delta_{k,y_j})$ in (1), where $\mathbf{f}_j = [f_{j,1},\ldots,f_{j,K}] \in R^K$. Assume that there exist positive constants a, b, and c such that*

- *$\phi_0(x,y)$ is non-negative and convex in x.*
- *When $y = 0, 1$, and $\phi_0(x,y) \leq a$, $|\nabla_1 \phi_0(x,y)| \leq b$, where $\nabla_1 \phi_0(x,y)$ denotes a sub-gradient of $\phi_0(x,y)$ with respect to x.*

- $c = \inf\{x : \phi_0(x,1) \le a\} - \sup\{x : \phi_0(x,0) \le a\}$.

The formulation presented here corresponds to the one-versus-all method for multi-category classification, and standard binary loss functions such as least squares, logistic regression, and SVMs can be used. For the SVM loss function $\phi_0(x,y) = \max(0, 1-(2x-1)(2y-1))$, we may take $a = 0.5$, $b = 2$, and $c = 0.5$. For the least squares function $\phi_0(x,y) = (x-y)^2$, we may choose $a = 1/16$, $b = 0.5, c = 0.5$.

We are interested in the generalization behavior of (1) compared to a properly defined optimal regularized risk. This type of inequality is often referred to as "oracle inequality" in the learning theory literature and is particularly useful for analyzing the quality of the underlying learning method. The following theorem gives an oracle inequality, and its proof can be found in Appendix A.

Theorem 1 *Consider (1) with loss function ϕ satisfying Assumption 1. Then $\forall p > 0$, the expected generalization error of the learning method (1) over the training samples Z_n, uniformly drawn without replacement from graph nodes $\{1,\ldots,m\}$, can be bounded by:*

$$\mathbf{E}_{Z_n} \frac{1}{m-n} \sum_{j \in \bar{Z}_n} \mathbf{err}(\hat{\mathbf{f}}_j(Z_n), y_j) \le \frac{1}{a} \inf_{\mathbf{f} \in R^{mK}} \left[\frac{1}{m} \sum_{j=1}^{m} \phi(\mathbf{f}_j, y_j) + \lambda \mathbf{f}^T \mathbf{Q}_{\mathbf{K}} \mathbf{f} \right] + \left(\frac{b \mathbf{tr}_p(\mathbf{K})}{\lambda n c} \right)^p,$$

where $\bar{Z}_n = \{1,\ldots,m\} - Z_n$,

$$\mathbf{tr}_p(\mathbf{K}) = \left(\frac{1}{m} \sum_{j=1}^{m} \mathbf{K}_{j,j}^p \right)^{1/p},$$

and $\mathbf{K}_{j,j}$ denotes the j-th diagonal entry of matrix $\mathbf{K}$.

If we take $p = 1$ in Theorem 1, then the bound depends on the trace of matrix $\mathbf{K}$: $\mathbf{tr}(\mathbf{K}) = m\mathbf{tr}_1(\mathbf{K})$. The trace of a kernel matrix has been employed in a number of previous studies to characterize generalization ability of kernel methods. The generalized quantity in Theorem 1 with $p \neq 1$ has non-trivial consequences which we will investigate in the paper.

Although we consider a specific form of loss function in this paper, one can obtain similar bounds with other forms of loss functions such as $\phi(\mathbf{f}_j, y_j) = \sup_{k \neq y_j} \phi_0(f_{j,y_j} - f_{j,k})$. What is important in our analysis are the two quantities $\mathbf{f}^T \mathbf{Q}_{\mathbf{K}} \mathbf{f}$ and $\mathbf{tr}_p(\mathbf{K})$ that determine the generalization performance. We will focus on the interpretation of these quantities.

3. Margin and Graph Cut

Consider an undirected graph $G = (V,E)$ defined on the nodes $V = \{v_j : j = 1,\ldots,m\}$, with edges $E \subset \{1,\ldots,m\} \times \{1,\ldots,m\}$, and weights $w_{j,j'} \ge 0$ associated with edges $(j,j') \in E$. For simplicity, we assume that $(j,j) \notin E$ and $w_{j,j'} = 0$ when $(j,j') \notin E$. Let $\deg_j(G) = \sum_{j'=1}^{m} w_{j,j'}$ be the degree of node j of graph G. We consider the following definition of normalized Laplacian.

Definition 2 *Consider a graph $G = (V,E)$ of m nodes with weights $w_{j,j'}$ $(j,j' = 1,\ldots,m)$. The unnormalized Laplacian matrix $\mathcal{L}(G) \in R^{m \times m}$ is defined as: $\mathcal{L}_{j,j'}(G) = -w_{j,j'}$ if $j \neq j'$; $\deg_j(G)$ otherwise. Given m scaling factors $\mathbf{S}_j$ $(j = 1,\ldots,m)$, let $\mathbf{S} = \mathrm{diag}(\{\mathbf{S}_j\})$. The $\mathbf{S}$-normalized Laplacian matrix is defined as: $\mathcal{L}_{\mathbf{S}}(G) = \mathbf{S}^{-1/2} \mathcal{L}(G) \mathbf{S}^{-1/2}$. The corresponding regularization is based on:*

$$\mathbf{f}_{\cdot,k}^T \mathcal{L}_{\mathbf{S}}(G) \mathbf{f}_{\cdot,k} = \frac{1}{2} \sum_{j,j'=1}^{m} w_{j,j'} \left(\frac{f_{j,k}}{\sqrt{\mathbf{S}_j}} - \frac{f_{j',k}}{\sqrt{\mathbf{S}_{j'}}} \right)^2.$$

A common choice of $\mathbf{S}$ is $\mathbf{S} = \mathbf{I}$, corresponding to regularizing with the unnormalized Laplacian $\mathcal{L}$. The idea is natural: we assume that the predictive values $f_{j,k}$ and $f_{j',k}$ should be close when $(j, j') \in E$ with a strong link. Another common choice is to normalize by $\mathbf{S}_j = \deg_j(G)$, as in Ng et al. (2002), Shi and Malik (2000), Zhou et al. (2004) and Chung (1998), which we refer to as degree-based normalization. At first sight, the need for normalization is not immediately clear. However, as we will show later, normalization using appropriate scaling factors can improve performance.

3.1 Generalization Analysis Using Graph-Cut

We will adapt Theorem 1 in Section 2 to analyze graph learning using graph properties such as graph-cut. We now introduce a learning theoretical definition of $\mathbf{S}$-normalized graph cut as follows.

Definition 3 *Given label $y = \{y_j\}_{j=1,\ldots,m}$ on V, we define the cut for the $\mathbf{S}$-normalized Laplacian $\mathcal{L}_{\mathbf{S}}$ in Definition 2 as:*

$$\mathbf{cut}(\mathcal{L}_{\mathbf{S}}, y) = \sum_{j,j':y_j \neq y_{j'}} \frac{w_{j,j'}}{2}\left(\frac{1}{\mathbf{S}_j} + \frac{1}{\mathbf{S}_{j'}}\right) + \sum_{j,j':y_j = y_{j'}} \frac{w_{j,j'}}{2}\left(\frac{1}{\sqrt{\mathbf{S}_j}} - \frac{1}{\sqrt{\mathbf{S}_{j'}}}\right)^2.$$

Note that unlike typical graph-theoretical definitions of graph-cut in the literature, the learning theoretical definition of cut not only penalizes a normalized version of between-class edge weights, but also penalizes within-class edge weights when such an edge connects two nodes with different scaling factors. This difference has important consequences, which we will investigate later in the paper. For unnormalized Laplacian, the second term on the right hand side of Definition 3 vanishes, which means that it only penalizes weights corresponding to edges connecting nodes with different labels. In this case, the learning theoretical definition corresponds to the graph-theoretical definition: $\mathbf{cut}(\mathcal{L}, y) = \sum_{j,j':y_j \neq y_{j'}} w_{j,j'}$.

It is worth noting that in our framework, cut is used to indicate the absolute amount of perturbation from the idealized case with zero-cut. In spectral clustering, the absolute cut is often scaled and the resulting quantity is used as a quality measure for the clusters. In comparison, our quality measure is always to minimize the classification error. In particular, the unnormalized Laplacian is used in spectral clustering to approximately minimize the *ratio cut* $= \sum_{j\in A, j'\in B} w_{j,j'}/(|A| \cdot |B|)$ (Hagen and Kahng, 1992) instead of $\sum_{j\in A, j'\in B} w_{j,j'}$. The scaling in the ratio cut (when $K = 2$) corresponds to the normalization of a specific encoding of the target vectors ($\mathbf{f}_{\cdot,k} \in R^m$ which encodes the true output values in our setting); it is used in the computation of the second smallest eigenvalue of the unnormalized Laplacian. Our analysis throughout this paper assumes unmodified target vectors with components taking values in $\{0, 1\}$, which differs from Hagen and Kahng (1992). As such, the discrepancy of cut definition is merely due to a difference in representation of the target vectors, and both lead to the same unnormalized Laplacian matrix algorithmically. Although our definition is motivated by classification problems, it may also be useful for clustering problems because there is a strong relationship between clustering and transductive classification (where class labels indicate the underlying clusters). This means that the normalization method we propose in the paper might be useful in spectral clustering as well.

Using the learning theoretical graph-cut definition, we can obtain a generalization result for the estimator in (1) with $\mathbf{K}$ defined as follows:

$$\mathbf{K}^{-1} = \alpha \mathbf{S}^{-1} + \mathcal{L}_{\mathbf{S}}(G) = \mathbf{S}^{-1/2}(\alpha \mathbf{I} + \mathcal{L}(G))\mathbf{S}^{-1/2}\,, \tag{2}$$

where $\mathbf{I}$ is the identity matrix. Note that $\alpha > 0$ is a tuning parameter to ensure that $\mathbf{K}$ is strictly positive definite. As we will see later, this parameter is important. The corresponding regularization condition is

$$\mathbf{f}^T \mathbf{Q}_{\mathbf{K}} \mathbf{f} = \sum_{k=1}^{K} \left(\alpha \sum_{j=1}^{m} \frac{f_{k,j}^2}{\mathbf{S}_j} + \frac{1}{2} \sum_{j,j'=1}^{m} \left(\frac{f_{j,k}}{\sqrt{\mathbf{S}_j}} - \frac{f_{j',k}}{\sqrt{\mathbf{S}_{j'}}} \right)^2 w_{j,j'} \right).$$

Another possibility is to let $\mathbf{K}^{-1} = \alpha \mathbf{I} + \mathcal{L}_{\mathbf{S}}(G)$. The conclusions, which we will not include in this paper, are similar to that of (2).

For simplicity, we state the generalization bound based on Theorem 1 with optimal λ. Note that in applications, λ is usually tuned through cross validation. Therefore assuming optimal λ will simplify the bound so that we can focus on the more essential characteristics of generalization performance. The following assumption is used to simplify the bound

Assumption 2 *Consider (1) with regularization condition (2), loss function ϕ satisfying Assumption 1, and assume that $\phi_0(0,0) = \phi_0(1,1) = 0$.*

It is easy to check that the conditions on the loss function in Assumption 2 hold for the least squares method (which we focus on in this paper) as well as other standard loss functions such as SVM.

Theorem 4 *Consider (1) such that Assumption 2 is satisfied. Then $\forall p > 0$, there exists a sample independent regularization parameter λ in (1) such that the expected generalization error is bounded by:*

$$\mathbf{E}_{Z_n} \frac{1}{m-n} \sum_{j \in \bar{Z}_n} \mathbf{err}(\hat{\mathbf{f}}_j(Z_n), y_j) \leq \frac{C_p(a,b,c)}{n^{p/(p+1)}} (\alpha s + \mathbf{cut}(\mathcal{L}_{\mathbf{S}}, y))^{p/(p+1)} \mathbf{tr}_p(\mathbf{K})^{p/(p+1)},$$

$$C_p(a,b,c) = (b/ac)^{p/(p+1)} (p^{1/(p+1)} + p^{-p/(p+1)}), \tag{3}$$

where $s = \sum_{j=1}^{m} \mathbf{S}_j^{-1}$.

Proof Let $f_{j,k} = \delta_{y_j,k}$. It can be easily verified that

$$\frac{1}{m} \sum_{j=1}^{m} \phi(\mathbf{f}_j, y_j) + \lambda \mathbf{f}^T \mathbf{Q}_{\mathbf{K}} \mathbf{f} = \lambda(\alpha s + \mathbf{cut}(\mathcal{L}_{\mathbf{S}}, y)).$$

Now, using this expression in Theorem 1, and then optimizing over λ, we obtain the desired inequality. ■

Note that with the least squares loss, we can take $b/ac = 16$ in Theorem 1. With a fixed p, the generalization error decreases at the rate $O(n^{-p/(p+1)})$ when the sample size n increases. This rate of convergence is faster when p increases. However in general, $\mathbf{tr}_p(\mathbf{K})$ is an increasing function of p. Therefore we have a trade-off between the two terms, and without appropriate normalization (which

we will consider later in the paper), one may prefer a smaller p in order to optimize the bound. An analysis will be provided in the next section. The bound also suggests that if we normalize $\mathbf{K}$ so that its diagonal entries $\mathbf{K}_{j,j}$ become a constant, then $\mathbf{tr}_p(\mathbf{K})$ is independent of p, and thus a larger p can be used in the bound. This motivates the idea of normalizing the diagonals of $\mathbf{K}$, which we will further investigate later in the paper. The generalization bound in Theorem 4 is closely related to the margin analysis for binary linear classification. Specifically, the right hand side can be viewed as a margin-like-quantity associated with the target function $f_{j,k} = \delta_{y_j,k}$ that separates the data. Here it is related to the concept of graph cut. Our goal is to better understand the quantity $(\alpha s + \mathbf{cut}(\mathcal{L}_{\mathbf{S}}, y))^{p/(p+1)}\mathbf{tr}_p(\mathbf{K})^{p/(p+1)}$ using graph properties, which gives better understanding of graph based learning.

In the following, we will give example applications of Theorem 4. They illustrate that theoretically it is important to tune the parameter α to achieve good performance, which is also empirically observed in our experiments.

3.2 Zero-cut and Geometric Margin Separation

We consider an application of Theorem 4 for the unnormalized Laplacian under the zero-cut assumption that each connected component of the graph has a single label. With this assumption, the task is simply to estimate what label each connected component has.

Theorem 5 *Consider (1) such that Assumption 2 is satisfied and the regularization condition is* $\mathbf{K}^{-1} = \alpha\mathbf{I} + \mathcal{L}$. *Assume that* $\mathbf{cut}(\mathcal{L}, y) = 0$, *and the graph has q connected components of sizes* $m_1 \leq \cdots \leq m_q$ ($\sum_\ell m_\ell = m$). *For all $p > 0$, let $\alpha \to 0$, and with optimal λ, we have the generalization bound*

$$\mathbf{E}_{Z_n}\frac{1}{m-n}\sum_{j\in\bar{Z}_n}\mathbf{err}(\hat{\mathbf{f}}_j, y_j) \leq \frac{C_p(a,b,c)}{n^{p/(p+1)}}\left(\sum_{\ell=1}^{q}(m/m_\ell)^{p-1}\right)^{1/(p+1)} + O(\alpha),$$

where C_p is defined in (3). In particular, we have

$$\mathbf{E}_{Z_n}\frac{1}{m-n}\sum_{j\in\bar{Z}_n}\mathbf{err}(\hat{\mathbf{f}}_j, y_j) \leq \min\left[2\sqrt{\frac{b}{ac}\cdot\frac{q}{n}}, \frac{b}{ac}\cdot\frac{m}{nm_1}\right] + O(\alpha).$$

Proof Since the graph has q connected components, $\mathcal{L}$ has q eigenvectors $\mathbf{v}_\ell$ ($\ell = 1,\ldots,q$) associated with zero-eigenvalues, where each eigenvector $\mathbf{v}_\ell$ is the indicator function of the ℓ-th connected component in the graph, that is, the j-th entry of vector $\mathbf{v}_\ell$ is 1 if j belongs to the ℓ-th connected component and 0 otherwise. It is not hard to check that as $\alpha \to 0$, $\alpha\mathbf{K} \to \sum_{\ell=1}^{q}\frac{1}{m_\ell}\mathbf{v}_\ell\mathbf{v}_\ell^T + O(\alpha)$. Therefore $\alpha\mathbf{tr}_p(\mathbf{K}) \to m^{-1/p}(\sum_{\ell=1}^{q} m_\ell^{1-p})^{1/p}$. Now, we can use Theorem 4 to obtain the first inequality. The second inequality is obtained by setting $p = 1$ and by letting $p \to \infty$ on the right hand side. ■

Under the zero-cut assumption, the generalization performance can be bounded as $O(\sqrt{q/n})$ when $\alpha \to 0$. However, we can also achieve a faster convergence rate of $O(1/n)$, although the generalization performance depends on the inverse of the smallest component size through $m/m_1 \geq$

q. This implies that we will achieve better convergence at the $O(1/n)$ level if the sizes of the components are balanced. If the component sizes are significantly different, the convergence may behave like $O(\sqrt{q/n})$.

We discuss a concrete example in which Theorem 5 is applicable. Assume that each node v_j is associated with a data point x_j that belongs to the d-dimensional unit ball $B = \{x \in R^d : \|x\|_2 \leq 1\}$. We form a graph by connecting all nodes v_j to their nearest neighbors. In particular, we may consider an ε-ball centered at each v_j: $B_j(\varepsilon) = \{x : \|x - x_j\|_2 \leq \varepsilon\}$. We then form a graph by connecting each j with all points within the ball $B_j(\varepsilon)$ and with unit weights.

We say that the data points are separable with geometric margin γ if for each node v_j the ball $B_j(\gamma)$ only contains points in class y_j. Now assume we use a ball of size $\varepsilon \leq \gamma$. In this case, $\mathbf{cut}(\mathcal{L}, y) = 0$, and there is a constant $q \leq \varepsilon^{-d}$ such that the graph has at most q connected components, and we have:

$$\mathbf{E}_{Z_n} \frac{1}{m-n} \sum_{j \in \bar{Z}_n} \mathbf{err}(\hat{\mathbf{f}}_j, y_j) \leq 2\sqrt{\frac{b}{ac} \cdot \frac{q}{n}} + O(\alpha).$$

This bound does not depend on margin γ but depends only on q, the number of connected components. So even if the margin γ is small, the bound can still be good as long as q is small. This result can be used to understand why graph based semi-supervised learning may work better than standard kernel learning. In fact, it is not possible to derive similar generalization bounds for supervised learning because one needs unlabeled data (in addition to labeled data) to define such connected components. This means that graph semi-supervised learning can take advantage of the new quantity q to characterize its generalization performance, and this quantity cannot be used by standard supervised learning.

Note that we have assumed a very specific generative model for the data. In particular, if the data are generated in a way such that the number of connected components q is small, and each connected component belongs to a single class, then graph based semi-supervised learning can work better than supervised kernel learning. If this assumption does not hold (at least approximately), then graph based learning methods may fail. However, for many practical applications, the geometric margin separation assumption does appear quite reasonable. Therefore for such problems, graph based semi-supervised learning, which can take advantage of the underlying data generation model, may become helpful.

This section only considers a special case where the graph has q connected components. In this particular situation, the learning method (1) and the analysis provided here may not be optimal. The best method is just to identify each connected component to be a cluster and then determine its label by looking at one point of the cluster. However, this idea won't generalize to graphs with components that are weakly connected. In comparison, our analysis can easily generalize to that situation, as we shall investigate in the next section.

3.3 Non-Zero Cut and Pure Components

It is often too restrictive to assume that each connected component has only one label (that is, the cut is zero). In this section, we show that similar bounds can be obtained when this data generation assumption is relaxed. We are still interested in giving a characterization of the performance of (1) in terms of properties of the graph and introduce the following definition.

Definition 6 *A subgraph $G_0 = (V_0, E_0)$ of $G = (V, E)$ is called a pure component if G_0 is connected, E_0 is induced by restricting E on V_0, and the labels y have identical values on V_0. A pure subgraph*

$G' = \cup_{\ell=1}^{q} G_\ell$ of G divides V into q disjoint sets $V = \cup_{\ell=1}^{q} V_\ell$ such that each subgraph $G_\ell = (V_\ell, E_\ell)$ is a pure component. Denote by $\lambda_i(G_\ell) = \lambda_i(\mathcal{L}(G_\ell))$ the i-th smallest eigenvalue of $\mathcal{L}(G_\ell)$.

For instance, if we remove all edges of G that connect nodes with different labels, then the resulting subgraph is a pure subgraph (though it may not be the only one). For each pure component G_ℓ, its first eigenvalue $\lambda_1(G_\ell)$ is always zero. The second eigenvalue $\lambda_2(G_\ell) > 0$ because G_ℓ is connected. This $\lambda_2(G_\ell)$ can be regarded as a measurement of how well G_ℓ is connected. We use it together with graph cut to derive a generalization bound. The proof is given in Appendix B.

Theorem 7 *Consider (1) such that Assumption 2 is satisfied. Let $G' = \cup_{\ell=1}^{q} G_\ell$ ($G_\ell = (V_\ell, E_\ell)$) be a pure subgraph of G. For all $p \geq 1$, there exist sample-independent regularization parameter λ and a fixed tuning parameter α, such that*

$$\begin{aligned}
&\mathbf{E}_{Z_n} \frac{1}{m-n} \sum_{j \in Z_n} \mathbf{err}(\hat{\mathbf{f}}_j, y_j) \\
\leq &\frac{C_p(a,b,c)}{n^{p/(p+1)}} \left(s^{1/2} \left(\sum_{\ell=1}^{q} \frac{s_\ell(p)/m}{m_\ell^p} \right)^{1/2p} + \mathbf{cut}(\mathcal{L}_{\mathbf{S}}, y)^{1/2} \left(\sum_{\ell=1}^{q} \frac{s_\ell(p)/m}{\lambda_2(G_\ell)^p} \right)^{1/2p} \right)^{2p/(p+1)},
\end{aligned}$$

where C_p is defined in (3), $m_\ell = |V_\ell|$, $s = \sum_{j=1}^{m} \mathbf{S}_j^{-1}$, and $s_\ell(p) = \sum_{j \in V_\ell} \mathbf{S}_j^p$.

Theorem 7 is a natural generalization of Theorem 5 when $p \geq 1$. It quantitatively illustrates the importance of analyzing graph learning using a partition of the original graph into well-connected pure components. The second eigenvalue $\lambda_2(G_i)$ measures how well-connected G_i is. A more intuitive quantity that measures the connectedness of graph $G = (V, E)$ is the *isoperimetric number* h_G defined as

$$h_G = \inf_{S \subset V} \sum_{j \in S, j' \in V - S} w_{j,j'} / \min(|S|, |V - S|).$$

It is well-known that $\lambda_2(G_i) \geq h_{G_i}^2 / (2 \max_j \deg_j(G_i))$ (Chung, 1998). The isoperimetric number of a graph is large when the nodes are well-connected everywhere. In particular, if $\deg_j(G)$ is of the order $|V|$, and $w_{i,j} = 1$ when $(i,j) \in E$, then for a well-connected graph, $\sum_{j \in S, j' \in V-S} w_{j,j'}$ is of the order $|S||V - S|$, and $h_G = O(|V|)$. Let G' be a well-behaved pure-subgraph of G, such that each pure component G_ℓ of G' is well-connected in the above sense. We thus have the condition

$$\lambda_2(G_\ell)/m_\ell \geq u(G')$$

for some constant $u(G')$ that does not depend on the size of the pure components (but only how well-connected each pure component is). Under this condition, we may replace $\sum_{\ell=1}^{q} m_\ell \lambda_2(G_\ell)^{-p}$ by $u(G')^{-p} \sum_{\ell=1}^{q} m_\ell^{1-p}$ in Theorem 7 and obtain a simplified bound:

$$\mathbf{E}_{Z_n} \frac{1}{m-n} \sum_{j \in Z_n} \mathbf{err}(\hat{\mathbf{f}}_j, y_j) \leq \frac{C_p(a,b,c)}{n^{p/(p+1)}} \left(\sum_{\ell=1}^{q} \frac{s_\ell(p)/m}{(m_\ell/m)^p} \right)^{1/(p+1)} \left(\sqrt{\frac{s}{m}} + \sqrt{\frac{\mathbf{cut}(\mathcal{L}_{\mathbf{S}}, y)}{u(G')m}} \right)^{2p/(p+1)},$$

where we define $u(G') = \min_\ell(\lambda_2(G_\ell)/m_\ell)$. We consider two special cases: $p = 1$ and $p \to \infty$:

$$\mathbf{E}_{Z_n} \frac{1}{m-n} \sum_{j \in Z_n} \mathbf{err}(\hat{\mathbf{f}}_j, y_j) \leq 2\sqrt{\frac{b}{ac} \cdot \frac{\sum_{\ell=1}^q (s_\ell(1)/m_\ell)}{n}} \left(\sqrt{\frac{s}{m}} + \sqrt{\frac{\mathbf{cut}(\mathcal{L}_\mathbf{S}, y)}{u(G')m}} \right), \tag{4}$$

$$\mathbf{E}_{Z_n} \frac{1}{m-n} \sum_{j \in Z_n} \mathbf{err}(\hat{\mathbf{f}}_j, y_j) \leq \frac{b}{ac} \cdot \frac{\max_\ell \max_{j \in V_\ell}(\mathbf{S}_j/m_\ell)}{n} \left(\sqrt{s} + \sqrt{\frac{\mathbf{cut}(\mathcal{L}_\mathbf{S}, y)}{u(G')}} \right)^2. \tag{5}$$

These bounds are generalizations of those in Theorem 5. Suppose that we take $\mathbf{S} = \mathbf{I}$. Then the number of pure components q affects the $O(1/\sqrt{n})$ convergence rate in (4) as $\sum_{\ell=1}^q s_\ell(1)/m_\ell = q$. If the sizes of the components are balanced, we can achieve better convergence at the $O(1/n)$ level as in (5); otherwise, the convergence may behave like $O(\sqrt{q/n})$. This observation motivates a scaling matrix $\mathbf{S}$ that compensates for the unbalanced pure component sizes, which we will investigate next.

3.4 Optimal Normalization for Near-zero-cut Partition

As discussed in the introduction, the common practice of the normalization of the adjacency matrix ($\mathbf{W}$) or the graph Laplacian ($\mathbf{D} - \mathbf{W}$) is based on degrees, which corresponds to setting $\mathbf{S} = \mathbf{D}$. Although such normalization may significantly affect the performance, to our knowledge, there is no learning theory analysis on the effect of normalization. The purpose of this section is to fill this gap using the theoretical tools developed earlier. We shall focus on a near ideal situation to gain intuition.

Consider a pure subgraph $G' = \cup_{\ell=1}^q G_\ell$ ($G_\ell = (V_\ell, E_\ell)$) of G. If scaling factors $\mathbf{S}_j$ are approximately constant within each pure component, then using the Laplacian in Definition 2, we have a small regularization penalty for the edges within a pure component and between the nodes that have close output values (i.e., $f_{j,k} \approx f_{j',k}$). Therefore, in the following we focus on finding the optimal scaling matrix $\mathbf{S}$ such that $\mathbf{S}_j$ is constant within each pure component V_ℓ, and assume that $\mathbf{S}$ is quantified by q numbers $[\bar{s}_\ell]_{\ell=1,\ldots,q}$, such that $\mathbf{S}_j = \bar{s}_\ell$ when $j \in V_\ell$.

Consider the following quantity:

$$\mathbf{cut}(G', y) = \sum_{j,j': y_j \neq y_{j'}} w_{j,j'} + \sum_{\ell \neq \ell'} \sum_{j \in V_\ell, j' \in V_{\ell'}} \frac{w_{j,j'}}{2}.$$

It is easy to check that

$$\mathbf{cut}(\mathcal{L}_\mathbf{S}, y) \leq \mathbf{cut}(G', y) / \min_\ell \bar{s}_\ell.$$

Assume that weights are small between pure components, and therefore, $\mathbf{cut}(G', y)$ is small.

With the $O(1/n)$ convergence rate, we obtain from (5) that

$$\frac{1}{m-n} \sum_{j \in Z_n} \mathbf{err}(\hat{\mathbf{f}}_j, y_j) \leq \frac{b}{ac} \cdot \frac{\max_\ell(\bar{s}_\ell/m_\ell)}{n} \left(\sqrt{\sum_{\ell=1}^q m_\ell/\bar{s}_\ell} + \sqrt{\frac{\mathbf{cut}(G', y)}{u(G') \min_\ell \bar{s}_\ell}} \right)^2.$$

If $\mathbf{cut}(G', y)$ is small, then the dominating term on the right hand side is

$$\frac{\max_\ell(\bar{s}_\ell/m_\ell)}{n} \sum_{\ell=1}^q \frac{m_\ell}{\bar{s}_\ell},$$

which can be optimized with the choice $\bar{s}_\ell = m_\ell$, and the resulting bound becomes:

$$\frac{1}{m-n}\sum_{j\in Z_n}\mathbf{err}(\hat{\mathbf{f}}_j, y_j) \leq \frac{b}{ac}\cdot\frac{1}{n}\left(\sqrt{q}+\sqrt{\frac{\mathbf{cut}(G',y)}{u(G')\min_\ell m_\ell}}\right)^2.$$

That is, if $\mathbf{cut}(G',y)$ is small, then we can choose scaling factor $\bar{s}_\ell \propto m_\ell$ for each pure component ℓ so that the generalization performance is approximately $(ac)^{-1}b\cdot q/n$, which is of the order $O(1/n)$.

The analysis provided here not only proves the importance of normalization under the learning theoretical framework, but also suggests that the good normalization factor for each node j is approximately the size of the well-connected pure component that contains node j (assuming that nodes belonging to different pure components are only weakly connected). Our analysis focused on the case that the scaling factors are a constant within each pure component. This condition is quite natural if we look at the normalized Laplacian regularization condition in Definition 2, where $f_{j,k}/\sqrt{\mathbf{S}_j}$ should be similar to $f_{j',k}/\sqrt{\mathbf{S}_{j'}}$ when $w_{j,j'}$ is large. If j and j' belongs to the same class, then $f_{j,k}$ should be similar to $f_{j',k}$. Therefore for such a pair (j,j'), we want to have $\mathbf{S}_j \approx \mathbf{S}_{j'}$ if $w_{j,j'}$ is large. Note that this requirement is not enforced by the standard degree-based normalization method $\mathbf{S}_j = \deg_j(G)$ because a well-connected pure component may contain nodes with quite different degrees. The assumption is satisfied under a simplified "box model", which is related to the models used by some previous researchers to derive the standard normalization method (e.g., Shi and Malik, 2000). In this model, a pure component is completely connected, and each node connects to all other nodes and itself with edge weight $w_{j,j'} = 1$. The degree is thus $\deg_j(G_\ell) = |V_\ell| = m_\ell$, which gives the optimal scaling in our analysis.

In general, the box model may not be a good approximation for practical problems. A more realistic approximation, which we call core-satellite model, will be introduced in the experimental section. For such a model, the degree-based normalization can fail because the $\deg_j(G_\ell)$ within each pure component G_ℓ is not approximately constant, and it may not be proportional to m_ℓ. In general, this approximation using degrees causes $\mathbf{S}_j$ to potentially vary significantly within a pure component because each $\mathbf{S}_j$ is only determined by its immediate neighbors.

Our analysis suggests that it is necessary to modify the degree-based scaling method $\mathbf{S}_j = \deg_j(G)$ so that the scaling factor is approximately a constant within each pure component, which should be proportional to m_ℓ. Our remedy is to look for connected components at a larger distance scale. Although there could be various methods to achieve this effect, we shall focus on a specific method motivated by the proofs of Theorem 5 and Theorem 7. Let $\bar{\mathbf{K}} = (\alpha\mathbf{I}+\mathcal{L})^{-1}$ be the kernel matrix corresponding to the unnormalized Laplacian. Using the terminology in the proofs, we observe that for small α:

$$\alpha\bar{\mathbf{K}} = \sum_{\ell=1}^{q}\mathbf{v}_\ell\mathbf{v}_\ell^T/m_\ell + O(1),$$

and thus $\bar{\mathbf{K}}_{j,j} \propto m_\ell^{-1}$ for each $j \in V_\ell$. Therefore with small α, the scaling factor $\mathbf{S}_j = 1/\bar{\mathbf{K}}_{j,j}$ is near optimal for all j. For $\alpha > 0$, the effect of this scaling factor is essentially equivalent to looking for connected components at a scale of at most $O(1/\alpha)$ number of nodes. We call this method of normalization $\mathbf{K}$-*scaling* in this paper. It is equivalent to a normalization of the kernel matrix $\mathbf{K}$, so that each $\mathbf{K}_{j,j} = 1$. Although this method coincides with a common practice in standard kernel learning, it is important to notice that to show this method behaves well in the graph learning setting is highly non-trivial and novel. To our best knowledge, no one has proposed this normalization

method in the graph learning setting before. In fact, without learning theoretical results developed in this paper, it is not obvious that this method should work better than the more standard degree-based normalization method. In our framework, the main advantage of $\mathbf{K}$-scaling (compared to the standard degree-scaling, which we call $\mathbf{L}$-scaling) is twofold:

- The resulting $\mathbf{S}_j$ does not vary significantly within a well-connected pure component.
- The resulting scaling is approximately m_ℓ (at a scale of $1/\alpha$), which is predicted by our theory to be desirable.[1]

The superiority of this method will be demonstrated in our experiments. The main drawback of this method is the computational cost of directly inverting $(\alpha\mathbf{I}+\mathcal{L})$. For large scale problems, approximation methods are required.

3.5 Dimension Reduction

Normalization and dimension reduction have been commonly used in spectral clustering such as Ng et al. (2002) and Shi and Malik (2000). For semi-supervised learning, dimension reduction (without normalization) is known to improve performance (Belkin and Niyogi, 2004, Zhang and Ando, 2006) while the degree-based normalization (without dimension reduction) has also been explored (Zhou et al., 2004). In this section, we present a brief high-level argument that an appropriate combination of normalization and dimension reduction (as commonly used in spectral clustering) can improve classification performance. Detailed analysis can be found in Appendix C.

Let us first introduce dimension reduction with normalized Laplacian $\mathcal{L}_{\mathbf{S}}(G)$. Denote by $\mathbf{P}^r_{\mathbf{S}}(G)$ the projection operator onto the eigenspace of $\alpha\mathbf{S}^{-1}+\mathcal{L}_{\mathbf{S}}(G)$ corresponding to the r smallest eigenvalues. Now, we may define the following regularizer on the reduced subspace:

$$\mathbf{f}^T_{\cdot,k}\mathbf{K}^{-1}\mathbf{f}_{\cdot,k} = \begin{cases} \mathbf{f}^T_{\cdot,k}\mathbf{K}_0^{-1}\mathbf{f}_{\cdot,k} & \text{if } \mathbf{P}^r_{\mathbf{S}}(G)\mathbf{f}_{\cdot,k} = \mathbf{f}_{\cdot,k}, \\ +\infty & \text{otherwise.} \end{cases} \tag{6}$$

The benefit of dimension reduction in graph learning has been investigated in Zhang and Ando (2006), under the spectral kernel design framework. The idea is to modify the kernel eigenvalues so that the target spectral coefficient matches the kernel coefficients. Note that the normalization issue, which will change the eigenvectors and their ordering, wasn't investigated there. However, with a fixed scaling matrix $\mathbf{S}$, the reasoning given in Zhang and Ando (2006) can also be applied here. It was shown there that if noise is added into the kernel matrix, then in general kernel eigenvalues will decay slower than the target spectral coefficients. Because of this, dimension reduction, which makes kernel eigenvalues better match the decay of target spectral coefficients, will become helpful. For Laplacian regularization investigated here, we may regard noise as edges connecting pure components of different classes, which increase the cut in Definition 3. Such noise can be significantly reduced if we project it into a low-dimensional space, and if the target functions approximately lie in this low-dimensional space. In this context, the effect of modification of eigenspaces through appropriate Laplacian normalization is to achieve faster decay of the target spectral coefficients in the

1. Although "the scaling factor $\mathbf{S}_j = m_\ell$" might be reminiscent of the ratio cut in spectral clustering, note that, as mentioned earlier, the ratio cut corresponds to the unnormalized Laplacian. $\mathbf{K}$-scaling suggested here normalizes the Laplacian matrix, which is in the same spirit of normalized cut (degree-scaling) but fixes some of its short-comings based on learning theoretical insights developed here.

	classes #1, #2	classes #3–#10
graph1	(4,2)	(2,1)
graph2	(6,3)	(2,1)
graph3	(8,4)	(2,1)

Figure 1: Generation of graph 1–5. (c,e) in the table indicates that for each node, we randomly chose c nodes of the same class and connect it to them, and we randomly chose e nodes of other classes (introducing errors) and connect it to them. Edge weights are fixed to 1.

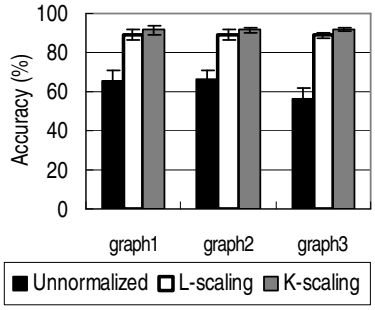


Figure 2: Classification accuracy (%) on the graphs where degrees are nearly constant within the class. $n = 40, m = 2000$. With dimension reduction (dim ≤ 20; chosen by cross validation). Average over 10 random splits with one standard deviation.

first few eigenvectors of the kernel. Therefore, under certain conditions, dimension reduction can reduce noise (corresponding to a small cut), which essentially makes normalization more effective as shown in Section 3.4.

We show our formal analysis of the combination of dimension reduction (as in (6) above) and normalization of Laplacian, for completeness, in Appendix C and empirical results in the next section.

4. Experiments

We experiment with the Laplacian regularization with the normalization methods discussed above, on synthesized data sets generated by controlling graph properties as well as three real-world data sets.

4.1 Experimental Framework

The Laplacian matrix $\mathcal{L}$ is generated from a graph G so that $\mathcal{L}_{j,j'} = -w_{j,j'}$ for $j \neq j'$ and $\mathcal{L}_{j,j} = \deg_j(G)$. Using $\mathcal{L}$, we define matrix $\mathbf{K}$ as follows:

- *Unnormalized:* $\mathbf{K} = (\alpha\mathbf{I} + \mathcal{L})^{-1}$. That is, $\mathbf{S} = \mathbf{I}$. No scaling.

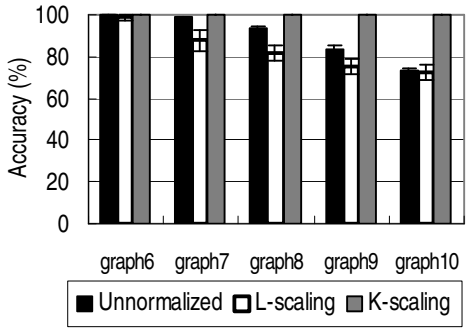


Figure 3: Classification accuracy on the core-satellite graphs. $n = 40, m = 2000$. With dimension reduction (dim ≤ 20; chosen by cross validation). Average over 10 random splits with one standard deviation.

- $\mathbf{K}$-*scaling:* $\mathbf{K} = (\mathbf{S}^{-1/2}(\alpha\mathbf{I} + \mathcal{L})\mathbf{S}^{-1/2})^{-1}$ where $\mathbf{S} = \mathrm{diag}_j(\bar{\mathbf{K}}_{j,j}^{-1})$ with $\bar{\mathbf{K}} = (\alpha\mathbf{I} + \mathcal{L})^{-1}$. The diagonal entries of $\mathbf{K}$ are all ones.
- $\mathbf{L}$-*scaling:* $\mathbf{K} = (\alpha\mathbf{I} + \mathbf{S}^{-1/2}\mathcal{L}\mathbf{S}^{-1/2})^{-1}$ where $\mathbf{S} = \mathrm{diag}_j(\deg_j(G))$. The diagonal entries of $\mathbf{K}^{-1}$ are constant $(\alpha + 1)$. This is the standard degree-based scaling.

Using these three types of matrix $\mathbf{K}$, we test the following two types of regularization. One regularizes by $\mathbf{f}_{\cdot,k}^T\mathbf{K}^{-1}\mathbf{f}_{\cdot,k}$ using $\mathbf{K}$ without dimension reduction, as in Section 3. The other reduces the dimension of $\mathbf{K}^{-1}$ to r by leaving out all but several eigenvectors corresponding to the smallest r eigenvalues to obtain the eigenspace projector $\mathbf{P}_{\mathbf{S}}^r(G)$ and regularizes by:

$$\begin{cases} \mathbf{f}_{\cdot,k}^T\mathbf{K}^{-1}\mathbf{f}_{\cdot,k} & \text{if } \mathbf{P}_{\mathbf{S}}^r(G)\mathbf{f}_{\cdot,k} = \mathbf{f}_{\cdot,k} \\ +\infty & \text{otherwise} \end{cases}$$

as in Section 3.5. We use the one-versus-all strategy and use least squares as our loss function: $\phi_k(a,b) = (a - \delta_{k,b})^2$.

From m data points, n training labeled examples are randomly chosen while ensuring that at least one training example is chosen from each class. The remaining $m - n$ data points serve as test data. The regularization parameter λ is chosen by cross validation on the n training labeled examples. We will show performance when the rest of the parameters (α and dimensionality r) are also chosen by cross validation on the training labeled examples and when they are set to the optimum. The dimensionality r is chosen from $K, K+5, K+10, \cdots, 100$ where K is the number of classes unless otherwise specified. Our focus is on small n close to the number of classes. Throughout this section, we conduct 10 runs with random training/test splits and report the average accuracy.

4.2 Controlled Data Experiments

The purpose of the controlled data experiments is to observe the correlation of the effectiveness of the normalization methods with graph properties. The graphs we generate contain 2000 nodes, each of which is assigned one of 10 classes.

First, we show the results when dimension reduction is applied to the three types of matrix $\mathbf{K}$. Figure 2 shows classification accuracy on three graphs that were generated so that the node degrees

(of either correct edges or erroneous edges) are close to constant within each class but vary across classes. Details of their generation are described in Figure 1. We observe that on these graphs, both **K**-scaling and **L**-scaling significantly improve classification accuracy over the unnormalized baseline. There is no prominent difference between **K**-scaling's and **L**-scaling's performance.

Observe that **K**-scaling and **L**-scaling perform differently on the graphs used in Figure 3. These graphs have the following properties. Each class consists of *core nodes* and *satellite nodes*. Core nodes of the same class are tightly connected with each other and do not have any erroneous edges. Satellite nodes are relatively weakly connected to core nodes of the same class. The satellite nodes are also connected to some other classes' satellite nodes (i.e., introducing errors). This core-satellite model is intended to simulate real-world data in which some data points are close to the class boundaries (satellite nodes). More precisely, graphs 6–10 were generated as follows. Each graph consists of 2000 nodes ($m = 2000$) uniformly distributed over 10 classes ($K = 10$). 10% of the nodes are the core nodes. For every core node, we randomly choose 10 other core nodes of the same class and connect it to them with edge weight 1 (that is, each core node is connected to at least 10 core nodes of the same class). For every satellite node, we randomly choose one core node of the same class and connect them with edge weight 0.01. Also, for each satellite node, we randomly choose one satellite node of some other class (i.e., introducing error) and connect them with edge weight w_e. We set the error edge weight $w_e = 0.002, 0.004, \cdots, 0.01$ for graphs $6, 7, \cdots, 10$, respectively. Note that although classes are uniformly distributed, pure components that optimize the generalization bound may be non-uniform in size. For graphs generated in this manner, degrees vary within the same class since the satellite nodes have smaller degrees than the core nodes. Our analysis suggests that **L**-scaling will do poorly. Figure 3 shows that on the five core-satellite graphs, **K**-scaling indeed produces higher performance than **L**-scaling. In particular, **K**-scaling does well even when **L**-scaling rather underperforms the unnormalized baseline.

Our analysis suggests that **K**-scaling should work well when the graph has relatively small error. This trend is more clearly observed on these core-satellite graphs without dimension reduction. As shown in Figure 4, the advantage of **K**-scaling over **L**-scaling is more prominent on the graphs with smaller error edge weights. On the other hand, the theory suggests that when the graph has large error (large cut), the benefit of normalization is less clear (since the derivation of **K**-scaling assumes near-zero cut). This is especially so when dimension reduction is not applied because as pointed out in Section 3.5, dimension reduction reduces error. This trend can be observed in Figure 5, which shows that on graphs 1–3 (having larger errors than the core-satellite graphs), neither **L**-scaling nor **K**-scaling prominently improves performance over the unnormalized Laplacian without dimension reduction though **L**-scaling seems to perform slightly better. Note that the performance without dimension reduction (Figure 5) is significantly worse than the performance with dimension reduction (Figure 2). This means that dimension reduction, which reduces error, is important when we try to apply graph based methods.

Additionally, we show illustrative toy examples based on the core-satellite model. Given the original graph as in Figure 6 (a), Figure 6 (b)–(d) show the graphs corresponding to the scaled adjacency matrices $\mathbf{S}^{-1/2}\mathbf{W}\mathbf{S}^{-1/2}$ where $\mathbf{S}$ is derived from **L**-scaling, **K**-scaling with $\alpha = 0.01$, and **K**-scaling with $\alpha = 0.1$, respectively. We observe that, compared with the unnormalized case, K-scaling and L-scaling essentially balance the edge weights between the two classes (i.e., "normalizing") by relatively lowering the weights of class2 which is more "massive". However, in this example, **L**-scaling in a sense overdoes it and so is rather harmful as it amplifies the error edge weights over the weights of the within-class edges. **K**-scaling does not suffer from this problem.

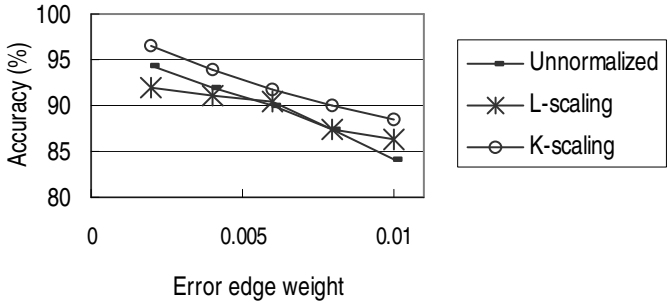


Figure 4: Classification accuracy on the core-satellite graphs. x-axis: error edge weight w_e. $n = 40, m = 2000$. Without dimension reduction. Average over 10 random splits.

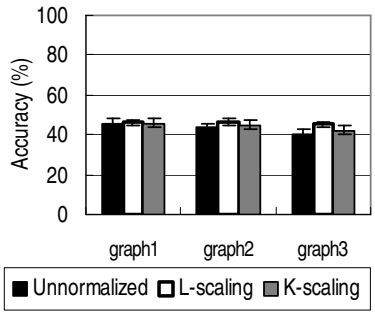


Figure 5: Classification accuracy (%) on the graphs where degrees are nearly constant within the class. Average over 10 random splits. $n = 40, m = 2000$. Without dimension reduction.

4.3 Real-world Data Experiments

Our real-world data experiments use two image data sets (MNIST and UMIST) and one text data set (RCV1).

4.3.1 DATA AND BASELINE

The MNIST data set, downloadable from http://yann.lecun.com/exdb/mnist/, consists of hand-written digit image data (representing 10 classes, from digit "0" to "9"). For our experiments, we randomly choose 2000 images (i.e., $m = 2000$). The UMIST data set, downloadable from http://images.ee.umist.ac.uk/danny/database.html, consists of 575 face images taken from several angles of 20 people (representing 20 classes). The details of this data are described in Graham and Allinson (1998). We use all the images (i.e., $m = 575$). Reuters Corpus Version 1 (RCV1) consists of news articles labeled with topics. For our experiments, we chose 10 topics (representing 10 classes) that have relatively large populations and randomly chose 2000 articles that are labeled with exactly one of those 10 topics. The class distribution over these 2000 articles is non-uniform as shown in Figure 7.

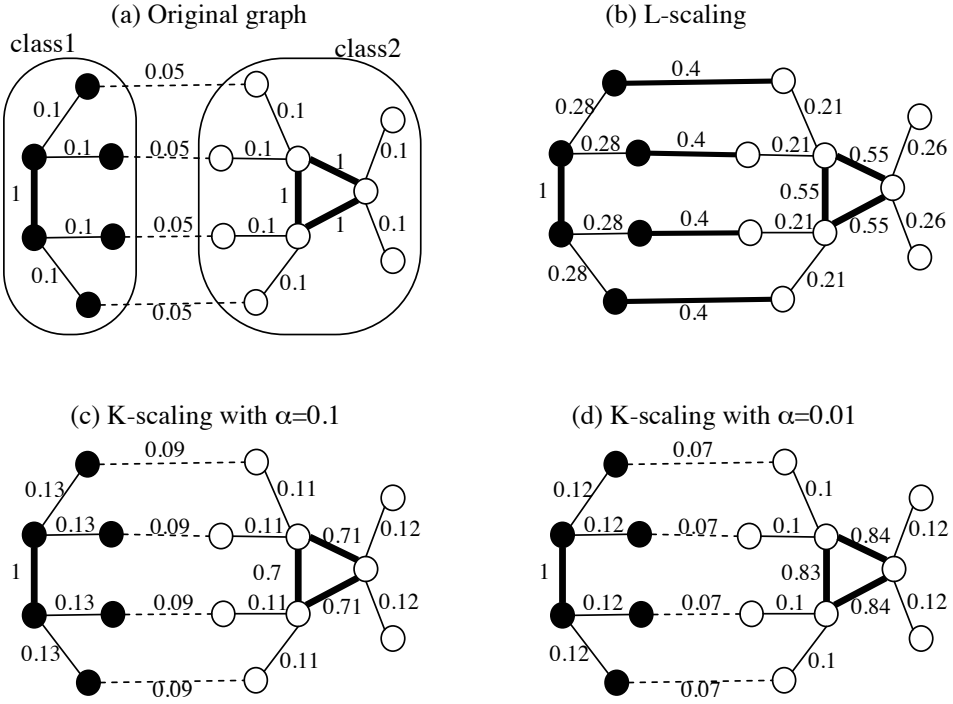


Figure 6: Illustrative toy examples of scaled adjacency matrices $\mathbf{S}^{-1/2}\mathbf{W}\mathbf{S}^{-1/2}$ where $\mathbf{S}$ is derived from $\mathbf{L}$-scaling or $\mathbf{K}$-scaling. For the two-class core-satellite graph in (a), $\mathbf{L}$-scaling makes the weights of error edges larger than the edge weights between the core nodes and satellite nodes of the same class as in (b). $\mathbf{K}$-scaling does not suffer from this problem ((c),(d)). (For an easy comparison, in (b)–(d), edge weights are multiplied with constants so that the largest weight becomes one.)

To generate graphs from the image data, as is commonly done, we first generate the vectors of the gray-scale values of the pixels, and produce the edge weight between the i-th and the j-th data points $\mathbf{X}_i$ and $\mathbf{X}_j$ by $w_{i,j} = \exp(-||\mathbf{X}_i - \mathbf{X}_j||^2/t)$ where $t > 0$ is a parameter (radial basis function (RBF) kernels). To generate graphs from the text data, we first create the bag-of-word vectors using content words only[2] and then set $w_{i,j}$ based on RBF as above or set $w_{i,j}$ to the inner product of $\mathbf{X}_i$ and $\mathbf{X}_j$ (linear kernels). Optionally, we zero out all $w_{i,j}$ but k nearest neighbors (i.e., i is j's k nearest neighbors or j is i's k nearest neighbors) to reduce error in graphs and refer to it as the RBF (or linear) kernel with kNN.

As our baseline, we also test the supervised configuration by letting $\mathbf{W} + \beta\mathbf{I}$ (where $\mathbf{W}$ is a weight matrix whose (i, j)-entry is $w_{i,j}$) be the kernel matrix and using the same least squares loss function. We set β to the optimum, which was 0.001 for the RBF kernel for RCV1 and 1 for the other graphs.

2. To generate a bag-of-word vector from a document, as is commonly done, we remove function words (such as "a", "the", and so on), set word frequencies of the document to the corresponding vector entries, and then scale the vector into a unit vector.

GPOL	Domestic politics	486
GSPO	Sports	407
GDIP	International relations	299
GCRIM	Crime, law enforcement	224
GJOB	Labor issues	206
GVIO	War, civil war	142
GDIS	Disasters and accidents	89
GHEA	Health	57
GENT	Arts, culture, entertainment	47
GENV	Environments	43
Total		2000

Figure 7: 10 RCV1 categories and their populations used in our experiments.

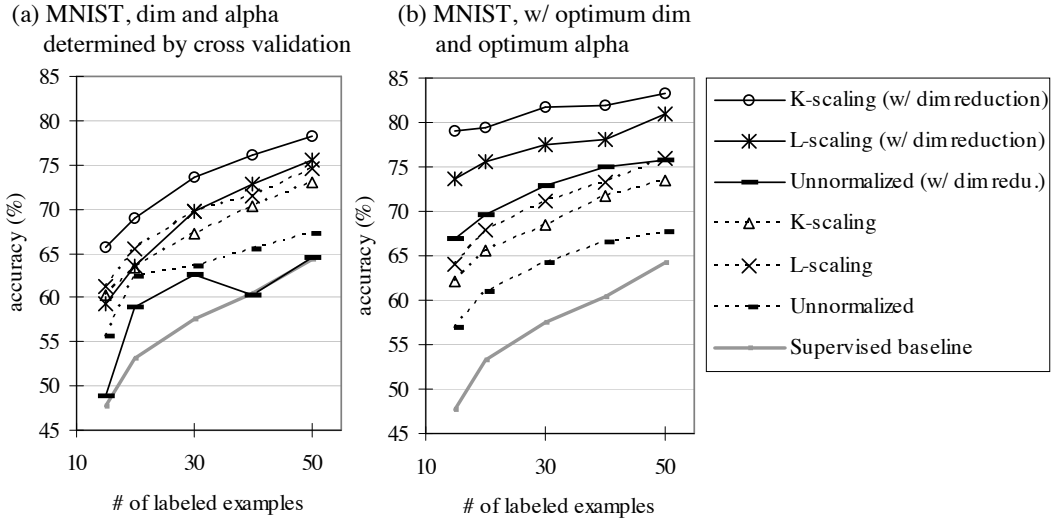


Figure 8: Classification accuracy (%) in relation to the number of labeled examples (n) on MNIST. $m = 2000$. (a) Dimensionality and α were determined by cross validation. (b) Dimensionality and α were set to the optimum. Average over 10 random splits.

4.3.2 RESULTS

Figure 8 shows performance in relation to the number of labeled examples (n) on the MNIST data set. The comparison of the three bold lines (representing the methods with dimension reduction) in Figure 8 (a) shows that when the dimensionality and α are determined (performing simple 2-dimensional grid search) by cross validation, **K**-scaling outperforms **L**-scaling, and **L**-scaling outperforms the unnormalized Laplacian. These performance differences are statistically significant ($p \leq 0.01$) based on the paired t test. The performance of the unnormalized Laplacian (with dimension reduction) is roughly consistent with the performance with similar (m,n) with heuristic dimension selection in Belkin and Niyogi (2004). Although without dimension reduction, **L**-scaling

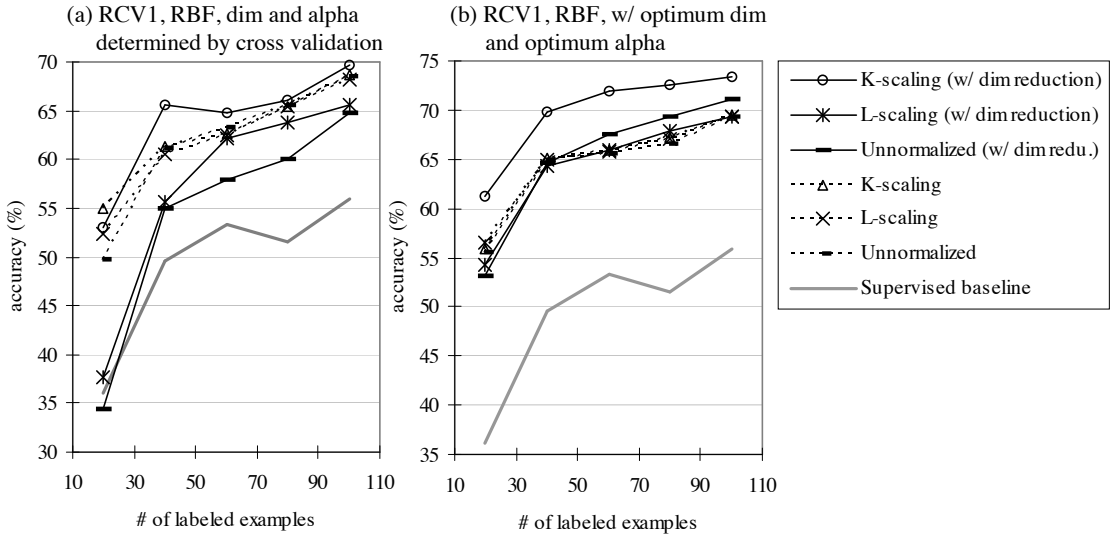


Figure 9: Classification accuracy (%) in relation to the number of labeled examples (n) on RCV1. RBF kernel (with $t = 0.25$). $m = 2000$. (a) Dimensionality and α were determined by cross validation. (b) Dimensionality and α were set to the optimum. Performance differences of the best performing method '**K**-scaling (w/ dim reduction)' from '**L**-scaling (w/ dim reduction)' and 'Unnormalized (w/ dim redu.)' are statistically significant ($p \leq 0.01$) in both the settings (a) and (b).

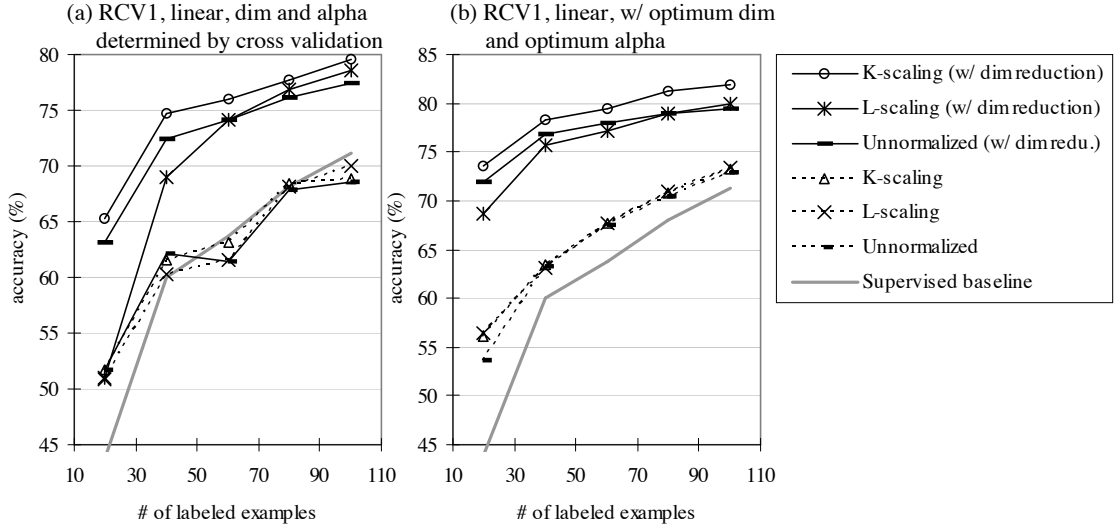


Figure 10: Classification accuracy (%) in relation to the number of labeled examples (n) on RCV1. Linear kernel. $m = 2000$. (a) Dimensionality and α were determined by cross validation. (b) Dimensionality and α were set to the optimum. Performance differences of the best performing method '**K**-scaling (w/ dim reduction)' from the second and third best '**L**-scaling (w/ dim reduction)' and 'Unnormalized (w/ dim redu.)' are statistically significant ($p \leq 0.01$) in both the settings (a) and (b).

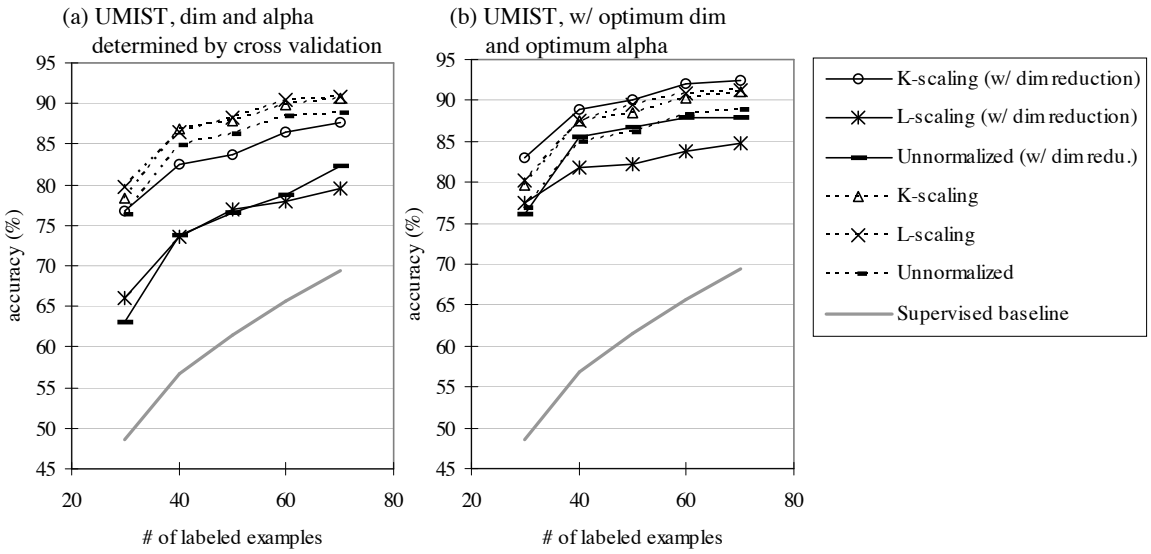


Figure 11: Classification accuracy (%) in relation to the number of labeled examples (n) on UMIST. $m = 575$. (a) Dimensionality and α were determined by cross validation. (b) Dimensionality and α were set to the optimum. In (b), performance differences of the best performing method '**K**-scaling (w/ dim reduction)' from the second and third best '**K**-scaling' and '**L**-scaling' are statistically significant ($p \leq 0.01$).

and **K**-scaling still improve performance over the unnormalized Laplacian, the best performance is always obtained by **K**-scaling with dimension reduction (the bold line with circles).

In Figure 8 (a), the unnormalized Laplacian with dimension reduction underperforms the unnormalized Laplacian without dimension reduction, indicating that dimension reduction rather degrades performance in this case. By comparing Figure 8 (a) and (b), we observe that this seemingly counter-intuitive performance trend is caused by the difficulty in choosing the right dimensionality by cross validation. Figure 8 (b) shows the performance at the optimum dimensionality and the optimum α. As observed, if the optimum dimensionality is known (as in (b)), dimension reduction improves performance either with or without normalization by **K**-scaling and **L**-scaling, and that all the transductive configurations outperform the supervised baseline. We also note that the comparison of Figure 8 (a) and (b) shows that choosing good dimensionality by cross validation is much harder than choosing α by cross validation especially when the number of labeled examples is small. This trend was observed also on the other data sets on which we experimented.

On the RCV1 data set, the performance trend is essentially similar to that of MNIST. Figure 9 shows the performance on RCV1 using the RBF kernel ($t = 0.25$, 100NN). In the setting of Figure 9 (a) where the dimensionality and α were determined by cross validation, **K**-scaling with dimension reduction generally performs the best. By setting the dimensionality and α to the optimum, the benefit of **K**-scaling with dimension reduction is even clearer (Figure 9 (b)).

On text data like RCV1, linear kernels (instead of RBF) are often used. Figure 10 shows the performance with linear kernels with 100NN. Again, **K**-scaling with dimension reduction performs the best. Its performance differences from the second and third best '**L**-scaling (w/ dim reduction)'

and 'Unnormalized (w/ dim redu.)' are statistically significant ($p \leq 0.01$) in both Figure 10 (a) and (b).

In Figure 11, we observe that dimension reduction seems less useful on the UMIST data set. We conjecture that this may be because UMIST differs from our other data sets in that it is much more 'sparse'; UMIST has a smaller number of data points ($m = 575$ vs. $m = 2000$) while it has more classes ($K = 20$ vs. $K = 10$). Nevertheless, when the dimensionality and α are set to the optimum (Figure 11 (b)), again, **K**-scaling with dimension reduction performs the best. Its differences from the second and the third best methods (**K**-scaling without dimension reduction and **L**-scaling without dimension reduction) are statistically significant ($p \leq 0.01$).

Overall, on these graphs generated from image and text data sets, **K**-scaling with dimension reduction consistently outperformed the others. But without dimension reduction, **K**-scaling and **L**-scaling were not always effective. Transductive learning (either with or without normalization) generally improved performance.

4.4 Approximation of K-scaling

Although **K**-scaling consistently improves performance as shown above, its drawback is the relatively large runtime as it involves the computation of the inverse of an m-by-m matrix. We propose a less computationally-intensive approximation method using a known fact that $(\mathbf{I}-\mathbf{A})^{-1} = \sum_{k=0}^{\infty} \mathbf{A}^k$ if $||\mathbf{A}||_2 < 1$. As in the introduction, let $\mathbf{D} = \mathrm{diag}_i(\deg_i(G))$, and let $\mathbf{W}$ be a weight matrix such that $\mathbf{W}_{i,j} = w_{i,j}$ so that we can write $\mathcal{L} = \mathbf{D} - \mathbf{W}$. Let $\hat{\mathbf{D}} = \mathbf{D} + \alpha\mathbf{I}$. We define $\hat{\mathbf{K}}(h)$ to be the h-th order approximation of $\bar{\mathbf{K}} = (\mathcal{L} + \alpha\mathbf{I})^{-1}$ as follows:

$$\hat{\mathbf{K}}(h) = \hat{\mathbf{D}}^{-1/2} \left(\sum_{k=0}^{h} \left(\hat{\mathbf{D}}^{-1/2} \mathbf{W} \hat{\mathbf{D}}^{-1/2} \right)^k \right) \hat{\mathbf{D}}^{-1/2} .$$

We then set the i-th scaling factor $\mathbf{S}_i$ so that:

$$\mathbf{S}_i = \hat{\mathbf{K}}(h)_{i,i}^{-1} .$$

Since $\lim_{h\to\infty} \hat{\mathbf{K}}(h) = \bar{\mathbf{K}}$, the scaling factors produced with a sufficiently large h closely approximate **K**-scaling. On the other hand, since $\hat{\mathbf{K}}(0) = \hat{\mathbf{D}}^{-1} = (\mathbf{D} + \alpha\mathbf{I})^{-1}$, the scaling factors produced by $\hat{\mathbf{K}}(0)$ with $\alpha = 0$ are exactly the same as **L**-scaling (or the standard degree-scaling).

Figure 12 shows the performance of this approximation method with $h = 0, 2, 5, 10$ with dimension reduction in comparison with corresponding **K**-scaling and **L**-scaling on MNIST. In Figure 12 (b), we observe that at the optimum dimensionality and α, the performance of the approximation method lies exactly between that of **L**-scaling and **K**-scaling, and it approaches to **K**-scaling as the order h increases. Intuitively, with a larger h, this approximation method takes more and more global connections into account and improves performance.

5. Conclusion

We derived generalization bounds for multi-category classification on graphs with Laplacian regularization, using geometric properties of the graph. In particular, we used this analysis to obtain a better understanding of the role of normalization of the graph Laplacian matrix. We argued that the standard **L**-scaling normalization method has the undesirable property that the normalization factors

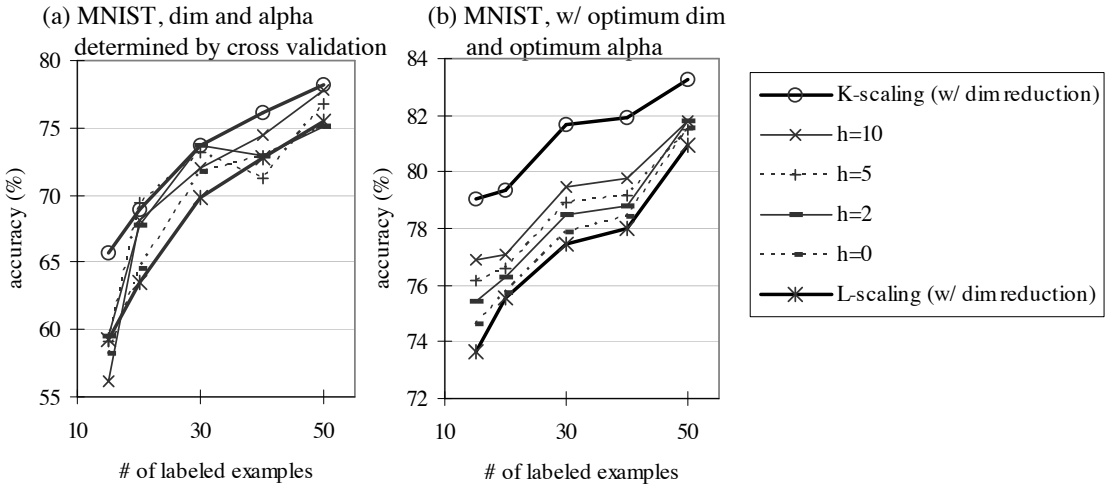


Figure 12: Classification accuracy (%) of the approximation method using $\hat{\mathbf{K}}(h)$. MNIST. (a) Dimensionality and α were determined by cross validation. (b) Dimensionality and α were set to the optimum.

can vary significantly within a pure component. An alternate normalization method, which we call **K**-scaling, is proposed to remedy the problem. Experiments confirm the superiority of **K**-scaling combined with dimension reduction. Finally, there are possible extensions of this work that require further investigation, for example, how to use the **K**-scaling for other types of graphs such as direct graphs, and how to apply this idea to spectral clustering.

Appendix A. Proof of Theorem 1

The proof employs the stability analysis of Zhang (2003), and is similar to the proof of a related bound for binary-classification in Zhang and Ando (2006). We shall introduce the following notation. let $i_{n+1} \neq i_1, \ldots, i_n$ be an integer randomly drawn from $\bar{Z}_n$, and let $Z_{n+1} = Z_n \cup \{i_{n+1}\}$. Let $\hat{\mathbf{f}}(Z_{n+1})$ be the semi-supervised learning method (1) using training data in Z_{n+1}:

$$\hat{\mathbf{f}}(Z_{n+1}) = \arg\min_{\mathbf{f} \in R^{mK}} \left[\frac{1}{n} \sum_{j \in Z_{n+1}} \phi(\mathbf{f}_j, y_j) + \lambda \mathbf{f}^T \mathbf{Q}_{\mathbf{K}} \mathbf{f} \right].$$

We have the following stability lemma (a related result can be found in Zhang, 2003);

Lemma 8 *The following inequality holds for each $k = 1, \ldots, K$:*

$$|\hat{f}_{i_{n+1},k}(Z_{n+1}) - \hat{f}_{i_{n+1},k}(Z_n)| \leq |\nabla_{1,k}\phi(\hat{f}_{i_{n+1}}(Z_{n+1}), y_{i_{n+1}})| \mathbf{K}_{i_{n+1},i_{n+1}} / (2\lambda n),$$

where $\nabla_{1,k}\phi(f_i, y)$ denotes a sub-gradient of $\phi(f_i, y)$ with respect to $f_{i,k}$, where $\mathbf{f}_i = [f_{i,1}, \ldots, f_{i,K}]$.

Proof From Rockafellar (1970), we know that there exist sub-gradients $\nabla_{1,k}\phi$ such that the following first-order condition for the optimization problem (1) holds:

$$-2\lambda n \mathbf{K}^{-1} \hat{\mathbf{f}}_{\cdot,k}(Z_n) = \sum_{j \in Z_n} \nabla_{1,k}\phi(\hat{\mathbf{f}}_j(Z_n), y_j) \mathbf{e}_j,$$

where $\mathbf{e}_j$ is the m-dimensional vector with all zeros except for the j-component with value one. Similarly, we have

$$-2\lambda n\mathbf{K}^{-1}\hat{\mathbf{f}}_{\cdot,k}(Z_{n+1}) = \sum_{j\in Z_{n+1}} \nabla_{1,k}\phi(\hat{\mathbf{f}}_j(Z_{n+1}), y_j)\mathbf{e}_j.$$

Now, for simplicity, let $\mathbf{g} = \hat{\mathbf{f}}(Z_n)$ and $\mathbf{h} = \hat{f}(Z_{n+1})$. By subtracting the above two equations, and then taking the inner product with $\mathbf{h}_{\cdot,k} - \mathbf{g}_{\cdot,k}$, we obtain

$$\begin{aligned}-2\lambda n(\mathbf{h}_{\cdot,k} - \mathbf{g}_{\cdot,k})^T\mathbf{K}^{-1}(\mathbf{h}_{\cdot,k} - \mathbf{g}_{\cdot,k}) =& \nabla_{1,k}\phi(h_{i_{n+1}}, y_{i_{n+1}})(h_{i_{n+1},k} - g_{i_{n+1},k}) \\ &+ \sum_{j\in Z_n}(\nabla_{1,k}\phi(\mathbf{h}_j, y_j) - \nabla_{1,k}\phi(\mathbf{g}_j, y_j))(h_{j,k} - g_{j,k}).\end{aligned}$$

Note that if $c(s)$ is a convex function of s, then it is easy to verify that $(\nabla c(s_1) - \nabla c(s_2))(s_1 - s_2) \geq 0$. Therefore we have $\sum_{j\in Z_n}(\nabla_{1,k}\phi(\mathbf{h}_j, y_j) - \nabla_{1,k}\phi(\mathbf{g}_j, y_j))(h_{j,k} - g_{j,k}) \geq 0$. This implies that

$$2\lambda n(\mathbf{h}_{\cdot,k} - \mathbf{g}_{\cdot,k})^T\mathbf{K}^{-1}(\mathbf{h}_{\cdot,k} - \mathbf{g}_{\cdot,k}) \leq -\nabla_{1,k}\phi(\mathbf{h}_{i_{n+1}}, y_{i_{n+1}})(h_{i_{n+1},k} - g_{i_{n+1},k}).$$

Using Cauchy-Schwartz inequality, we have

$$\begin{aligned}2\lambda n(h_{i_{n+1},k} - g_{i_{n+1},k})^2 =& 2\lambda n((\mathbf{h}_{\cdot,k} - \mathbf{g}_{\cdot,k})^T\mathbf{e}_{i_{n+1}})^2 \\ \leq& 2\lambda n(\mathbf{h}_{\cdot,k} - \mathbf{g}_{\cdot,k})^T\mathbf{K}^{-1}(\mathbf{h}_{\cdot,k} - \mathbf{g}_{\cdot,k})\mathbf{e}_{i_{n+1}}^T\mathbf{K}\mathbf{e}_{i_{n+1}} \\ \leq& |\nabla_{1,k}\phi(h_{i_{n+1}}, y_{i_{n+1}})| \cdot |h_{i_{n+1},k} - g_{i_{n+1},k}|\mathbf{K}_{i_{n+1},i_{n+1}}.\end{aligned}$$

Therefore we have $|h_{i_{n+1},k} - g_{i_{n+1},k}| \leq |\nabla_{1,k}\phi(h_{i_{n+1}}, y_{i_{n+1}})|\mathbf{K}_{i_{n+1},i_{n+1}}/(2\lambda n)$. ■

Lemma 9 *The following inequality holds*

$$\mathbf{err}(\hat{\mathbf{f}}_{i_{n+1}}(Z_n), y_{i_{n+1}}) \leq \sup_{k=k_0,i_{n+1}}\left[\frac{1}{a}\phi_0(\hat{f}_{i_{n+1},k}(Z_{n+1}), \delta_{i_{n+1},k}) + \left(\frac{b}{c\lambda n}\mathbf{K}_{i_{n+1},i_{n+1}}\right)^p\right].$$

Proof If $\hat{\mathbf{f}}(Z_n)$ does not make an error on the i_{n+1}-th example. That is, if $\mathbf{err}(\hat{\mathbf{f}}_{i_{n+1}}(Z_n), y_{i_{n+1}}) = 0$, then the inequality automatically holds.

Now, assume that $\hat{\mathbf{f}}(Z_n)$ makes an error on the i_{n+1}-th example: $\mathbf{err}(\hat{\mathbf{f}}_{i_{n+1}}(Z_n), y_{i_{n+1}}) = 1$. Then there exists $k_0 \neq y_{i_{n+1}}$ such that $\hat{f}_{i_{n+1},y_{i_{n+1}}}(Z_n) \leq \hat{f}_{i_{n+1},k_0}(Z_n)$. This means that for any d, either $\hat{f}_{i_{n+1},y_{i_{n+1}}}(Z_n) \leq d$ or $\hat{f}_{i_{n+1},k_0}(Z_n) \geq d$. We simple let $d = (\inf\{x : \phi_0(x,1) \leq a\} + \sup\{x : \phi_0(x,0) \leq a\})/2$. By the definition of c, either we have $\inf\{x : \phi_0(x,1) \leq a\} - \hat{f}_{i_{n+1},y_{i_{n+1}}}(Z_n) \geq c/2$ or we have $\hat{f}_{i_{n+1},k_0}(Z_n) - \sup\{x : \phi_0(x,0) \leq a\} \geq c/2$. It follows that there exists $k = k_0$ or $k = y_{i_{n+1}}$ such that either $\phi_0(\hat{f}_{i_{n+1},k}(Z_{n+1}), \delta_{y_{i_{n+1}},k}) \geq a$ or $\left|\hat{f}_{i_{n+1},k}(Z_{n+1}) - \hat{f}_{i_{n+1},k}(Z_n)\right| \geq c/2$.

Using Lemma 8, we have either $\phi_0(\hat{f}_{i_{n+1},k}(Z_{n+1}), \delta_{y_{i_{n+1}},k}) \geq a$ or $b\mathbf{K}_{i_{n+1},i_{n+1}}/(2\lambda n) \geq c/2$, implying that

$$\frac{1}{a}\phi_0(\hat{f}_{i_{n+1},k}(Z_{n+1}), \delta_{y_{i_{n+1}},k}) + \left(\frac{b\mathbf{K}_{i_{n+1},i_{n+1}}}{c\lambda n}\right)^p \geq 1 = \mathbf{err}(\hat{\mathbf{f}}_{i_{n+1}}(Z_n), y_{i_{n+1}}).$$

■

We are now ready to prove Theorem 1. For every $j \in Z_{n+1}$, denote by $Z_{n+1}^{(j)}$ the subset of n samples in Z_{n+1} with the j-th data point left out. From Lemma 9, we have

$$\mathbf{err}(\hat{\mathbf{f}}_j(Z_{n+1}^{(j)}), y_j) \leq \frac{1}{a}\phi(\hat{\mathbf{f}}_j(Z_{n+1}), y_j) + \left(\frac{b}{c\lambda n}\mathbf{K}_{j,j}\right)^p.$$

We thus obtain for all $\mathbf{f} \in R^{mK}$:

$$\begin{aligned}\sum_{j\in Z_{n+1}} \mathbf{err}(\hat{\mathbf{f}}_j(Z_{n+1}^{(j)}), y_j) \leq & \frac{1}{a}\sum_{j\in Z_{n+1}} \phi(\hat{\mathbf{f}}_j(Z_{n+1}), y_j) + \sum_{j\in Z_{n+1}} \left(\frac{b}{c\lambda n}\mathbf{K}_{j,j}\right)^p \\ \leq & \frac{1}{a}\left[\sum_{j\in Z_{n+1}} \phi(\mathbf{f}_j, y_j) + \lambda \mathbf{f}^T \mathbf{Q}_{\mathbf{K}}\mathbf{f}\right] + \sum_{j\in Z_{n+1}} \left(\frac{b}{c\lambda n}\mathbf{K}_{j,j}\right)^p.\end{aligned}$$

Therefore

$$\begin{aligned}&\mathbf{E}_{Z_n}\frac{1}{m-n}\sum_{j\in Z_n} \mathbf{err}(\hat{\mathbf{f}}_j(Z_n), y_j) \\ =&\mathbf{E}_{Z_{n+1}}\mathbf{err}(\hat{\mathbf{f}}_{n+1}(Z_{n+1}^{(n+1)}), y_{n+1}) = \frac{1}{n+1}\mathbf{E}_{Z_{n+1}}\sum_{j\in Z_{n+1}} \mathbf{err}(\hat{\mathbf{f}}_j(Z_{n+1}^{(j)}), y_j) \\ \leq & \frac{n}{a(n+1)}\mathbf{E}_{Z_{n+1}}\left[\frac{1}{n}\sum_{j\in Z_{n+1}} \phi(\mathbf{f}_j, y_j) + \lambda \mathbf{f}^T \mathbf{Q}_{\mathbf{K}}\mathbf{f}\right] + \frac{1}{n+1}\mathbf{E}_{Z_{n+1}}\sum_{j\in Z_{n+1}} \left(\frac{b}{c\lambda n}\mathbf{K}_{j,j}\right)^p \\ =&\frac{1}{a}\left[\frac{1}{m}\sum_{j=1}^{m} \phi(\mathbf{f}_j, y_j) + \frac{\lambda n}{n+1}\mathbf{f}^T \mathbf{Q}_{\mathbf{K}}\mathbf{f}\right] + \frac{1}{m}\sum_{j=1}^{m}\left(\frac{b\mathbf{K}_{j,j}}{c\lambda n}\right)^p.\end{aligned}$$

Appendix B. Proof of Theorem 7

The idea is similar to that of Theorem 5. We use the same notation, and let $\mathbf{v}_\ell$ be the indicator function of V_ℓ in V. Let $\mathbf{I}_\ell$ be the diagonal matrix with value ones for nodes corresponding to V_ℓ and zeros elsewhere. We have $\forall \mathbf{u} = [u_1, \ldots, u_m] \in R^m$, $(\mathbf{u} - \mathbf{u}^T\mathbf{v}_\ell\mathbf{v}_\ell/m_\ell)^T\mathbf{v}_\ell = 0$. Therefore by the definition of $\lambda_2(G_\ell)$, we have

$$\begin{aligned}\frac{1}{2}\sum_{j,j'\in V_\ell} w_{j,j'}(u_j - u_k)^2 =& (\mathbf{I}_\ell\mathbf{u} - \mathbf{u}^T\mathbf{v}_\ell\mathbf{v}_\ell/m_\ell)^T\mathcal{L}(G')(\mathbf{I}_\ell\mathbf{u} - \mathbf{u}^T\mathbf{v}_\ell\mathbf{v}_\ell/m_\ell) \\ \geq & \lambda_2(G_\ell)\|\mathbf{I}_\ell\mathbf{u} - \mathbf{u}^T\mathbf{v}_\ell\mathbf{v}_\ell/m_\ell\|_2^2 = \lambda_2(G_\ell)\mathbf{u}^T[\mathbf{I}_\ell - \mathbf{v}_\ell\mathbf{v}_\ell/m_\ell]\mathbf{u}.\end{aligned}$$

We thus obtain

$$\begin{aligned}\mathbf{u}^T\mathcal{L}\mathbf{u} =& \frac{1}{2}\sum_{j,j'=1}^{m} w_{j,j'}(u_j - u_k)^2 \geq \sum_{\ell=1}^{q}\frac{1}{2}\sum_{j,j'\in V_\ell} w_{j,j'}(u_j - u_k)^2 \\ \geq & \mathbf{u}^T\left[\sum_{\ell=1}^{q}\lambda_2(G_\ell)(\mathbf{I}_\ell - \mathbf{v}_\ell\mathbf{v}_\ell^T/m_\ell)\right]\mathbf{u}.\end{aligned}$$

Therefore $\mathcal{L} - \sum_{\ell=1}^{q} \lambda_2(G_\ell)(\mathbf{I}_\ell - \mathbf{v}_\ell \mathbf{v}_\ell^T/m_\ell)$ is positive semi-definite, and thus

$$\left(\alpha \mathbf{I} + \sum_{\ell=1}^{q} \lambda_2(G_\ell)(\mathbf{I}_\ell - \mathbf{v}_\ell \mathbf{v}_\ell^T/m_\ell)\right)^{-1} - (\alpha \mathbf{I} + \mathcal{L})^{-1}$$

is positive-semi-definite. Also, from (2) we have $\mathbf{S}^{-1/2}\mathbf{K}\mathbf{S}^{-1/2} = (\alpha \mathbf{I} + \mathcal{L}(G))^{-1}$, so we know that the diagonal entries of $\mathbf{S}^{-1/2}\mathbf{K}\mathbf{S}^{-1/2}$ can be upper-bounded by those of

$$\left(\alpha \mathbf{I} + \sum_{\ell=1}^{q} \lambda_2(G_\ell)(\mathbf{I}_\ell - \mathbf{v}_\ell \mathbf{v}_\ell^T/m_\ell)\right)^{-1} = \sum_{\ell=1}^{q} (\alpha + \lambda_2(G_\ell))^{-1} \left(\mathbf{I}_\ell + \alpha^{-1}\lambda_2(G_\ell)\mathbf{v}_\ell \mathbf{v}_\ell^T/m_\ell\right).$$

For the latter, its m_ℓ diagonal entries for each pure component ℓ can be upper bounded by $\lambda_2(G_\ell)^{-1} + (\alpha m_\ell)^{-1}$. Therefore:

$$\begin{aligned} m^{1/p}\mathbf{tr}_p(\mathbf{K}) &\leq \left(\sum_{\ell=1}^{q} s_\ell(p)(\alpha^{-1}m_\ell^{-1} + \lambda_2(G_\ell)^{-1})^p\right)^{1/p} \\ &\leq \left[\alpha^{-1}\left(\sum_{\ell=1}^{q} s_\ell(p) m_\ell^{-p}\right)^{1/p} + \left(\sum_{\ell=1}^{q} s_\ell(p)\lambda_2(G_\ell)^{-p}\right)^{1/p}\right]. \end{aligned}$$

Substitute this estimate into Theorem 4, and we have

$$\mathbf{E}_{Z_n} \frac{1}{m-n} \sum_{j \in \bar{Z}_n} \mathbf{err}(\hat{\mathbf{f}}_j(Z_n), y_j) \leq \frac{C_p(a,b,c)}{n^{p/(p+1)}} \left[m^{-1/p}(\alpha s + C)(\alpha^{-1}A + B)\right]^{p/(p+1)},$$

where $A = (\sum_{\ell=1}^{q} s_\ell(p) m_\ell^{-p})^{1/p}$, $B = (\sum_{\ell=1}^{q} s_\ell(p)\lambda_2(G_\ell)^{-p})^{1/p}$, and $C = \mathbf{cut}(\mathcal{L}_{\mathbf{S}}, y)$. Now optimize over α (let $\alpha = \sqrt{AC/(sB)}$), and simplify to obtain the desired inequality.

Appendix C. Dimension Reduction

As in Section 3.5, denote by $\mathbf{P}_{\mathbf{S}}^r(G)$ the projection operator onto the eigenspace of $\alpha\mathbf{S}^{-1} + \mathcal{L}_{\mathbf{S}}(G)$ corresponding to the r smallest eigenvalues, and define the following regularizer on the reduced subspace:

$$\mathbf{f}_{\cdot,k}^T \mathbf{K}^{-1} \mathbf{f}_{\cdot,k} = \begin{cases} \mathbf{f}_{\cdot,k}^T \mathbf{K}_0^{-1} \mathbf{f}_{\cdot,k} & \text{if } \mathbf{P}_{\mathbf{S}}^r(G)\mathbf{f}_{\cdot,k} = \mathbf{f}_{\cdot,k}, \\ +\infty & \text{otherwise.} \end{cases}$$

Note that in the following, we will focus on bounding the generalization complexity using the reduced dimensionality r. In such context, the choice of $\mathbf{K}_0$ is not important as far as our analysis is concerned. We may simply choose $\mathbf{K}_0 = \mathbf{I}$ (or we may let $\mathbf{K}_0^{-1} = \alpha\mathbf{S}^{-1} + \mathcal{L}_{\mathbf{S}}(G)$).

The following theorem shows that the target vectors can be well approximated by their projection via $\mathbf{P}_{\mathbf{S}}^q(G)$.

Theorem 10 *Let $G' = \cup_{\ell=1}^{q} G_\ell$ ($G_\ell = (V_\ell, E_\ell)$) be a pure subgraph of G. Consider $r \geq q$. Then we have:*

$$\lambda_{r+1}(\mathcal{L}_{\mathbf{S}}(G)) \geq \lambda_{r+1}(\mathcal{L}_{\mathbf{S}}(G')) \geq \min_{\ell} \lambda_2(\mathcal{L}_{\mathbf{S}}(G_\ell)).$$

For each k, let $\bar{f}_{j,k} = \delta_{y_j,k}$ be the target (encoding of the true labels) for class k ($j = 1,\ldots,m$). Then $\|\mathbf{P}^r_{\mathbf{S}}(G)\bar{\mathbf{f}}_{\cdot,k} - \bar{\mathbf{f}}_{\cdot,k}\|_2^2 \le \delta_r(\mathbf{S})\|\bar{\mathbf{f}}_{\cdot,k}\|_2^2$, where

$$\delta_r(\mathbf{S}) = \frac{\|\mathcal{L}_{\mathbf{S}}(G) - \mathcal{L}_{\mathbf{S}}(G')\|_2 + d(\mathbf{S})}{\lambda_{r+1}(\mathcal{L}_{\mathbf{S}}(G))}, \quad d(\mathbf{S}) = \max_{\ell} \frac{1}{2|V_\ell|} \sum_{j,j' \in V_\ell} w_{j,j'}(\mathbf{S}_j^{-1/2} - \mathbf{S}_{j'}^{-1/2})^2.$$

Proof Let $\mathbf{E} = \mathcal{L}_{\mathbf{S}}(G) - \mathcal{L}_{\mathbf{S}}(G')$, then $\mathbf{E}$ is a positive semi-definite matrix. Therefore, we know that the eigenvalues of $\mathcal{L}_{\mathbf{S}}(G)$ are no less than the corresponding eigenvalues of $\mathcal{L}_{\mathbf{S}}(G')$. This follows easily from the minimax variational formulation for the $q+1$-th smallest eigenvalue of a matrix $\mathcal{L} \in R^{m\times m}$, which can be written as

$$\max_{V_q \in R^{m\times q}: V_q^T V_q = I_{q\times q}} \min_{\mathbf{v}\in R^m: \|\mathbf{v}\|_2 = 1, V_q^T\mathbf{v} = 0} \mathbf{v}^T \mathcal{L}\mathbf{v},$$

together with the fact that $\mathbf{v}^T\mathcal{L}_{\mathbf{S}}(G)\mathbf{v} \ge \mathbf{v}^T\mathcal{L}_{\mathbf{S}}(G')\mathbf{v}$.

Since the $(q+1)$-th smallest eigenvalue of $\mathcal{L}_{\mathbf{S}}(G')$ is $\min_\ell \lambda_2(\mathcal{L}(G_\ell))$, we obtain the first displayed inequalities. Moreover,

$$\bar{\mathbf{f}}_{\cdot,k}^T\mathcal{L}_{\mathbf{S}}(G)\bar{\mathbf{f}}_{\cdot,k} = \bar{\mathbf{f}}_{\cdot,k}^T\mathbf{E}\bar{\mathbf{f}}_{\cdot,k} + \bar{\mathbf{f}}_{\cdot,k}^T\mathcal{L}_{\mathbf{S}}(G')\bar{\mathbf{f}}_{\cdot,k} \le (\|\mathbf{E}\|_2 + d(\mathbf{S}))\bar{\mathbf{f}}_{\cdot,k}^T\bar{\mathbf{f}}_{\cdot,k}.$$

Since $\mathbf{P}^r_{\mathbf{S}}(G)\bar{\mathbf{f}}_{\cdot,k}$ belongs to the subspace spanned by the smallest r eigenvectors of $\mathcal{L}_{\mathbf{S}}(G)$, and $(\mathbf{I} - \mathbf{P}^r_{\mathbf{S}}(G))\bar{\mathbf{f}}_{\cdot,k}$ belongs to the subspace spanned by the remaining eigenvectors, we obtain

$$\begin{aligned}
\lambda_{r+1}(\mathcal{L}_{\mathbf{S}}(G))\bar{\mathbf{f}}_{\cdot,k}^T(\mathbf{I} - \mathbf{P}^r_{\mathbf{S}}(G))\bar{\mathbf{f}}_{\cdot,k} \le& \bar{\mathbf{f}}_{\cdot,k}^T(\mathbf{I} - \mathbf{P}^r_{\mathbf{S}}(G))\mathcal{L}_{\mathbf{S}}(G)(\mathbf{I} - \mathbf{P}^r_{\mathbf{S}}(G))\bar{\mathbf{f}}_{\cdot,k} \\
=& \bar{\mathbf{f}}_{\cdot,k}^T\mathcal{L}_{\mathbf{S}}(G)\bar{\mathbf{f}}_{\cdot,k} - (\mathbf{P}^r_{\mathbf{S}}(G)\bar{\mathbf{f}}_{\cdot,k})^T\mathcal{L}_{\mathbf{S}}(G)(\mathbf{P}^r_{\mathbf{S}}(G)\bar{\mathbf{f}}_{\cdot,k}) \\
\le& \bar{\mathbf{f}}_{\cdot,k}^T\mathcal{L}_{\mathbf{S}}(G)\bar{\mathbf{f}}_{\cdot,k} \\
\le& [\|\mathcal{L}_{\mathbf{S}}(G) - \mathcal{L}_{\mathbf{S}}(G'))\|_2 + d(\mathbf{S})]\|\bar{\mathbf{f}}_{\cdot,k}\|_2^2.
\end{aligned}$$

The result follows from the observation that $\bar{\mathbf{f}}_{\cdot,k}^T(\mathbf{I} - \mathbf{P}^r_{\mathbf{S}}(G))\bar{\mathbf{f}}_{\cdot,k} = \|\bar{\mathbf{f}}_{\cdot,k} - \mathbf{P}^r_{\mathbf{S}}(G)\bar{\mathbf{f}}_{\cdot,k}\|_2^2$. ■

In Theorem 10, normalization plays a direct role because $\mathbf{S}$ affects $\delta_r(\mathbf{S})$. Note that the eigenvalue inequality implies that $\delta_r(\mathbf{S})$ can be bounded by $1/\min_\ell \lambda_2(\mathcal{L}_{\mathbf{S}}(G_\ell))$. If $\mathcal{L}_{\mathbf{S}}(G) \approx \mathcal{L}_{\mathbf{S}}(G')$, then $d(\mathbf{S}) \approx 0$, which means that $\mathbf{S}_j \approx \mathbf{S}_{j'}$ if j and j' belongs to the same V_ℓ. Moreover, $\lambda_2(\mathcal{L}_{\mathbf{S}}(G_\ell))$ is approximately a constant; otherwise, we may reduce the largest eigenvalue $\lambda_2(G_\ell)$ by increasing the corresponding scaling factor, which reduces $\|\mathcal{L}_{\mathbf{S}}(G) - \mathcal{L}_{\mathbf{S}}(G'))\|_2$, thus reduces $\delta_r(\mathbf{S})$. Using reasoning that is analogous to Section 3.4, we can obtain similar conclusion under the condition $\mathcal{L}_{\mathbf{S}}(G) \approx \mathcal{L}_{\mathbf{S}}(G')$. That is, $\delta_r(\mathbf{S})$ is approximately minimized when $\mathbf{S}_j = \bar{s}_\ell = m_\ell$ for each $j \in V_\ell$.

Similar to Theorem 5, we can prove the following generalization bound using Theorem 10. For simplicity, we only consider a simple kernel $\mathbf{K}_0 = \mathbf{I}$, and take $p = 1$.

Theorem 11 *Let the assumptions of Theorem 10 hold. Consider the least squares loss $\phi(\mathbf{f}_j, y_j) = \sum_{k=1}^K (f_{j,k} - \delta_{k,y_j})^2$ in (1) using the regularization condition (6) and $\mathbf{K}_0 = \mathbf{I}$. The generalization error with optimal λ can be bounded as:*

$$\mathbf{E}_{Z_n} \frac{1}{m-n} \sum_{j \in Z_n} \mathbf{err}(\hat{\mathbf{f}}_j, y_j) \le 16\delta_r(\mathbf{S}) + 8\sqrt{r/n}.$$

Proof Using Theorem 10, it can be easily verified that

$$\frac{1}{m}\sum_{j=1}^{m}\phi(\mathbf{P}^r_{\mathbf{S}}(G)\bar{\mathbf{f}}_j,y_j)+\lambda\sum_{k=1}^{K}(\mathbf{P}^r_{\mathbf{S}}(G)\bar{\mathbf{f}}_{\cdot,k})^T\mathbf{K}^{-1}(\mathbf{P}^r_{\mathbf{S}}(G)\bar{\mathbf{f}}_{\cdot,k})\leq\delta_r(\mathbf{S})+\lambda m.$$

Since regularizing with $\mathbf{K}_0=\mathbf{I}$ is equivalent with regularizing with $\mathbf{K}_0=\mathbf{P}^r_{\mathbf{S}}(G)$, we can use $\mathbf{tr}(\mathbf{K})=r$. Now using this estimate in Theorem 1, we have

$$\mathbf{E}_{Z_n}\frac{1}{m-n}\sum_{j\in\bar{Z}_n}\mathbf{err}(\hat{\mathbf{f}}_j,y_j)\leq 16(\delta_r(\mathbf{S})+\lambda m)+\frac{r}{\lambda nm}.$$

Optimizing over λ gives the desired bound. ■

Similar to Theorem 7, it is possible to prove a bound for general p in Theorem 11, but the estimation of $\mathbf{tr}_p(\mathbf{K})$ is more complicated than that of $\mathbf{tr}(\mathbf{K})$. We skip the derivation because the extra complication is not important for the purpose of this paper. Compared to Theorem 7, the advantage of dimension reduction in Theorem 11 is that the quantity $\mathbf{cut}(\mathcal{L}_{\mathbf{S}},y)$ is replaced by $\|\mathcal{L}_{\mathbf{S}}(G)-\mathcal{L}_{\mathbf{S}}(G')\|_2$, which is typically much smaller. Instead of a rigorous analysis, we shall just give a brief intuition. For simplicity we take $\mathbf{S}=\mathbf{I}$ so that we can ignore the variations caused by $\mathbf{S}$. The 2-norm of the symmetric error matrix $\mathcal{L}_{\mathbf{S}}(G)-\mathcal{L}_{\mathbf{S}}(G')$ is its largest eigenvalue, which is no more than the largest 1-norm of one of its row vectors. In contrast, $\mathbf{cut}(\mathcal{L}_{\mathbf{S}},y)$ behaves similar to the absolute sum of entries of the error matrix, which is m times more than the averaged 1-norm of its row vectors. Therefore if error is relatively uniform across rows, then $\mathbf{cut}(\mathcal{L}_{\mathbf{S}},y)$ can be at an order of m times more than $\|\mathcal{L}_{\mathbf{S}}(G)-\mathcal{L}_{\mathbf{S}}(G')\|_2$.

References

Rie K. Ando and Tong Zhang. Learning on graph with Laplacian regularization. In *NIPS 19*, 2007.

Mikhail Belkin and Partha Niyogi. Semi-supervised learning on Riemannian manifolds. *Machine Learning*, Special Issue on Clustering:209–239, 2004.

Fan R.K. Chung. *Spectral Graph Theory*. Regional Conference Series in Mathematics. American Mathematical Society, Rhode Island, 1998.

Chris Ding, Xiaofeng He, Hongyuan Zha, Ming Gu, and Horst Simon. A min-max cut algorithm for graph partitioning and data clustering. In *IEEE Int'l Conf. Data Mining*, pages 107–114, 2001.

Daniel B. Graham and Nigel M. Allinson. Characterizing virtual eigensignatures for general purpose face recognition. *Face Recognition: From Theory to Applications, NATO ASI Series F, Computer and Systems Sciences*, 163:446–456, 1998.

Lars Hagen and Andrew B. Kahng. New spectral methods for ratio cut partitioning and clustering. *IEEE Transactions on Computer-Aided Design*, 11(9):1074–1085, 1992.

Gert R.G. Lanckriet, Nello Cristianini, Peter Bartlett, Laurent El Ghaoui, and Michael I. Jordan. Learning the kernel matrix with semidefinite programming. *Journal of Machine Learning Research*, 5:27–72, 2004.

Marina Meilă, Susan Shortreed, and Liang Xu. Regularized spectral learning. In *AISTATS*, 2005.

Andrew Y. Ng, Michael I. Jordan, and Yair Weiss. On spectral clustering: Analysis and an algorithm. In *NIPS 14*, pages 849–856, 2002.

Ralph Tyrrell Rockafellar. *Convex analysis*. Princeton University Press, Princeton, NJ, 1970.

Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *IEEE Trans. Pattern Anal. Mach. Intell*, 22:888–905, 2000.

Martin Szummer and Tommi Jaakkola. Partially labeled classification with Markov random walks. In *Advances in Neural Information Processing Systems 14*, 2002.

Ulrike von Luxburg, Olivier Bousquet, and Mikhail Belkin. Limits of spectral clustering. In Lawrence K. Saul, Yair Weiss, and Léon Bottou, editors, *Advances in Neural Information Processing Systems 17*, pages 857–864. MIT Press, Cambridge, MA, 2005.

Eric P. Xing, Andrew Y. Ng, Michael I. Jordan, and Stuart Russell. Distance metric learning, with application to clustering with side-information. In *NIPS 15*, 2003.

Tong Zhang. Leave-one-out bounds for kernel methods. *Neural Computation*, 15:1397–1437, 2003.

Tong Zhang and Rie K. Ando. Analysis of spectral kernel design based semi-supervised learning. In *NIPS 18*, 2006.

Dengyong Zhou, Olivier Bousquet, Thomas Navin Lal, Jason Weston, and Bernhard Schlkopf. Learning with local and global consistency. In *Advances in Neural Information Processing Systems 16*, pages 321–328, 2004.

Xiaojin Zhu, Zoubin Ghahramani, and John Lafferty. Semi-supervised learning using Gaussian fields and harmonic functions. In *ICML 2003*, 2003.

Journal of Machine Learning Research 8 (2007) 1519-1555 Submitted 7/06; Revised 1/07; Published 7/07

An Interior-Point Method for Large-Scale ℓ_1-Regularized Logistic Regression

Kwangmoo Koh DENEB1@STANFORD.EDU
Seung-Jean Kim SJKIM@STANFORD.EDU
Stephen Boyd BOYD@STANFORD.EDU
Information Systems Laboratory
Electrical Engineering Department
Stanford University
Stanford, CA 94305-9510, USA

Editor: Yi Lin

Abstract

Logistic regression with ℓ_1 regularization has been proposed as a promising method for feature selection in classification problems. In this paper we describe an efficient interior-point method for solving large-scale ℓ_1-regularized logistic regression problems. Small problems with up to a thousand or so features and examples can be solved in seconds on a PC; medium sized problems, with tens of thousands of features and examples, can be solved in tens of seconds (assuming some sparsity in the data). A variation on the basic method, that uses a preconditioned conjugate gradient method to compute the search step, can solve very large problems, with a million features and examples (e.g., the 20 Newsgroups data set), in a few minutes, on a PC. Using warm-start techniques, a good approximation of the entire regularization path can be computed much more efficiently than by solving a family of problems independently.

Keywords: logistic regression, feature selection, ℓ_1 regularization, regularization path, interior-point methods.

1. Introduction

In this section we describe the basic logistic regression problem, the ℓ_2- and ℓ_1-regularized versions, and the regularization path. We set out our notation, and review existing methods and literature. Finally, we give an outline of this paper.

1.1 Logistic Regression

Let $x \in \mathbf{R}^n$ denote a vector of explanatory or feature variables, and $b \in \{-1,+1\}$ denote the associated binary output or outcome. The logistic model has the form

$$\mathrm{Prob}(b|x) = \frac{1}{1+\exp\left(-b(w^T x+v)\right)} = \frac{\exp\left(b(w^T x+v)\right)}{1+\exp\left(b(w^T x+v)\right)},$$

where $\mathrm{Prob}(b|x)$ is the conditional probability of b, given $x \in \mathbf{R}^n$. The logistic model has parameters $v \in \mathbf{R}$ (the intercept) and $w \in \mathbf{R}^n$ (the weight vector). When $w \neq 0$, $w^T x + v = 0$ defines the neutral hyperplane in feature space, on which the conditional probability of each outcome is $1/2$. On the shifted parallel hyperplane $w^T x + v = 1$, which is a distance $1/\|w\|_2$ from the neutral hyperplane,

the conditional probability of outcome $b = 1$ is $1/(1+1/e) \approx 0.73$, and the conditional probability of $b = -1$ is $1/(1+e) \approx 0.27$. On the hyperplane $w^T x + v = -1$, these conditional probabilities are reversed. As $w^T x + v$ increases above one, the conditional probability of outcome $b = 1$ rapidly approaches one; as $w^T x + v$ decreases below -1, the conditional probability of outcome $b = -1$ rapidly approaches one. The slab in feature space defined by $|w^T x + v| \leq 1$ defines the *ambiguity region*, in which there is substantial probability of each outcome; outside this slab, one outcome is much more likely than the other.

Suppose we are given a set of (observed or training) examples,

$$(x_i, b_i) \in \mathbf{R}^n \times \{-1, +1\}, \quad i = 1, \ldots, m,$$

assumed to be independent samples from a distribution. We use $p_{\log}(v, w) \in \mathbf{R}^m$ to denote the vector of conditional probabilities, according to the logistic model,

$$p_{\log}(v, w)_i = \text{Prob}(b_i | x_i) = \frac{\exp(w^T a_i + v b_i)}{1 + \exp(w^T a_i + v b_i)}, \quad i = 1, \ldots, m,$$

where $a_i = b_i x_i$. The likelihood function associated with the samples is $\prod_{i=1}^m p_{\log}(v, w)_i$, and the log-likelihood function is given by

$$\sum_{i=1}^m \log p_{\log}(v, w)_i = -\sum_{i=1}^m f(w^T a_i + v b_i),$$

where f is the *logistic loss function*

$$f(z) = \log(1 + \exp(-z)). \tag{1}$$

This loss function is convex, so the log-likelihood function is concave. The negative of the log-likelihood function is called the *(empirical) logistic loss*, and dividing by m we obtain the *average logistic loss*,

$$l_{\text{avg}}(v, w) = (1/m) \sum_{i=1}^m f(w^T a_i + v b_i).$$

We can determine the model parameters w and v by maximum likelihood estimation from the observed examples, by solving the convex optimization problem

$$\text{minimize} \quad l_{\text{avg}}(v, w), \tag{2}$$

with variables $v \in \mathbf{R}$ and $w \in \mathbf{R}^n$, and problem data $A = [a_1 \ \cdots \ a_m]^T \in \mathbf{R}^{m \times n}$ and the vector of binary outcomes $b = [b_1 \ \cdots \ b_m]^T \in \mathbf{R}^m$. The problem (2) is called the *logistic regression problem* (LRP).

The average logistic loss is always nonnegative, that is, $l_{\text{avg}}(v, w) \geq 0$, since $f(z) \geq 0$ for any z. For the choice $w = 0$, $v = 0$, we have $l_{\text{avg}}(0, 0) = \log 2 \approx 0.693$, so the optimal value of the LRP lies between 0 and $\log 2$. In particular, the optimal value can range (roughly) between 0 and 1. The optimal value is 0 only when the original data are linearly separable, that is, there exist w and v such that $w^T x_i + v > 0$ when $b_i = 1$, and $w^T x_i + v < 0$ when $b_i = -1$. In this case the optimal value of the LRP (2) is not achieved (except in the limit with w and v growing arbitrarily large). The optimal value is $\log 2$, that is, $w = 0$, $v = 0$ are optimal, only if $\sum_{i=1}^m b_i = 0$ and $\sum_{i=1}^m a_i = 0$. (This follows from the expression for ∇l_{avg}, given in Section 2.1.) This occurs only when the number of positive

examples (i.e., those for which $b_i = 1$) is equal to the number of negative examples, and the average of x_i over the positive examples is the negative of the average value of x_i over the negative examples.

The LRP (2) is a smooth convex optimization problem, and can be solved by a wide variety of methods, such as gradient descent, steepest descent, Newton, quasi-Newton, or conjugate-gradients (CG) methods (see, for example Hastie et al., 2001, § 4.4).

Once we find maximum likelihood values of v and w, that is, a solution of (2), we can predict the probability of the two possible outcomes, given a new feature vector $x \in \mathbf{R}^n$, using the associated logistic model. For example, when $w \neq 0$, we can form the logistic classifier

$$\phi(x) = \mathrm{sgn}(w^T x + v), \tag{3}$$

where

$$\mathrm{sgn}(z) = \begin{cases} +1 & z > 0 \\ -1 & z \leq 0, \end{cases}$$

which picks the more likely outcome, given x, according to the logistic model. This classifier is linear, meaning that the boundary between the two decision outcomes is a hyperplane (defined by $w^T x + v = 0$).

1.2 ℓ_2-Regularized Logistic Regression

When m, the number of observations or training examples, is not large enough compared to n, the number of feature variables, simple logistic regression leads to over-fit. That is, the classifier found by solving the LRP (2) performs perfectly (or very well) on the training examples, but may perform poorly on unseen examples. Over-fitting tends to occur when the fitted model has many feature variables with (relatively) large weights in magnitude, that is, w is large.

A standard technique to prevent over-fitting is *regularization*, in which an extra term that penalizes large weights is added to the average logistic loss function. The *ℓ_2-regularized logistic regression problem* is

$$\text{minimize} \quad l_{\mathrm{avg}}(v,w) + \lambda \|w\|_2^2 = (1/m) \textstyle\sum_{i=1}^m f(w^T a_i + v b_i) + \lambda \sum_{i=1}^n w_i^2. \tag{4}$$

Here $\lambda > 0$ is the regularization parameter, used to control the trade-off between the average logistic loss and the size of the weight vector, as measured by the ℓ_2-norm. No penalty term is imposed on the intercept, since it is a parameter for thresholding the weighted sum $w^T x$ in the linear classifier (3). The solution of the ℓ_2-regularized regression problem (4) (which exists and is unique) can be interpreted in a Bayesian framework, as the maximum a posteriori probability (MAP) estimate of w and v, when w has a Gaussian prior distribution on $\mathbf{R}^n$ with zero mean and covariance λI and v has the (improper) uniform prior on $\mathbf{R}$; see, for example, Chaloner and Larntz (1989) or Jaakkola and Jordan (2000).

The objective function in the ℓ_2-regularized LRP is smooth and convex, and so (like the ordinary, unregularized LRP) can be minimized by standard methods such as gradient descent, steepest descent, Newton, quasi-Newton, truncated Newton, or CG methods; see, for example, Luenberger (1984), Lin et al. (2007), Minka (2003), Nocedal and Wright (1999), and Nash (2000). Other methods that have been used include optimization transfer (Krishnapuram and Hartemink, 2005; Zhang et al., 2004) and iteratively re-weighted least squares (Komarek, 2004). Newton's method and variants are very effective for small and medium sized problems, while conjugate-gradients and limited memory Newton (or truncated Newton) methods can handle very large problems. In Minka (2003)

the author compares several methods for ℓ_2-regularized LRPs with large data sets. The fastest methods turn out to be conjugate-gradients and limited memory Newton methods, outperforming IRLS, gradient descent, and steepest descent methods. Truncated Newton methods have been applied to large-scale problems in several other fields, for example, image restoration (Fu et al., 2006) and support vector machines (Keerthi and DeCoste, 2005). For large-scale iterative methods such as truncated Newton or CG, the convergence typically improves as the regularization parameter λ is increased, since (roughly speaking) this makes the objective more quadratic, and improves the conditioning of the problem.

1.3 ℓ_1-Regularized Logistic Regression

More recently, *ℓ_1-regularized logistic regression* has received much attention. The ℓ_1-regularized logistic regression problem is

$$\text{minimize} \quad l_{\text{avg}}(v,w) + \lambda \|w\|_1 = (1/m) \sum_{i=1}^m f(w^T a_i + v b_i) + \lambda \sum_{i=1}^n |w_i|, \tag{5}$$

where $\lambda > 0$ is the regularization parameter. The only difference with ℓ_2-regularized logistic regression is that we measure the size of w by its ℓ_1-norm, instead of its ℓ_2-norm. A solution of the ℓ_1-regularized logistic regression must exist, but it need not be unique. Any solution of the ℓ_1-regularized logistic regression problem (5) can be interpreted in a Bayesian framework as a MAP estimate of w and v, when w has a Laplacian prior distribution and v has the (improper) uniform prior. The objective function in the ℓ_1-regularized LRP (5) is convex, but not differentiable (specifically, when any of the weights is zero), so solving it is more of a computational challenge than solving the ℓ_2-regularized LRP (4).

Despite the additional computational challenge posed by ℓ_1-regularized logistic regression, compared to ℓ_2-regularized logistic regression, interest in its use has been growing. The main motivation is that ℓ_1-regularized LR typically yields a *sparse* vector w, that is, w typically has relatively few nonzero coefficients. (In contrast, ℓ_2-regularized LR typically yields w with all coefficients nonzero.) When $w_j = 0$, the associated logistic model does not use the jth component of the feature vector, so sparse w corresponds to a logistic model that uses only a few of the features, that is, components of the feature vector. Indeed, we can think of a sparse w as a *selection* of the relevant or important features (i.e., those associated with nonzero w_j), as well as the choice of the intercept value and weights (for the selected features). A logistic model with sparse w is, in a sense, simpler or more parsimonious than one with nonsparse w. It is not surprising that ℓ_1-regularized LR can outperform ℓ_2-regularized LR, especially when the number of observations is smaller than the number of features (Ng, 2004; Wainwright et al., 2007).

We refer to the number of nonzero components in w as its *cardinality*, denoted $\text{card}(w)$. Thus, ℓ_1-regularized LR tends to yield w with $\text{card}(w)$ small; the regularization parameter λ roughly controls $\text{card}(w)$, with larger λ typically (but not always) yielding smaller $\text{card}(w)$.

The general idea of ℓ_1 regularization for the purposes of model or feature selection (or just sparsity of solution) is quite old, and widely used in other contexts such as geophysics (Claerbout and Muir, 1973; Taylor et al., 1979; Levy and Fullagar, 1981; Oldenburg et al., 1983). In statistics, it is used in the well-known Lasso algorithm (Tibshirani, 1996) for ℓ_1-regularized linear regression, and its extensions such as the fused Lasso (Tibshirani et al., 2005), the grouped Lasso (Kim et al., 2006; Yuan and Lin, 2006; Zhao et al., 2007), and the monotone Lasso (Hastie et al., 2007). The idea also comes up in signal processing in basis pursuit (Chen and Donoho, 1994; Chen et al.,

2001), signal recovery from incomplete measurements (Candès et al., 2006, 2005; Donoho, 2006), and wavelet thresholding (Donoho et al., 1995), decoding of linear codes (Candès and Tao, 2005), portfolio optimization (Lobo et al., 2005), controller design (Hassibi et al., 1999), computer-aided design of integrated circuits (Boyd et al., 2001), computer vision (Bhusnurmath and Taylor, 2007), sparse principal component analysis (d'Aspremont et al., 2005; Zou et al., 2006), graphical model selection (Wainwright et al., 2007), maximum likelihood estimation of graphical models (Banerjee et al., 2006; Dahl et al., 2005), boosting (Rosset et al., 2004), and ℓ_1-norm support vector machines (Zhu et al., 2004). A recent survey of the idea can be found in Tropp (2006). Donoho and Elad (2003) and Tropp (2006) give some theoretical analysis of why ℓ_1 regularization leads to a sparse model in linear regression. Recently, theoretical properties of ℓ_1-regularized linear regression have been studied by several researchers; see, for example, Zou (2006), Zhao and Yu (2006), and Zou et al. (2007).

To solve the ℓ_1-regularized LRP (5), generic methods for nondifferentiable convex problems can be used, such as the ellipsoid method or subgradient methods (Shor, 1985; Polyak, 1987). These methods are usually very slow in practice, however. (Because ℓ_1-regularized LR typically results in a weight vector with (many) zero components, we cannot simply ignore the nondifferentiability of the objective in the ℓ_1-regularized LRP (5), hoping to not encounter points of nondifferentiability.)

Another approach is to transform the problem to one with differentiable objective and constraint functions. We can solve the ℓ_1-regularized LRP (5), by solving an equivalent formulation, with linear inequality constraints,

$$\begin{array}{ll} \text{minimize} & l_{\text{avg}}(v,w) + \lambda \mathbf{1}^T u \\ \text{subject to} & -u_i \leq w_i \leq u_i, \quad i = 1, \ldots, n, \end{array} \tag{6}$$

where the variables are the original ones $v \in \mathbf{R}$, $w \in \mathbf{R}^n$, along with $u \in \mathbf{R}^n$. Here $\mathbf{1}$ denotes the vector with all components one, so $\mathbf{1}^T u$ is the sum of the components of u. (To see the equivalence with the ℓ_1-regularized LRP (5), we note that at the optimal point for (6), we must have $u_i = |w_i|$, in which case the objectives in (6) and (5) are the same.) The reformulated problem (6) is a convex optimization problem, with a smooth objective, and linear constraint functions, so it can be solved by standard convex optimization methods such as SQP, augmented Lagrangian, interior-point, and other methods. High quality solvers that can directly handle the problem (6) (and therefore, carry out ℓ_1-regularized LR) include for example LOQO (Vanderbei, 1997), LANCELOT (Conn et al., 1992), MOSEK (MOSEK ApS, 2002), and NPSOL (Gill et al., 1986). These general purpose solvers can solve small and medium scale ℓ_1-regularized LRPs quite effectively.

Other recently developed computational methods for ℓ_1-regularized logistic regression include the IRLS method (Lee et al., 2006; Lokhorst, 1999), a generalized LASSO method (Roth, 2004) that extends the LASSO method proposed in Osborne et al. (2000) to generalized linear models, generalized iterative scaling (Goodman, 2004), bound optimization algorithms (Krishnapuram et al., 2005), online algorithms (Balakrishnan and Madigan, 2006; Perkins and Theiler, 2003), coordinate descent methods (Friedman et al., 2007; Genkin et al., 2006), and the Gauss-Seidel method (Shevade and Keerthi, 2003). Some of these methods can handle very large problems (assuming some sparsity in the data) with modest accuracy. But the additional computational cost required for these methods to achieve higher accuracy can be very large.

The main goal of this paper is to describe a specialized interior-point method for solving the ℓ_1-regularized logistic regression problem that is very efficient, for all size problems. In particular our method handles very large problems, attains high accuracy, and is not much slower than the fastest

large-scale methods (conjugate-gradients and limited memory Newton) applied to the ℓ_2-regularized LRP.

Numerical experiments show that our method is as fast as, or faster than, other methods, and reliably provides very accurate solutions. Compared with high-quality implementations of general purpose primal-dual interior-point methods, our method is far faster, especially for large problems. Compared with first-order methods such as coordinate descent methods, our method is comparable in solving large problems with modest accuracy, but is able to solve them with high accuracy with relatively small additional computational cost.

In this paper we focus on methods for *solving* the ℓ_1-regularized LRP; we do not discuss the benefits or advantages of ℓ_1-regularized LR, compared to ℓ_2-regularized LR or other methods for modeling or constructing classifiers for two-class data.

1.4 Regularization Path

Let $(v_\lambda^\star, w_\lambda^\star)$ be a solution for the ℓ_1-regularized LRP with regularization parameter λ. The family of solutions, as λ varies over $(0,\infty)$, is called the (ℓ_1-) *regularization path*. In many applications, the regularization path (or some portion) needs to be computed, in order to determine an appropriate value of λ. At the very least, the ℓ_1-regularized LRP must be solved for multiple, and often many, values of λ.

In ℓ_1-regularized linear regression, which is the problem of minimizing

$$\|Fz-g\|_2^2+\lambda\|z\|_1$$

over the variable z, where $\lambda>0$ is the regularization parameter, $F\in\mathbf{R}^{p\times n}$ is the covariate matrix, and $g\in\mathbf{R}^p$ is the vector of responses, it can be shown that the regularization path is piecewise linear, with kinks at each point where any component of the variable z transitions from zero to nonzero, or vice versa. Using this fact, the entire regularization path in a (small or medium size) ℓ_1-regularized linear regression problem can be computed efficiently (Hastie et al., 2004; Efron et al., 2004; Rosset, 2005; Rosset and Zhu, 2007; Osborne et al., 2000). These methods are related to numerical continuation techniques for following piecewise smooth curves, which have been well studied in the optimization literature (Allgower and Georg, 1993).

Path following methods have been applied to several problems (Hastie et al., 2004; Park and Hastie, 2006a,b; Rosset, 2005). Park and Hastie (2006a) describe an algorithm for (approximately) computing the entire regularization path for general linear models (GLMs) including logistic regression models. In Rosset (2005), a general path-following method based on a predictor-corrector method is described for general regularized convex loss minimization problems. Path-following methods can be slow for large-scale problems, where the number of kinks or events is very large (at least n). When the number of kinks on the portion of the regularization path of interest is modest, however, these path-following methods can be very fast, requiring just a small multiple of the effort needed to solve one regularized problem to compute the whole path (or a portion).

In this paper we describe a fast method for computing a large number of points on the regularization path, using a warm-start technique and our interior-point method. Unlike the methods mentioned above, our method does not attempt to track the path exactly (i.e., jumping from kink to kink on the path); it remains efficient even when successive values of λ jump over many kinks. This is essential when computing the regularization path in a large-scale problem. Our method allows us to compute a large number of points (but much fewer than n, when n is very large) on the

regularization path, much more efficiently than by solving a family of the problems independently. Our method is far more efficient than path following methods in computing a good approximation of the regularization for a medium-sized or large data set.

1.5 Outline

In Section 2, we give (necessary and sufficient) optimality conditions, and a dual problem, for the ℓ_1-regularized LRP. Using the dual problem, we show how to compute a lower bound on the suboptimality of any pair (v,w). We describe our basic interior-point method in Section 3, and demonstrate its performance in Section 4 with small and medium scale synthetic and machine learning benchmark examples. We show that ℓ_1-regularized LR can be carried out within around 35 or so iterations, where each iteration has the same complexity as solving an ℓ_2-regularized linear regression problem.

In Section 5, we describe a variation on our basic method that uses a preconditioned conjugate gradient approach to compute the search direction. This variation on our method can solve very large problems, with a million features and examples (e.g., the 20 Newsgroups data set), in under an hour, on a PC, provided the data matrix is sufficiently sparse.

In Section 6, we consider the problem of computing the regularization path efficiently, at a variety of values of λ (but potentially far fewer than the number of kinks on the path). Using warm-start techniques, we show how this can done much more efficiently than by solving a family of problems independently. In Section 7, we compare the interior-point method with several existing methods for ℓ_1-regularized logistic regression. In Section 8, we describe generalizations of our method to other ℓ_1-regularized convex loss minimization problems.

2. Optimality Conditions and Dual

In this section we derive a necessary and sufficient optimality condition for the ℓ_1-regularized LRP, as well as a Lagrange dual problem, from which we obtain a lower bound on the objective that we will use in our algorithm.

2.1 Optimality Conditions

The objective function of the ℓ_1-regularized LRP, $l_{\rm avg}(v,w)+\lambda\|w\|_1$, is convex but not differentiable, so we use a first-order optimality condition based on subdifferential calculus (see Bertsekas, 1999, Prop. B.24 or Borwein and Lewis, 2000, §2). The average logistic loss is differentiable, with

$$\begin{aligned}\nabla_v l_{\rm avg}(v,w) &= (1/m)\sum_{i=1}^m f'(w^T a_i + v b_i) b_i \\ &= -(1/m)\sum_{i=1}^m (1 - p_{\log}(v,w)_i) b_i \\ &= -(1/m) b^T (\mathbf{1} - p_{\log}(v,w)),\end{aligned}$$

and

$$\begin{aligned}\nabla_w l_{\mathrm{avg}}(v,w) &= (1/m)\sum_{i=1}^m f'(w^T a_i + v b_i) a_i \\ &= -(1/m)\sum_{i=1}^m (1 - p_{\log}(v,w)_i) a_i \\ &= -(1/m) A^T (\mathbf{1} - p_{\log}(v,w)).\end{aligned}$$

The subdifferential of $\|w\|_1$ is given by

$$(\partial \|w\|_1)_i = \begin{cases} \{1\} & w_i > 0, \\ \{-1\} & w_i < 0, \\ [-1,1] & w_i = 0. \end{cases}$$

The necessary and sufficient condition for (v,w) to be optimal for the ℓ_1-regularized LRP (5) is

$$\nabla_v l_{\mathrm{avg}}(v,w) = 0, \qquad 0 \in \nabla_w l_{\mathrm{avg}}(v,w) + \lambda \partial \|w\|_1,$$

which can be expressed as

$$b^T (\mathbf{1} - p_{\log}(v,w)) = 0, \tag{7}$$

and

$$(1/m)\left(A^T(\mathbf{1} - p_{\log}(v,w))\right)_i \in \begin{cases} \{+\lambda\} & w_i > 0, \\ \{-\lambda\} & w_i < 0, \\ [-\lambda,\lambda] & w_i = 0, \end{cases} \qquad i = 1,\ldots,n. \tag{8}$$

Let us analyze when a pair of the form $(v,0)$ is optimal. This occurs if and only if

$$b^T (\mathbf{1} - p_{\log}(v,0)) = 0, \qquad \|(1/m) A^T (\mathbf{1} - p_{\log}(v,0))\|_\infty \le \lambda.$$

The first condition is equivalent to $v = \log(m_+/m_-)$, where m_+ is the number of training examples with outcome 1 (called positive) and m_- is the number of training examples with outcome -1 (called negative). Using this value of v, the second condition becomes

$$\lambda \ge \lambda_{\max} = \|(1/m) A^T (\mathbf{1} - p_{\log}(\log(m_+/m_-),0))\|_\infty.$$

The number $\lambda_{\max}$ gives us an upper bound on the useful range of the regularization parameter λ: For any larger value of λ, the logistic model obtained from ℓ_1-regularized LR has weight zero (and therefore has no ability to classify). Put another way, for $\lambda \ge \lambda_{\max}$, we get a maximally sparse weight vector, that is, one with $\mathrm{card}(w) = 0$.

We can give a more explicit formula for $\lambda_{\max}$:

$$\lambda_{\max} = (1/m)\left\| \frac{m_-}{m} \sum_{b_i=1} a_i + \frac{m_+}{m} \sum_{b_i=-1} a_i \right\|_\infty = (1/m)\left\| X^T \tilde{b} \right\|_\infty,$$

where

$$\tilde{b}_i = \begin{cases} m_-/m & b_i = 1 \\ -m_+/m & b_i = -1, \end{cases} \qquad i = 1,\ldots,m.$$

Thus, $\lambda_{\max}$ is a maximum correlation between the individual features and the (weighted) output vector $\tilde{b}$. When the features have been standardized, we have $\sum_{i=1}^m x_i = 0$, so we get the simplified expression

$$\lambda_{\max} = (1/m)\left\| \sum_{b_i=1} x_i \right\|_\infty = (1/m)\left\| \sum_{b_i=-1} x_i \right\|_\infty.$$

2.2 Dual Problem

To derive a Lagrange dual of the ℓ_1-regularized LRP (5), we first introduce a new variable $z \in \mathbf{R}^m$, as well as new equality constraints $z_i = w^T a_i + v b_i$, $i = 1, \ldots, m$, to obtain the equivalent problem

$$\begin{array}{ll} \text{minimize} & (1/m)\sum_{i=1}^m f(z_i) + \lambda \|w\|_1 \\ \text{subject to} & z_i = w^T a_i + v b_i, \quad i = 1, \ldots, m. \end{array} \tag{9}$$

Associating dual variables $\theta_i \in \mathbf{R}$ with the equality constraints, the Lagrangian is

$$L(v, w, z, \theta) \;=\; (1/m)\sum_{i=1}^m f(z_i) + \lambda \|w\|_1 + \theta^T(-z + Aw + bv).$$

The dual function is

$$\begin{aligned} \inf_{v,w,z} L(v, w, z, \theta) &= (1/m)\inf_z \sum_{i=1}^m (f(z_i) - m\theta_i z_i) + \inf_w \left(\lambda \|w\|_1 + \theta^T A w\right) + \inf_v \theta^T b v \\ &= \begin{cases} -(1/m)\sum_{i=1}^m f^*(-m\theta_i) & \|A^T\theta\|_\infty \le \lambda, \quad b^T\theta = 0, \\ -\infty & \text{otherwise,} \end{cases} \end{aligned}$$

where f^* is the *conjugate* of the logistic loss function f:

$$f^*(y) = \sup_{u \in \mathbf{R}} (yu - f(u)) = \begin{cases} (-y)\log(-y) + (1+y)\log(1+y), & -1 < y < 0 \\ 0 & y = -1 \text{ or } y = 0 \\ \infty, & \text{otherwise.} \end{cases}$$

For general background on convex duality and conjugates, see, for example, Boyd and Vandenberghe (2004, Chap. 5) or Borwein and Lewis (2000).

Thus, we have the following Lagrange dual of the ℓ_1-regularized LRP (5):

$$\begin{array}{ll} \text{maximize} & G(\theta) \\ \text{subject to} & \|A^T\theta\|_\infty \le \lambda, \quad b^T\theta = 0, \end{array} \tag{10}$$

where

$$G(\theta) = -(1/m)\sum_{i=1}^m f^*(-m\theta_i)$$

is the dual objective. The dual problem (10) is a convex optimization problem with variable $\theta \in \mathbf{R}^m$, and has the form of an ℓ_∞-norm constrained maximum generalized entropy problem. We say that $\theta \in \mathbf{R}^m$ is *dual feasible* if it satisfies $\|A^T\theta\|_\infty \le \lambda$, $b^T\theta = 0$.

From standard results in convex optimization we have the following.

- *Weak duality.* Any dual feasible point θ gives a lower bound on the optimal value $p^\star$ of the (primal) ℓ_1-regularized LRP (5):
$$G(\theta) \le p^\star. \tag{11}$$
- *Strong duality.* The ℓ_1-regularized LRP (5) satisfies a variation on Slater's constraint qualification, so there is an optimal solution of the dual (10) $\theta^\star$, which satisfies
$$G(\theta^\star) = p^\star.$$
In other words, the optimal values of the primal (5) and dual (10) are equal.

We can relate a primal optimal point $(v^\star, w^\star)$ and a dual optimal point $\theta^\star$ to the optimality conditions (7) and (8). They are related by

$$\theta^\star = (1/m)(\mathbf{1} - p_{\log}(v^\star, w^\star)).$$

We also note that the dual problem (10) can be derived starting from the equivalent problem (6), by introducing new variables z_i (as we did in (9)), and associating dual variables $\theta_+ \geq 0$ for the inequalities $w \leq u$, and $\theta_- \geq 0$ for the inequalities $-u \leq w$. By identifying $\theta = \theta_+ - \theta_-$ we obtain the dual problem (10).

2.3 Suboptimality Bound

We now derive an easily computed bound on the suboptimality of a pair (v, w), by constructing a dual feasible point $\bar{\theta}$ from an arbitrary w. Define $\bar{v}$ as

$$\bar{v} = \arg\min_v l_{\text{avg}}(v, w), \tag{12}$$

that is, $\bar{v}$ is the optimal intercept for the weight vector w, characterized by $b^T(\mathbf{1} - p_{\log}(\bar{v}, w)) = 0$. Now, we define $\bar{\theta}$ as

$$\bar{\theta} = (s/m)(\mathbf{1} - p_{\log}(\bar{v}, w)), \tag{13}$$

where the scaling constant s is

$$s = \min\left\{m\lambda / \|A^T(\mathbf{1} - p_{\log}(\bar{v}, w))\|_\infty, 1\right\}.$$

Evidently $\bar{\theta}$ is dual feasible, so $G(\bar{\theta})$ is a lower bound on $p^\star$, the optimal value of the ℓ_1-regularized LRP (5).

To compute the lower bound $G(\bar{\theta})$, we first compute $\bar{v}$. This is a one-dimensional smooth convex optimization problem, which can be solved very efficiently, for example, by a bisection method on the optimality condition

$$b^T(\mathbf{1} - p_{\log}(v, w)) = 0,$$

since the lefthand side is a monotone function of v. Newton's method can be used to ensure extremely fast terminal convergence to $\bar{v}$. From $\bar{v}$, we compute $\bar{\theta}$ using (13), and then evaluate the lower bound $G(\bar{\theta})$.

The difference between the primal objective value of (v, w), and the associated lower bound $G(\bar{\theta})$, is called the *duality gap*, and denoted η:

$$\begin{array}{rcl} \eta(v, w) & = & l_{\text{avg}}(v, w) + \lambda\|w\|_1 - G(\bar{\theta}) \\ & = & (1/m)\sum_{i=1}^m \left(f(w^T a_i + v b_i) + f^*(-m\theta_i)\right) + \lambda\|w\|_1. \end{array} \tag{14}$$

We always have $\eta \geq 0$; and (by weak duality (11)) the point (v, w) is no more than η-suboptimal. At the optimal point $(v^\star, w^\star)$, we have $\eta = 0$.

3. An Interior-Point Method

In this section we describe an interior-point method for solving the ℓ_1-regularized LRP (5), in the equivalent formulation

$$\begin{array}{ll} \text{minimize} & l_{\text{avg}}(v, w) + \lambda \mathbf{1}^T u \\ \text{subject to} & -u_i \leq w_i \leq u_i, \quad i = 1, \ldots, n, \end{array}$$

with variables $w, u \in \mathbf{R}^n$ and $v \in \mathbf{R}$.

3.1 Logarithmic Barrier and Central Path

The *logarithmic barrier* for the bound constraints $-u_i \leq w_i \leq u_i$ is

$$\Phi(w,u) = -\sum_{i=1}^{n} \log(u_i + w_i) - \sum_{i=1}^{n} \log(u_i - w_i) = -\sum_{i=1}^{n} \log(u_i^2 - w_i^2),$$

with domain

$$\mathbf{dom}\ \Phi = \{(w,u) \in \mathbf{R}^n \times \mathbf{R}^n \mid |w_i| < u_i,\ i = 1,\ldots,n\}.$$

The logarithmic barrier function is smooth and convex. We augment the weighted objective function by the logarithmic barrier, to obtain

$$\phi_t(v,w,u) = t l_{\mathrm{avg}}(v,w) + t\lambda \mathbf{1}^T u + \Phi(w,u),$$

where $t > 0$ is a parameter. This function is smooth, strictly convex, and bounded below, and so has a unique minimizer which we denote $(v^\star(t), w^\star(t), u^\star(t))$. This defines a curve in $\mathbf{R} \times \mathbf{R}^n \times \mathbf{R}^n$, parametrized by t, called the *central path*. (See Boyd and Vandenberghe, 2004, Chap. 11 for more on the central path and its properties.)

With the point $(v^\star(t), w^\star(t), u^\star(t))$ we associate

$$\theta^\star(t) = (1/m)(\mathbf{1} - p_{\log}(v^\star(t), w^\star(t))),$$

which can be shown to be dual feasible. (Indeed, it coincides with the dual feasible point $\bar{\theta}$ constructed from $w^\star(t)$ using the method of Section 2.3.) The associated duality gap satisfies

$$l_{\mathrm{avg}}(v^\star(t), w^\star(t)) + \lambda \|w^\star(t)\|_1 - G(\theta^\star(t)) \leq l_{\mathrm{avg}}(v^\star(t), w^\star(t)) + \lambda \mathbf{1}^T u^\star(t) - G(\theta^\star(t)) = 2n/t.$$

In particular, $(v^\star(t), w^\star(t))$ is no more than $2n/t$-suboptimal, so the central path leads to an optimal solution.

In a primal interior-point method, we compute a sequence of points on the central path, for an increasing sequence of values of t, using Newton's method to minimize $\phi_t(v,w,u)$, starting from the previously computed central point. A typical method uses the sequence $t = t_0, \mu t_0, \mu^2 t_0, \ldots$, where μ is between 2 and 50 (see Boyd and Vandenberghe, 2004, §11.3). The method can be terminated when $2n/t \leq \varepsilon$, since then we can guarantee ε-suboptimality of $(v^\star(t), w^\star(t))$. The reader is referred to Nesterov and Nemirovsky (1994), Wright (1997), and Ye (1997) for more on (primal) interior-point methods.

3.2 A Custom Interior-Point Method

Using our method for cheaply computing a dual feasible point and associated duality gap for *any* (v,w) (and not just for (v,w) on the central path, as in the general case), we can construct a custom interior-point method that updates the parameter t at each iteration.

CUSTOM INTERIOR-POINT METHOD FOR ℓ_1-REGULARIZED LR.

given tolerance $\varepsilon > 0$, line search parameters $\alpha \in (0, 1/2)$, $\beta \in (0,1)$

Set initial values. $t := 1/\lambda$, $v := \log(m_+/m_-)$, $w := 0$, $u := \mathbf{1}$.

repeat

1. *Compute search direction*.
   Solve the Newton system $\nabla^2\phi_t(v,w,u)\begin{bmatrix}\Delta v\\ \Delta w\\ \Delta u\end{bmatrix} = -\nabla\phi_t(v,w,u)$.
2. *Backtracking line search*. Find the smallest integer $k \geq 0$ that satisfies
   $\phi_t(v+\beta^k\Delta v, w+\beta^k\Delta w, u+\beta^k\Delta u) \leq \phi_t(v,w,u) + \alpha\beta^k\nabla\phi_t(v,w,u)^T\begin{bmatrix}\Delta v\\ \Delta w\\ \Delta u\end{bmatrix}$.
3. *Update*. $(v,w,u) := (v,w,u) + \beta^k(\Delta v, \Delta w, \Delta u)$.
4. Set $v := \bar{v}$, the optimal value of the intercept, as in (12).
5. Construct dual feasible point θ from (13).
6. Evaluate duality gap η from (14).
7. **quit** if $\eta \leq \varepsilon$.
8. *Update* t.

This description is complete, except for the rule for updating the parameter t, which will be described below. Our choice of initial values for v, w, u, and t can be explained as follows. The choice $w = 0$ and $u = \mathbf{1}$ seems to work very well, especially when the original data are standardized. The choice $v = \log(m_+/m_-)$ is the optimal value of v when $w = 0$ and $u = \mathbf{1}$, and the choice $t = 1/\lambda$ minimizes $\|(1/t)\nabla\phi_t(\log(m_+/m_-), 0, \mathbf{1})\|_2$. (In any case, the choice of the initial values does not greatly affect performance.) The construction of a dual feasible point and duality gap, in steps 4–6, is explained in Section 2.3. Typical values for the line search parameters are $\alpha = 0.01$, $\beta = 0.5$, but here too, these parameter values do not have a large effect on performance. The computational effort per iteration is dominated by step 1, the search direction computation.

There are many possible update rules for the parameter t. In a classical primal barrier method, t is held constant until ϕ_t is (approximately) minimized, that is, $\|\nabla\phi_t\|_2$ is small; when this occurs, t is increased by a factor typically between 2 and 50. More sophisticated update rules can be found in, for example, Nesterov and Nemirovsky (1994), Wright (1997), and Ye (1997).

The update rule we propose is

$$t := \begin{cases} \max\{\mu\min\{\hat{t}, t\}, t\}, & s \geq s_{\min} \\ t, & s < s_{\min} \end{cases} \tag{15}$$

where $\hat{t} = 2n/\eta$, and $s = \beta^k$ is the step length chosen in the line search. Here $\mu > 1$ and $s_{\min} \in (0,1]$ are algorithm parameters; we have found good performance with $\mu = 2$ and $s_{\min} = 0.5$.

To explain the update rule (15), we first give an interpretation of $\hat{t}$. If (v,w,u) is on the central path, that is, ϕ_t is minimized, the duality gap is $\eta = 2n/t$. Thus $\hat{t}$ is the value of t for which the associated central point has the same duality gap as the current point. Another interpretation is that if t were held constant at $t = \hat{t}$, (v,w,u) would converge to $(v^\star(\hat{t}), w^\star(\hat{t}), u^\star(\hat{t}))$, at which point the duality gap would be exactly η.

We use the step length s as a crude measure of proximity to the central path. When the current point is near the central path, that is, ϕ_t is nearly minimized, we have $s = 1$; far from the central path, we typically have $s \ll 1$. Now we can explain the update rule (15). When the current point is near the central path, as judged by $s \geq s_{\min}$ and $\hat{t} \approx t$, we increase t by a factor μ; otherwise, we keep t at its current value.

We can give an informal justification of convergence of the custom interior-point algorithm. (A formal proof of convergence would be quite long.) Assume that the algorithm does not terminate. Since t never decreases, it either increases without bound, or converges to some value $\bar{t}$. In the first case, the duality gap η converges to zero, so the algorithm must exit. In the second case, the algorithm reduces (roughly) to Newton's method for minimizing $\phi_{\bar{t}}$. This must converge, which means that (v,w,u) converges to $(v^\star(\bar{t}),w^\star(\bar{t}),u^\star(\bar{t}))$. Therefore the duality gap converges to $\bar{\eta} = 2n/\bar{t}$. A basic property of Newton's method is that near the solution, the step length is one. At the limit, we therefore have

$$\bar{t} = \max\{\mu \min\{2n/\bar{\eta},\bar{t}\},\bar{t}\} = \mu\bar{t},$$

which is a contradiction since $\mu > 1$.

3.3 Gradient and Hessian

In this section we give explicit formulas for the gradient and Hessian of ϕ_t. The gradient $g = \nabla\phi_t(v,w,u)$ is given by

$$g = \begin{bmatrix} g_1 \\ g_2 \\ g_3 \end{bmatrix} \in \mathbf{R}^{2n+1},$$

where

$$\begin{aligned}
g_1 &= \nabla_v\phi_t(v,w,u) = -(t/m)b^T(\mathbf{1} - p_{\log}(v,w)) \in \mathbf{R}, \\
g_2 &= \nabla_w\phi_t(v,w,u) = -(t/m)A^T(\mathbf{1} - p_{\log}(v,w)) + \begin{bmatrix} 2w_1/(u_1^2 - w_1^2) \\ \vdots \\ 2w_n/(u_n^2 - w_n^2) \end{bmatrix} \in \mathbf{R}^n, \\
g_3 &= \nabla_u\phi_t(v,w,u) = t\lambda\mathbf{1} - \begin{bmatrix} 2u_1/(u_1^2 - w_1^2) \\ \vdots \\ 2u_n/(u_n^2 - w_n^2) \end{bmatrix} \in \mathbf{R}^n.
\end{aligned}$$

The Hessian $H = \nabla^2\phi_t(v,w,u)$ is given by

$$H = \begin{bmatrix} tb^TD_0b & tb^TD_0A & 0 \\ tA^TD_0b & tA^TD_0A + D_1 & D_2 \\ 0 & D_2 & D_1 \end{bmatrix} \in \mathbf{R}^{(2n+1)\times(2n+1)},$$

where

$$\begin{aligned}
D_0 &= (1/m)\operatorname{diag}(f''(w^Ta_1 + vb_1),\ldots,f''(w^Ta_m + vb_m)), \\
D_1 &= \operatorname{diag}\left(2(u_1^2 + w_1^2)/(u_1^2 - w_1^2)^2,\ldots,2(u_n^2 + w_n^2)/(u_n^2 - w_n^2)^2\right), \\
D_2 &= \operatorname{diag}\left(-4u_1w_1/(u_1^2 - w_1^2)^2,\ldots,-4u_nw_n/(u_n^2 - w_n^2)^2\right).
\end{aligned}$$

Here, we use $\operatorname{diag}(z_1,\ldots,z_m)$ to denote the diagonal matrix with diagonal entries $z_1,\ldots,z_m$, where $z_i \in \mathbf{R},\ i = 1,\ldots,m$. The Hessian H is symmetric and positive definite.

3.4 Computing the Search Direction

The search direction is defined by the linear equations (Newton system)

$$\begin{bmatrix} tb^TD_0b & tb^TD_0A & 0 \\ tA^TD_0b & tA^TD_0A+D_1 & D_2 \\ 0 & D_2 & D_1 \end{bmatrix} \begin{bmatrix} \Delta v \\ \Delta w \\ \Delta u \end{bmatrix} = - \begin{bmatrix} g_1 \\ g_2 \\ g_3 \end{bmatrix}.$$

We first eliminate Δu to obtain the reduced Newton system

$$H_{\mathrm{red}} \begin{bmatrix} \Delta v \\ \Delta w \end{bmatrix} = -g_{\mathrm{red}}, \tag{16}$$

where

$$H_{\mathrm{red}} = \begin{bmatrix} tb^TD_0b & tb^TD_0A \\ tA^TD_0b & tA^TD_0A+D_3 \end{bmatrix}, \quad g_{\mathrm{red}} = \begin{bmatrix} g_1 \\ g_2 - D_2D_1^{-1}g_3 \end{bmatrix}, \quad D_3 = D_1 - D_2D_1^{-1}D_2.$$

Once this reduced system is solved, Δu can be recovered as

$$\Delta u = -D_1^{-1}(g_3 + D_2\Delta w).$$

Several methods can be used to solve the reduced Newton system (16), depending on the relative sizes of n and m and the sparsity of the data A.

3.4.1 More Examples than Features

We first consider the case when $m \geq n$, that is, there are more examples than features. We form H_{red}, at a cost of $O(mn^2)$ flops (floating-point operations), then solve the reduced system (16) by Cholesky factorization of H_{red}, followed by back and forward substitution steps, at a cost of $O(n^3)$ flops. The total cost using this method is $O(mn^2+n^3)$ flops, which is the same as $O(mn^2)$ when there are more examples than features.

When A is sufficiently sparse, the matrix $tA^TD_0A+D_3$ is sparse, so H_{red} is sparse, with a dense first row and column. By exploiting sparsity in forming $tA^TD_0A+D_3$, and using a sparse Cholesky factorization to factor H_{red}, the complexity can be much smaller than $O(mn^2)$ flops (see Boyd and Vandenberghe, 2004, App. C or George and Liu, 1981).

3.4.2 Fewer Examples than Features

When $m \leq n$, that is, there are fewer examples than features, the matrix H_{red} is a diagonal matrix plus a rank $m+1$ matrix, so we can use the Sherman-Morrison-Woodbury formula to solve the reduced Newton system (16) at a cost of $O(m^2n)$ flops (see Boyd and Vandenberghe, 2004, §4.3). We start by eliminating Δw from (16) to obtain

$$(tb^TD_0b - t^2b^TD_0AS^{-1}A^TD_0b)\Delta v = -g_1 + tb^TD_0AS^{-1}(g_2 - D_2D_1^{-1}g_3),$$

where $S = tA^TD_0A+D_3$. By the Sherman-Morrison-Woodbury formula (Golub and Van Loan, 1996, p. 50), the inverse of S is given by

$$S^{-1} = D_3^{-1} - D_3^{-1}A^T\left((1/t)D_0^{-1} + AD_3^{-1}A^T\right)^{-1}AD_3^{-1}.$$

We can now calculate Δv via Cholesky factorization of the matrix $\left((1/t)D_0^{-1} + AD_3^{-1}A^T\right)$ and two backsubstitutions (Boyd and Vandenberghe, 2004, App. C). Once we compute Δv, we can compute the other components of the search direction as

$$\begin{aligned} \Delta w &= -S^{-1}(g_2 - D_2 D_1^{-1} g_3 + tA^T D_0 b \Delta v), \\ \Delta u &= -D_1^{-1}(g_3 + D_2 \Delta w). \end{aligned}$$

The total cost of computing the search direction is $O(m^2 n)$ flops. We can exploit sparsity in the Cholesky factorization, whenever $(1/t)D_0^{-1} + AD_3^{-1}A^T$ is sufficiently sparse, to reduce the complexity.

3.4.3 SUMMARY

In summary, the number of flops needed to compute the search direction is

$$O(\min(n,m)^2 \max(n,m)),$$

using dense matrix methods. If $m \geq n$ and $A^T A$ is sparse, or $m \leq n$ and AA^T is sparse, we can use (direct) sparse matrix methods to compute the search direction with less effort. In each of these cases, the computational effort per iteration of the interior-point method is the same as the effort of solving one ℓ_2-regularized linear regression problem.

4. Numerical Examples

In this section we give some numerical examples to illustrate the performance of the interior-point method described in Section 3, using algorithm parameters

$$\alpha = 0.01, \qquad \beta = 0.5, \qquad s_{\min} = 0.5, \qquad \mu = 2, \qquad \varepsilon = 10^{-8}.$$

(The algorithm performs well for much smaller values of ε, but this accuracy is more than adequate for any practical use.) The algorithm was implemented in both Matlab and C, and run on a 3.2GHz Pentium IV under Linux. The C implementation, which is more efficient than the Matlab implementation (especially for sparse problems), is available online (`www.stanford.edu/~boyd/l1_logreg`).

4.1 Benchmark Problems

The data are four small or medium standard data sets taken from the UCI machine learning benchmark repository (Newman et al., 1998) and other sources. The first data set is leukemia cancer gene expression data (Golub et al., 1999), the second is colon tumor gene expression data (Alon et al., 1999), the third is ionosphere data (Newman et al., 1998), and the fourth is spambase data (Newman et al., 1998).

For each data set, we considered four values of the regularization parameter: $\lambda = 0.5\lambda_{\max}$, $\lambda = 0.1\lambda_{\max}$, $\lambda = 0.05\lambda_{\max}$, and $\lambda = 0.01\lambda_{\max}$. We discarded examples with missing data, and standardized each data set. The dimensions of each problem, along with the number of interior-point method iterations (IP iterations) needed, and the execution time, are given in Table 1. In reporting $\mathrm{card}(w)$, we consider a component w_i to be zero when

$$\left|(1/m)\left(A^T(\mathbf{1} - p_{\log}(v,w))\right)_i\right| \leq \tau\lambda,$$

Data	Features n	Examples m	$\lambda/\lambda_{\max}$	card(w)	IP iterations	Time (sec)
Leukemia	7129	38	0.5	6	37	0.60
(Golub et al., 1999)			0.1	14	38	0.62
			0.05	14	39	0.63
			0.01	18	37	0.60
Colon	2000	62	0.5	7	35	0.26
(Alon et al., 1999)			0.1	22	32	0.25
			0.05	25	33	0.26
			0.01	28	32	0.25
Ionosphere	34	351	0.5	3	30	0.02
(Newman et al., 1998)			0.1	11	29	0.02
			0.05	14	30	0.02
			0.01	24	33	0.03
Spambase	57	4061	0.5	8	31	0.63
(Newman et al., 1998)			0.1	28	32	0.66
			0.05	38	33	0.69
			0.01	52	36	0.75

Table 1: Performance of the interior-point method on 4 data sets, each for 4 values of λ.

where $\tau = 0.9999$. This rule is inspired by the optimality condition in (8).

In all sixteen examples, around 35 iterations were required. We have observed this behavior over a large number of other examples as well. The execution times are well predicted by the complexity order $\min(m,n)^2 \max(m,n)$.

Figure 1 shows the progress of the interior-point method on the four data sets, for the same four values of λ. The vertical axis shows duality gap, and the horizontal axis shows iteration number, which is the natural measure of computational effort when dense linear algebra methods are used. The figures show that the algorithm has linear convergence, with duality gap decreasing by a factor around 1.85 in each iteration.

4.2 Randomly Generated Problems

To examine the effect of problem size on the number of iterations required, we generate 100 random problem instances for each of 20 values of n, ranging from $n = 100$ to $n = 10000$, with $m = 0.1n$, that is, 10 times more features than examples. Each problem has an equal number of positive and negative examples, that is, $m_+ = m_- = m/2$. Features of positive (negative) examples are independent and identically distributed, drawn from a normal distribution $\mathcal{N}(v,1)$, where v is in turn drawn from a uniform distribution on $[0,1]$ ($[-1,0]$).

For each of the 2000 data sets, we solve the ℓ_1-regularized LRP for $\lambda = 0.5\lambda_{\max}$, $\lambda = 0.1\lambda_{\max}$, and $\lambda = 0.05\lambda_{\max}$. The lefthand plot in Figure 2 shows the mean and standard deviation of the number of iterations required to solve the 100 problem instances associated with each value of n and λ. It can be seen that the number of iterations required is very near 35, for all 6000 problem instances.

In the same way, we generate a family of data sets with $m = 10n$, that is, 10 times more examples than features, with 100 problem instances for each of 20 values of n ranging from $n = 10$ to $n = 1000$, and for the same 3 values of λ. The righthand plot in Figure 2 shows the mean and standard deviation

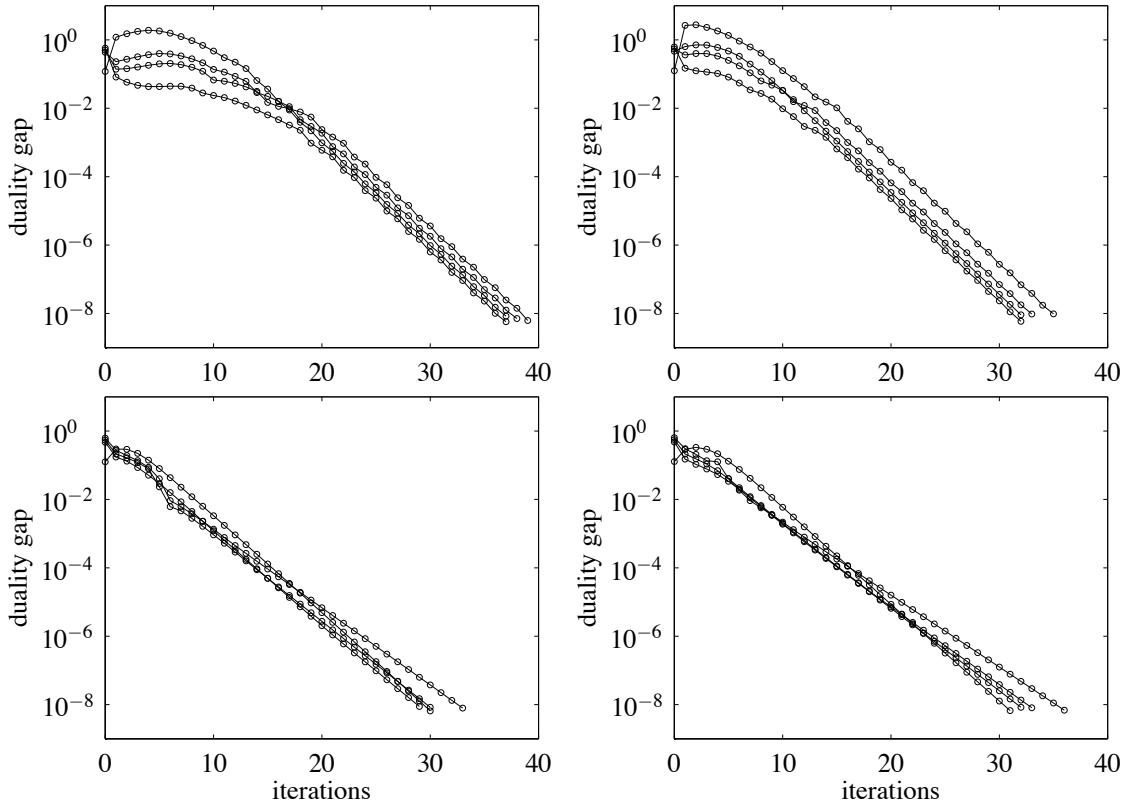


Figure 1: Progress of the interior-point method on 4 data sets, showing duality gap versus iteration number. *Top left*: Leukemia cancer gene data set. *Top right*: Colon tumor gene data set. *Bottom left*: Ionosphere data set. *Bottom right*: Spambase data set.

of the number of iterations required to solve the 100 problem instances associated with each value of n and λ. The results are quite similar to the case with $m = 0.1n$.

5. Truncated Newton Interior-Point Method

In this section we describe a variation on our interior-point method that can handle very large problems, provided the data matrix A is sparse, at the cost of having a run time that is less predictable. The basic idea is to compute the search direction approximately, using a preconditioned conjugate gradients (PCG) method. When the search direction in Newton's method is computed approximately, using an iterative method such as PCG, the overall algorithm is called a *conjugate gradient Newton method*, or a *truncated Newton method* (Ruszczynski, 2006; Dembo and Steihaug, 1983). Truncated Newton methods have been applied to interior-point methods (see, for example, Vandenberghe and Boyd, 1995 and Portugal et al., 2000).

5.1 Preconditioned Conjugate Gradients

The PCG algorithm (Demmel, 1997, §6.6) computes an approximate solution of the linear equations $Hx = -g$, where $H \in \mathbf{R}^{N\times N}$ is symmetric positive definite. It uses a preconditioner $P \in \mathbf{R}^{N\times N}$, also symmetric positive definite.

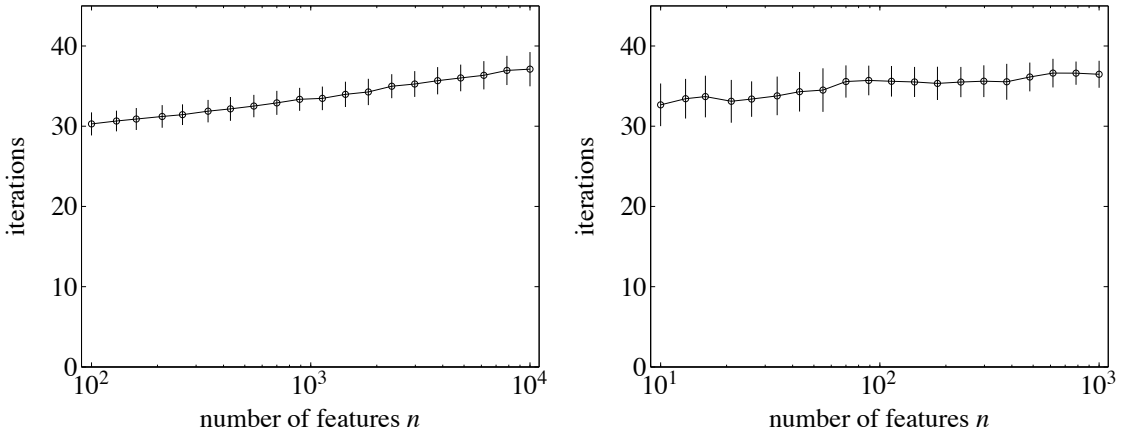


Figure 2: Average number of iterations required to solve 100 randomly generated ℓ_1-regularized LRPs with different problem size and regularization parameter. *Left:* $n = 10m$. *Right:* $n = 0.1m$. Error bars show standard deviation.

PRECONDITIONED CONJUGATE GRADIENTS ALGORITHM

given relative tolerance $\varepsilon_{\text{pcg}} > 0$, iteration limit N_{pcg}, and $x_0 \in \mathbf{R}^k$

$k := 0, r_0 := Hx_0 - g, p_1 := -P^{-1}g, y_0 := P^{-1}r_0$.

repeat

$k := k+1$
$z := Hp_k$
$\theta_k := y_{k-1}^T r_{k-1}/p_k^T z$
$x_k := x_{k-1} + \theta_k p_k$
$r_k := r_{k-1} - \theta_k z$
$y_k := P^{-1}r_k$
$\mu_{k+1} := y_k^T r_k / y_{k-1}^T r_{k-1}$
$p_{k+1} := y_k + \mu_{k+1} p_k$

until $\|r_k\|_2/\|g\|_2 \leq \varepsilon_{\text{pcg}}$ or $k = N_{\text{pcg}}$.

Each iteration of the PCG algorithm involves a handful of inner products, the matrix-vector product Hp_k and a solve step with P in computing $P^{-1}r_k$. With exact arithmetic, and ignoring the stopping condition, the PCG algorithm is guaranteed to compute the exact solution $x = -H^{-1}g$ in N steps. When $P^{-1/2}HP^{-1/2}$ is well conditioned, or has just a few extreme eigenvalues, the PCG algorithm can compute an approximate solution in a number of steps that can be far smaller than N. Since $P^{-1}r_k$ is computed in each step, we need this computation to be efficient.

5.2 Truncated Newton Interior-Point Method

The truncated Newton interior-point method is the same as the interior-point algorithm described in Section 3, with the search direction computed using the PCG algorithm.

We can compute Hp_k in the PCG algorithm using

$$\begin{aligned} Hp_k &= \begin{bmatrix} tb^TD_0b & tb^TD_0A & 0 \\ tA^TD_0b & tA^TD_0A+D_1 & D_2 \\ 0 & D_2 & D_1 \end{bmatrix} \begin{bmatrix} p_{k1} \\ p_{k2} \\ p_{k3} \end{bmatrix} \\ &= \begin{bmatrix} b^Tu \\ A^Tu+D_1p_{k2} \\ D_2p_{k2}+D_1p_{k3} \end{bmatrix}, \end{aligned}$$

where $u = tD_0(bp_{k1} + Ap_{k2}) \in \mathbf{R}^m$. The cost of computing Hp_k is $O(p)$ flops when A is sparse with p nonzero elements. (We assume $p \geq n$, which holds if each example has at least one nonzero feature.)

We now describe a simple choice for the preconditioner P. The Hessian can be written as

$$H = t\nabla^2 l_{\text{avg}}(v,w) + \nabla^2\Phi(w,u).$$

To obtain the preconditioner, we replace the first term with its diagonal part, to get

$$P = \text{diag}\left(t\nabla^2 l_{\text{avg}}(v,w)\right) + \nabla^2\Phi(w,u) = \begin{bmatrix} d_0 & 0 & 0 \\ 0 & D_3 & D_2 \\ 0 & D_2 & D_1 \end{bmatrix}, \tag{17}$$

where

$$d_0 = tb^TD_0b, \qquad D_3 = \text{diag}(tA^TD_0A) + D_1.$$

(Here $\text{diag}(S)$ is the diagonal matrix obtained by setting the off-diagonal entries of the matrix S to zero.) This preconditioner approximates the Hessian of tl_{avg} with its diagonal entries, while retaining the Hessian of the logarithmic barrier. For this preconditioner, $P^{-1}r_k$ can be computed cheaply as

$$\begin{aligned} P^{-1}r_k &= \begin{bmatrix} d_0 & 0 & 0 \\ 0 & D_3 & D_2 \\ 0 & D_2 & D_1 \end{bmatrix}^{-1} \begin{bmatrix} r_{k1} \\ r_{k2} \\ r_{k3} \end{bmatrix} \\ &= \begin{bmatrix} r_{k1}/d_0 \\ (D_1D_3 - D_2^2)^{-1}(D_1r_{k2} - D_2r_{k3}) \\ (D_1D_3 - D_2^2)^{-1}(-D_2r_{k2} + D_3r_{k3}) \end{bmatrix}, \end{aligned}$$

which requires $O(n)$ flops.

We can now explain how *implicit standardization* can be carried out. When using standardized data, we work with the matrix A^{std} defined in (20), instead of A. As mentioned in Appendix A, A^{std} is in general dense, so we should not form the matrix. In the truncated Newton interior-point method we do not need to form the matrix A^{std}; we only need a method for multiplying a vector by A^{std} and a method for multiplying a vector by $A^{\text{std}\,T}$. But this is easily done efficiently, using the fact that A^{std} is a sparse matrix (i.e., A) times a diagonal matrix, plus a rank-one matrix; see (20) in Appendix A.

There are several good choices for the initial point in the PCG algorithm (labeled x_0 in Section 5.1), such as the negative gradient, or the previous search direction. We have found good performance with both, with a small advantage in using the previous search direction.

The PCG relative tolerance parameter ε_{pcg} has to be carefully chosen to obtain good efficiency in a truncated Newton method. If the tolerance is too small, too many PCG steps are needed to compute each search direction; if the tolerance is too high, then the computed search directions do not give adequate reduction in duality gap per iteration. We experimented with several methods of adjusting the PCG relative tolerance, and found good results with the adaptive rule

$$\varepsilon_{\text{pcg}} = \min\{0.1, \xi\eta/\|g\|_2\}, \tag{18}$$

where g is the gradient and η is the duality gap at the current iterate. Here, ξ is an algorithm parameter. We have found that $\xi = 0.3$ works well for a wide range of problems. In other words, we solve the Newton system with low accuracy (but never worse than 10%) at early iterations, and solve it more accurately as the duality gap decreases. This adaptive rule is similar in spirit to standard methods used in inexact and truncated Newton methods (see Nocedal and Wright, 1999).

The computational effort of the truncated Newton interior-point algorithm is the product of s, the total number of PCG steps required over all iterations, and the cost of a PCG step, which is $O(p)$, where p is the number of nonzero entries in A, that is, the total number of (nonzero) features appearing in all examples. In extensive testing, we found the truncated Newton interior-point method to be very efficient, requiring a total number of PCG steps ranging between a few hundred (for medium size problems) and several thousand (for large problems). For medium size (and sparse) problems it was faster than the basic interior-point method; moreover the truncated Newton interior-point method was able to solve very large problems, for which forming the Hessian H (let alone computing the search direction) would be prohibitively expensive.

While the total number of iterations in the basic interior-point method is around 35, and nearly independent of the problem size and problem data, the total number of PCG iterations required by the truncated Newton interior-point method can vary significantly with problem data and the value of the regularization parameter λ. In particular, for small values of λ (which lead to large values of $\text{card}(w)$), the truncated Newton interior-point method requires a larger total number of PCG steps. Algorithm performance that depends substantially on problem data, as well as problem dimension, is typical of all iterative (i.e., non direct) methods, and is the price paid for the ability to solve very large problems.

5.3 Numerical Examples

In this section we give some numerical examples to illustrate the performance of the truncated Newton interior-point method. We use the same algorithm parameters for line search, update rule, and stopping criterion as those used in Section 4, and the PCG tolerance given in (18) with $\xi = 0.3$. We chose the parameter N_{pcg} to be large enough (5000) that the iteration limit was never reached in our experiments; the typical number of PCG iterations was far smaller. The algorithm is implemented in both Matlab and C, on a 3.2GHz Pentium IV running Linux, except for very large problems. For very large problems whose data could not be handled on this computer, the method was run on AMD Opteron 254 with 8GB main memory. The C implementation is available online at `www.stanford.edu/~boyd/l1_logreg`.

5.3.1 A MEDIUM SPARSE PROBLEM

We consider the Internet advertisements data set (Newman et al., 1998) with 1430 features and 2359 examples (discarding examples with missing data). The total number of nonzero entries in the data

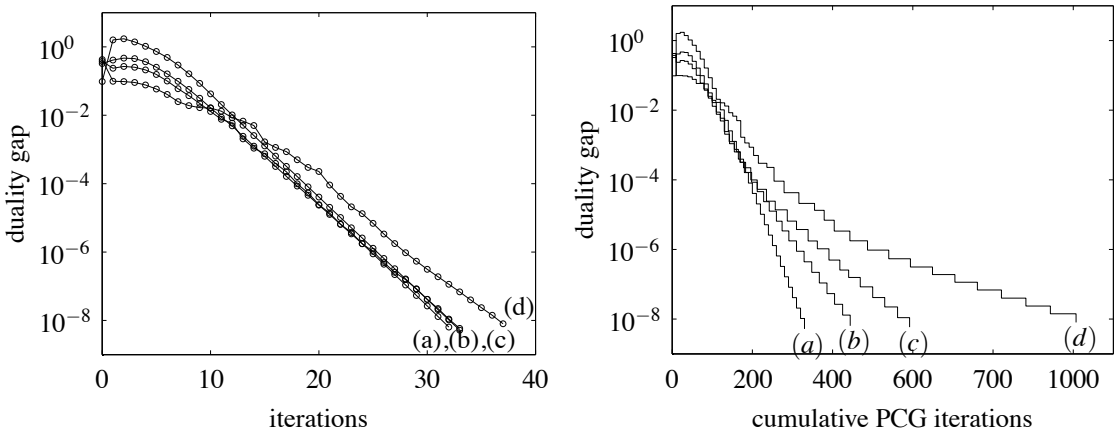


Figure 3: Progress of the truncated Newton interior-point method on the Internet advertisements data set with four regularization parameters: (a) $\lambda = 0.5\lambda_{\max}$, (b) $\lambda = 0.1\lambda_{\max}$, (c) $\lambda = 0.05\lambda_{\max}$, and (d) $\lambda = 0.01\lambda_{\max}$.

matrix A is $p = 39011$. We standardized the data set using implicit standardization, as explained in Section 5.2, solving four ℓ_1-regularized LRPs, with $\lambda = 0.5\lambda_{\max}$, $\lambda = 0.1\lambda_{\max}$, $\lambda = 0.05\lambda_{\max}$, and $\lambda = 0.01\lambda_{\max}$. Figure 3 shows the convergence behavior. The lefthand plot shows the duality gap versus outer iterations; the righthand plot shows duality gap versus cumulative PCG iterations, which is the more accurate measure of computational effort.

The lefthand plot shows that the number of Newton iterations required to solve the problem is not much more than in the basic interior-point method described in Section 3. The righthand plot shows that the total number of PCG steps is several hundred, and depends substantially on the value of λ. Thus, the search directions are computed using on the order of ten PCG iterations.

To give a very rough comparison with the direct method applied to this sparse problem, the truncated Newton interior-point method is much more efficient than the basic interior-point method that does not exploit the sparsity of the data. It is comparable to or faster than the basic interior-point method that uses sparse linear algebra methods, when the regularization parameter is not too small.

5.3.2 A LARGE SPARSE PROBLEM

Our next example uses the 20 Newsgroups data set (Lang, 1995). We processed the data set in a way similar to Keerthi and DeCoste (2005). The positive class consists of the 10 groups with names of form sci.*, comp.*, and misc.forsale, and the negative class consists of the other 10 groups. We used McCallum's Rainbow program (McCallum, 1996) with the command

```
rainbow -g 3 -h -s -O 2 -i
```

to tokenize the (text) data set. These options specify trigrams, skip message headers, no stoplist, and drop terms occurring fewer than two times. The resulting data set has $n = 777811$ features (trigrams) and $m = 11314$ examples (articles). Each example contains an average of 425 nonzero features. The total number of nonzero entries in the data matrix A is $p = 4802169$. We standardized the data set using implicit standardization, as explained in Section 5.2, solving three ℓ_1-regularized LRPs, with $\lambda = 0.5\lambda_{\max}$, $\lambda = 0.1\lambda_{\max}$, and $\lambda = 0.05\lambda_{\max}$. (For the value $\lambda = 0.01\lambda_{\max}$, the runtime is on the order of one hour. This case is not of practical interest, and so not reported here, since

$\lambda/\lambda_{\max}$	card(w)	Iterations	PCG iterations	Time (sec)
0.5	9	43	558	134
0.1	544	60	1036	256
0.05	2531	58	2090	501

Table 2: Performance of truncated Newton interior-point method on the 20 newsgroup data set ($n = 777811$ features, $m = 11314$ examples) for 3 values of λ.

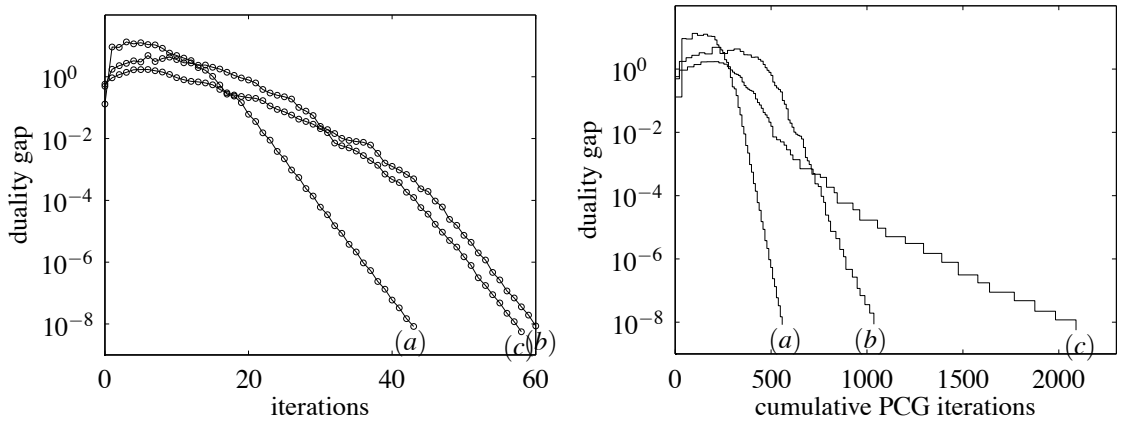


Figure 4: Progress of the truncated Newton interior-point method on the 20 Newsgroups data set for (a) $\lambda = 0.5\lambda_{\max}$, (b) $\lambda = 0.1\lambda_{\max}$, and (c) $\lambda = 0.05\lambda_{\max}$. *Left*. Duality gap versus iterations. *Right*. Duality gap versus cumulative PCG iterations.

the cardinality of the optimal solution is around 10000 and comparable to the number of examples.) The performance of the algorithm, and the cardinality of the weight vectors, is given in Table 2. Figure 4 shows the progress of the algorithm, with duality gap versus iteration (lefthand plot), and duality gap versus cumulative PCG iteration (righthand plot).

The number of iterations required to solve the problems ranges between 43 and 60, depending on λ. The more relevant measure of computational effort is the total number of PCG iterations, which ranges between around 500 and 2000, again, increasing with decreasing λ, which corresponds to increasing card(w). The average number of PCG iterations, per iteration of the truncated Newton interior-point method, is around 13 for $\lambda = 0.5\lambda_{\max}$, 17 for $\lambda = 0.1\lambda_{\max}$, and 36 for $\lambda = 0.05\lambda_{\max}$. (The variance in the number of PCG iterations required per iteration, however, is large.) The running time is consistent with a cost of around 0.24 seconds per PCG iteration. The increase in running time, for decreasing λ, is due primarily to an increase in the average number of PCG iterations required per iteration, but also from an increase in the overall number of iterations required.

5.3.3 RANDOMLY GENERATED PROBLEMS

We generated a family of 21 data sets, with the number of features n varying from one hundred to ten million, and $m = 0.1n$ examples. The data were generated using the same general method described in Section 4.2, but with A sparse, with an average number of nonzero features per example around 30. Thus, the total number of nonzero entries in A is $p \approx 30m$. We standardized the data

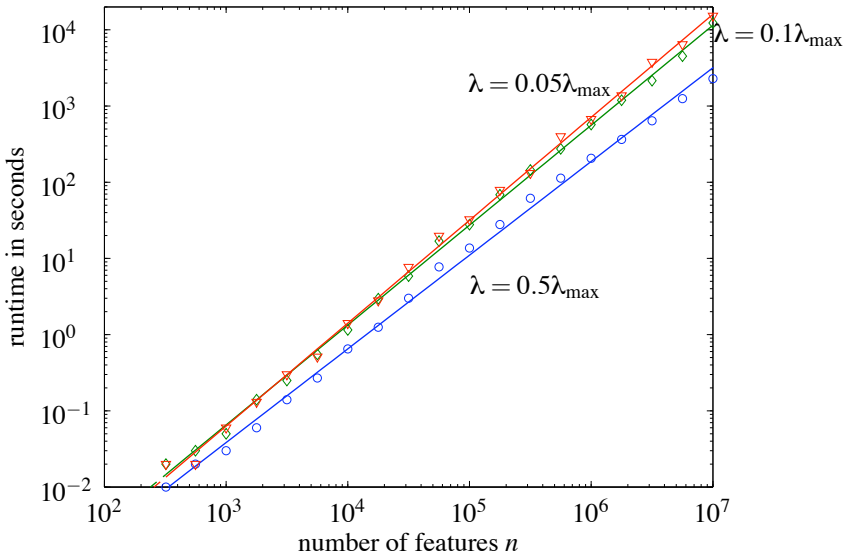


Figure 5: Runtime of the truncated Newton interior-point method, for randomly generated sparse problems, with three values of λ.

set using implicit standardization, as explained in Section 5.2, solving each problem instance for the three values $\lambda = 0.5\lambda_{\max}$, $\lambda = 0.1\lambda_{\max}$, and $\lambda = 0.05\lambda_{\max}$. The total runtime, for the 63 ℓ_1-regularized LRPs, is shown in Figure 5. The plot shows that runtime increases as λ decreases, and grows approximately linearly with problem size.

We compare the runtimes of the truncated Newton interior-point and the basic interior-point method using dense linear algebra methods to compute the search direction. Figure 6 shows the results for $\lambda = 0.1\lambda_{\max}$. The truncated Newton interior-point method is far more efficient for medium problems. For large problems, the basic interior-point method fails due to memory limitations, or extremely long computation times.

By fitting an exponent to the data over the range from $n = 320$ to the largest problem successfully solved by each method, we find that the basic interior-point method scales as $O(n^{2.8})$ (which is consistent with the basic flop count analysis, which predicts $O(n^3)$). For the truncated Newton interior-point method, the empirical complexity is $O(n^{1.3})$.

When sparse matrix methods are used to compute the search direction in the basic interior-point method, we get an empirical complexity of $O(n^{2.2})$ for the Matlab implementation of the basic interior-point method that uses sparse matrix methods, showing a good efficiency gain over dense methods, for medium scale problems. The C implementation would have the same empirical complexity as the Matlab one with a smaller constant hidden in the $O(\cdot)$ notation.

5.3.4 PRECONDITIONER PERFORMANCE

To examine the effect of the preconditioner (17) on the efficiency of the approximate search direction computation, we compare the eigenvalue distributions of the Hessian H and the preconditioned Hessian $P^{-1/2}HP^{-1/2}$, for the colon gene tumor problem ($n = 2000$ features, $m = 62$ examples) at the 15th iterate, in Figure 7. The eigenvalues of the preconditioned Hessian are tightly clustered,

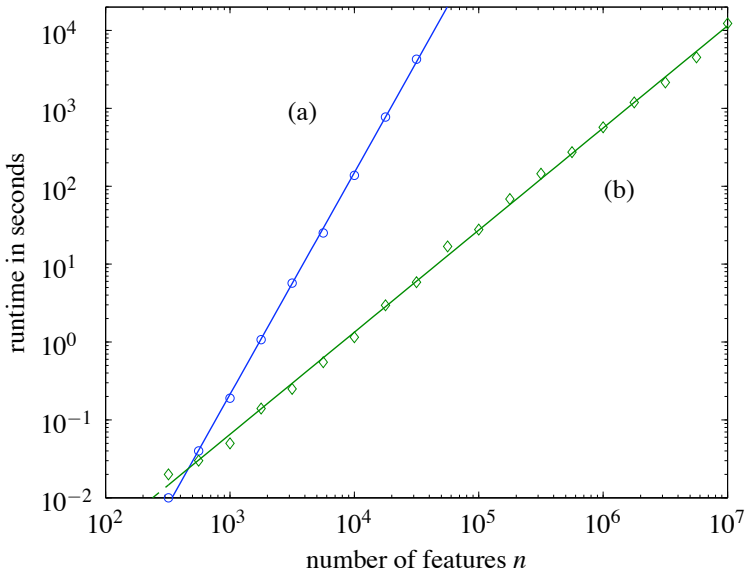


Figure 6: Runtime of (a) the basic interior-point method and (b) the truncated Newton interior-point method, for a family of randomly generated sparse problems.

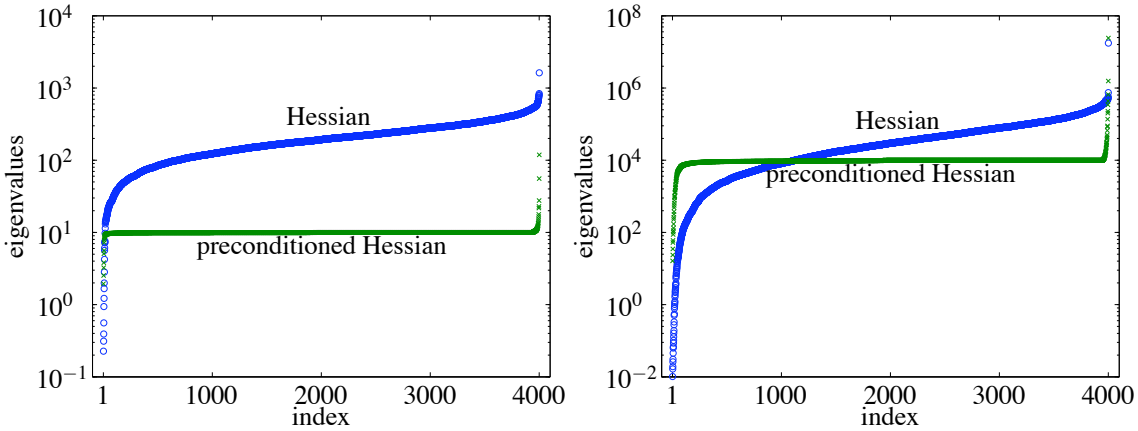


Figure 7: Eigenvalue distributions of Hessian and preconditionned Hessian, at the 15th iterate, for the colon gene tumor problem, for $\lambda = 0.5\lambda_{\max}$ (left) and $\lambda = 0.05\lambda_{\max}$ (right).

with just a few extreme eigenvalues, which explains the good performance with relatively few PCG iterations per iteration (Demmel, 1997, §6.6).

6. Computing the Regularization Path

In this section we consider the problem of solving the ℓ_1-regularized LRP for M values of the regularization parameter λ,

$$\lambda_{\max} = \lambda_1 > \lambda_2 > \cdots > \lambda_M > 0.$$

This can be done by applying the methods described above, for each of the M problems. This is called a *cold-start* approach, since each problem is solved independently of the others. This

is efficient when multiple processors are used, since the LRPs can be solved simultaneously, on different processors. But when one processor is used, we can solve these M problems much more efficiently by solving them sequentially, using the previously computed solution as a starting point for the next computation. This is called a *warm-start* approach.

We first note that the solution for $\lambda = \lambda_1 = \lambda_{\max}$ is $(\log(m_+/m_-), 0, 0)$. Since this point does not satisfy $|w_i| < u_i$, it cannot be used to initialize the computation for $\lambda = \lambda_2$. We modify it by adding a small increment to u to get

$$(v^{(1)}, w^{(1)}, u^{(1)}) = (\log(m_+/m_-), 0, (\varepsilon_{\text{abs}}/(n\lambda))\mathbf{1}),$$

which is strictly feasible. In fact, it is on the central path with parameter $t = 2n/\varepsilon_{\text{abs}}$, and so is ε_{abs}-suboptimal. Note that so far we have expended no computational effort.

Now for $k = 2, \ldots, M$ we compute the solution $(v^{(k)}, w^{(k)}, u^{(k)})$ of the problem with $\lambda = \lambda_k$, by applying the interior-point method, with starting point modified to be

$$(v_{\text{init}}, w_{\text{init}}, u_{\text{init}}) = (v^{(k-1)}, w^{(k-1)}, u^{(k-1)}),$$

and initial value of t set to $t = 2n/\varepsilon_{\text{abs}}$.

In the warm-start technique described above, the number of grid points, M, is fixed in advance. The grid points (and M) can be chosen adaptively on the fly, while taking into account the curvature of the regularization path trajectories, as described in Park and Hastie (2006a).

6.1 Numerical Results

Our first example is the leukemia cancer gene expression data, for $M = 100$ values of λ, uniformly distributed on a logarithmic scale over the interval $[0.001\lambda_{\max}, \lambda_{\max}]$. (For this example, $\lambda_{\max} = 0.37$.) The left plot in Figure 8 shows the regularization path, that is, $w^{(k)}$, versus regularization parameter λ. The right plot shows the number of iterations required to solve each problem from a warm-start, and from a cold-start.

The number of cold-start iterations required is always near 36, while the number of warm-start iterations varies, but is always smaller, and typically much smaller, with an average value of 3.1. Thus the computational savings for this example is over 11 : 1.

Our second example is the 20 Newsgroups data set, with $M = 100$ values of λ uniformly spaced on a logarithmic scale over $[0.05\lambda_{\max}, \lambda_{\max}]$. For this problem we have $\lambda_{\max} = 0.12$. The top plot in Figure 9 shows the regularization path. The bottom left plot shows the total number of PCG iterations required to solve each problem, with the warm-start and cold-start methods. The bottom right plot shows the cardinality of w as a function of λ.

Here too the warm-start method gives a substantial advantage over the cold-start method, at least for λ not too small, that is, as long as the optimal weight vector is relatively sparse. The total runtime using the warm-start method is around 2.8 hours, and the total runtime using the cold-start method is around 6.2 hours, so the warm-start methods gives a savings of around 2 : 1. If we consider only the range from $0.1\lambda_{\max}$ to $\lambda_{\max}$, the savings increases to 5 : 1.

We note that for this example, the number of events (i.e., a weight transitioning between zero and nonzero) along the regularization path is very large, so methods that attempt to track every event will be very slow.

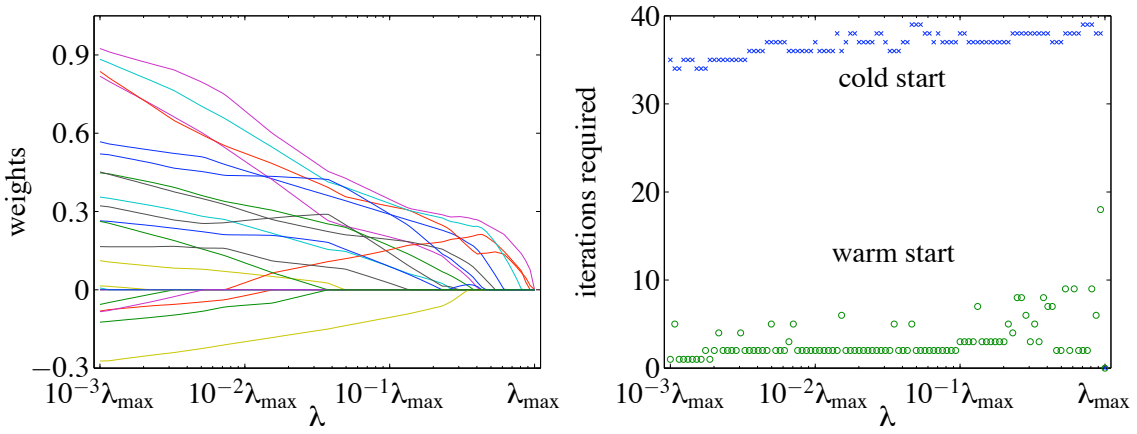


Figure 8: *Left*. Regularization path for leukemia cancer gene expression data. *Right*. Iterations required for cold-start and warm-start methods.

7. Comparison

In this section we compare the performance of our basic and truncated Newton interior-point methods, implemented in C (called `l1_logreg`), with several existing methods for ℓ_1-regularized logistic regression, We make comparisons with `MOSEK` (MOSEK ApS, 2002), `IRLS-LARS` (Lee et al., 2006), `BBR` (Genkin et al., 2006), and `glmpath` (Park and Hastie, 2006a).

`MOSEK` is a general purpose primal-dual interior-point solver, which is known to be quite efficient compared to other standard solvers. `MOSEK` can solve ℓ_1-regularized LRPs using the separable convex formulation (9), or by treating the problem as a geometric program (GP) (see Boyd et al., 2006). We used both formulations and report the better results here in each case. `MOSEK` uses a stopping criterion based on the duality gap, like our method.

`IRLS-LARS` alternates between approximating the average logistic loss by a quadratic approximation at the current iterate, and solving the resulting ℓ_1-regularized least squares problem using the LARS method (Efron et al., 2004) to update the iterate. `IRLS-LARS` outperforms many existing methods for ℓ_1-regularized logistic regression including `GenLASSO` (Roth, 2004), `SCGIS` (Goodman, 2004), `Gl1ce` (Lokhorst, 1999), and `Grafting` (Perkins and Theiler, 2003). `IRLS-LARS` used in our comparison is implemented in Matlab and C, with the LARS portion implemented in C. The hybrid implementation is called `IRLS-LARS-MC` and available from `http://ai.stanford.edu/~silee/softwares/irlslars.htm`. We ran it until the primal objective is within tolerance from the optimal objective value, which is computed using `l1_logreg`, with small (10^{-12}) duality gap.

`BBR`, implemented in C, uses the cyclic coordinate descent method for Bayesian logistic regression. The C implementation is available from `http://www.stat.rutgers.edu/~madigan/BBR/`. The stopping criterion is based on lack of progress, and not on a suboptimality bound or duality gap. Tightening the tolerance for `BBR` greatly increased its running time, and only had a minor effect on the final accuracy.

`glmpath` uses a path-following method for generalized linear models, including logistic models, and computes a portion of the regularization path. It is implemented in the R environment and available from `http://cran.r-project.org/src/contrib/Descriptions/glmpath.html`. We compared `glmpath` to our warm-start method, described in Section 6.

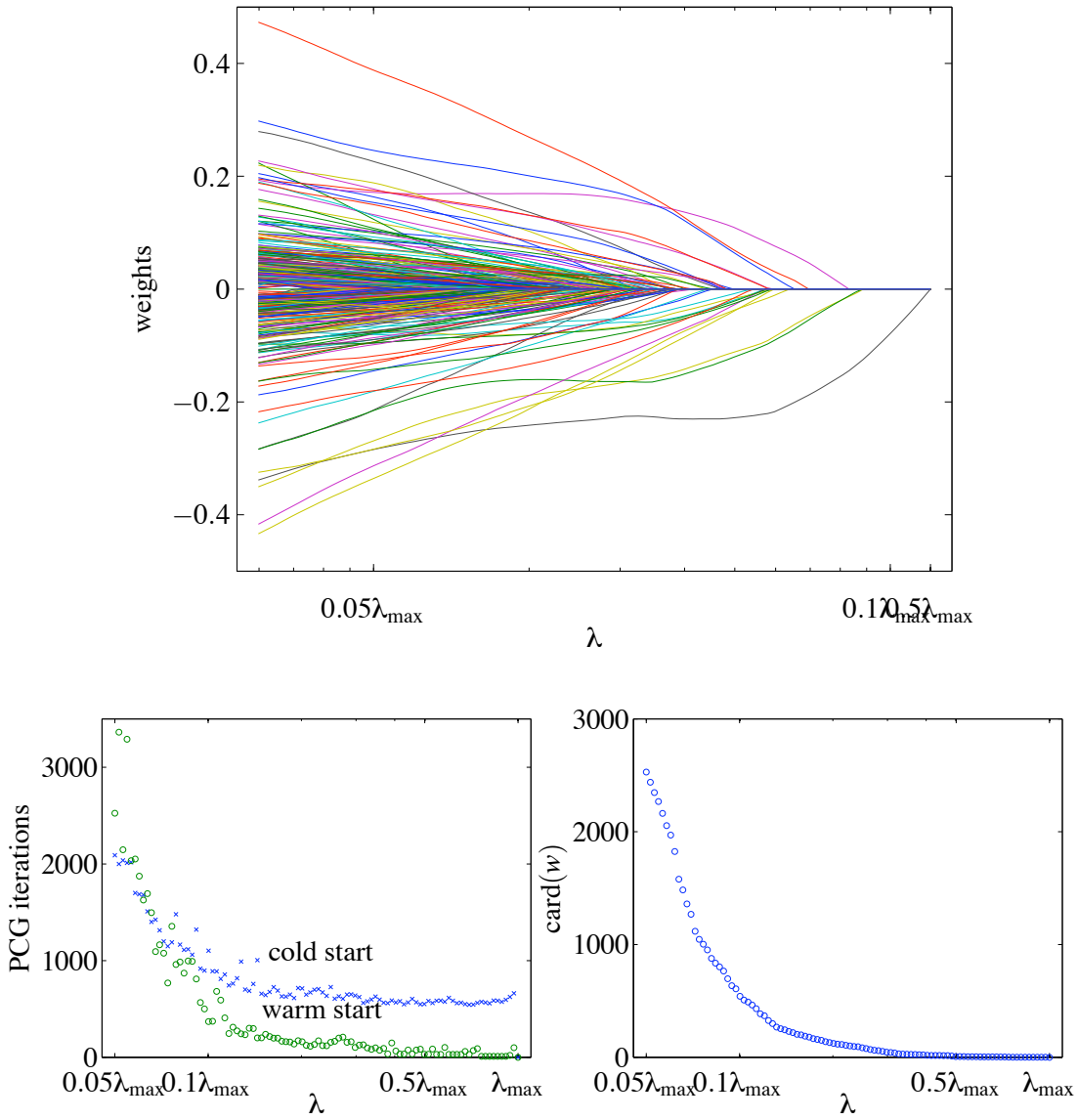


Figure 9: *Top*. Regularization path for 20 newsgroup data. *Bottom left*. Total PCG iterations required by cold-start and warm-start methods. *Bottom right*. card(w) versus λ.

We report the run times of the methods, using the different stopping criteria described above. We also report the actual accuracy achieved, that is, the difference between the achieved primal objective and the optimal objective value (as computed by `l1_logreg` with duality gap 10^{-12}). However, it is important to point out that it is very difficult, if not impossible, to carry out a fair comparison of solution methods, due to the issue of implementation (which can have a great influence on the algorithm performance), the choice of algorithm parameters, and the different stopping criteria. Therefore, the comparison results reported below should be interpreted with caution.

We report comparison results using four data sets: the leukemia cancer gene expression and spambase data sets, two dense benchmark data sets used in Section 4.1, the Internet advertisements data set, the medium sparse data set used in Section 5.3, and the 20 Newsgroups data set, the large sparse data set used in Section 5.3. When the large 20 Newsgroups data set was standardized, the three existing solvers could not handle a data set of this size, so it was not standardized. The solvers could handle the standardized Internet advertisements data set but do not exploit the sparse plus rank-one structure of the standardized data matrix. Therefore, the Internet advertisements data set was not standardized as well. For small problems (leukemia and spambase), the solvers were all run on a 3.2GHz Pentium IV under Linux; for medium and large problem (Internet advertisements and 20 Newsgroups), the solvers were run on an AMD Opteron 254 (with 8GB RAM) under Linux.

The regularization parameter λ can strongly affect the runtime of the methods, including ours. For each data set, we considered many values of the regularization parameter λ over the interval $[0.0001\lambda_{\max}, \lambda_{\max}]$ (which appears to cover the range of interest for many standardized problems). We report here the results with $\lambda = 0.001\lambda_{\max}$ for the two small benchmark data. The cardinality of the optimal solution is 21 for the leukemia data and 54 for the spambase data, larger than half of the minimum of the number of features and the number of examples. For the two unstandardized problems, ℓ_1-regularized LR with a regularization parameter in the interval $[0.0001\lambda_{\max}, \lambda_{\max}]$ yields a very sparse solution. We used $\lambda = 0.0001\lambda_{\max}$ for the Internet advertisements and $\lambda = 0.001\lambda_{\max}$ for the 20 Newsgroups data. The cardinalities of optimal solutions are relatively very small (19 for the Internet advertisements and 247 for the 20 Newsgroups) compared with the sizes of the problems.

Table 3 summarizes the comparison results for the four problems. Here, the tolerance has a different meaning depending on the method, as described above. As shown in this table, our method is as fast as, or faster than, existing methods for the small two data sets, but the discrepancy in performance is not significant, since the problems are small. For the unstandardized Internet advertisements data set, our method is most efficient. `MOSEK` could not handle the unstandardized large 20 Newsgroups data set, so we compared `l1_logreg` with `BBR` and `IRLS-LARS-MC` on the unstandardized 20 Newsgroups data set, for the regularization parameter $\lambda = 0.001\lambda_{\max}$. The truncated Newton interior-point method solves the problem to the accuracy 2×10^{-6} in around 100 seconds and to a higher accuracy with relatively small additional run time; `BBR` solves them to this ultimate accuracy in a comparable time, but slows down when attempting to compute a solution with higher accuracy.

Finally, we compare the runtimes of the warm-start method and `glmpath`. We consider the leukemia cancer gene expression data set and the Internet advertisements data set as benchmark examples of small dense and medium sparse problems, respectively. For each data set, the warm-start method finds M points ($w^{(k)}$, $k = 1, \ldots, M$) on the regularization path, with λ uniformly spaced on a logarithmic scale over the interval $[0.001\lambda_{\max}, \lambda_{\max}]$. `glmpath` finds an approximation of the regularization path by choosing the kink points adaptively over the same interval. The results are

Program	Tol.	Leukemia		Spambase		Internet adv.		Newsgroups	
		time	accuracy	time	accuracy	time	accuracy	time	accuracy
`l1_logreg`	10^{-4}	0.37	2×10^{-6}	0.34	2×10^{-5}	0.14	4×10^{-5}	90	2×10^{-6}
	10^{-8}	0.57	1×10^{-10}	0.74	1×10^{-9}	0.27	5×10^{-9}	140	2×10^{-10}
`IRLS-LARS-MC`	10^{-4}	0.75	6×10^{-6}	0.29	9×10^{-6}	1.2	1×10^{-5}	450	8×10^{-5}
	10^{-8}	0.81	6×10^{-11}	0.37	3×10^{-11}	2.5	1×10^{-10}	1200	7×10^{-9}
`MOSEK`	10^{-4}	9	8×10^{-6}	10	8×10^{-8}	1.3	3×10^{-9}	-	-
	10^{-8}	10	3×10^{-9}	11	5×10^{-13}	1.4	4×10^{-13}	-	-
`BBR`	10^{-4}	15	1×10^{-7}	39	3×10^{-6}	0.44	1×10^{-7}	140	2×10^{-6}
	10^{-11}	73	4×10^{-8}	300	1×10^{-11}	1.1	1×10^{-12}	850	1×10^{-6}

Table 3: Comparison results with two standardized data sets (leukemia and spambase) and two unstandardized data sets (Internet advertisements and 20 Newsgroups). The regularization parameter is taken as $\lambda = 0.001\lambda_{\max}$ for leukemia and spambase, $\lambda = 0.0001\lambda_{\max}$ for Internet advertisements and $\lambda = 0.001\lambda_{\max}$ for 20 Newsgroups.

Data	warm-start ($M = 25$)	warm-start ($M = 100$)	`glmpath`
Leukemia	2.8	6.4	1.9
Internet	5.2	13	940

Table 4: Regularization path computation time (in seconds) of the warm-start method and `glmpath` for standardized leukemia cancer gene expression data and Internet advertisements data.

shown in Table 4. For the small leukemia data set, `glmpath` is faster than the warm-start method. The warm-start method is more efficient than `glmpath` for the Internet advertisements data set, a medium-sized sparse problem. The performance discrepancy is partially explained by the fact that our warm-start method exploits the sparse plus rank-one structure of the standardized data matrix, whereas `glmpath` does not.

8. Extensions and Variations

The basic interior-point method and the truncated Newton variation can be extended to general ℓ_1-regularized convex loss minimization problems, with twice differentiable loss functions, that have the form

$$\begin{array}{ll} \text{minimize} & (1/m)\sum_{i=1}^m \phi(z_i) + \lambda\|w\|_1 \\ \text{subject to} & z_i = w^T a_i + v b_i + c_i, \quad i = 1,\ldots,m, \end{array} \tag{19}$$

with variables are $v \in \mathbf{R}$, $w \in \mathbf{R}^n$, and $z \in \mathbf{R}^m$, and problem data $c_i \in \mathbf{R}$, $a_i \in \mathbf{R}^n$, and $b_i \in \mathbf{R}$ determined by a set of given (observed or training) examples

$$(x_i, y_i) \in \mathbf{R}^n \times \mathbf{R}, \quad i = 1,\ldots,m.$$

Here $\phi : \mathbf{R} \to \mathbf{R}$ is a loss function which is convex and twice differentiable. Prior work related to the extension includes Park and Hastie (2006a), Rosset (2005), and Tibshirani (1997).

In ℓ_1-regularized (binary) classification, we have $y_i \in \{-1,+1\}$ (binary labels), and z_i has the form $z_i = y_i(w^T x_i + v)$, so we have the form (19) with $a_i = y_i x_i$, $b_i = y_i$, and $c_i = 0$. The associated classifier is given by $y = \mathrm{sgn}(w^T x + v)$. The loss function ϕ is small for positive arguments,

and grows for negative arguments. When ϕ is the logistic loss function f in (1), this general ℓ_1-regularized classification problem reduces to the ℓ_1-regularized LRP. When ϕ is the convex loss function $\phi(u) = -\log\Phi(u)$, where Φ is the cumulative distribution function of the standard normal distribution, this ℓ_1-regularized classification problem reduces to the ℓ_1-regularized probit regression problem. More generally, ℓ_1-regularized estimation problems that arise in generalized linear models (McCullagh and Nelder, 1989; Hastie et al., 2001) for binary response variables (which include logistic and probit models) can be formulated problems of the form (19); see Park and Hastie (2006a) for the precise formulation.

In ℓ_1-regularized linear regression, we have $y_i \in \mathbf{R}$, and z_i has the form $z_i = w^T x_i + v - y_i$, which is the difference between y_i and its predicted value, $w^T x_i + v$. Thus ℓ_1-regularized regression problems have the form (19) with $a_i = x_i$, $b_i = 1$, and $c_i = -y_i$. Typically ϕ is symmetric, with $\phi(0) = 0$. When the loss function is quadratic, that is, $\phi(u) = u^2$, the convex loss minimization problem (19) is the ℓ_1-regularized least squares regression problem studied extensively in the literature.

The dual of the ℓ_1-regularized convex loss minimization problem (19) is

$$\begin{array}{ll} \text{maximize} & -(1/m)\sum_{i=1}^m \phi^*(-m\theta_i) + \theta^T c \\ \text{subject to} & \|A^T\theta\|_\infty \leq \lambda, \quad b^T\theta = 0, \end{array}$$

where $A = [a_1 \ \cdots \ a_m]^T \in \mathbf{R}^{m\times n}$, the variable is $\theta \in \mathbf{R}^m$, and ϕ^* is the conjugate of the loss function ϕ,

$$\phi^*(y) = \sup_{u\in\mathbf{R}} \left(yu - \phi(u)\right).$$

As with ℓ_1-regularized logistic regression, we can derive a bound on the suboptimality of (v,w), by constructing a dual feasible point $\bar{\theta}$, from an arbitrary w,

$$\bar{\theta} = (s/m)p(\bar{v},w), \qquad p(\bar{v},w) = \begin{bmatrix} \phi'(w^T a_1 + \bar{v}b_1 + c_1) \\ \vdots \\ \phi'(w^T a_m + \bar{v}b_m + c_m) \end{bmatrix},$$

where $\bar{v}$ is the optimal intercept for the offset w,

$$\bar{v} = \arg\min_v (1/m)\sum_{i=1}^m \phi(w^T a_i + v b_i + c_i),$$

and the scaling constant s is given by $s = \min\left\{m\lambda/\|A^T p(\bar{v},w))\|_\infty, 1\right\}$.

Using this method for cheaply computing a dual feasible point and associated duality gap for *any* (v,w), we can extend the custom interior-point method for ℓ_1-regularized LRPs to general ℓ_1-regularized convex (twice differentiable) loss problems.

Other possible extensions include ℓ_1-regularized Cox proportional hazards models (Cox, 1972). The associated ℓ_1-regularized problem does not have the form (19), but the idea behind the custom interior-point method for ℓ_1-regularized LRPs can be readily extended. The reader is referred to Park and Hastie (2006a) and Tibshirani (1997) for related work on computational methods for ℓ_1-regularized Cox proportional hazards models.

Acknowledgments

This work is supported in part by the National Science Foundation under grants #0423905 and (through October 2005) #0140700, by the Air Force Office of Scientific Research under grant #F49620-01-1-0365, by MARCO Focus center for Circuit & System Solutions contract #2003-CT-888, and by MIT DARPA contract #N00014-05-1-0700. The authors thank Michael Grant, Trevor Hastie, Honglak Lee, Suin Lee, Mee Young Park, Robert Tibshirani, Yinyu Ye, and Sungroh Yoon for helpful comments and suggestions. The authors thank anonymous reviewers for helpful comments and suggestions (especially on comparison with existing methods).

Appendix A. Standardization

Standardization is a widely used pre-processing step applied to the feature vector, so that each (transformed) feature has zero mean and unit variance (over the examples) (Ryan, 1997). The mean feature vector is $\mu = (1/m)\sum_{i=1}^m x_i$, and the vector of feature standard deviations σ is defined by

$$\sigma_j = \left((1/m) \sum_{i=1}^m (x_{ij} - \mu_j)^2 \right)^{1/2}, \quad j = 1, \ldots, n,$$

where x_{ij} is the jth component of x_i. The *standardized feature vector* is defined as

$$x^{\mathrm{std}} = \mathrm{diag}(\sigma)^{-1}(x - \mu).$$

When the examples are standardized, we obtain the standardized data matrix

$$A^{\mathrm{std}} = \mathrm{diag}(b)(X - \mathbf{1}\mu^T)\,\mathrm{diag}(\sigma)^{-1} = A\,\mathrm{diag}(\sigma)^{-1} - b\mu^T\,\mathrm{diag}(\sigma)^{-1}, \tag{20}$$

where $X = [x_1 \cdots x_m]^T$. We carry out logistic regression (possibly regularized) using the data matrix A^{std} in place of A, to obtain (standardized) logistic model parameters w^{std}, v^{std}. In terms of the original feature vector, our logistic model is

$$\mathrm{Prob}(b|x) = \frac{1}{1 + \exp\left(-b(w^{\mathrm{std}\,T} x^{\mathrm{std}} + v^{\mathrm{std}})\right)} = \frac{1}{1 + \exp\left(-b(w^T x + v)\right)}$$

where

$$w = \mathrm{diag}(\sigma)^{-1} w^{\mathrm{std}}, \qquad v = v^{\mathrm{std}} - w^{\mathrm{std}\,T}\,\mathrm{diag}(\sigma)^{-1}\mu.$$

We point out one subtlety here related to sparsity of the data matrix. For small or medium sized problems, or when the original data matrix A is dense, forming the standardized data matrix A^{std} does no harm. But when the original data matrix A is sparse, which is the key to efficient solution of large-scale ℓ_1-regularized LRPs, forming A^{std} is disastrous, since A^{std} is in general dense, even when A is sparse.

But we can get around this problem, when working with very large problems, by never actually forming the matrix A^{std}, as explained in Section 5.

References

E. Allgower and K. Georg. Continuation and path following. *Acta Numerica*, 2:1–64, 1993.

U. Alon, N. Barkai, D. Notterman, K. Gish, S. Ybarra, D. Mack, and A. Levine. Broad patterns of gene expression revealed by clustering of tumor and normal colon tissues probed by oligonucleotide arrays. *Proceedings of the National Academy of Sciences*, 96:6745–6750, 1999.

S. Balakrishnan and D. Madigan. Algorithms for sparse linear classifiers in the massive data setting, 2006. Manuscript. Available from `www.stat.rutgers.edu/~madigan/papers/`.

O. Banerjee, L. El Ghaoui, A. d'Aspremont, and G. Natsoulis. Convex optimization techniques for fitting sparse Gaussian graphical models. In *Proceedings of the 23rd International Conference on Machine Learning*, 2006.

D. Bertsekas. *Nonlinear Programming*. Athena Scientific, second edition, 1999.

A. Bhusnurmath and C. Taylor. Solving the graph cut problem via ℓ_1 norm minimization, 2007. University of Pennsylvania CIS Tech Report number MS-CIS-07-10.

J. Borwein and A. Lewis. *Convex Analysis and Nonlinear Optimization*. Springer, 2000.

S. Boyd, S.-J. Kim, L. Vandenberghe, and A. Hassibi. A tutorial on geometric programming, 2006. To appear in *Optimization and Engineering*.

S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

S. Boyd, L. Vandenberghe, A. El Gamal, and S. Yun. Design of robust global power and ground networks. In *Proceedings of ACM/SIGDA International Symposium on Physical Design (ISPD)*, pages 60–65, 2001.

E. Candès, J. Romberg, and T. Tao. Stable signal recovery from incomplete and inaccurate measurements. *Communications on Pure and Applied Mathematics*, 59(8):1207–1223, 2005.

E. Candès, J. Romberg, and T. Tao. Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information. *IEEE Transactions on Information Theory*, 52 (2):489–509, 2006.

E. Candès and T. Tao. Decoding by linear programming. *IEEE Transactions on Information Theory*, 51(12):4203–4215, 2005.

K. Chaloner and K. Larntz. Optimal Bayesian design applied to logistic regression experiments. *Journal of Statistical Planning and Inferenc*, 21:191–208, 1989.

S. Chen and D. Donoho. Basis pursuit. In *Proceedings of the Twenty-Eighth Asilomar Conference on Signals, Systems and Computers*, volume 1, pages 41–44, 1994.

S. Chen, D. Donoho, and M. Saunders. Atomic decomposition by basis pursuit. *SIAM Review*, 43 (1):129–159, 2001.

J. Claerbout and F. Muir. Robust modeling of erratic data. *Geophysics*, 38(5):826–844, 1973.

A. Conn, N. Gould, and Ph. Toint. *LANCELOT: A Fortran package for large-scale nonlinear optimization (Release A)*, volume 17 of *Springer Series in Computational Mathematics*. Springer-Verlag, 1992.

D. Cox. Regression models and life-tables. *Journal of the Royal Statistical Society. Series B*, 34(2): 187–220, 1972.

J. Dahl, V. Roychowdhury, and L. Vandenberghe. Maximum likelihood estimation of Gaussian graphical models: Numerical implementation and topology selection. Submitted. Available from `www.ee.ucla.edu/~vandenbe/covsel.html`, 2005.

A. d'Aspremont, L. El Ghaoui, M. Jordan, and G. Lanckriet. A direct formulation for sparse PCA using semidefinite programming, 2005. In L. Saul, Y. Weiss and L. Bottou, editors, *Advances in Neural Information Processing Systems*, 17, pp. 41-48, MIT Press.

R. Dembo and T. Steihaug. Truncated-Newton algorithms for large-scale unconstrained optimization. *Math. Program.*, 26:190–212, 1983.

J. Demmel. *Applied Numerical Linear Algebra*. Society for Industrial and Applied Mathematics, 1997.

D. Donoho. Compressed sensing. *IEEE Transactions on Information Theory*, 52(4):1289–1306, 2006.

D. Donoho and M. Elad. Optimally sparse representation in general (non-orthogonal) dictionaries via ℓ^1 minimization. *Proc. Nat. Aca. Sci.*, 100(5):2197–2202, March 2003.

D. Donoho, I. Johnstone, G. Kerkyacharian, and D. Picard. Wavelet shrinkage: Asymptopia? *J. R. Statist. Soc. B.*, 57(2):301–337, 1995.

B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani. Least angle regression. *Annals of Statistics*, 32 (2):407–499, 2004.

J. Friedman, T. Hastie T, and R. Tibshirani. Pathwise coordinate optimization, 2007. Manuscript available from `www-stat.stanford.edu/~hastie/pub.htm`.

H. Fu, M. Ng, M. Nikolova, and J. Barlow. Efficient minimization methods of mixed ℓ_1-ℓ_1 and ℓ_2-ℓ_1 norms for image restoration. *SIAM Journal on Scientific computing*, 27(6):1881–1902, 2006.

A. Genkin, D. Lewis, and D. Madigan. Large-scale Bayesian logistic regression for text categorization, 2006. To appear in *Technometrics*. Available from `www.stat.rutgers.edu/~madigan/papers/`.

A. George and J. Liu. *Computer Solution of Large Sparse Positive Definite Systems*. Prentice-Hall, 1981.

P. Gill, W. Murray, M. Saunders, and M. Wright. User's guide for NPSOL (Version 4.0): A FORTRAN package for nonlinear programming. Technical Report SOL 86-2, Operations Research Dept., Stanford University, Stanford, California 94305, January 1986.

G. Golub and C. Van Loan. *Matrix Computations*, volume 13 of *Studies in Applied Mathematics*. John Hopkins University Press, third edition, 1996.

T. Golub, D. Slonim, P. Tamayo, C. Gaasenbeek, J. Mesirov, H. Coller, M. Loh, J. Downing, M. Caligiuri, C. Bloomfield, and E. Lander. Molecular classification of cancer: Class discovery and class prediction by gene expression monitoring. *Science*, 286:531–537, 1999.

J. Goodman. Exponential priors for maximum entropy models. In *Proceedings of the Annual Meetings of the Association for Computational Linguistics*, 2004.

A. Hassibi, J. How, and S. Boyd. Low-authority controller design via convex optimization. In *Proceedings of the IEEE Conference on Decision and Control*, pages 140–145, 1999.

T. Hastie, S. Rosset, R. Tibshirani, and J. Zhu. The entire regularization path for the support vector machine. *Journal of Machine Learning Research*, 5:1391–1415, 2004.

T. Hastie, J. Taylor, R. Tibshirani, and G. Walther. Forward stagewise regression and the monotone lasso. *Electronic Journal of Statistics*, 1:1–29, 2007.

T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer Series in Statistics. Springer-Verlag, New York, 2001. ISBN 0-387-95284-5.

T. Jaakkola and M. Jordan. Bayesian parameter estimation via variational methods. *Statistics and Computing*, 10:25–37, 2000.

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

Y. Kim, J. Kim, and Y. Kim. Blockwise sparse regression. *Statistica Sinica*, 16:375–390, 2006.

P. Komarek. *Logistic Regression for Data Mining and High-Dimensional Classification*. PhD thesis, Carnegie Mellon University, 2004.

B. Krishnapuram, L. Carin, M. Figueiredo, and A. Hartemink. Sparse multinomial logistic regression: Fast algorithms and generalization bounds. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(6):957–968, 2005.

B. Krishnapuram and A. Hartemink. Sparse multinomial logistic regression: Fast algorithms and generalization bounds. *IEEE Transactions on Pattern Analysis and Mach. Intelligence*, 27(6): 957–968, 2005. ISSN 0162-8828.

K. Lang. Newsweeder: Learning to filter netnews. In *Proceedings of the Twenty-First International Conference on Machine learning (ICML)*, pages 331–339, 1995.

S. Lee, H. Lee, P. Abeel, and A. Ng. Efficient l_1-regularized logistic regression. In *Proceedings of the 21st National Conference on Artificial Intelligence (AAAI-06)*, 2006.

S. Levy and P. Fullagar. Reconstruction of a sparse spike train from a portion of its spectrum and application to high-resolution deconvolution. *Geophysics*, 46(9):1235–1243, 1981.

C.-J. Lin, R. Weng, and S. Keerthi. Trust region Newton methods for large-scale logistic regression, 2007. To appear in Proceedings of the 24th International Conference on Machine Learning (ICML).

M. Lobo, M. Fazel, and S. Boyd. Portfolio optimization with linear and fixed transaction costs. *Annals of Operations Research*, 2005.

J. Lokhorst. The LASSO and generalised linear models, 1999. Honors Project, Department of Statistics, The University of Adelaide, South Australia, Australia.

D. Luenberger. *Linear and Nonlinear Programming*. Addison-Wesley, second edition, 1984.

A. McCallum. Bow: A toolkit for statistical language modeling, text retrieval, classification and clustering. Available from `www.cs.cmu.edu/~mccallum/bow`, 1996.

P. McCullagh and J. Nelder. *Generalized Linear Models*. Chapmand & Hall/CRC, second edition, 1989.

T. Minka. A comparison of numerical optimizers for logistic regression, 2003. Technical report. Available from `research.microsoft.com/~minka/papers/logreg/`.

MOSEK ApS. *The MOSEK Optimization Tools Version 2.5. User's Manual and Reference*, 2002. Available from `www.mosek.com`.

S. Nash. A survey of truncated-Newton methods. *Journal of Computational and Applied Mathematics*, 124:45–59, 2000.

Y. Nesterov and A. Nemirovsky. *Interior-Point Polynomial Methods in Convex Programming*, volume 13 of *Studies in Applied Mathematics*. SIAM, Philadelphia, PA, 1994.

D. Newman, S. Hettich, C. Blake, and C. Merz. UCI repository of machine learning databases, 1998. Available from `www.ics.uci.edu/~mlearn/MLRepository.html`.

A. Ng. Feature selection, ℓ_1 vs. ℓ_2 regularization, and rotational invariance. In *Proceedings of the Twenty-First International Conference on Machine learning (ICML)*, pages 78–85, New York, NY, USA, 2004. ACM Press. ISBN 1-58113-828-5.

J. Nocedal and S. Wright. *Numerical Optimization*. Springer Series in Operations Research. Springer, 1999.

D. Oldenburg, T. Scheuer, and S. Levy. Recovery of the acoustic impedance from reflection seismograms. *Geophysics*, 48(10):1318–1337, 1983.

M. Osborne, B. Presnell, and B. Turlach. A new approach to variable selection in least squares problems. *IMA Journal of Numerical Analysis*, 20(3):389–403, 2000.

M.-Y. Park and T. Hastie. An ℓ_1 regularization-path algorithm for generalized linear models, 2006a. To appear in *Journal of the Royal Statistical Society, Series B*. Available from `www-stat.stanford.edu/~hastie/pub.htm`.

M.-Y. Park and T. Hastie. Regularization path algorithms for detecting gene interactions, 2006b. Manuscript. Available from `www-stat.stanford.edu/~hastie/Papers/glasso.pdf`.

S. Perkins and J. Theiler. Online feature selection using grafting. In *Proceedings of the Twenty-First International Conference on Machine learning (ICML)*, pages 592–599. ACM Press, 2003.

B. Polyak. *Introduction to Optimization*. Optimization Software, 1987. Translated from Russian.

L. Portugal, M. Resende, G. Veiga, and J. Júdice. A truncated primal-infeasible dual-feasible network interior point method. *Networks*, 35:91–108, 2000.

S. Rosset. Tracking curved regularized optimization solution paths. In L. Saul, Y. Weiss, and L. Bottou, editors, *Advances in Neural Information Processing Systems 17*. MIT Press, Cambridge, MA, 2005.

S. Rosset and J. Zhu. Piecewise linear regularized solution paths, 2007. To appear in *Annals of Statistics*.

S. Rosset, J. Zhu, and T. Hastie. Boosting as a regularized path to a maximum margin classifier. *Journal of Machine Learning Research*, 5:941–973, 2004.

V. Roth. The generalized LASSO. *IEEE Transactions on Neural Networks*, 15(1):16–28, 2004.

A. Ruszczynski. *Nonlinear Optimization*. Princeton university press, 2006.

T. Ryan. *Modern Regression Methods*. Wiley, 1997.

S. Shevade and S. Keerthi. A simple and efficient algorithm for gene selection using sparse logistic regression. *Bioinformatics*, 19(17):2246–2253, 2003.

N. Z. Shor. *Minimization Methods for Non-differentiable Functions*. Springer Series in Computational Mathematics. Springer, 1985.

H. Taylor, S. Banks, and J. McCoy. Deconvolution with the l_1 norm. *Geophysics*, 44(1):39–52, 1979.

R. Tibshirani. Regression shrinkage and selection via the Lasso. *Journal of the Royal Statistical Society, Series B*, 58(1):267–288, 1996.

R. Tibshirani. The Lasso for variable selection in the Cox model. *Statistics in Medicine*, 16:385–395, 1997.

R. Tibshirani, M. Saunders, S. Rosset, and J. Zhu. Sparsity and smoothness via the fused Lasso. *Journal of the Royal Statistical Society Series B*, 67(1):91–108, 2005.

J. Tropp. Just relax: Convex programming methods for identifying sparse signals in noise. *IEEE Transactions on Information Theory*, 52(3):1030–1051, 2006.

L. Vandenberghe and S. Boyd. A primal-dual potential reduction method for problems involving matrix inequalities. *Math. Program.*, 69:205–236, 1995.

R. Vanderbei. *LOQO User's Manual — Version 3.10*, 1997. Available from `www.orfe.princeton.edu/loqo`.

M. Wainwright, P. Ravikumar, and J. Lafferty. High-dimensional graphical model selection using ℓ_1-regularized logistic regression., 2007. To appear in *Advances in Neural Information Processing Systems (NIPS) 19*. Available from `http://www.eecs.berkeley.edu/~wainwrig/Pubs/publist.html#High-dimension%al`.

S. Wright. *Primal-dual Interior-point Methods*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1997. ISBN 0-89871-382-X.

Y. Ye. *Interior Point Algorithms: Theory and Analysis*. John Wiley & Sons, 1997.

M. Yuan and L. Lin. Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society, Series B*, 68(1):49–67, 2006.

Z. Zhang, J. Kwok, and D. Yeung. Surrogate maximization/minimization algorithms for AdaBoost and the logistic regression model. In *Proceedings of the Twenty-First International Conference on Machine learning (ICML)*, pages 927–934, New York, NY, USA, 2004. ACM Press.

P. Zhao, G. Rocha, and B. Yu. Grouped and hierarchical model selection through composite absolute penalties, 2007. Tech Report 703. Stat Dept. UCB. Available from `www.stat.berkeley.edu/~binyu/ps/703.pdf`.

P. Zhao and B. Yu. On model selection consistency of Lasso. *Journal of Machine Learning Research*, 7:2541–2563, 2006.

J. Zhu, S. Rosset, T. Hastie, and R. Tibshirani. 1-norm support vector machines. In S. Thrun, L Saul, and B. Schölkopf, editors, *Advances in Neural Information Processing Systems 16*, pages 49–56, Cambridge, MA, 2004. MIT Press.

H. Zou. The adaptive Lasso and its oracle properties. *Journal of the American Statistical Association*, 101(476):1418–1429, 2006.

H. Zou, T. Hastie, and R. Tibshirani. Sparse principal component analysis. *Journal of Computational and Graphical Statistics*, 15(2):262–286, 2006.

H. Zou, T. Hastie, and R. Tibshirani. On the degrees of freedom of the Lasso, 2007. To appear in Annals of Statistics.

Journal of Machine Learning Research 8 (2007) 1557-1581 Submitted 8/06; Revised 4/07; Published 7/07

Multi-class Protein Classification Using Adaptive Codes

Iain Melvin* IAIN@NEC-LABS.COM
NEC Laboratories of America
Princeton, NJ 08540, USA

Eugene Ie* TIE@UCSD.EDU
Department of Computer Science and Engineering
University of California
San Diego, CA 92093-0404, USA

Jason Weston JASONW@NEC-LABS.COM
NEC Laboratories of America
Princeton, NJ 08540, USA

William Stafford Noble NOBLE@GS.WASHINGTON.EDU
Department of Genome Sciences
Department of Computer Science and Engineering
University of Washington
Seattle, WA 98195, USA

Christina Leslie CLESLIE@CS.COLUMBIA.EDU
Center for Computational Learning Systems
Columbia University
New York, NY 10115, USA

Editor: Nello Cristianini

Abstract

Predicting a protein's structural class from its amino acid sequence is a fundamental problem in computational biology. Recent machine learning work in this domain has focused on developing new input space representations for protein sequences, that is, string kernels, some of which give state-of-the-art performance for the binary prediction task of discriminating between one class and all the others. However, the underlying protein classification problem is in fact a huge multi-class problem, with over 1000 protein folds and even more structural subcategories organized into a hierarchy. To handle this challenging many-class problem while taking advantage of progress on the binary problem, we introduce an adaptive code approach in the output space of one-vs-the-rest prediction scores. Specifically, we use a ranking perceptron algorithm to learn a weighting of binary classifiers that improves multi-class prediction with respect to a fixed set of output codes. We use a cross-validation set-up to generate output vectors for training, and we define codes that capture information about the protein structural hierarchy. Our code weighting approach significantly improves on the standard one-vs-all method for two difficult multi-class protein classification problems: remote homology detection and fold recognition. Our algorithm also outperforms a previous code learning approach due to Crammer and Singer, trained here using a perceptron, when the dimension of the code vectors is high and the number of classes is large. Finally, we compare against PSI-BLAST, one of the most widely used methods in protein sequence analysis, and find that our method strongly outperforms it on every structure clas-

*. The first two authors contributed equally to this work.

sification problem that we consider. Supplementary data and source code are available at `http://www.cs.columbia.edu/compbio/adaptive`.

Keywords: multi-class classification, error-correcting output codes, structured outputs

1. Introduction

Numerous statistical and supervised learning methods have been developed for detecting protein structural classes from primary sequence information alone. These methods can be categorized into three major types of approaches: pairwise sequence comparison algorithms (Altschul et al., 1990; Smith and Waterman, 1981), generative models for protein families (Krogh et al., 1994; Park et al., 1998), and discriminative classifiers (Jaakkola et al., 2000; Leslie et al., 2002b; Liao and Noble, 2002; Ben-Hur and Brutlag, 2003; Saigo et al., 2004). Many recent studies (see, e.g., Leslie et al., 2004) have shown that discriminative classifiers such as support vector machines (SVMs) used with appropriate sequence representations outperform the other two types of protein classification methods in the context of binary classification, that is, prediction of whether a sequence belongs to a particular structural class or not. The binary classification performance of semi-supervised discriminative methods, which incorporate unlabeled protein sequence data into the learning algorithm, is particularly strong (Kuang et al., 2005; Weston et al., 2005). However, it is uncertain how best to leverage these accurate binary classifiers to solve the more important multi-class problem of classifying protein sequences into one of a vast number structural classes. Currently, for example, the manually curated Structural Classification of Proteins (SCOP, Murzin et al., 1995) contains more than 1000 distinct 3D conformation classes called folds and even more structural subcategories (protein superfamilies and families). This complex prediction task provides a challenging problem for multi-class algorithms.

In the machine learning literature, two main strategies have been devised to tackle multi-class problems (reviewed in Rifkin and Klautau, 2004): formulating multi-class optimization problems that generalize binary classifiers like support vector machines (Vapnik, 1998; Weston and Watkins, 1999), or reducing multi-class problems to a set of binary classification problems and processing the output vectors of binary predictions to obtain a multi-class prediction (Allwein et al., 2000; Dietterich and Bakiri, 1995). The difficulty with the first method is that one usually ends up with a complex optimization problem that is computationally expensive. We therefore focus on the second, more computationally tractable approach, which encompasses standard methods like one-vs-all, all-vs-all, and error-correcting output codes. By "one-vs-all," we refer to the procedure of training N one-vs-the-rest real-valued classifiers to obtain a length-N output vector and testing new examples by predicting the class with the largest binary prediction score. All-vs-all is similar, except that one trains all pairwise binary classifiers to obtain a length $N(N-1)/2$ output vector (Allwein et al., 2000). In error-correcting output codes (ECOC), one represents different classes by binary vectors, called output codes, in the output vector space and predicts the class based on which output code is closest to the binary output vector for the example (Dietterich and Bakiri, 1995; Crammer and Singer, 2000). Despite the wide range of proposed multi-class solutions, a recent empirical study suggests that the simple one-vs-all approach performs as well or better than all other methods in most cases (Rifkin and Klautau, 2004).

One failing of one-vs-all is that it assumes that the prediction scores of the component binary classifiers are comparable, so that the individual classifier with the largest prediction corresponds to the best class. This assumption is often invalid in practice. One proposed remedy for SVM

classifiers in particular is to fit a sigmoid function to the predicted margins for each classifier (Platt, 1999). After this procedure, the output probabilities rather than the margins are compared in one-vs-all. However, in many applications, the training data may be insufficient to fit the sigmoids accurately, or the sigmoids may be poor models for the margin distributions. Moreover, one-vs-all and the other standard output vector approaches do not take advantage of known relationships between classes, such as hierarchical relationships in the protein structural taxonomy, although there has been some recent work on hierarchical classification (Dekel et al., 2004; Cesa-Bianchi et al., 2006; Barutcuoglu et al., 2006). We further note that within the Bayesian learning community, alternative probabilistic strategies have been proposed for the multi-class problem, for example the multinomial probit model for multi-class Gaussian process classification (Girolami and Rogers, 2006).

In this work, we present a simple but effective multi-class method for protein structural classification that combines the predictions of state-of-the-art one-vs-the-rest SVM protein classifiers by supervised learning in the output space. In order to solve the problem that prediction scores from different classifiers are not on the same scale, we pose an optimization problem to learn a weighting of the real-valued binary classifiers that make up the components of the output vector. Instead of using *ad hoc* output codes as in ECOC, we design codes that are directly related to the structural hierarchy of a known taxonomy, such as SCOP, with components that correspond to fold, superfamily, and family detectors. We use a cross-validation set-up to generate output vectors as training data for learning weights, which we accomplish with a simple ranking perceptron approach. We note that Rätsch et al. (2002) considered a more general and difficult problem of adapting codes and embeddings, that is, learning both the code vectors and the embedding of the vector of prediction scores in output space via a non-convex optimization problem. In addition, Crammer and Singer (2000) formulated another more general problem of learning a mapping of all inputs to all outputs. By restricting ourselves to the simpler problem of reweighting the output space so that our fixed codes perform well, we are able to define a convex large-margin optimization problem that is tractable in very large-scale settings. We can also choose which loss function we wish to optimize. For example, in protein classification, we can use the balanced loss, so that performance on the large classes does not dominate the results.

The rest of the paper is organized as follows. In Section 2, we provide background on the protein classification problem, including our choice of base classifiers and construction of hierarchical codes. We then present our algorithmic approach for learning code weights in the output space using the ranking perceptron, describe different perceptron update rules, and compare to the code learning method of Crammer and Singer (2000) and other related work in Section 3. We provide large-scale experimental results on the multi-class remote homology detection and fold recognition problems in Section 4, comparing our approach with a number of alternatives: standard one-vs-all, sigmoid fitting, PSI-BLAST (Altschul et al., 1997) used in a nearest neighbor approach to make multi-class predictions, and a perceptron version of Crammer and Singer's code learning method. We find that our adaptive code approach significantly outperforms one-vs-all in both multi-class problem settings and over all choices of code elements. We also strongly outperform PSI-BLAST for every structural classification problem that we consider. Finally, we find that our code learning algorithm obtains significantly better results than the higher capacity scheme of Crammer and Singer in the setting where the number of classes and dimension of the output space are both high. The current work is an expanded version of a conference proceedings paper (Ie et al., 2005). For this version, we have provided a much larger-scale experimental validation, added results on the fold recognition

problem, introduced improved perceptron update rules and extended code vectors, and included a comparison with the Crammer and Singer method.

Our adaptive code algorithm is used in a newly deployed protein fold recognition web server available called SVM-Fold, available at `http://svm-fold.c2b2.columbia.edu`.

2. Background on Protein Classification: Problems, Representations, and Codes

In this section, we first discuss protein classification problems in general. We then give an overview of our method.

2.1 Remote Homology Detection and Fold Recognition

Protein classification is the prediction of a protein's structural class from its primary sequence of amino acids. This prediction problem is of fundamental importance in computational biology for a number reasons. First, a protein's structure is closely linked to its biological function, so knowledge of the structural category can allow improved prediction of function. Moreover, experimental methods for determining the full 3D structure of a protein (X-ray crystallography, NMR) are time consuming and difficult and cannot keep pace with the rapid accumulation of unannotated protein sequences from newly sequenced genomes. Indeed, the complete repository of known protein structures, deposited in the Protein Data Bank, contains just 27K structures, while there are about 1.5M protein sequences in the Non-redundant Database of protein sequences. Second, prediction of a protein sequence's structural class enables the selection of a template structure from the database, which can then used with various comparative modeling techniques to predict a full 3D structure for the protein. Predicted structures are important for more detailed biochemical analysis and in particular for drug design. Note that template-based modeling approaches far outperform *ab ini-*

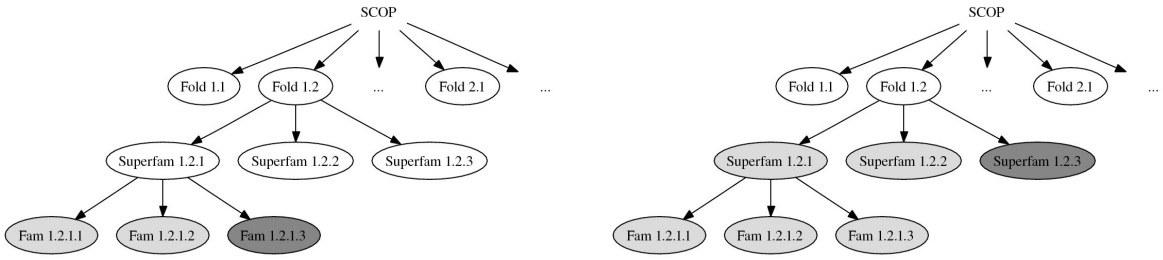


Figure 1: Two protein classification problems. (Left) In the SCOP database, we simulate the remote homology detection problem by holding out a test family (shown in dark gray) from a superfamily and using the other families as positive training data (shown in light gray). The task is to correctly predict the superfamily or fold membership of the held-out sequences. (Right) We simulate the fold recognition problem by holding out a test superfamily (dark gray) from a fold and using the other superfamilies as training data (light gray). The task is to correctly recognize the fold of the held-out sequences.

tio techniques for protein structure, that is, methods that search for conformations that optimize an energy function without any template structure.

In this work, we focus on two protein classification problems that are considered unsolved in the structural biology community: remote homology detection and fold recognition. In remote homology detection, we wish to recognize when a new protein sequence has a distant evolutionary relationship to a protein sequence in a database (e.g., one whose structure is known). Due to a distant common ancestor, the protein sequences exhibit subtle sequence similarities (remote homology) that cannot generally be detected by statistical, alignment-based methods (Altschul et al., 1990, 1997). In fold recognition, we wish to recognize when a new protein sequence will exhibit the same fold as a protein from the structure database, even is there is no evidence of any evolutionary relationship between the proteins.

We base our experiments on SCOP, a manually curated hierarchical classification system for known protein structures. At the top level of the hierarchy are SCOP folds, consisting of sequences that have the same general 3D structural architecture. SCOP folds are divided into superfamilies, containing sequences that are at least remotely homologous (evolutionarily related). Each superfamily is further divided into families, consisting of homologous sequences with an easily detectable level of sequence similarity. We can design experiments based on the SCOP hierarchy to test performance on both the remote homology detection and the fold recognition problem, as depicted in Figure 1.

2.2 Profile-Based Fold, Superfamily and Family Detectors

For our base binary classifiers, we use profile-based string kernel SVMs (Kuang et al., 2005) that are trained to recognize SCOP fold, superfamily, and family classes. We call these trained SVMs *fold detectors*, *superfamily detectors*, and *family detectors*. The profile kernel is a function that measures the similarity of two protein sequence profiles based on their representation in a high-dimensional vector space indexed by all k-mers (k-length subsequences of amino acids). A sequence profile is based on a multiple alignment of protein sequences to the input sequence and simply refers to the position-specific distribution of amino acids estimated from each column of the alignment. Intuitively, we use each k-length window of the sequence profile to define a *positional mutation neighborhood* of k-mers that satisfy a likelihood threshold, and the underlying feature map counts the k-mers from all the positional neighborhoods.

Specifically, for a sequence x and its sequence profile $P(x)$, the *positional mutation neighborhood* at position j and with threshold σ is defined to be the set of k-mers $\beta = b_1 b_2 \ldots b_k$ satisfying a likelihood inequality with respect to the corresponding block of the profile $P(x)$, as follows:

$$M_{(k,\sigma)}(P(x[j+1:j+k])) =$$
$$\{\beta = b_1 b_2 \ldots b_k : -\sum_{i=1}^{k} \log p_{j+i}(b_i) < \sigma\}.$$

Note that the emission probabilities, $p_{j+i}(b), i = 1 \ldots k$, come from the profile $P(x)$; for notational simplicity, we do not explicitly indicate the dependence on x.

We now define the profile feature mapping as

$$\Phi^{\text{Profile}}_{(k,\sigma)}(P(x)) = \sum_{j=0\ldots|x|-k} (\phi_\beta(P(x[j+1:j+k])))_{\beta\in\Sigma^k}$$

where the coordinate $\phi_\beta(P(x[j+1:j+k])) = 1$ if β belongs to the mutation neighborhood $M_{(k,\sigma)}$ $(P(x[j+1:j+k]))$, and otherwise the coordinate is 0. The profile kernel between two protein sequences, that is, the inner product of feature vectors, can be efficiently computed from the original pair of profiles using a trie data structure (Kuang et al., 2005).

The use of profile-based string kernels is an example of semi-supervised learning, since unlabeled data in the form of a large sequence database is used in the discrimination problem (specifically, to estimate the probabilistic profiles).

A large variety of kernels have been designed specifically for protein sequences (e.g., Jaakkola et al., 2000; Liao and Noble, 2002; Leslie et al., 2002a,b; Weston et al., 2005; Saigo et al., 2004; Ben-Hur and Brutlag, 2003; Rangwala and Karypis, 2005). For this work, we selected the profile kernel because it is state-of-the-art. However, we have no reason to suspect that our primary conclusions regarding various multi-class classification methods depend on the choice of kernel function.

2.3 PSI-BLAST Family Detectors

PSI-BLAST (Altschul et al., 1997) is a widely used sequence comparison algorithm that builds a probabilistic profile around a query sequence, based on iterative alignment to database sequences. The resulting profile is then used to evaluate pairwise sequence similarities between the query and target sequences in the database. PSI-BLAST reports the significance of the similarity as an E-value (defined as the expected number of times that a similarity as strong or stronger than the observed similarity would be observed in a random protein sequence database of the given size), based on a profile-sequence alignment score. In theory, the E-value calculation makes PSI-BLAST results from different queries comparable to each other.

In this work, we use PSI-BLAST as a baseline method for comparison as well as a tool for generating sequence profiles for the profile kernel. In addition, we use PSI-BLAST to define an additional set of base classifiers for extended components of the output vectors in our multi-class approach.

For a given input sequence, we compute 1.0 - (the smallest PSI-BLAST E-value from the input sequence to training proteins in the family) and use this value as the component for the PSI-BLAST family detector in the discriminant vectors. Sequences with high similarity to one of the training sequences in the family receive a prediction score close to 1; if the E-value is highly insignificant, the score will be negative and large.

2.4 Output Vectors and Codes

To incorporate hierarchical labels into our output representation, we simply concatenate into a single output vector the one-vs-the-rest classification scores for classes at all relevant levels of the hierarchy. For example, in either the remote homology detection or fold recognition setting, both fold and superfamily detectors may be relevant for making fold-level predictions on test examples. Suppose the number of superfamilies and folds in a SCOP-based data set is k and q respectively. Then the real-valued output vector for each test sequence x would be $\vec{f}(x) = (f_1(x), ..., f_{k+q}(x))$, where the f_i are binary SVM superfamily or fold detectors trained using profile string kernels as described above. One can also extend the output vector to include binary family detectors. More generally, in this work we consider code elements that correspond to binary class detectors at the same level in the hierarchy as the target class (e.g., the fold level for fold predictions) as well as sublevels of the target class.

In the same output space, we define binary vectors that represent the hierarchical labels relevant for the problem. For the fold-level prediction example above, we define for superfamily classes $j \in \{1,...,k\}$ the code vectors $\mathbf{C}_j = (\text{superfam}_j, \text{fold}_j)$, where superfam_j and fold_j are vectors with length equal to the number of known superfamilies (k) and folds (q), and each of these two vectors has exactly one non-zero component corresponding to structural class identity.

Our main idea is to learn a weight vector $\mathbf{W} = (W_1, \ldots, W_{k+q})$ to perform multi-class predictions with the weighted code prediction rule, $\hat{y} = \arg\max_j (\mathbf{W} * \vec{f}(x)) \cdot \mathbf{C}_j$, where $\mathbf{W} * \vec{f}(x)$ denotes component-wise multiplication. In the next section, we describe how to learn $\mathbf{W}$ by using a ranking perceptron with a cross-validation set-up on the training set, and we develop update rules suited to the hierarchical problem.

3. Multi-Class Algorithms

In this section we describe methods for combining binary classification models into multiclass algorithms.

3.1 Motivation: Optimizing Weights in Output Space

Given a fixed set of binary codes $\mathbf{C}_j$ and real-valued output vectors $\vec{f}(x)$ in the same output space $\mathbb{R}^N$ (see Section 2.4 for an example where $N = k+q$=#folds + #superfamilies), we want to adapt the coding system by learning a weight vector $\mathbf{W}$ so that the multi-class prediction rule $\hat{y} = \arg\max_j (\mathbf{W} * \vec{f}(x)) \cdot \mathbf{C}_j$ gives good empirical loss on the training data and generalizes well.

To learn $\mathbf{W}$, we first propose a hard margin optimization problem as

$$\min_{\mathbf{W}} ||\mathbf{W}||_2^2, \tag{1}$$

subject to

$$\left(\mathbf{W} * \vec{f}(x_i)\right) \cdot (\mathbf{C}_{y_i} - \mathbf{C}_j) \geq 1, \ \forall j \neq y_i.$$

Intuitively, our problem is to find an optimal weighting of the output vector elements such that the re-weighted embedding of examples in the output space $\mathbb{R}^N$ will exhibit a large margin between correct and incorrect codes.

We use the ranking perceptron to find an approximate solution to this optimization problem, though a structured SVM approach (see Section 3.4.4) is also possible. Since for the SVM base classifiers in particular, discriminant scores on training sequences are not very informative, we use a cross-validation set-up to produce prediction scores for the weight learning optimization. The full methodology consists of five steps: (1) split the training data into 10 cross-validation sets; (2) for each held-out fold, train a collection of fold-, superfamily-, and family-level detectors on the remaining data and use them to generate real-valued predictions on the held-out fold; (3) using the cross-validation scores to form output vectors, learn code weights with the ranking perceptron algorithm; (4) re-train fold, superfamily, and family detectors on the full training set; and (5) test on the final untouched test set.

3.2 Learning Weights with the Ranking Perceptron

The ranking perceptron algorithm (Collins and Duffy, 2002) is a variant of the well-known perceptron linear classifier (Rosenblatt, 1958). In our experiments, the ranking perceptron receives as input

(A) Code weights learning

Define $F(x,y) = \mathbf{W} \cdot (\vec{f}(x) * \mathbf{C}_y)$
Input ν:
$\mathbf{W} \leftarrow \vec{\mathbf{0}}$
for $i = 1$ to n **do**
  $k = \arg\max_{p \in \{Y - y_i\}} F(x_i, p)$
  if $F(x_i, y_i) - m < F(x_i, k)$ **then**
    $\mathbf{W} \leftarrow \mathbf{W} + \nu n_i^{-1} \left(\vec{f}(x_i) * \mathbf{C}_{y_i} - \vec{f}(x_i) * \mathbf{C}_k \right)$
  end if
end for
Return $\mathbf{W}$

(B) Class prediction

Define $F(x,y) = \mathbf{W} \cdot (\vec{f}(x) * \mathbf{C}_y)$
Input $\mathbf{W}, x_i$:
Return $\hat{y} \leftarrow \arg\max_j F(x_i, j)$

Figure 2: Pseudocode for the ranking perceptron algorithm used to learn code weighting. In the pseudocode, ν is the learning parameter; $n_i = |\{y_j : y_j = y_i\}|$ for balanced-loss, and $n_i = 1$, for zero-one loss.

the discriminant vectors for training sequences (generated through a cross-validation procedure) and produces as output a weight vector $\mathbf{W}$ which is a linear combination of the discriminant vectors projected onto the non-zero components of codes. We modify the ranking perceptron algorithm such that it will learn our weight vector $\mathbf{W}$ by satisfying n constraints:

$$\mathbf{W} \cdot (\vec{f}(x_i) * \mathbf{C}_{y_i} - \vec{f}(x_i) * \mathbf{C}_j) \geq m, \ \forall j \neq y_i, \tag{2}$$

where m is the size of the margin that we enforce (Figure 2).

The update rule of the ranking perceptron algorithm depends upon what loss function one is aiming to optimize. In standard *zero-one loss* (or classification loss), one counts all prediction mistakes equally,

$$l_z(y, \hat{y}) = \begin{cases} 1 & \text{if } \hat{y} \neq y; \\ 0 & \text{otherwise.} \end{cases}$$

The final zero-one empirical loss is $\frac{1}{n} \sum_i l_z(y_i, \hat{y}_i)$. In *balanced loss*, the cost of each mistake is inversely proportional to the true class size,

$$l_b(y, \hat{y}) = \begin{cases} \frac{1}{|y_i : y_i = y|} & \text{if } \hat{y} \neq y; \\ 0 & \text{otherwise.} \end{cases}$$

The final balanced empirical loss is $\frac{1}{|Y|} \sum_i l_b(y_i, \hat{y}_i)$, where Y denotes the set of output labels.

Balanced loss is relevant to the protein structure prediction because class sizes are unbalanced, and we do not want to perform well only on the largest classes. The particular ranking perceptron training and prediction algorithms that we use are summarized in the pseudocode in Figure 2, including update rules for both zero-one and balanced loss.

3.3 The Friend/Foe and Mean Friend/Foe Update Rules

When using codes representing multiple levels of the label hierarchy, we can also use relationships between codes to redefine the perceptron update rule. For a given label y, let friends(y) be the

set of codes for classes belonging to the same superclass as y. For example, if we are using both superfamily and fold detectors for fold-level predictions and $C_y = (\text{superfam}_y, \text{fold}_y)$, the set of friends would be the codes for the superfamilies in the same fold as y (in particular, y itself belongs to friends(y)). We let foes(y) be all the codes that are not in friends(y). Then we can use the following update rule that updates all of friends(y) when the weakest friend does not beat the strongest foe by a margin:

1: $k = \arg\min_{p \in \{\text{friends}(y_i)\}} F(x_i, p)$
2: $l = \arg\max_{p \in \{\text{foes}(y_i)\}} F(x_i, p)$
3: **if** $F(x_k, \text{k}) - m < F(x_i, \text{l})$ **then**
4: $\quad \mathbf{W} \leftarrow \mathbf{W} + \nu n_i^{-1} \left(\vec{f}(x_i) * \mathbf{C}_{\text{friends}(y_i)} - \vec{f}(x_i) * \mathbf{C}_l \right)$
5: **end if**

In this rule, the vector $\mathbf{C}_{\text{friends(y)}}$ is the binary OR of all code vectors belonging to the superclass of y. We also implement a "mean friend/foe" rule whereby each element of $\mathbf{C}_{\text{friends(n)}}$ is the arithmetic mean of the occurrences of the code in all code vectors belonging to the superclass.

3.4 Comparison with Existing Approaches

In this section we compare our method to a range of existing approaches.

3.4.1 THE RANKING PERCEPTRON AND STRUCTURED OUTPUT LEARNING

The ranking perceptron (Collins, 2000; Freund and Schapire, 1999) has been used to solve structured learning problems, particularly in natural language processing. Collins and Duffy (2002) trained parsers on the Penn Treebank that output a parse tree given a input sentence. This mapping from structured input to structured output is achieved by embedding both input and output objects into a joint feature space and computing for a test example:

$$\hat{y} = \arg\max_{y \in Y} \langle \mathbf{W}, \psi(x, y) \rangle.$$

Our approach follows the same framework, but we compute $\hat{y} = \arg\max_j (\mathbf{W} * \vec{f}(x)) \cdot \mathbf{C}_j$, where $\mathbf{W} * \vec{f}(x)$ denotes component-wise multiplication. That is, we choose the joint feature embedding:

$$\psi(x, y) = \vec{f}(x) * \mathbf{C}_y = \sum_i (\vec{f}(x))_i (C_y)_i.$$

Our approach is thus an instance of structured output learning where the joint feature space captures dependencies between input and output variables. Firstly, inputs are modeled by $\vec{f}$ which uses classifiers relating to the levels of the label hierarchy in SCOP. Secondly, the outputs are modeled such that they are only dependent on classifiers from the relevant node or its ancestors.

3.4.2 MULTI-CLASS SVMS

Multi-class SVMs (Weston and Watkins, 1999; Vapnik, 1998) are a generalization of SVMs that handle the multi-class classification case by optimizing a single objective function. They use the rule $\hat{y} = \arg\max_{y \in Y} \langle w_i \cdot x \rangle$ as in the one-versus-all approach but enforce constraints of the form:

$$\langle w_{y_i} \cdot x \rangle - \langle w_y \cdot x \rangle > 1 \quad \forall y \in \mathcal{Y}, y \neq y_i.$$

1: Define $F(x,y) = \mathbf{W_y} \cdot (\vec{f}(x))$
2: $k = \arg\max_{p \in \{Y - y_i\}} F(x_i, p)$
3: **if** $F(x_i, y_i) - m < F(x_i, k)$ **then**
4: $\quad \mathbf{W_{y_i}} \leftarrow \mathbf{W_{y_i}} + \nu n_{y_i}^{-1} \vec{f}(x_i)$
5: $\quad \mathbf{W_k} \leftarrow \mathbf{W_k} - \nu n_j^{-1} \vec{f}(x_i)$
6: **end if**

Figure 3: Pseudo-code for the perceptron implementation of the Crammer-Singer code learning method detailed in Crammer and Singer (2000).

Crammer and Singer (2002) later extended this approach by simplifying computations in the non-separable case (although the separable case remains the same). These constraints are identical to the ranking perceptron approach if, for that method, one selects the embedding (see Tsochantaridis et al., 2004):

$$\psi(x,y) = \phi(x) \otimes \Lambda(y), \tag{3}$$

where $\Lambda(y) = (\delta_{1,y}, .., \delta_{N,y})^\top \in \{0,1\}^K$, K is the number of clases, $\delta_{\cdot,\cdot}$ is the Kronecker delta, and $(a \otimes b)_{j+(i-1)K)} = a_i \cdot b_j$.

Multi-class SVMs are rather slow to train for non-linear systems which use kernels, since training in the dual has worst case complexity $O((mK)^3)$, where m is the number of training examples and K is the number of classes (Hsu and Lin, 2002). For this reason, we do not compare our approach to this method. However, our two-stage approach of training one-vs-all classifiers and then improving their performance with adaptive codes is tractable, because we train in primal variables and only have N input features, equal to the number of classifiers trained.

3.4.3 THE CRAMMER-SINGER METHOD

Crammer and Singer (2000) also suggested a method for learning codes in a two-step fashion, which represents the closest algorithmic approach to ours that we know of. In their method, N weight vectors (one for each class) are optimized simultaneously but with no dependency between input and output features. The classification rule they use is $\hat{y} = \arg\max_i \mathbf{W_i} \cdot \vec{f}(x)$, and the feature space they use is identical to (3). In other words, they use the multi-class SVM to learn the second stage of their two-stage approach. This formulation means that the prediction of a given label i could depend on the output of any of the classifiers from the first stage, if the weight vector learned is not sparse. By contrast, in our approach, the i^{th} label only depends on classifiers related to that label in the label hierarchy. In the case of a flat hierarchy (pure multi-class classification), our approach only rescales the classifiers in the one-vs-all approach, whereas the Crammer-Singer method learns, for each output, a weighted combination from all the one-vs-all classifiers. That is, they learn the more difficult problem of a mapping of all inputs to all outputs. Because of this, we hypothesise that the Crammer-Singer approach is likely to fail in the case of a large number of classes, as uncorrelated classes are essentially noisy features in the second stage of learning.

In Section 4 we make a comparison between our method and the Crammer-Singer approach. To facilitate comparisons, we implemented a perceptron-style learning of their algorithm (Figure 3), both with and without balanced loss. In our experiments, which use a very large number of classes, our approach indeed does outperform the one of Crammer and Singer.

3.4.4 USING STRUCTURED SVMS TO LEARN CODE WEIGHTS

Support vector machines have been applied to problems with interdependent and structured output spaces in Tsochantaridis et al. (2004). These authors make use of a combined input-output feature representation $\psi(x,y)$ as training vectors to learn a linear classification rule $\hat{y} = \arg\max_{y \in Y} \langle \mathbf{W}, \psi(x,y) \rangle$. Specifically, they use the $\psi(\cdot,\cdot)$ relation to discover input-output relations by forming $n|Y| - n$ linear constraints. These linear constraints specify that all correct input-output structures must be clearly separated from all incorrect input-output structures,

$$\langle \mathbf{W}, \delta\psi_i(y) \rangle > 0 \quad \forall i, y \neq y_i,$$

where $\delta\psi_i(y) \equiv \psi(x_i,y_i) - \psi(x_i,y)$. By defining, $\psi(x_i,y) = \vec{f}(x_i) * \mathbf{C}_y$, we arrive at linear constraints that are a special case of Equation 2. Using standard maximum-margin methods like SVMs, we obtain the hard margin problem described by (1) above and the soft margin problem

$$\min_{\mathbf{W},\xi} \tfrac{1}{2}||\mathbf{W}||_2^2 + \tfrac{C}{n}\textstyle\sum_{i=1}^{n} \xi_i$$
$$\forall i, \xi_i \geq 0; \forall i, \forall y \in \{Y - y_i\} : \langle \mathbf{W}, \delta\psi(y) \rangle \geq 1 - \xi_i,$$

where the ξ_i correspond to slack variables (the amount an example can violate the margin), and C corresponds to the trade-off between maximizing the margin and the degree to which noisy examples are allowed to violate the margin.

Intuitively, our definition of ψ defines the distance between two different protein embeddings in code space, and we are using large margin SVM methods to find the relative weighting of the dimensions in code space. Moreover, one can optimize the balanced loss by rescaling the slack variables $\xi_i \leftarrow \dfrac{\xi_i}{l_b(y_i,y)}$ in the constraint inequalities. However, in preliminary results (Ie et al., 2005), we found that the structured SVM gave similar performance to the ranking perceptron when used with our joint input-output embedding ψ, so we focus on perceptron approaches in the current study.

3.4.5 LOSS FUNCTIONS AND ENERGY BASED LEARNING

Another general approach to structured output learning, called Energy Based Learning (EBL), was suggested by LeCun and Huang (2005), which is derived from the earliest approach to structured output prediction that we know of (Bottou et al., 1997). In EBL, for a given input, one chooses the output with the lowest energy:

$$\hat{y} = \arg\min_{y \in \mathcal{Y}} E(x,y).$$

One therefore seeks to use a loss function that pushes "down" the energy of the correct output(s) and pushes "up" the energy of other outputs. The authors show how different choices of loss function lead to different existing algorithms, including the ranking perceptron, which only pushes up the incorrect answers produced by the model, and negative log-likelihood, which pushes up the energies for all the examples with a force proportional to the likelihood of each answer under the model. Our friend/foe update rule can be seen in this framework as a different loss function that takes account of multiple output values that all give the same original loss.

3.4.6 RÄTSCH ET AL.'S ADAPTIVE CODE LEARNING

Rätsch et al. (2002) also considered the problem of adaptive code learning and proposed a general approach consisting of learning both code vectors and the embedding of the vector of prediction scores in output space. Their algorithm involves iteration between learning the codes and learning the embedding, resulting in a difficult non-convex optimization problem.

By restricting ourselves to the simpler problem of reweighting the output space so that our fixed codes perform well, we are able to define a convex large-margin optimization problem that is tractable in very large scale settings. Furthermore, by training our second-stage code learning by first running cross-validation on the first-stage predictors, we attempt to learn correcting codes which minimize the cross-validation classification error.

3.4.7 PLATT'S SIGMOID METHOD

Platt (1999) proposed a method for estimating posterior probabilities from SVM outputs in order to enable various kinds of post-processing. By converting the outputs of one-vs-the-rest SVM classifiers to class-specific posterior probabilities, in principle the probabilities are comparable and multi-class prediction through a one-vs-all strategy should improve. Platt's approach involves fitting a sigmoid for the posterior distribution,

$$P(y=1|f) = \frac{1}{1+\exp(Af+B)},$$

using training data of the form $\{(f_i,t_i)\}$, where f_i is the output of a trained SVM and t_i is the 0 or 1 target probability. The parameters A and B are found by maximizing the log likelihood of this training data. Typically, one would want to use a held-out set or cross-validation to generate outputs for fitting the sigmoid, since the outputs for the examples used to train the SVM give a biased estimate of the true output distribution. In addition to Platt's original sigmoid fitting algorithm, Lin et al. (2003) have proposed a more robust procedure.

However, in some cases, the sigmoid may be a poor description of the posterior probability, or there may be too little positive training data to properly fit the sigmoid. In our preliminary results (Ie et al., 2005), we found that sigmoid fitting performed poorly in our problem setting on a smaller data set. We retest here on larger benchmarks and again find that sigmoid fitting does not improve over one-vs-all for this problem (see Section 4).

3.4.8 IN DEFENSE OF ONE-VS-ALL

A recent empirical study suggests that the simple one-vs-all approach performs as well or better than all other multi-class methods in most cases (Rifkin and Klautau, 2004) when all the methods are well-tuned. However, the authors use only data sets with relatively few classes (between 4 and 48) for comparison and readily admit that that they use "toy data sets" from the UCI repository. Intuitively, the one-vs-all approach can be quite brittle in the many-class case: if only a single classifier is "corrupted" and always outputs a high score, then all of the examples can be misclassified. The more classes one has, the more chance that such corruptions can take place. In multi-class protein prediction, one has hundreds or thousands of classes. We present experimental results to show that the adaptive code approach improves over the "one-vs-all" by reweighting and effectively correcting such mistakes (see Section 4). Moreover, our approach also offers control of the loss function (such as using balanced loss) and use of hierarchical labels, which are not possible in one-vs-all.

4. Experimental Results

In this section we describe our data sets, methods and experimental results.

4.1 Data Sets

We assembled benchmark data sets for the remote homology detection and fold recognition problems using sequences from the SCOP 1.65 protein database (see Section 2.1 for a definition of these problems). We used ASTRAL (Brenner et al., 2000) to filter these sequences so that no two sequences share greater than 95% identity.

For the fold recognition problem, we designed our experiments so that the test set consists of held-out superfamilies belonging to folds that are represented in the training data. We prepared a data set by first removing all superfamilies that have less than 5 sequence examples. We then removed all folds that have less than 3 superfamilies. We selected superfamilies for testing at random from the remaining superfamilies such that the test set for the superfamily contains no more than 40% of the remaining sequences for the fold. If at least one suitable superfamily could not be found, then the fold was removed from the experiment. The resulting fold detection data set contains of 26 folds, 303 superfamilies, and 652 families for training. We completely hold out 614 sequences from 46 superfamilies for testing.

For the remote homology detection, the test set should contain held-out families belonging to superfamilies that are represented in the training data. One can evaluate performance for multi-class prediction of fold or superfamily levels, and it is natural to try different codes for these two tasks; therefore, we prepared a separate data set for remote homology superfamily and fold detection. For the superfamily data set, we used the same selection scheme as for fold recognition, except the minimum number of sequences for the children of the superfamilies is relaxed to 3, and we selected random families for testing instead of superfamilies. The resulting superfamily detection data set contains of 74 superfamilies, and 544 families for training. We completely hold out 802 sequences from 110 families for testing.

For the remote homology fold detection data set, we first removed all superfamilies with less than 2 families. We then selected families from the remaining superfamilies for testing. We selected families at random from each superfamily such that we never selected more than 40% of the parent superfamily for testing. If no such families were found then the superfamily was removed from the data set. If a fold was then found to have no superfamilies with held out families for testing, it was removed from the data set. The resulting remote homology detection set contains 44 folds, 424 superfamilies, and 809 families for training. We completely hold out 381 sequences from 136 families for testing.

We use the training sequences in a cross-validation set-up to obtain classification scores and learn code weights. When training base classifiers, we only use negative data from outside of the target class of the experiment. For fold recognition, this means that when we train superfamily or family detectors, we exclude negative example sequences that come from the parent fold. We then retrain the base classifiers on all the training data to generate prediction scores for the test sequences, and then use the weighted code vectors to obtain multi-class predictions.

4.2 Methods

We test our weight learning approach using the ranking perceptron with the class-based, friend/foe, and mean class update rules for a variety of code sets for the remote homology detection and fold recognition problems. For each choice of codes, we compare against standard one-vs-all, sigmoid fitting using the robust procedure described in Lin et al. (2003), and a ranking perceptron version of the Crammer and Singer code learning method (Crammer and Singer, 2000). We do not test the SVM-struct implementation of our code learning optimization problem, since our preliminary results showed little difference in performance between the perceptron and SVM-struct on this problem (Ie et al., 2005).

As an additional baseline method, we also test PSI-BLAST, a widely used pairwise sequence comparison algorithm. In order to produce multi-class predictions, we use PSI-BLAST E-values as a distance measure for a nearest neighbor approach. PSI-BLAST E-values are not symmetric, since PSI-BLAST obtains somewhat different results depending on whether it builds a profile around each training sequence or each test sequence; however, preliminary results suggested that nearest neighbor performance was not significantly affected by this choice (Ie et al., 2005). Therefore, we use PSI-BLAST E-values based on training sequence profiles, which is the more computationally efficient choice.

For all ranking perceptron experiments, we train the perceptron algorithm for 200 iterations. When using SVM one-vs-all classifier codes, the learning parameter for all ranking perceptron experiments is set to 0.01, and the required margin is chosen to be $m = 2$. For ranking perceptron on one-vs-all classifier codes with PSI-BLAST extension, we set the initial weights on the PSI-BLAST portion of the codes to 0.1. We also use two learning parameters, 0.01 for the SVM portion and 0.001 for the PSI-BLAST portion. This choice effectively stops our algorithm from adjusting weights in the PSI-BLAST part of the code. We take this approach because the individual PSI-BLAST codes are derived from E-values and hence should already be comparable to each other. We use the same required margin of $m = 2$ in the ranking perceptron algorithm.

4.3 Remote Homology Detection Results

For the remote homology detection data set, where the test set consists of held-out protein families that belong to superfamilies represented in the training data, we evaluate performance both for the superfamily-level and fold-level prediction tasks. Results for multi-class superfamily and fold prediction are provided in Tables 1 and 2, respectively. Significance tests are given comparing the methods in Tables 4, 5, 7, and 8. The last two tables use a balanced error measure by averaging the error rates over each prediction label before computing the significance test.

We compare our adaptive code method to PSI-BLAST, a standard homology detection method based on sequence alignment, as well as simple one-vs-all, sigmoid fitting, and the Crammer-Singer method, using various choices of code vectors. In addition to reporting classification loss and balanced loss results, we give "top 5" classification and balanced loss performance, which evaluates whether the correct class was found in the top 5 class predictions. The motivation for top 5 loss results is that a structural biologist might be willing to investigate a small number of false positives if it was likely that the list also contained the true structural class.

For the superfamily prediction task, we find that the adaptive codes method significantly outperforms one-vs-all both in terms of classification and balanced error, even when superfamily-only codes are used, and performance improves as more elements are added to the codes. By contrast,

Method (and optimization target)	Error	Balanced Error	Top 5 Error	Balanced Top 5 Error
PSI-BLAST	0.399	0.457	0.273	0.365
one-vs-all: Sfams	0.271	0.445	0.105	0.197
one-vs-all: Sfams,Fams	0.271	0.445	0.110	0.207
Sigmoid Fitting: Sfams	0.365	0.547	0.197	0.369
Adaptive Codes: Sfams (zero-one)	0.247	0.385	0.096	0.148
Adaptive Codes: Sfams (balanced)	0.247	0.362	0.110	0.161
Adaptive Codes: Sfams,Fams (zero-one)	0.243	0.382	**0.090**	**0.141**
Adaptive Codes: Sfams,Fams (balanced)	0.239	0.352	0.107	0.162
Adaptive Codes: Sfams,Fams,PSI-Fams (zero-one)	0.223	0.338	0.094	0.142
Adaptive Codes: Sfams,Fams,PSI-Fams (balanced)	**0.217**	**0.320**	0.103	0.153

Table 1: Results for multi-class superfamily prediction in the remote homology detection set-up. Results for the adaptive code method are reported for a SCOP benchmark data set (67 folds, 74 superfamilies, 544 families, with 802 test sequences) and compared to nearest neighbor using PSI-BLAST, standard one-vs-all, and a perceptron version of the Crammer and Singer method. The mean class update rule is used to train the adaptive weights method.

Method (and optimization target)	Error	Balanced Error	Top 5 Error	Balanced Top 5 Error
PSI-BLAST	0.409	0.443	0.297	0.367
one-vs-all: Folds	0.331	0.456	0.126	0.195
one-vs-all: Folds,Sfams	0.331	0.456	0.126	0.195
Sigmoid Fitting: Folds	0.339	0.514	0.163	0.329
Adaptive Codes: Folds (zero-one)	0.307	0.383	0.121	0.177
Adaptive Codes: Folds (balanced)	0.336	0.378	0.165	0.186
Adaptive Codes: Folds,Sfams (zero-one)	0.276	0.370	0.118	0.182
Adaptive Codes: Folds,Sfams (balanced)	0.297	0.351	0.134	0.173
Adaptive Codes: Folds,Sfams,Fams (zero-one)	**0.252**	0.351	**0.100**	0.168
Adaptive Codes: Folds,Sfams,Fams (balanced)	0.265	**0.340**	0.115	**0.142**

Table 2: Results for multi-class fold prediction in the remote homology detection set-up. Results for the adaptive codes method are reported for a SCOP benchmark data set (44 folds, 424 superfamilies, 809 families, with 381 test sequences) and compared to nearest neighbor using PSI-BLAST, standard one-vs-all, and a perceptron version of the Crammer and Singer method. The mean class update rule is used to train the adaptive weights method.

the Crammer-Singer code-learning method does not beat simple one-vs-all for this task, and performance tends to degrade as more elements are added to the codes. We also note that sigmoid fitting

gives substantially worse performance than one-vs-all for this task. When compared to the widely-used PSI-BLAST method, even simple one-vs-all outperforms PSI-BLAST strongly in terms of classification error and slightly in terms of balanced error; adaptive codes outperforms PSI-BLAST very strongly by both measures and also when considering "top 5" prediction performance.

For the fold prediction task, we use a different set of codes, including code elements corresponding to protein fold detectors. We observe a similar trend, but with better results for Crammer-Singer when compared to one-vs-all. In this case, both Crammer-Singer and adaptive codes beat one-vs-all with respect to classification and balanced loss when fold-only codes are used; in fact, for fold-only codes, performance of Crammer-Singer is slightly better than adaptive codes. However, as we add more code elements, the performance of Crammer-Singer degrades while adaptive codes continues to improve, so that the best result for our method (corresponding to the longest code that we tried) is better than the best result for Crammer-Singer (the shortest code). The best results for both methods are significantly better than PSI-BLAST. Finally, sigmoid fitting slightly degrades performance as compared to one-vs-all.

Overall, we observe that when the individual code elements are helpful, as seems to be the case in remote homology detection, our adaptive code method can successfully improve performance by adding elements without overfitting. By contrast, the Crammer-Singer method, which learns a matrix of weights from the discriminant vectors to the label vectors, can perform well when codes are short but is susceptible to overfitting as they grow.

4.4 Fold Recognition Results

For the more difficult fold recognition task, where the data set consists of held-out superfamilies from protein folds represented in the training data, we expect that code elements from subclasses (i.e., superfamilies and families) will provide less information, since protein sequences from different superfamilies in principle have no detectable sequence similarity.

Results for the fold recognition problem are provided in Table 3. Note first that the errors for PSI-BLAST, even for the top 5 fold predictions, are very high, underscoring the difficulty of the problem. Sigmoid fitting appears to slightly help reduce one-vs-all error in this case, though balanced error is unaffected. We find that the adaptive codes method can again beat one-vs-all and strongly outperform PSI-BLAST, but we see no trend of improvement as more code elements are added, with various length codes leading to similar error rates. The best classification error rate for adaptive codes is somewhat lower than the best one for Crammer-Singer. Interestingly, in this case, Crammer-Singer with fold-only codes outperforms the best adaptive codes result in terms of balanced loss, though the top 5 results for adaptive codes are uniformly better than Crammer-Singer by either loss function. We conclude that in this case, since the code elements corresponding to subclasses are not very helpful, the adaptive code method cannot leverage longer codes to achieve much higher accuracy. However, the weight learning approach does significantly outperform one-vs-all by all evaluation measures. Significance tests are given comparing the methods in Tables 6 and 9.

Method (and optimization target)	Error	Balanced Error	Top 5 Error	Balanced Top 5 Error
PSI-BLAST	0.648	0.703	0.518	0.543
one-vs-all: Folds	0.463	0.628	0.145	0.235
one-vs-all: Folds,Sfams	0.463	0.628	0.145	0.235
Sigmoid Fitting: Folds	0.451	0.628	0.169	0.287
Adaptive Codes: Folds (zero-one)	0.406	0.558	0.107	0.156
Adaptive Codes: Folds (balanced)	0.371	0.512	0.112	**0.145**
Adaptive Codes: Folds,Sfams (zero-one)	0.409	0.552	0.117	0.172
Adaptive Codes: Folds,Sfams (balanced)	**0.357**	0.508	0.109	0.146
Adaptive Codes: Folds,Sfams,Fams (zero-one)	0.401	0.535	**0.106**	0.173
Adaptive Codes: Folds,Sfams,Fams (balanced)	0.370	**0.499**	0.114	0.155

Table 3: Results for multi-class fold prediction in the fold recognition set-up. Results for the adaptive codes method are reported on a SCOP benchmark data set (26 folds, 303 superfamilies, 614 test sequences) and compared to nearest neighbor using PSI-BLAST, standard one-vs-all, and a perceptron version of the Crammer and Singer method. The adaptive codes method was trained using the mean class update rule.

	PSI-BLAST	one-vs-all: Folds	one-vs-all: Folds,Sfams	Sigmoid Fitting: Folds	C&S: Sf (balanced)	C&S: Sf,f (balanced)	C&S: Sf,f,PSI-f (balanced)	A-Codes: Sf (balanced)	A-Codes: Sf,f (balanced)	A-Codes: Sf,f,PSI-f (balanced)
PSI-BLAST	1	-	-	-	-	-	0.61	-	-	-
one-vs-all: Folds	0	1	1	0	0	0	0	-	-	-
one-vs-all: Folds,Sfams	0	1	1	0	0	0	0	-	-	-
Sigmoid Fitting: Folds	0.06	-	-	1	-	0.25	0	-	-	-
C&S: Sf (balanced)	0	-	-	0	1	0	0	-	-	-
C&S: Sf,f (balanced)	0.25	-	-	-	-	1	0.05	-	-	-
C&S: Sf,f,PSI-f (balanced)	-	-	-	-	-	-	1	-	-	-
A-Codes: Sf (balanced)	0	0.01	0.01	0	0	0	0	1	-	-
A-Codes: Sf,f (balanced)	0	0	0	0	0	0	0	0.13	1	-
A-Codes: Sf,f,PSI-f (balanced)	0	0	0	0	0	0	0	0	0	1

Table 4: P-values from the Wilcoxon signed rank test for superfamily prediction in the remote homology setup. The table shows, at the 0.05 significance level, whether a method in a given row beats a method in a given column (numbers with gray background are significant). Dashes represent when a method in a given row did not beat the method in the given column.

	PSI-BLAST	one-vs-all: Folds	one-vs-all: Folds,Sfams	Sigmoid Fitting: Folds	C&S: F (balanced)	C&S: F,Sf (balanced)	C&S: F,Sf,f (balanced)	A-Codes: F (balanced)	A-Codes: F,Sf (balanced)	A-Codes: F,Sf,f (balanced)
PSI-BLAST	1	-	-	-	-	0	0	-	-	-
one-vs-all: Folds	0	1	1	0.59	0.05	0	0	0.77	-	-
one-vs-all: Folds,Sfams	0	1	1	0.59	0.05	0	0	0.77	-	-
Sigmoid Fitting: Folds	0.01	-	-	1	0.1	0	0	-	-	-
C&S: F (balanced)	0.24	-	-	-	1	0	0	-	-	-
C&S: F,Sf (balanced)	-	-	-	-	-	1	-	-	-	-
C&S: F,Sf,f (balanced)	-	-	-	-	-	0	1	-	-	-
A-Codes: F (balanced)	0.01	-	-	0.89	0.03	0	0	1	-	-
A-Codes: F,Sf (balanced)	0	0.05	0.05	0.04	0	0	0	0	1	-
A-Codes: F,Sf,f (balanced)	0	0	0	0	0	0	0	0	0	1

Table 5: P-values from the Wilcoxon signed rank test for fold prediction in the remote homology setup. The table shows, at the 0.05 significance level, whether a method in a given row beats a method in a given column (numbers with gray background are significant). Dashes represent when a method in a given row did not beat the method in the given column.

	PSI-BLAST	one-vs-all: Folds	one-vs-all: Folds,Sfams	Sigmoid Fitting: Folds	C&S: F (balanced)	C&S: F,Sf (balanced)	C&S: F,Sf,f (balanced)	A-Codes: F (balanced)	A-Codes: F,Sf (balanced)	A-Codes: F,Sf,f (balanced)
PSI-BLAST	1	-	-	-	-	-	-	-	-	-
one-vs-all: Folds	0	1	1	-	-	0.07	0	-	-	-
one-vs-all: Folds,Sfams	0	1	1	-	-	0.07	0	-	-	-
Sigmoid Fitting: Folds	0	0.39	0.39	1	-	0.02	0	-	-	-
C&S: F (balanced)	0	0	0	0	1	0	0	-	-	-
C&S: F,Sf (balanced)	0	-	-	-	-	1	0	-	-	-
C&S: F,Sf,f (balanced)	0	-	-	-	-	-	1	-	-	-
A-Codes: F (balanced)	0	0	0	0	0.56	0	0	1	-	-
A-Codes: F,Sf (balanced)	0	0	0	0	0.15	0	0	0.03	1	0.21
A-Codes: F,Sf,f (balanced)	0	0	0	0	0.48	0	0	0.88	-	1

Table 6: P-values from the Wilcoxon signed rank test for fold recognition. The table shows, at the 0.05 significance level, whether a method in a given row beats a method in a given column (numbers with gray background are significant.) Dashes represent when a method in a given row did not beat the method in the given column.

	PSI-BLAST	one-vs-all: Folds	one-vs-all: Folds,Sfams	Sigmoid Fitting: Folds	C&S: Sf (balanced)	C&S: Sf,f (balanced)	C&S: Sf,f,PSI-f (balanced)	A-Codes: Sf (balanced)	A-Codes: Sf,f (balanced)	A-Codes: Sf,f,PSI-f (balanced)
PSI-BLAST	1	-	-	0.03	-	0.07	0.08	-	-	-
one-vs-all: Folds	0.79	1	1	0	0.4	0.01	0.02	-	-	-
one-vs-all: Folds,Sfams	0.79	1	1	0	0.4	0.01	0.02	-	-	-
Sigmoid Fitting: Folds	-	-	-	1	-	-	-	-	-	-
C&S: Sf (balanced)	0.86	-	-	0	1	0.01	0.07	-	-	-
C&S: Sf,f (balanced)	-	-	-	0.55	-	1	-	-	-	-
C&S: Sf,f,PSI-f (balanced)	-	-	-	0.35	-	0.44	1	-	-	-
A-Codes: Sf (balanced)	0.01	0	0	0	0	0	0	1	-	-
A-Codes: Sf,f (balanced)	0.01	0	0	0	0	0	0	0.09	1	-
A-Codes: Sf,f,PSI-f (balanced)	0	0	0	0	0	0	0	0.02	0.05	1

Table 7: P-values from the Wilcoxon signed rank test for balanced superfamily prediction in the remote homology setup. The table shows, at the 0.05 significance level, whether a method in a given row beats a method in a given column (numbers with gray background are significant). Dashes represent when a method in a given row did not beat the method in the given column.

	PSI-BLAST	one-vs-all: Folds	one-vs-all: Folds,Sfams	Sigmoid Fitting: Folds	C&S: F (balanced)	C&S: F,Sf (balanced)	C&S: F,Sf,f (balanced)	A-Codes: F (balanced)	A-Codes: F,Sf (balanced)	A-Codes: F,Sf,f (balanced)
PSI-BLAST	1	0.95	0.95	0.34	-	0.34	0.73	-	-	-
one-vs-all: Folds	-	1	1	0.39	-	0.34	-	-	-	-
one-vs-all: Folds,Sfams	-	1	1	0.39	-	0.34	-	-	-	-
Sigmoid Fitting: Folds	-	-	-	1	-	-	-	-	-	-
C&S: F (balanced)	0.04	0.05	0.05	0.01	1	0.01	0.04	0.84	0.67	-
C&S: F,Sf (balanced)	-	-	-	1	-	1	-	-	-	-
C&S: F,Sf,f (balanced)	-	0.78	0.78	0.38	-	0.39	1	-	-	-
A-Codes: F (balanced)	0.12	0.01	0.01	0.01	-	0.02	0.13	1	-	-
A-Codes: F,Sf (balanced)	0.02	0	0	0	-	0.01	0.03	0.03	1	-
A-Codes: F,Sf,f (balanced)	0.01	0	0	0	0.43	0.01	0.02	0.01	0.09	1

Table 8: P-values from the Wilcoxon signed rank test for balanced fold prediction in the remote homology setup. The table shows, at the 0.05 significance level, whether a method in a given row beats a method in a given column (numbers with gray background are significant). Dashes represent when a method in a given row did not beat the method in the given column.

	PSI-BLAST	one-vs-all: Folds	one-vs-all: Folds,Sfams	Sigmoid Fitting: Folds	C&S: F (balanced)	C&S: F,Sf (balanced)	C&S: F,Sf,f (balanced)	A-Codes: F (balanced)	A-Codes: F,Sf (balanced)	A-Codes: F,Sf,f (balanced)
PSI-BLAST	1	-	-	-	-	-	-	-	-	-
one-vs-all: Folds	0.16	1	1	0.92	-	-	-	-	-	-
one-vs-all: Folds,Sfams	0.16	1	1	0.92	-	-	-	-	-	-
Sigmoid Fitting: Folds	0.36	-	-	1	-	-	-	-	-	-
C&S: F (balanced)	0	0.01	0.01	0.01	1	0.03	0	0.12	0.19	0.21
C&S: F,Sf (balanced)	0.16	0.58	0.58	0.49	-	1	0.24	-	-	-
C&S: F,Sf,f (balanced)	0.7	0.66	0.66	0.96	-	-	1	-	-	-
A-Codes: F (balanced)	0	0	0	0.01	-	0.21	0.02	1	-	-
A-Codes: F,Sf (balanced)	0	0	0	0.01	-	0.17	0.03	0.81	1	-
A-Codes: F,Sf,f (balanced)	0	0	0	0.01	-	0.12	0.03	0.73	0.52	1

Table 9: P-values from the Wilcoxon signed rank test for balanced fold recognition. The table shows, at the 0.05 significance level, whether a method in a given row beats a method in a given column (numbers with gray background are significant). Dashes represent when a method in a given row did not beat the method in the given column.

4.5 Modified Perceptron Update Rules

Finally, for all multi-class prediction class, we evaluate the effectiveness of our modified perceptron update rules: the friend/foe rule and the mean class update rule. Results are shown in Table 10. Significance tests are given comparing the methods in Table 11.

We find that the new update rules consistently and significantly improve performance for both remote homology prediction tasks when evaluated in terms of classification error, with the most dramatic improvements occurring when training the perceptron using balanced loss in the remote homology fold prediction task. The same performance improvement is true when measured in terms of balanced error for the remote homology fold prediction task; however, for remote homology superfamily prediction, the improvement in balanced error only holds when the perceptron is also trained with balanced error.

In the case of fold recognition, previous results indicate that the subclass code elements are less useful, so we expect that update rules which respect the code structure may be less effective. Indeed, we get mixed results here, with a neutral or slightly weakening effect when the perceptrons are trained using classification loss. However, even for fold recognition, the new update rules significantly improve classification error when the perceptrons are trained using balanced loss.

	Error			Balanced Error		
Method (and optimization target)	single codes	friend/foe	mean friend/foe	single codes	friend/foe	mean friend/foe
Fold recognition						
Folds,Sfams (zero-one)	0.402	0.414	0.409	0.547	0.558	0.552
Folds,Sfams (balanced)	0.412	0.368	0.357	0.535	0.509	0.508
Folds,Sfams,Fams (zero-one)	0.404	0.406	0.401	0.548	0.553	0.535
Folds,Sfams,Fams (balanced)	0.406	0.378	0.370	0.497	0.508	0.499
Remote Homology Superfamily Prediction						
Sfams,Fams (zero-one)	0.251	0.239	0.243	0.372	0.380	0.382
Sfams,Fams (balanced)	0.266	0.241	0.239	0.385	0.347	0.352
Sfams,Fams,PSI-Fams (zero-one)	0.232	0.219	0.223	0.330	0.340	0.338
Sfams,Fams,PSI-Fams (balanced)	0.241	0.213	0.217	0.346	0.313	0.320
Remote Homology Fold Prediction						
Folds,Sfams (zero-one)	0.310	0.283	0.276	0.391	0.372	0.370
Folds,Sfams (balanced)	0.375	0.312	0.297	0.401	0.360	0.351
Folds,Sfams,Fams (zero-one)	0.252	0.247	0.252	0.363	0.347	0.351
Folds,Sfams,Fams (balanced)	0.315	0.278	0.265	0.373	0.349	0.340

Table 10: Results for multi-class prediction comparing different perceptron update rules. Results for the friend/foe and mean friend/foe update rules are compared with the standard perceptron update rule for the fold recognition and remote homology fold and superfamily prediction tasks when using hierarchical codes. Experiments shown with a gray background are those for which the modified update rule gives poorer performance than the standard rule, usually by an insignificant amount. In all other experiments, the modified rules consistently outperform the regular rule, usually by a significant amount (but with one case of a tie).

5. Discussion

We have presented a novel and effective method for multi-class classification that uses the ranking perceptron to learn a reweighting of components of output vectors. Our application domain is the highly multi-class protein structural classification problem, where there are typically hundreds of classes arranged in a hierarchical taxonomy. In this domain, we focus on two difficult subproblems: remote homology detection and fold recognition. We exploit hierarchical information in this problem by training one-vs-the-rest SVM classifiers to recognize classes at different levels of the hierarchy and using these classifiers to define different components of the output vectors. We then use fixed binary codes to represent the hierarchy of labels associated with each class, and we adapt our output vector embedding in order to improve classification relative to these fixed codes.

Unlike the results of a recent empirical study of multi-class classification algorithms that used smaller "toy data sets" (Rifkin and Klautau, 2004), we find that we can significantly outperform one-vs-all in our problem setting. We also convincingly beat PSI-BLAST, which is a widely-used alignment-based method for detecting relationships between protein sequences.

Many authors have presented "output code" methods for multi-class classification. We compare our approach to a perceptron version of the recent Crammer-Singer code-learning approach, which seeks to learn a mapping from the vector of prediction scores for an input example to the vector of

	Superfamily			Fold			Fold (Remote Homology)		
	single Codes	friend/foe	mean friend/foe	single Codes	friend/foe	mean friend/foe	single Codes	friend/foe	mean friend/foe
single Codes	1	-	-	1	0.73	0.92	1	-	-
friend/foe	0.03	1	0.85	-	1	-	0.01	1	-
mean friend/foe	0.08	-	1	-	0.22	1	0	0.12	1

Table 11: P-values from the Wilcoxon signed rank test for different perceptron update rules. The table shows, at the 0.05 significance level, whether a method in a given row beats a method in a given column (numbers with gray background are significant). Dashes represent when a method in a given row did not beat the method in the given column. All measurments are for balanced error.

output classes. We find that when there are a smaller number of classes and when relatively few code elements are used, the Crammer-Singer method can tie or slightly outperform (but not statistically significantly) our adaptive code approach. However, as the number of code elements grows, Crammer-Singer performance deteriorates, often giving much poorer results than one-vs all, while our performance continues to improve. Therefore, as we add more base classifiers, we can almost always beat the best Crammer-Singer result. Moreover, we also present results using modified update rules for the ranking perceptron which take into consideration multi-class predictions that lead to the same loss. These update rules, called the friend/foe and mean friend/foe updates, lead to small but significant performance advantages across multiple experiments.

A large body of recent work has focused on finding good representations for the *inputs* in the protein classification problem, in particular in the form of novel string kernels for protein sequence data. Our current study focuses on the complementary problem of adapting the embedding of the *outputs*. Our experimental results provide a promising indication that new kernel methods combined with novel multi-class "output space" algorithms can truly achieve state-of-the-art performance in a large-scale multi-class protein classification setting.

As we scale to the full-scale protein classification problem, with on the order of 1000 folds, one issue with our approach is limited coverage: for many small SCOP folds, there are not enough labeled sequences to train an SVM fold detector. In ongoing work, we are considering two strategies for increasing coverage. First, there is a standard method for increasing the positive training set size in this problem, namely using PSI-BLAST or another alignment-based method to pull in additional sequences from the non-redundant database that are homologous to the known fold members. Adding domain homologs creates a larger, if biased, training set, and one could investigate the trade-off between coverage and multi-class accuracy as one applies this strategy to very small classes. Second, we are investigating a strategy of "punting" from one prediction method to another based on a prediction score threshold. The goal is to combine a method with weaker performance but full coverage, such as PSI-BLAST, with a higher accuracy method with reduced coverage, such

as SVM adaptive codes, to produce a hybrid method with full coverage that in general outperforms both component methods. Details and results of this approach will be reported elsewhere.

Acknowledgments

We would like to thank Thorsten Joachims for helpful suggestions on the implementation of SVM-Struct and Asa Ben-Hur for helpful comments on the manuscript. This work was supported by NSF grant EIA-0312706 and NIH grant GM74257 and by the NIH Roadmap Initiative, National Centers for Biomedical Computing Grant 1U54CA121852.

References

Erin L. Allwein, Robert E. Schapire, and Yoram Singer. Reducing multiclass to binary: A unifying approach for margin classifiers. In *Proceedings of the 17th International Conference on Machine Learning*, pages 9–16. Morgan Kaufmann, San Francisco, CA, 2000.

Stephen F. Altschul, Warren Gish, Webb Miller, Eugene W. Myers, and David J. Lipman. A basic local alignment search tool. *Journal of Molecular Biology*, 215(3):403–410, 1990.

Stephen F. Altschul, Thomas L. Madden, Alejandro A. Schaffer, Jinghui Zhang, Zheng Zhang, Webb Miller, and David J. Lipman. Gapped BLAST and PSI-BLAST: A new generation of protein database search programs. *Nucleic Acids Research*, 25:3389–3402, 1997.

Zafer Barutcuoglu, Robert E. Schapire, and Olga G. Troyanskaya. Hierarchical multi-label prediction of gene function. *Bioinformatics*, 22(7):830–836, 2006.

Asa Ben-Hur and Douglas Brutlag. Remote homology detection: a motif based approach. *Proceedings of the Eleventh International Conference on Intelligent Systems for Molecular Biology*, 19 suppl 1:i26–i33, 2003.

Léon Bottou, Yann LeCun, and Yoshua Bengio. Global training of document processing systems using graph transformer networks. In *Proc. of Computer Vision and Pattern Recognition*, pages 490–494, Puerto-Rico, 1997. IEEE.

Steven E. Brenner, Patrice Koehl, and Michael Levitt. The ASTRAL compendium for sequence and structure analysis. *Nucleic Acids Research*, 28:254–256, 2000.

Nicolò Cesa-Bianchi, Claudio Gentile, and Luca Zaniboni. Incremental algorithms for hierarchical classification. *Journal of Machine Learning Research*, 7:31–54, 2006.

Michael Collins. Discriminative reranking for natural language parsing. In *Proceedings of the 17th International Conference on Machine Learning*, pages 175 – 182. Morgan Kaufmann, San Francisco, CA, 2000.

Michael Collins and Nigel Duffy. New ranking algorithms for parsing and tagging: Kernels over discrete structures, and the voted perceptron. *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 263–270, 2002.

Koby Crammer and Yoram Singer. On the learnability and design of output codes for multiclass problems. In *Computational Learning Theory*, pages 35–46, 2000.

Koby Crammer and Yoram Singer. On the algorithmic implementation of multiclass kernel-based vector machines. *Journal Machine Learning Research*, 2:265–292, 2002. ISSN 1533-7928.

Ofer Dekel, Joseph Keshet, and Yoram Singer. Large margin hierarchical classification. In *Proceedings of the 21st International Conference on Machine Learning*, 2004.

Thomas G. Dietterich and Ghulum Bakiri. Solving multiclass learning problems via error-correcting output codes. *Journal of Artificial Intelligence Research*, 2:263–286, 1995.

Yoav Freund and Robert E. Schapire. Large margin classification using the perceptron algorithm. *Machine Learning*, 37(3):277 – 296, 1999.

Mark Girolami and Simon Rogers. Variational Bayesian multinomial probit regression with Gaussian process priors. *Neural Computation*, 18(8):1790–1817, 2006.

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

Eugene Ie, Jason Weston, William Stafford Noble, and Christina Leslie. Multi-class protein fold recognition using adaptive codes. *Proceedings of the 22nd International Conference on Machine Learning*, 2005.

Tommi Jaakkola, Mark Diekhans, and David Haussler. A discriminative framework for detecting remote protein homologies. *Journal of Computational Biology*, 7(1–2):95–114, 2000.

Anders Krogh, Michael Brown, I. Saira Mian, Kimmen Sjölander, and David Haussler. Hidden Markov models in computational biology: Applications to protein modeling. *Journal of Molecular Biology*, 235:1501–1531, 1994.

Rui Kuang, Eugene Ie, Ke Wang, Kai Wang, Mahira Siddiqi, Yoav Freund, and Christina Leslie. Profile kernels for detecting remote protein homologs and discriminative motifs. *Journal of Bioinformatics and Computational Biology*, 2005. To appear.

Yann LeCun and Fu Jie Huang. Loss functions for discriminative training of energy-based models. In *Proceedings of the 10th International Workshop on Artificial Intelligence and Statistics*, 2005.

Christina Leslie, Eleazar Eskin, and William S. Noble. The spectrum kernel: A string kernel for SVM protein classification. *Proceedings of the Pacific Biocomputing Symposium*, pages 564–575, 2002a.

Christina Leslie, Eleazar Eskin, Jason Weston, and William S. Noble. Mismatch string kernels for SVM protein classification. *Advances in Neural Information Processing Systems 15*, pages 1441–1448, 2002b.

Christina Leslie, Eleazar Eskin, Adiel Cohen, Jason Weston, and William S. Noble. Mismatch string kernels for discriminative protein classification. *Bioinformatics*, 20(4):467–476, 2004.

Li Liao and William S. Noble. Combining pairwise sequence similarity and support vector machines for remote protein homology detection. *Proceedings of the 6th Annual International Conference on Research in Computational Molecular Biology*, pages 225–232, 2002.

Hsuan-Tien Lin, Chih-Jen Lin, and Ruby C. Weng. A note on Platt's probabilistic outputs for support vector machines. Technical report, Department of Computer Science and Information Engineering, National Taiwan University, 2003.

Alexey G. Murzin, Steven E. Brenner, Tim Hubbard, and Cyrus Chothia. SCOP: A structural classification of proteins database for the investigation of sequences and structures. *Journal of Molecular Biology*, 247(4):536–540, 1995.

Jong Park, Kevin Karplus, Christian Barrett, Richard Hughey, David Haussler, Tim Hubbard, and Cyrus Chothia. Sequence comparisons using multiple sequences detect twice as many remote homologues as pairwise methods. *Journal of Molecular Biology*, 284(4):1201–1210, 1998.

John Platt. Probabilities for support vector machines. *Advances in Large Margin Classifiers*, pages 61–74, 1999.

Huzefa Rangwala and George Karypis. Profile-based direct kernels for remote homology detection and fold recognition. *Bioinformatics*, 21(23):4239–4247, 2005.

Gunnar Rätsch, Alexander J. Smola, and Sebastian Mika. Adapting codes and embeddings for polychotomies. *Advances in Neural Information Processing Systems*, 15:513–520, 2002.

Ryan Rifkin and Aldebaro Klautau. In defense of one-vs-all classification. *Journal Machine Learning Research*, 5:101–141, 2004. ISSN 1533-7928.

Frank Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65:386–407, 1958.

Hiroto Saigo, Jean-Philippe Vert, Nobuhisa Ueda, and Tatsuya Akutsu. Protein homology detection using string alignment kernels. *Bioinformatics*, 20(11):1682–1689, 2004.

Temple Smith and Michael Waterman. Identification of common molecular subsequences. *Journal of Molecular Biology*, 147(1):195–197, 1981.

Ioannis Tsochantaridis, Thomas Hofmann, Thorsten Joachims, and Yasemin Altun. Support vector learning for interdependent and structured output spaces. *Proceedings of the 21st International Conference on Machine Learning*, pages 823–830, 2004.

Vladimir N. Vapnik. *Statistical Learning Theory*. John Wiley and Sons, New York, 1998.

Jason Weston and Chris Watkins. Support vector machines for multiclass pattern recognition. In *Proceedings of the 7th European Symposium On Artificial Neural Networks*, 1999.

Jason Weston, Christina Leslie, Eugene Ie, Dengyong Zhou, Andre Elisseeff, and William S. Noble. Semi-supervised protein classification using cluster kernels. *Bioinformatics*, 21(15):3241–3247, 2005.

Journal of Machine Learning Research 8 (2007) 1583-1623 Submitted 4/06; Revised 1/07; Published 7/07

Spherical-Homoscedastic Distributions: The Equivalency of Spherical and Normal Distributions in Classification

Onur C. Hamsici HAMSICIO@ECE.OSU.EDU
Aleix M. Martinez ALEIX@ECE.OSU.EDU
Department of Electrical and Computer Engineering
The Ohio State University
Columbus, OH 43210, USA

Editor: Greg Ridgeway

Abstract

Many feature representations, as in genomics, describe directional data where all feature vectors share a common norm. In other cases, as in computer vision, a norm or variance normalization step, where all feature vectors are normalized to a common length, is generally used. These representations and pre-processing step map the original data from $\mathbb{R}^p$ to the surface of a hypersphere S^{p-1}. Such representations should then be modeled using spherical distributions. However, the difficulty associated with such spherical representations has prompted researchers to model their spherical data using Gaussian distributions instead—as if the data were represented in $\mathbb{R}^p$ rather than S^{p-1}. This opens the question to whether the classification results calculated with the Gaussian approximation are the same as those obtained when using the original spherical distributions. In this paper, we show that in some particular cases (which we named spherical-homoscedastic) the answer to this question is positive. In the more general case however, the answer is negative. For this reason, we further investigate the additional error added by the Gaussian modeling. We conclude that the more the data deviates from spherical-homoscedastic, the less advisable it is to employ the Gaussian approximation. We then show how our derivations can be used to define optimal classifiers for spherical-homoscedastic distributions. By using a kernel which maps the original space into one where the data adapts to the spherical-homoscedastic model, we can derive non-linear classifiers with potential applications in a large number of problems. We conclude this paper by demonstrating the uses of spherical-homoscedasticity in the classification of images of objects, gene expression sequences, and text data.

Keywords: directional data, spherical distributions, normal distributions, norm normalization, linear and non-linear classifiers, computer vision

1. Introduction

Many problems in science and engineering involve spherical representations or directional data, where the sample vectors lie on the surface of a hypersphere. This is typical, for example, of some genome sequence representations (Janssen et al., 2001; Audit and Ouzounis, 2003), in text analysis and clustering (Dhillon and Modha, 2001; Banerjee et al., 2005), and in morphometrics (Slice, 2005). Moreover, the use of some kernels (e.g., radial basis function) in machine learning algorithms, will reshape all sample feature vectors to have a common norm. That is, the original data is mapped into the surface of a hypersphere. Another area where spherical representations are common is in computer vision, where spherical representations emerge after the common norm-

normalization step is incorporated. This pre-processing step guarantees that all vectors have a common norm and it is used in systems where the representation is based on the shading properties of the object to make the algorithm invariant to changes of the illumination intensity, and when the representation is shape-based to provide scale and rotation invariance. Typical examples are in object and face recognition (Murase and Nayar, 1995; Belhumeur and Kriegman, 1998), pose estimation (Javed et al., 2004), shape analysis (Dryden and Mardia, 1998) and gait recognition (Wang et al., 2003; Veeraraghavan et al., 2005).

Figure 1 provides two simple computer vision examples. On the left hand side of the figure the two p-dimensional feature vectors $\hat{\mathbf{x}}_i$, $i = \{1,2\}$, correspond to the same face illuminated from the same angle but with different intensities. Here, $\hat{\mathbf{x}}_1 = \alpha\hat{\mathbf{x}}_2$, $\alpha \in \mathbb{R}$, but normalizing these vectors to a common norm results in the same representation $\mathbf{x}$; that is, the resulting representation is invariant to the intensity of the light source. On the right hand side of Figure 1, we show a classical application to shape analysis. In this case, each of the p elements in $\hat{\mathbf{y}}_i$ represents the Euclidean distances from the centroid of the 2D shape to a set of p equally separated points on the shape. Normalizing each vector with respect to its norm, guarantees our representation is scale invariant.

The most common normalization imposes that all vectors have a unit norm, that is,

$$\mathbf{x} = \frac{\hat{\mathbf{x}}}{\|\hat{\mathbf{x}}\|},$$

where $\hat{\mathbf{x}} \in \mathbb{R}^p$ is the original feature vector, and $\|\mathbf{x}\|$ is the magnitude (2-norm length) of the vector $\mathbf{x}$. When the feature vectors have zero mean, it is common to normalize these with respect to their variances instead,

$$\mathbf{x} = \frac{\hat{\mathbf{x}}}{\sqrt{\frac{1}{p-1}\sum_{i=1}^{p}\hat{\mathbf{x}}^2}} = \sqrt{p-1}\frac{\hat{\mathbf{x}}}{\|\hat{\mathbf{x}}\|},$$

which generates vectors with norms equal to $\sqrt{p-1}$. This second option is usually referred to as variance normalization.

It is important to note that these normalizations enforce all feature vectors $\mathbf{x}$ to be at a common distance from the origin; that is, the original feature space is mapped to a spherical representation (see Figure 1). This means that the data now lays on the surface of the $(p-1)$-dimensional unit sphere S^{p-1}. [1]

Our description above implies that the data would now need to be interpreted as spherical. For example, while the illumination subspace of a (Lambertian) convex object illuminated by a single point source at infinity is known to be 3-dimensional (Belhumeur and Kriegman, 1998), this corresponds to the 2-dimensional sphere S^2 after normalization. The third dimension (not shown in the spherical representation) corresponds to the intensity of the source. Similarly, if we use norm-normalized images to define the illumination cone, the extreme rays that define the cone will be the extreme points on the corresponding hypersphere.

An important point here is that data would now need to be modeled using spherical distributions. However, the computation of the parameters that define spherical models is usually complex, very costly and, in many cases, impossible to obtain (see Section 2 for a review). This leaves us with an unsolved problem: To make a system invariant to some parameters, we want to use spherical

1. Since all spherical representations are invariant to the radius (i.e., there is an isomorphism connecting any two representations of distinct radius), selecting a specific value for the radius is not going to effect the end result. In this paper, we always impose this radius to be equal to one.

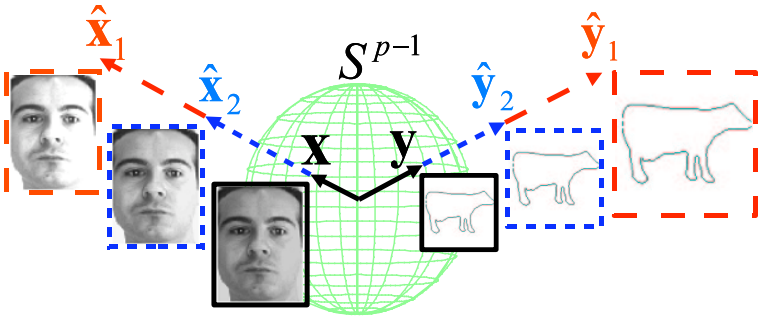


Figure 1: On the left hand side of this figure, we show two feature vectors corresponding to the same face illuminated from the same position but with different intensities. This means that $\hat{\mathbf{x}}_1 = \alpha\hat{\mathbf{x}}_2$, $\alpha \in \mathbb{R}$. Normalizing these two vectors with respect to their norm yields a common solution, $\mathbf{x} = \frac{\hat{\mathbf{x}}_1}{\|\hat{\mathbf{x}}_1\|} = \frac{\hat{\mathbf{x}}_2}{\|\hat{\mathbf{x}}_2\|}$. The norm-normalized vector $\mathbf{x}$ is on S^{p-1}, whereas $\hat{\mathbf{x}}_i \in \mathbb{R}^p$. On the right hand side of this figure we show a shape example where the elements of the feature vectors $\hat{\mathbf{y}}_i$ represent the Euclidean distance between the centroid of the 2D shape and p points on the shape contour. As above, $\hat{\mathbf{y}}_1 = \beta\hat{\mathbf{y}}_2$ (where $\beta \in \mathbb{R}$), and normalizing them with respect to their norm yields $\mathbf{y}$.

representations (as in genomics) or normalize the original feature vectors to such a representation (as in computer vision). But, in such cases, the parameter estimation of our distribution is impossible or very difficult. This means, we are left to approximate our spherical distribution with a model that is well-understood and easy to work with. Typically, the most convenient choice is the Gaussian (Normal) distribution.

The question arises: *how accurate are the classification results obtained when approximating spherical distributions with Gaussian distributions?*

Note that if the Bayes decision boundary obtained with Gaussians is very distinct to that found by the spherical distributions, our results will not generally be useful in practice. This would be catastrophic, because it would mean that by using spherical representations to solve one problem, we have created another problem that is even worse.

In this paper, we show that in almost all cases where the Bayes classifier is linear (which is the case when the data is what we will refer to as *spherical-homoscedastic*—a rotation-invariant extension of homoscedasticity) the classification results obtained on the true underlying spherical distributions and on those Gaussians that best approximate them are identical. We then show that for the general case (which we refer to as *spherical-heteroscedastic*) these classification results can vary substantially. In general, the more the data deviates from our spherical-homoscedastic definition, the more the classification results diverge from each other. This provides a mechanism to test when it makes sense to use the Gaussian approximation and when it does not.

Our definition of spherical-homoscedasticity will also allow us to define simple classification algorithms that provide the minimal Bayes classification error for two spherical homoscedastic distributions. This result can then be extended to the more general spherical-heteroscedastic case by incorporating the idea of the kernel trick. Here, we will employ a kernel to (intrinsically) map the data to a space where the spherical-homoscedastic model provides a good fit.

The rest of this paper is organized as follows. Section 2 presents several of the commonly used spherical distributions and describes some of the difficulties associated to their parameter estimation. In Section 3, we introduce the concept of spherical-homoscedasticity and show that whenever two spherical distributions comply with this model, the Gaussian approximation works well. Section 4 illustrates the problems we will encounter when the data deviates from our spherical-homoscedastic model. In particular, we study the classification error added when we model spherical-heteroscedastic distributions with the Gaussian model. Section 5 presents the linear and (kernel) non-linear classifiers for spherical-homoscedastic and -heteroscedastic distributions, respectively. Our experimental results are in Section 6. Conclusions are in Section 7. A summary of our notation is in Appendix A.

2. Spherical Data

In this section, we introduce some of the most commonly used spherical distributions and discuss their parameter estimation and associated problems. We follow with a description of the corresponding distance measurements derived from the Bayes classification rule.

2.1 Spherical Distributions

Spherical data can be modeled using a large variety of data distributions (Mardia and Jupp, 2000), most of which are analogous to distributions defined for the Cartesian representation. For example, the von Mises-Fisher (vMF) distribution is the spherical counterpart of those Gaussian distributions that can be represented with a covariance matrix of the form $\tau^2\mathbf{I}$; where $\mathbf{I}$ is the $p \times p$ identity matrix and $\tau > 0$. More formally, the probability density function (pdf) of the p-dimensional vMF model $M(\mu,\kappa)$ is defined as

$$f(\mathbf{x}|\mu,\kappa) = c_{MF}(p,\kappa)\exp\{\kappa\mu^T\mathbf{x}\}, \tag{1}$$

where $c_{MF}(p,\kappa)$ is a normalizing constant which guarantees that the integral of our density over the $(p-1)$-dimensional sphere S^{p-1} is one, $\kappa \geq 0$ is the concentration parameter, and μ is the mean direction vector (i.e., $\|\mu\| = 1$). Here, the concentration parameter κ is used to represent distinct types of data distributions—from uniformly distributed (for which κ is small) to very localized (for which κ is large). This means that when $\kappa = 0$ the data will be uniformly distributed over the sphere, and when $\kappa \to \infty$ the distribution will approach a point.

As mentioned above, Equation (1) can only be used to model circularly symmetric distributions around the mean direction. When the data does not conform to such a distribution type, one needs to use more flexible pdfs such as the Bingham distribution (Bingham, 1974). The pdf for the p-dimensional Bingham $B(\mathbf{A})$ is an antipodally symmetric function (i.e., $f(-\mathbf{x}) = f(\mathbf{x})$, $\forall\mathbf{x} \in S^{p-1}$) given by

$$f(\pm\mathbf{x}|\mathbf{A}) = c_B(p,\mathbf{A})\exp\{\mathbf{x}^T\mathbf{A}\mathbf{x}\}, \tag{2}$$

where $c_B(p,\mathbf{A})$ is the normalizing constant and $\mathbf{A}$ is a $p \times p$ symmetric matrix defining the parameters of the distribution. Note that since the feature vectors have been mapped onto the unit sphere, $\mathbf{x}^T\mathbf{x} = 1$. This means that substituting $\mathbf{A}$ for $\mathbf{A} + c\mathbf{I}$ with any $c \in \mathbb{R}$ would result in the same pdf as that shown in (2). To eliminate this redundancy, we need to favor a solution with an additional constraint. One such constraint is $\lambda_{MAX}(\mathbf{A}) = 0$, where $\lambda_{MAX}(\mathbf{A})$ is the largest eigenvalue of $\mathbf{A}$.

For many applications the assumption of antipodally symmetric is inconvenient. In such cases, we can use the Fisher-Bingham distribution (Mardia and Jupp, 2000) which combines the idea

of von Mises-Fisher with that of Bingham, yielding the following p-dimensional Fisher-Bingham $FB(\mu,\kappa,\mathbf{A})$ pdf

$$f(\mathbf{x}|\mu,\kappa,\mathbf{A}) = c_{FB}(\kappa,\mathbf{A})\exp\{\kappa\mu^T\mathbf{x}+\mathbf{x}^T\mathbf{A}\mathbf{x}\},$$

where $c_{FB}(\kappa,\mathbf{A})$ is the normalizing constant and $\mathbf{A}$ is a symmetric $p\times p$ matrix, with the constraint $tr(\mathbf{A})=0$. Note that the combination of the two components in the exponential function shown above, provides enough flexibility to represent a large variety of ellipsoidal distributions (same as Bingham) but without the antipodally symmetric constraint (same as in von Mises-Fisher).

2.2 Parameter Estimation

To use each of these distributions, we need to first estimate their parameters from a training data-set, $\mathbf{X}=(\mathbf{x}_1,\mathbf{x}_2,\ldots,\mathbf{x}_n)$; $\mathbf{X}$ a $p\times n$ matrix, with $\mathbf{x}_i\in S^{p-1}$ the sample vectors.

If one assumes that the samples in $\mathbf{X}$ arise from a von Mises-Fisher distribution, we will need to estimate the concentration parameter κ and the (unit) mean direction μ. The most common way to estimate these parameters is to use the maximum likelihood estimates (m.l.e.). The sample mean direction $\hat{\mu}$ is given by

$$\hat{\mu} = \frac{\bar{\mathbf{x}}}{\|\bar{\mathbf{x}}\|}, \tag{3}$$

where $\bar{\mathbf{x}}=\frac{1}{n}\sum_{i=1}^{n}\mathbf{x}_i$ is the average feature vector. It can be shown that

$$c_{MF} = \left(\frac{\kappa}{2}\right)^{p/2-1}\frac{1}{(2\pi)^{p/2}I_{p/2-1}(\kappa)}$$

in (1), where $I_v(.)$ denotes the modified Bessel function of the first kind and of order v (Banerjee et al., 2005).[2] By substituting this normalization constant in (1) and calculating the expected value of $\mathbf{x}$ (by integrating the pdf over the surface of S^{p-1}), we obtain $\|\bar{\mathbf{x}}\| = \frac{I_{p/2}(\hat{\kappa})}{I_{p/2-1}(\hat{\kappa})}$. Unfortunately, equations defining a ratio of Bessel functions cannot be inverted and, hence, approximation methods need to be defined for $\hat{\kappa}$. Banerjee et al. (2005) have recently proposed one such approximation which can be applied regardless of the dimensionality of the data,

$$\hat{\kappa} = \frac{\|\bar{\mathbf{x}}\|p-\|\bar{\mathbf{x}}\|^3}{1-\|\bar{\mathbf{x}}\|^2}.$$

This approximation makes the parameter $\hat{\kappa}$ directly dependent on the training data and, hence, can be easily computed.

For the Bingham distribution, the normalizing constant, $c_B^{-1}=\int_{S^{p-1}}\exp\left\{\mathbf{x}^T\mathbf{A}\mathbf{x}\right\}d\mathbf{x}$, requires that we estimate the parameters defined in $\mathbf{A}$. Since $\mathbf{A}$ is a symmetric matrix, its spectral decomposition can be written as $\mathbf{A}=\mathbf{Q}\Lambda\mathbf{Q}^T$, where $\mathbf{Q}=(\mathbf{q}_1,\mathbf{q}_2,\ldots,\mathbf{q}_p)$ is a matrix whose columns $\mathbf{q}_i$ correspond to the eigenvectors of $\mathbf{A}$ and $\Lambda=diag(\lambda_1,\ldots,\lambda_p)$ is the $p\times p$ diagonal matrix of corresponding eigenvalues. This allows us to calculate the log-likelihood of the data by adding the log version of (2) over all samples in $\mathbf{X}$, $\mathcal{L}(\mathbf{Q},\Lambda)=ntr(\mathbf{S}\mathbf{Q}\Lambda\mathbf{Q}^T)+n\ln(c_B(p,\Lambda))$; where $\mathbf{S}=n^{-1}\mathbf{X}\mathbf{X}^T$ is the sample autocorrelation matrix (sometimes also referred to as scatter matrix). Since the $tr(\mathbf{S}\mathbf{A})$ is maximized when the eigenvectors of $\mathbf{S}$ and $\mathbf{A}$ are the same, the m.l.e. of $\mathbf{Q}$ (denoted $\hat{\mathbf{Q}}$) is given

2. The modified Bessel function of the first kind is proportional to the contour integral of the exponential function defined in (1) over the $(p-1)$-dimensional sphere S^{p-1}.

by the eigenvector decomposition of the autocorrelation matrix, $\mathbf{S} = \hat{\mathbf{Q}}\hat{\Lambda}_S\hat{\mathbf{Q}}$; where $\hat{\Lambda}_S$ is the eigenvalue matrix of $\mathbf{S}$. Unfortunately, the same does not apply to the estimation of the eigenvalues Λ, because these depend on $\mathbf{S}$ and c_B. Note that in order to calculate the normalizing constant c_B we need to know Λ, but to compute Λ we need to know c_B. This chicken-and-egg problem needs to be solved using iterative approaches or optimization algorithms. To define such iterative approaches, we need to calculate the derivative of c_B. Since there are no known ways to express Λ as a function of the derivative of $c_B(p,\Lambda)$, approximations for c_B (which permit such a dependency) are necessary. Kume and Wood (2005) have recently proposed a saddlepoint approximation that can be used for this purpose. In their approach, the estimation of the eigenvalues is given by the following optimization problem

$$\arg\max_{\Lambda} ntr(\hat{\Lambda}_S\Lambda) - n\ln(\hat{c}_B(\Lambda)),$$

where $\hat{c}_B(\Lambda)$ is now the estimated normalizing constant given by the saddlepoint approximation of the density function of the 2-norm of $\mathbf{x}$.

The estimation of the parameters of the Fisher-Bingham distribution comes at an additional cost given by the large number of parameters that need to be estimated. For example, the normalizing constant c_{FB} depends on κ, μ and $\mathbf{A}$, making the problem even more complex than that of the Bingham distribution. Hence, approximation methods are once more required. One such approximation is given by Kent (1982), where it is assumed that the data is highly concentrated (i.e., κ is large) or that the data is distributed more or less equally about every dimension (i.e., the distribution is almost circularly symmetric). In this case, the mean direction μ is estimated using (3) and the estimate of the parameter matrix (for the 3-dimensional case) is given by $\mathbf{A} = \beta(\mathbf{q}_1\mathbf{q}_1^T - \mathbf{q}_2\mathbf{q}_2^T)$; where $\mathbf{q}_1$ and $\mathbf{q}_2$ are the two eigenvectors associated to the two largest eigenvalues ($\lambda_1 \geq \lambda_2$) of $\bar{\mathbf{S}}$, $\bar{\mathbf{S}}$ is the scatter matrix calculated on the null space of the mean direction, and β is the parameter that allows us to deviate from the circle defined by κ.[3] Note that the two eigenvectors $\mathbf{q}_1$ and $\mathbf{q}_2$ are constrained to be orthogonal to the unit vector $\hat{\mu}$ describing the mean direction. To estimate the value for κ and β, Kent proposes to further assume the data can be locally represented as Gaussian distributions on S^{p-1} and shows that in the three dimensional case $\hat{\beta} \cong \frac{1}{2}\left((2-2\|\bar{\mathbf{x}}\|-r_2)^{-1} + (2-2\|\bar{\mathbf{x}}\|+r_2)^{-1}\right)$ and $\hat{\kappa} \cong (2-2\|\bar{\mathbf{x}}\|-r_2)^{-1} + (2-2\|\bar{\mathbf{x}}\|+r_2)^{-1}$, where $r_2 = \lambda_1 - \lambda_2$.

The Kent distribution is one of the most popular distribution models for the estimation of 3-dimensional spherical data, since it has fewer parameters to be estimated than Fisher-Bingham and can model any ellipsoidally symmetric distribution. A more recent and general approximation for Fisher-Bingham distributions is given in Kume and Wood (2005), but this requires the estimate of a larger number of parameters, a cost to be considered.

As summarized above, the actual values of the parameters of spherical distributions can rarely be computed, and approximations are needed. Furthermore, we have seen that most of these approximation algorithms require assumptions that may not be applicable to our problem. In the case of the Kent distribution, these assumptions are quite restrictive. When the assumptions do not hold, we cannot make use of these spherical pdfs.

3. Recall that the first part of the definition of the Fisher-Bingham pdf is the same as that of the von Mises-Fisher, $\kappa\mu^T\mathbf{x}$, which defines a small circle on S^{p-1}, while the second component $\mathbf{x}^T\mathbf{A}\mathbf{x}$ allows us to deviate from the circle and represent a large variety of ellipsoidal pdfs.

2.3 Distance Calculation

The probability ("distance") of a new (test) vector $\mathbf{x}$ to belong to a given distribution can be defined as (inversely) proportional to the likelihood or log-likelihood of $\mathbf{x}$. For instance, the pdf of the p-dimensional Gaussian $N(\mathbf{m},\Sigma)$ is $f(\mathbf{x}|\mathbf{m},\Sigma) = c_N(\Sigma)\exp\{-\frac{1}{2}(\mathbf{x}-\mathbf{m})^T\Sigma^{-1}(\mathbf{x}-\mathbf{m})\}$, where $\mathbf{m}$ is the mean, Σ is the sample covariance matrix of the data and $c_N^{-1}(\Sigma) = (2\pi)^{p/2}|\Sigma|^{1/2}$ is the normalizing constant. When the priors of each class are the same, the optimum "distance" measurement (in the Bayes sense) of a point $\mathbf{x}$ to $f(\mathbf{x}|\mathbf{m},\Sigma)$ as derived by the Bayes rule is the negative of the log-likelihood

$$d_N^2(\mathbf{x}) = -\ln f(\mathbf{x}|\mathbf{m},\Sigma) = \frac{1}{2}(\mathbf{x}-\mathbf{m})^T\Sigma^{-1}(\mathbf{x}-\mathbf{m}) - \ln(c_N(\Sigma)). \quad (4)$$

Similarly, we can define the distance of a test sample to each of the spherical distributions defined above (i.e., von Mises-Fisher, Bingham and Fisher-Bingham) as

$$\begin{aligned} d_{MF}^2(\mathbf{x}) &= -\kappa\mu^T\mathbf{x} - \ln(c_{MF}(p,\kappa)), & (5)\\ d_B^2(\mathbf{x}) &= -\mathbf{x}^T\mathbf{A}\mathbf{x} - \ln(c_B(p,\mathbf{A})), & (6)\\ d_{FB}^2(\mathbf{x}) &= -\kappa\mu^T\mathbf{x} - \mathbf{x}^T\mathbf{A}\mathbf{x} - \ln(c_{FB}(\kappa,\mathbf{A})). & (7) \end{aligned}$$

As seen in Section 2.2, the difficulty with the distance measures defined in (5-7) will be given by the estimation of the parameters of our distribution (e.g., μ, κ, and $\mathbf{A}$), because this is usually complex and sometimes impossible. This is the reason why most researchers prefer to use the Gaussian model and its corresponding distance measure defined in (4) instead. The question remains: *are the classification errors obtained using the spherical distributions defined above lower than those obtained when using the Gaussian approximation? And, if so, when?*

The rest of this paper addresses this general question. In particular, we show that when the data distributions conform to a specific relation (which we call spherical-homoscedastic), the classification errors will be the same. However, in the most general case, they need not be.

3. Spherical-Homoscedastic Distributions

If the distributions of each class are known, the optimal classifier is given by the Bayes Theorem. Furthermore, when the class priors are equal, this decision rule simplifies to the comparison of the likelihoods (maximum likelihood classification), $p(\mathbf{x}|w_i)$; where w_i specifies the i^{th} class. In the rest of this paper, we will make the assumption of equal priors (i.e., $P(w_i) = P(w_j)\ \forall i,j$). An alternative way to calculate the likelihood of an observation $\mathbf{x}$ to belong to a class, is to measure the log-likelihood-based distance (e.g., d_N^2 in the case of a Gaussian distribution). In the spherical case, the distances defined in Section 2.3 can be used.

In the Gaussian case, we say that a set of r Gaussians, $\{N_1(\mathbf{m}_1,\Sigma_1),\ldots,N_r(\mathbf{m}_r,\Sigma_r)\}$, are *homoscedastic* if their covariance matrices are all the same (i.e., $\Sigma_1 = \cdots = \Sigma_r$). Homoscedastic Gaussian distributions are relevant, because their Bayes decision boundaries are given by hyperplanes.

However, when all feature vectors are restricted to lay on the surfaces of a hypersphere, the definition of homoscedasticity given above becomes too restrictive. For example, if we use Gaussian pdfs to model some spherical data that is ellipsoidally symmetric about its mean, then only those distributions that have the same mean up to a sign can be homoscedastic. This is illustrated in Figure 2. Although the three classes shown in this figure have the same covariance matrix up to a rotation, only the ones that have the same mean up to a sign (i.e., Class 1 and 3) have the exact

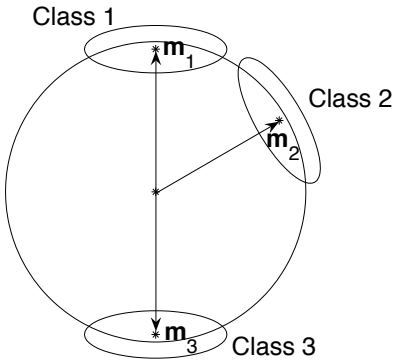


Figure 2: Assume we model the data of three classes laying on S^{p-1} using three Gaussian distributions. In this case, each set of Gaussian distributions can only be homoscedastic if the mean feature vector of one class is the same as that of the others up to a sign. In this figure, Class 1 and 3 are homoscedastic, but classes 1,2 and 3 are not. Classes 1, 2 and 3 are however spherical-homoscedastic.

same covariance matrix. Hence, only classes 1 and 3 are said to be homoscedastic. Nonetheless, the decision boundaries for each pair of classes in Figure 2 are all hyperplanes. Furthermore, we will show that these hyperplanes (given by approximating the original distributions with Gaussians) are generally the same as those obtained using the Bayes Theorem on the true underlying distributions. Therefore, it is important to define a new and more general type of homoscedasticity that is rotational invariant.

Definition 1 *Two distributions (f_1 and f_2) are said to be spherical-homoscedastic if the Bayes decision boundary between f_1 and f_2 is given by one or more hyperplanes and the variances in f_1 are the same as those in f_2.*

Recall that the variance of the vMF distribution is defined using a single parameter, κ. This means that in the vMF case, we will only need to impose that all concentration parameters be the same. For the other cases, these will be defined by κ and the eigenvalues of $\mathbf{A}$. This is what is meant by "the variances" in our definition above.

Further, in this paper, we will work on the case where the two distributions (f_1 and f_2) are of the same form, that is, two Gaussian, vMF, Bingham or Kent distributions.

Our main goal in the rest of this section, is to demonstrate that the linear decision boundaries (given by the Bayes Theorem) of a pair of spherical-homoscedastic von Mises-Fisher, Bingham or Kent, are the same as those obtained when these are assumed to be Gaussian. We start with the study of the Gaussian distribution.

Theorem 2 *Let two Gaussian distributions $N_1(\mathbf{m},\Sigma)$ and $N_2(\mathbf{R}^T\mathbf{m},\mathbf{R}^T\Sigma\mathbf{R})$ model the spherical data of two classes on S^{p-1}; where $\mathbf{m}$ is the mean, Σ is the covariance matrix (which is assumed to be full ranked), and $\mathbf{R}\in SO(p)$ is a rotation matrix. Let $\mathbf{R}$ be spanned by two of the eigenvectors of Σ, $\mathbf{v}_1$ and $\mathbf{v}_2$, and let one of these eigenvectors define the same direction as $\mathbf{m}$ (i.e., $\mathbf{v}_i=\mathbf{m}/\|\mathbf{m}\|$ for i equal to 1 or 2). Then, $N_1(\mathbf{m},\Sigma)$ and $N_2(\mathbf{R}^T\mathbf{m},\mathbf{R}^T\Sigma\mathbf{R})$ are spherical-homoscedastic.*

Proof We want to prove that the Bayes decision boundaries are hyperplanes. This boundary is given when the ratio of the log-likelihoods of $N_1(\mathbf{m},\Sigma)$ and $N_2(\mathbf{R}^T\mathbf{m},\mathbf{R}^T\Sigma\mathbf{R})$ equals one. Formally,

$$\ln(c_N(\Sigma)) - \frac{1}{2}(\mathbf{x}-\mathbf{m})^T\Sigma^{-1}(\mathbf{x}-\mathbf{m}) =$$
$$\ln(c_N(\mathbf{R}^T\Sigma\mathbf{R})) - \frac{1}{2}\left(\mathbf{x}-\mathbf{R}^T\mathbf{m}\right)^T(\mathbf{R}^T\Sigma\mathbf{R})^{-1}\left(\mathbf{x}-\mathbf{R}^T\mathbf{m}\right).$$

Since for any function f we know that $f(|\mathbf{R}^T\Sigma\mathbf{R}|) = f(|\Sigma|)$, the constant parameter $c_N(|\mathbf{R}^T\Sigma\mathbf{R}|) = c_N(|\Sigma|)$; where $|\mathbf{M}|$ is the determinant of $\mathbf{M}$. Furthermore, since the normalizing constant c_N only depends on the determinant of the covariance matrix, we know that $c_N(\mathbf{R}^T\Sigma\mathbf{R}) = c_N(\Sigma)$. This allows us to simplify our previous equation to

$$\begin{aligned}(\mathbf{x}-\mathbf{m})^T\Sigma^{-1}(\mathbf{x}-\mathbf{m}) &= \left(\mathbf{x}-\mathbf{R}^T\mathbf{m}\right)^T\mathbf{R}^T\Sigma^{-1}\mathbf{R}\left(\mathbf{x}-\mathbf{R}^T\mathbf{m}\right)\\ &= (\mathbf{R}\mathbf{x}-\mathbf{m})^T\Sigma^{-1}(\mathbf{R}\mathbf{x}-\mathbf{m}).\end{aligned}$$

Writing this equation in an open form,

$$\begin{aligned}\mathbf{x}^T\Sigma^{-1}\mathbf{x} - 2\mathbf{x}^T\Sigma^{-1}\mathbf{m} + \mathbf{m}^T\Sigma^{-1}\mathbf{m} &= (\mathbf{R}\mathbf{x})^T\Sigma^{-1}(\mathbf{R}\mathbf{x}) - 2(\mathbf{R}\mathbf{x})^T\Sigma^{-1}\mathbf{m} + \mathbf{m}^T\Sigma^{-1}\mathbf{m},\\ \mathbf{x}^T\Sigma^{-1}\mathbf{x} - 2\mathbf{x}^T\Sigma^{-1}\mathbf{m} &= (\mathbf{R}\mathbf{x})^T\Sigma^{-1}(\mathbf{R}\mathbf{x}) - 2(\mathbf{R}\mathbf{x})^T\Sigma^{-1}\mathbf{m}. \qquad (8)\end{aligned}$$

Let the spectral decomposition of Σ be $\mathbf{V}\Lambda\mathbf{V}^T = (\mathbf{v}_1,\cdots,\mathbf{v}_p)\,diag(\lambda_1,\cdots,\lambda_p)\,(\mathbf{v}_1,\cdots,\mathbf{v}_p)^T$.

Now use the assumption that $\mathbf{m}$ is orthogonal to all the eigenvectors of Σ except one (i.e., $\mathbf{v}_j = \mathbf{m}/\|\mathbf{m}\|$ for some j). More formally, $\mathbf{m}^T\mathbf{v}_i = 0$ for all $i \neq j$ and $\mathbf{m}^T\mathbf{v}_j = s\|\mathbf{m}\|$, where $s = \pm 1$. Without loss of generality, let $j = 1$, which yields $\Sigma^{-1}\mathbf{m} = \mathbf{V}\Lambda^{-1}\mathbf{V}^T\mathbf{m} = {\lambda_1}^{-1}\mathbf{m}$. Substituting this in (8) we get

$$\mathbf{x}^T\Sigma^{-1}\mathbf{x} - 2\lambda_1^{-1}\mathbf{x}^T\mathbf{m} = (\mathbf{R}\mathbf{x})^T\Sigma^{-1}(\mathbf{R}\mathbf{x}) - 2\lambda_1^{-1}(\mathbf{R}\mathbf{x})^T\mathbf{m}.$$

Writing Σ^{-1} in an open form,

$$\sum_{i=1}^{p}\lambda_i^{-1}\left(\mathbf{x}^T\mathbf{v}_i\right)^2 - 2\lambda_1^{-1}\mathbf{x}^T\mathbf{m} = \sum_{i=1}^{p}\lambda_i^{-1}\left((\mathbf{R}\mathbf{x})^T\mathbf{v}_i\right)^2 - 2\lambda_1^{-1}(\mathbf{R}\mathbf{x})^T\mathbf{m},$$
$$\sum_{i=1}^{p}\left(\lambda_i^{-1}\left[\left(\mathbf{x}^T\mathbf{v}_i\right)^2 - \left(\mathbf{x}^T\mathbf{R}^T\mathbf{v}_i\right)^2\right]\right) - 2\lambda_1^{-1}\mathbf{x}^T\mathbf{m} + 2\lambda_1^{-1}\mathbf{x}^T\mathbf{R}^T\mathbf{m} = 0.$$

Recall that the rotation is constrained to be in the subspace spanned by two of the eigenvectors of Σ. One of these eigenvectors must be $\mathbf{v}_1$. Let the other eigenvector be $\mathbf{v}_2$. Then, $\mathbf{x}^T\mathbf{R}^T\mathbf{v}_i = \mathbf{x}^T\mathbf{v}_i$ for $i \neq \{1,2\}$. This simplifies our last equation to

$$\lambda_1^{-1}\left[\left(\mathbf{x}^T\mathbf{v}_1\right)^2 - \left(\mathbf{x}^T\mathbf{R}^T\mathbf{v}_1\right)^2\right] + \lambda_2^{-1}\left[\left(\mathbf{x}^T\mathbf{v}_2\right)^2 - \left(\mathbf{x}^T\mathbf{R}^T\mathbf{v}_2\right)^2\right] + 2\lambda_1^{-1}\left(\mathbf{x}^T\mathbf{R}^T\mathbf{m} - \mathbf{x}^T\mathbf{m}\right) = 0. \qquad (9)$$

Noting that $\left(\mathbf{x}^T\mathbf{v}_1\right)^2 - \left(\mathbf{x}^T\mathbf{R}^T\mathbf{v}_1\right)^2 = (\mathbf{x}^T\mathbf{R}^T\mathbf{v}_1 - \mathbf{x}^T\mathbf{v}_1)(-\mathbf{x}^T\mathbf{v}_1 - \mathbf{x}^T\mathbf{R}^T\mathbf{v}_1)$, and that $\mathbf{m} = s\|\mathbf{m}\|\mathbf{v}_1$, allows us to rewrite (9) as

$$\begin{aligned}&\lambda_1^{-1}(\mathbf{x}^T\mathbf{R}^T\mathbf{v}_1 - \mathbf{x}^T\mathbf{v}_1)(2\|\mathbf{m}\|s - \mathbf{x}^T\mathbf{v}_1 - \mathbf{x}^T\mathbf{R}^T\mathbf{v}_1)\\ &+\lambda_2^{-1}\left(\left(\mathbf{x}^T\mathbf{v}_2\right)^2 - \left(\mathbf{x}^T\mathbf{R}^T\mathbf{v}_2\right)^2\right) = 0,\\ &\lambda_1^{-1}(\mathbf{x}^T\mathbf{R}^T\mathbf{v}_1 - \mathbf{x}^T\mathbf{v}_1)(2\|\mathbf{m}\|s - \mathbf{x}^T\mathbf{v}_1 - \mathbf{x}^T\mathbf{R}^T\mathbf{v}_1)\\ &+\lambda_2^{-1}(\mathbf{x}^T(\mathbf{v}_2 - \mathbf{R}^T\mathbf{v}_2))(\mathbf{x}^T(\mathbf{v}_2 + \mathbf{R}^T\mathbf{v}_2)) = 0. \qquad (10)\end{aligned}$$

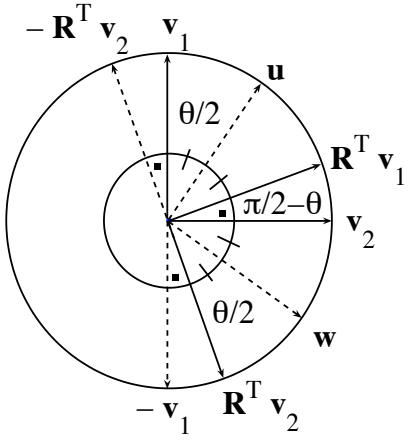


Figure 3: Shown here are two orthonormal vectors, $\mathbf{v}_1$ and $\mathbf{v}_2$, and their rotated versions, $\mathbf{R}^T\mathbf{v}_1$ and $\mathbf{R}^T\mathbf{v}_2$. We see that $\mathbf{R}^T\mathbf{v}_1+\mathbf{v}_1 = 2\mathbf{u}\cos\left(\frac{\theta}{2}\right)$, $\mathbf{R}^T\mathbf{v}_2+\mathbf{v}_2 = 2\mathbf{w}\cos\left(\frac{\theta}{2}\right)$, $\mathbf{R}^T\mathbf{v}_1-\mathbf{v}_1 = 2\mathbf{w}\cos\left(\frac{\pi}{2}-\frac{\theta}{2}\right)$ and $\mathbf{v}_2-\mathbf{R}^T\mathbf{v}_2 = 2\mathbf{u}\cos\left(\frac{\pi}{2}-\frac{\theta}{2}\right)$.

In addition, we know that the rotation matrix $\mathbf{R}$ defines a rotation in the $(\mathbf{v}_1,\mathbf{v}_2)$-plane. Assume that $\mathbf{R}$ rotates the vector $\mathbf{v}_1$ θ degrees in the clockwise direction yielding $\mathbf{R}^T\mathbf{v}_1$. Similarly, $\mathbf{v}_2$ becomes $\mathbf{R}^T\mathbf{v}_2$. From Figure 3 we see that

$$\begin{aligned}
\mathbf{v}_2-\mathbf{R}^T\mathbf{v}_2 &= 2\mathbf{u}\cos\left(\frac{\pi}{2}-\frac{\theta}{2}\right), \quad \mathbf{v}_2+\mathbf{R}^T\mathbf{v}_2 = 2\mathbf{w}\cos\left(\frac{\theta}{2}\right),\\
\mathbf{R}^T\mathbf{v}_1-\mathbf{v}_1 &= 2\mathbf{w}\cos\left(\frac{\pi}{2}-\frac{\theta}{2}\right), \quad \mathbf{R}^T\mathbf{v}_1+\mathbf{v}_1 = 2\mathbf{u}\cos\left(\frac{\theta}{2}\right),
\end{aligned}$$

where $\mathbf{u}$ and $\mathbf{w}$ are the unit vectors as shown in Figure 3. Therefore,

$$\mathbf{v}_2+\mathbf{R}^T\mathbf{v}_2 = (\mathbf{R}^T\mathbf{v}_1-\mathbf{v}_1)\cot\left(\frac{\theta}{2}\right) \quad \text{and} \quad \mathbf{v}_2-\mathbf{R}^T\mathbf{v}_2 = (\mathbf{R}^T\mathbf{v}_1+\mathbf{v}_1)\tan\left(\frac{\theta}{2}\right).$$

If we use these results in (10), we find that

$$\begin{aligned}
&\lambda_1^{-1}(\mathbf{x}^T\mathbf{R}^T\mathbf{v}_1-\mathbf{x}^T\mathbf{v}_1)(2\|\mathbf{m}\|s-\mathbf{x}^T\mathbf{v}_1-\mathbf{x}^T\mathbf{R}^T\mathbf{v}_1)\\
+\;&\lambda_2^{-1}(\mathbf{x}^T\mathbf{R}^T\mathbf{v}_1-\mathbf{x}^T\mathbf{v}_1)\cot\left(\frac{\theta}{2}\right)(\mathbf{x}^T\mathbf{R}^T\mathbf{v}_1+\mathbf{x}^T\mathbf{v}_1)\tan\left(\frac{\theta}{2}\right)=0,
\end{aligned}$$

which can be reorganized to

$$\left[\mathbf{x}^T(\mathbf{R}^T\mathbf{v}_1-\mathbf{v}_1)\right]\left[(\lambda_2^{-1}-\lambda_1^{-1})\mathbf{x}^T(\mathbf{R}^T\mathbf{v}_1+\mathbf{v}_1)+2\lambda_1^{-1}\|\mathbf{m}\|s\right]=0.$$

The two possible solutions of this equation provide the two hyperplanes for the Bayes classifier. The first hyperplane,

$$\mathbf{x}^T\frac{(\mathbf{R}^T\mathbf{v}_1-\mathbf{v}_1)}{2\sin\left(\frac{\theta}{2}\right)}=0, \tag{11}$$

passes through the origin and its normal vector is $(\mathbf{R}^T\mathbf{v}_1 - \mathbf{v}_1)/2\sin(\frac{\theta}{2})$. The second hyperplane,

$$\mathbf{x}^T \frac{\mathbf{R}^T\mathbf{v}_1 + \mathbf{v}_1}{2\cos\left(\frac{\theta}{2}\right)} + \frac{\lambda_1^{-1}\|\mathbf{m}\|s}{(\lambda_2^{-1} - \lambda_1^{-1})\cos\left(\frac{\theta}{2}\right)} = 0, \tag{12}$$

has a bias equal to

$$\frac{\lambda_1^{-1}\|\mathbf{m}\|s}{(\lambda_2^{-1} - \lambda_1^{-1})\cos\left(\frac{\theta}{2}\right)},$$

and its normal is $\left(\mathbf{R}^T\mathbf{v}_1 + \mathbf{v}_1\right)/2\cos(\theta/2)$. ■

The result above, shows that when the rotation matrix is spanned by two of the eigenvectors of Σ, then N_1 and N_2 are spherical-homoscedastic. The reader may have noted though, that there exist some Σ (e.g., $\tau^2\mathbf{I}$) which are less restrictive on $\mathbf{R}$. The same applies to spherical distributions. We start our study with the case where the true underlying distributions of the data are von Mises-Fisher.

Theorem 3 *Two von Mises-Fisher distributions* $M_1(\mu,\kappa)$ *and* $M_2(\mathbf{R}^T\mu,\kappa)$ *are spherical-homoscedastic if* $\mathbf{R} \in SO(p)$.

Proof As above, the Bayes decision boundary is given when the ratio between the log-likelihood of $M_1(\mu,\kappa)$ and that of $M_2(\mathbf{R}^T\mu,\kappa)$ is equal to one. This means that

$$\begin{aligned} \kappa\mu^T\mathbf{x} + \ln(c_{MF}(\kappa)) &= \kappa(\mathbf{R}^T\mu)^T\mathbf{x} + \ln(c_{MF}(\kappa)), \\ \kappa(\mathbf{x}^T\mathbf{R}^T\mu - \mathbf{x}^T\mu) &= 0. \end{aligned} \tag{13}$$

We see that (13) defines the decision boundary between $M_1(\mu,\kappa)$ and $M_2(\mathbf{R}^T\mu,\kappa)$ and that this is a hyperplane[4] with normal

$$\frac{\mathbf{R}^T\mu - \mu}{2\cos(\omega/2)},$$

where ω is the magnitude of the rotation angle.[5] Hence, $M_1(\mu,\kappa)$ and $M_2(\mathbf{R}^T\mu,\kappa)$ are spherical-homoscedastic. ■

We now want to show that the Bayes decision boundary of two spherical-homoscedastic vMF is the same as the classification boundary obtained when these distributions are modeled using Gaussian pdfs. However, Theorem 2 provides two hyperplanes—those given in Equations (11-12). We need to show that one of these equations is the same as the hyperplane given in Equation (13), and that the other equation gives an irrelevant decision boundary. The irrelevant hyperplane is that in (12). To show why this equation is not relevant for classification, we will demonstrate that this hyperplane is always outside S^{p-1} and, hence, cannot divide the spherical data into more than one region.

4. Since this hyperplane is constrained with $\|\mathbf{x}\| = 1$, the decision boundary will define a great circle on the sphere.
5. Note that since the variance about every direction orthogonal to μ is equal to τ^2, all rotations can be expressed as a planar rotation spanned by μ and any $\mu^\perp$ (where $\mu^\perp$ is a vector orthogonal to μ) .

Proposition 4 *When modeling the data of two spherical-homoscedastic von Mises-Fisher distributions, $M_1(\mu,\kappa)$ and $M_2(\mathbf{R}^T\mu,\kappa)$, using two Gaussian distributions, $N_1(\mathbf{m},\Sigma)$ and $N_2(\mathbf{R}^T\mathbf{m},\mathbf{R}^T\Sigma\mathbf{R})$, the Bayes decision boundary will be given by the two hyperplanes defined in Equations (11-12). However, the hyperplane given in (12) does not intersect with the sphere and can be omitted for classification purposes.*

Proof Recall that the bias of the hyperplane given in (12) was

$$b_2 = \frac{s\|\mathbf{m}\|}{((\lambda_1/\lambda_2)-1)\cos(\theta/2)}. \tag{14}$$

We need to show that the absolute value of this bias is greater than one; that is, $|b_2| > 1$.

We know from Dryden and Mardia (1998) that if $\mathbf{x}$ is distributed as $M(\mu,\kappa)$, then

$$\mathbf{m} = E(\mathbf{x}) = A_p(\kappa)\mu,$$

and the covariance matrix of $\mathbf{x}$ is given by

$$\Sigma = A_p'(\kappa)\mu\mu^T + \frac{A_p(\kappa)}{\kappa}(\mathbf{I}_p - \mu\mu^T),$$

where $A_p(\kappa) = I_{p/2}(\kappa)/I_{p/2-1}(\kappa)$ and $A_p'(\kappa) = 1 - A_p^2(\kappa) - \frac{p-1}{\kappa}A_p(\kappa)$. Note that the first eigenvector of the matrix defined above is aligned with the mean direction, and that the rest are orthogonal to it. Furthermore, the first eigenvalue of this matrix is

$$\lambda_1 = 1 - A_p^2(\kappa) - \frac{p-1}{\kappa}A_p(\kappa),$$

and the rest are all equal and defined as

$$\lambda_i = \frac{A_p(\kappa)}{\kappa}, \quad \forall i > 1.$$

Substituting the above calculated terms in (14) yields

$$\hat{b}_2(\kappa) = \frac{A_p(\kappa)}{\frac{1-A_p^2(\kappa)-\frac{p-1}{\kappa}A_p(\kappa)}{\frac{A_p(\kappa)}{\kappa}} - 1} = \frac{A_p^2(\kappa)}{\kappa\left(1 - A_p^2(\kappa) - \frac{p}{\kappa}A_p(\kappa)\right)} \tag{15}$$

with $b_2 = \frac{s\hat{b}_2(\kappa)}{\cos(\theta/2)}$.

Note that we can rewrite $A_p(\kappa)$ as

$$A_p(\kappa) = \frac{I_\nu(\kappa)}{I_{\nu-1}(\kappa)},$$

where $\nu = p/2$. Moreover, the recurrence relation between modified Bessel functions states that $I_{\nu-1}(\kappa) - I_{\nu+1}(\kappa) = \frac{2\nu}{\kappa}I_\nu(\kappa)$, which is the same as $1 - (I_{\nu+1}(\kappa)/I_{\nu-1}(\kappa)) = (2\nu I_\nu(\kappa))/(\kappa I_{\nu-1}(\kappa))$. This can be combined with the result shown above to yield

$$\frac{p}{\kappa}A_p(\kappa) = 1 - \frac{I_{\nu+1}(\kappa)}{I_{\nu-1}(\kappa)}.$$

By substituting these terms in (15), one obtains

$$\hat{b}_2(\kappa) = \frac{\left(\frac{I_\nu(\kappa)}{I_{\nu-1}(\kappa)}\right)^2}{\kappa\left(-\left(\frac{I_\nu(\kappa)}{I_{\nu-1}(\kappa)}\right)^2 + \frac{I_{\nu+1}(\kappa)I_{\nu-1}(\kappa)}{I^2_{\nu-1}(\kappa)}\right)} = \frac{I^2_\nu(\kappa)}{\kappa\left(-I^2_\nu(\kappa) + I_{\nu+1}(\kappa)I_{\nu-1}(\kappa)\right)}.$$

Using the bound defined by Joshi (1991, see Equation 3.14), $0 < I^2_\nu(\kappa) - I_{\nu-1}(\kappa)I_{\nu+1}(\kappa) < \frac{I^2_\nu(\kappa)}{\nu+\kappa}$ ($\forall\kappa > 0$), we have

$$0 < I^2_\nu(\kappa) - I_{\nu-1}(\kappa)I_{\nu+1}(\kappa) < \frac{I^2_\nu(\kappa)}{\nu+\kappa},$$

$$\frac{I^2_\nu(\kappa)}{\kappa(I^2_\nu(\kappa) - I_{\nu-1}(\kappa)I_{\nu+1}(\kappa))} > \frac{(\nu+\kappa)}{\kappa},$$

$$\frac{I^2_\nu(\kappa)}{\kappa(-I^2_\nu(\kappa) + I_{\nu-1}(\kappa)I_{\nu+1}(\kappa))} < \frac{(\nu+\kappa)}{-\kappa} < -1.$$

This upper-bound shows that $|b_2| = \left|\frac{\hat{b}_2(\kappa)}{\cos(\theta/2)}\right| > 1$. This means that the second hyperplane will not divide the data into more than one class and therefore can be ignored for classification purposes. ■

From our proof above, we note that Σ is spanned by μ and a set of $p-1$ basis vectors that are orthogonal to μ. Furthermore, since a vMF is circularly symmetric around μ, these basis vectors can be represented by any orthonormal set of vectors orthogonal to μ. Next, note that the mean direction of M_2 can be written as $\mu_2 = \mathbf{R}^T\mu$. This means that $\mathbf{R}$ can be spanned by μ and any unit vector orthogonal to μ (denoted $\mu^\perp$)—such as an eigenvector of Σ. Therefore, N_1 and N_2 are spherical-homoscedastic. We can summarize this results in the following.

Corollary 5 *If we model two spherical-homoscedastic von Mises-Fisher, $M_1(\mu,\kappa)$ and $M_2(\mathbf{R}^T\mu,\kappa)$, with their corresponding Gaussian approximations $N_1(\mathbf{m},\Sigma)$ and $N_2(\mathbf{R}^T\mathbf{m},\mathbf{R}^T\Sigma\mathbf{R})$, then N_1 and N_2 are also spherical-homoscedastic.*

This latest result is important to show that the Bayes decision boundaries of two spherical-homoscedastic vMFs can be calculated exactly using the Gaussian model.

Theorem 6 *The Bayes decision boundary of two spherical-homoscedastic von Mises-Fisher, $M_1(\mu,\kappa)$ and $M_2(\mathbf{R}^T\mu,\kappa)$, is the same as that given in (11), which is obtained when modeling $M_1(\mu,\kappa)$ and $M_2(\mathbf{R}^T\mu,\kappa)$ using the two Gaussian distributions $N_1(\mathbf{m},\Sigma)$ and $N_2(\mathbf{R}^T\mathbf{m},\mathbf{R}^T\Sigma\mathbf{R})$.*

Proof From Corollary 5 we know that $N_1(\mathbf{m},\Sigma)$ and $N_2(\mathbf{R}^T\mathbf{m},\mathbf{R}^T\Sigma\mathbf{R})$ are spherical-homoscedastic. And, from Proposition 4, we know that the hyperplane decision boundary given by Equation (12) is outside the sphere and can be eliminated. In the proof of Proposition 4 we also showed that $\mathbf{v}_1 = \mu$. This means, (11) can be written as

$$\mathbf{x}^T\frac{(\mathbf{R}^T\mu - \mu)}{2\sin(\theta/2)} = 0. \tag{16}$$

The decision boundary for two spherical-homoscedastic vMF was derived in Theorem 3, where it was shown to be

$$\kappa(\mathbf{x}^T R^T\mu - \mathbf{x}^T\mu) = 0. \tag{17}$$

We note that the normal vectors of Equations (16-17) are the same and that both biases are zero. Therefore, the two equations define the same great circle on S^{p-1}; that is, they yield the same classification results. ■

When the vMF model is not flexible enough to represent our data, we need to use a more general definition such as that given by Bingham. We will now study under which conditions two Bingham distributions are spherical-homoscedastic and, hence, can be efficiently approximated with Gaussians.

Theorem 7 *Two Bingham distributions, $B_1(\mathbf{A})$ and $B_2(\mathbf{R}^T\mathbf{A}\mathbf{R})$, are spherical-homoscedastic if $\mathbf{R} \in SO(p)$ defines a planar rotation in the subspace spanned by any two of the eigenvectors of $\mathbf{A}$, say $\mathbf{q}_1$ and $\mathbf{q}_2$.*

Proof Making the ratio of the log-likelihood equations equal to one yields

$$\mathbf{x}^T\mathbf{A}\mathbf{x} = \mathbf{x}^T\mathbf{R}^T\mathbf{A}\mathbf{R}\mathbf{x}. \tag{18}$$

Since the rotation is defined in the subspace spanned by $\mathbf{q}_1$ and $\mathbf{q}_2$ and $\mathbf{A} = \mathbf{Q}\Lambda\mathbf{Q}^T$, then the above equation can be expressed (in open form) as $\sum_{i=1}^{p}\lambda_i(\mathbf{x}^T\mathbf{q}_i)^2 = \sum_{i=1}^{p}\lambda_i(\mathbf{x}^T\mathbf{R}^T\mathbf{q}_i)^2$. In addition, $\mathbf{R}^T\mathbf{q}_i = \mathbf{q}_i$ for $i > 2$, which simplifies our equation to

$$\sum_{i=1}^{p}\lambda_i(\mathbf{x}^T\mathbf{q}_i)^2 = \sum_{i=1}^{2}\lambda_i(\mathbf{x}^T\mathbf{R}^T\mathbf{q}_i)^2 + \sum_{i=3}^{p}\lambda_i(\mathbf{x}^T\mathbf{q}_i)^2$$

$$\lambda_1\left((\mathbf{x}^T\mathbf{q}_1)^2 - (\mathbf{x}^T\mathbf{R}^T\mathbf{q}_1)^2\right) + \lambda_2\left((\mathbf{x}^T\mathbf{q}_2)^2 - (\mathbf{x}^T\mathbf{R}^T\mathbf{q}_2)^2\right) = 0. \tag{19}$$

From the proof of Theorem 2, we know that $\mathbf{q}_2$ can be expressed as a function of $\mathbf{q}_1$ as $\mathbf{q}_2 + \mathbf{R}^T\mathbf{q}_2 = \left(\mathbf{R}^T\mathbf{q}_1 - \mathbf{q}_1\right)\cot(\theta/2)$ and $\mathbf{q}_2 - \mathbf{R}^T\mathbf{q}_2 = \left(\mathbf{R}^T\mathbf{q}_1 + \mathbf{q}_1\right)\tan(\theta/2)$. This allows us to write the decision boundary given in (19) as

$$\mathbf{x}^T(\mathbf{R}^T\mathbf{q}_1 + \mathbf{q}_1) = 0, \tag{20}$$

and

$$\mathbf{x}^T(\mathbf{R}^T\mathbf{q}_1 - \mathbf{q}_1) = 0. \tag{21}$$

These two hyperplanes are necessary to successfully classify the antipodally symmetric data of two Bingham distributions. ■

Since antipodally symmetric distributions, such as Bingham distributions, have zero mean, the Gaussian distributions fitted to the data sampled from these distributions will also have zero mean. We now study the spherical-homoscedastic Gaussian pdfs when the mean vector is equal to zero.

Lemma 8 *Two zero-mean Gaussian distributions, $N_1(\mathbf{0},\Sigma)$ and $N_2(\mathbf{0},\mathbf{R}^T\Sigma\mathbf{R})$, are spherical-homoscedastic if $\mathbf{R} \in SO(p)$ defines a planar rotation in the subspace spanned by any two of the eigenvectors of Σ, say $\mathbf{v}_1$ and $\mathbf{v}_2$.*

Proof The Bayes classification boundary between these distributions can be obtained by making the ratio of the log-likelihood equations equal to one, $\mathbf{x}^T\Sigma\mathbf{x} = \mathbf{x}^T\mathbf{R}^T\Sigma\mathbf{R}\mathbf{x}$. Note that this equation is in the same form as that derived in (18). Furthermore, since the rotation is defined in the subspace spanned by $\mathbf{v}_1$ and $\mathbf{v}_2$ and $\Sigma = \mathbf{V}\Lambda\mathbf{V}^T$, we can follow the proof of Theorem 7 to show

$$\mathbf{x}^T(\mathbf{R}^T\mathbf{v}_1 + \mathbf{v}_1) = 0, \tag{22}$$

and

$$\mathbf{x}^T(\mathbf{R}^T\mathbf{v}_1 - \mathbf{v}_1) = 0. \tag{23}$$

And, therefore, N_1 and N_2 are also spherical-homoscedastic. ■

We are now in a position to prove that the decision boundaries obtained using two Gaussian distributions are the same as those defined by spherical-homoscedastic Bingham distributions.

Theorem 9 *The Bayes decision boundaries of two spherical-homoscedastic Bingham distributions, $B_1(\mathbf{A})$ and $B_2(\mathbf{R}^T\mathbf{A}\mathbf{R})$, are the same as those obtained when modeling $B_1(\mathbf{A})$ and $B_2(\mathbf{R}^T\mathbf{A}\mathbf{R})$ with two Gaussian distributions, $N_1(\mathbf{m},\Sigma)$ and $N_2(\mathbf{R}^T\mathbf{m},\mathbf{R}^T\Sigma\mathbf{R})$, where $\mathbf{m} = \mathbf{0}$ and $\Sigma = \mathbf{S}$.*

Proof Since the data sampled from a Bingham distribution is symmetric with respect to the origin, its mean will be the origin, $\mathbf{m} = \mathbf{0}$. Therefore, the sample covariance matrix will be equal to the sample autocorrelation matrix $\mathbf{S} = n^{-1}\mathbf{X}\mathbf{X}^T$. In short, the estimated Gaussian distribution of $B_1(\mathbf{A})$ will be $N_1(\mathbf{0},\mathbf{S})$. We also know from Section 2.2 that the m.l.e. of the orthonormal matrix $\mathbf{Q}$ (where $\mathbf{A} = \mathbf{Q}\Lambda\mathbf{Q}^T$) is given by the eigenvectors of $\mathbf{S}$. This means that the two Gaussian distributions representing the data sampled from two spherical-homoscedastic Bingham distributions, $B_1(\mathbf{A})$ and $B_2(\mathbf{R}^T\mathbf{A}\mathbf{R})$, are $N_1(\mathbf{0},\mathbf{S})$ and $N_2(\mathbf{0},\mathbf{R}^T\mathbf{S}\mathbf{R})$.

Following Lemma 8, $N_1(\mathbf{0},\mathbf{S})$ and $N_2(\mathbf{0},\mathbf{R}^T\mathbf{S}\mathbf{R})$ are spherical-homoscedastic if $\mathbf{R}$ is spanned by any two eigenvectors of $\mathbf{S}$. Since the eigenvectors of $\mathbf{A}$ and $\mathbf{S}$ are the same ($\mathbf{v}_i = \mathbf{q}_i$ for all i), these two Gaussian distributions representing the two spherical-homoscedastic Bingham distributions will also be spherical-homoscedastic. Furthermore, the hyperplanes of the spherical-homoscedastic Bingham distributions $B_1(\mathbf{A})$ and $B_2(\mathbf{R}^T\mathbf{A}\mathbf{R})$, Equations (20 - 21), and the hyperplanes of the spherical-homoscedastic Gaussian distributions $N_1(\mathbf{0},\mathbf{S})$ and $N_2(\mathbf{0},\mathbf{R}^T\mathbf{S}\mathbf{R})$, Equations (22 - 23), will be identical. ■

We now turn to study the similarity between the results obtained using the Gaussian distribution and the Kent distribution. We first define when two Kent distributions are spherical-homoscedastic.

Theorem 10 *Two Kent distributions $K_1(\mu,\kappa,\mathbf{A})$ and $K_2(\mathbf{R}^T\mu,\kappa,\mathbf{R}^T\mathbf{A}\mathbf{R})$ are spherical-homoscedastic, if the rotation matrix $\mathbf{R}$ is defined on the plane spanned by the mean direction μ and one of the eigenvectors of $\mathbf{A}$.*

Proof By making the two log-likelihood equations equal, we have $\kappa\mu^T\mathbf{x} + \mathbf{x}^T\mathbf{A}\mathbf{x} = \kappa(\mathbf{R}^T\mu)^T\mathbf{x} + \mathbf{x}^T\mathbf{R}^T\mathbf{A}\mathbf{R}\mathbf{x}$. Let the spectral decomposition of $\mathbf{A}$ be $\mathbf{A} = \mathbf{Q}\Lambda\mathbf{Q}^T$, then $\kappa(\mu^T\mathbf{x} - \mu^T\mathbf{R}\mathbf{x}) + \sum_{i=1}^{p}\lambda_i(\mathbf{x}^T\mathbf{q}_i)^2 - \sum_{i=1}^{p}\lambda_i(\mathbf{x}^T(\mathbf{R}^T\mathbf{q}_i))^2 = 0$. Since $\mathbf{R}$ is defined to be in the plane spanned by an eigenvector of $\mathbf{A}$ (say, $\mathbf{q}_1$) and the mean direction μ, one can simplify the above equation to $\kappa(\mu^T\mathbf{x} - \mu^T\mathbf{R}\mathbf{x}) + \lambda_1(\mathbf{x}^T\mathbf{q}_1)^2 -$

$\lambda_1(\mathbf{x}^T(\mathbf{R}^T\mathbf{q}_1))^2 = 0$. Since the first term of this equation is a constant, its transpose would yield the same result,

$$\kappa(\mathbf{x}^T(\mu - \mathbf{R}^T\mu)) + \lambda_1(\mathbf{x}^T(\mathbf{q}_1 - \mathbf{R}^T\mathbf{q}_1))(\mathbf{x}^T(\mathbf{q}_1 + \mathbf{R}^T\mathbf{q}_1)) = 0.$$

Using the relation between $\mathbf{v}_1$ and $\mathbf{v}_2$ (now μ and $\mathbf{q}_1$) given in the proof of Theorem 2,

$$\begin{aligned}(\mathbf{q}_1 + \mathbf{R}^T\mathbf{q}_1) &= (\mathbf{R}^T\mu - \mu)\cot\left(\frac{\theta}{2}\right),\\ (\mathbf{q}_1 - \mathbf{R}^T\mathbf{q}_1) &= (\mathbf{R}^T\mu + \mu)\tan\left(\frac{\theta}{2}\right),\end{aligned}$$

we can write $(\mathbf{x}^T(\mathbf{R}^T\mu - \mu))(\mathbf{x}^T(\mathbf{R}^T\mu + \mu)\lambda_1 - \kappa) = 0$, where θ is the rotation angle defined by $\mathbf{R}$. This equation gives us the hyperplane decision boundary equations,

$$\mathbf{x}^T\frac{(\mathbf{R}^T\mu - \mu)}{\sin\left(\frac{\theta}{2}\right)} = 0, \tag{24}$$

$$\mathbf{x}^T\left(\frac{\mathbf{R}^T\mu + \mu}{\cos\left(\frac{\theta}{2}\right)}\right) - \frac{\kappa}{\cos\left(\frac{\theta}{2}\right)\lambda_1} = 0. \tag{25}$$

■

Finally, we are in a position to define the relation between Kent and Gaussian pdfs.

Theorem 11 *The first hyperplane given by the Bayes decision boundary of two spherical-homoscedastic Kent distributions, $K_1(\mu,\kappa,\mathbf{A})$ and $K_2(\mathbf{R}^T\mu,\kappa,\mathbf{R}^T\mathbf{A}\mathbf{R})$, is equal to the first hyperplane obtained when modeling K_1 and K_2 with the two Gaussian distributions $N_1(\mathbf{m},\Sigma)$ and $N_2(\mathbf{R}^T\mathbf{m},\mathbf{R}^T\Sigma\mathbf{R})$ that best approximate them. Furthermore, when $\kappa > \lambda_{1_K}$ and $\|m\| > 1 - \lambda_{1_G}/\lambda_{2_G}$, then the second hyperplanes of the Kent and Gaussian distributions are outside the sphere and can be ignored in classification; where λ_{1_K} is the eigenvalue associated to the eigenvector of $\mathbf{A}$ defining the rotation $\mathbf{R}$, and λ_{1_G} and λ_{2_G} are the two eigenvalues of Σ defining the rotation $\mathbf{R}$.*

Proof If we fit a Gaussian distribution to an ellipsoidally symmetric pdf, then the mean direction of the data is described by one of the eigenvectors of the covariance matrix. Since the Kent distribution assumes the data is either concentrated or distributed more or less equally about every dimension, one can conclude that the eigenvectors of $\bar{\mathbf{S}}$ (the scatter matrix calculated on the null-space of the mean direction) are a good estimate of the orthonormal bases of $\mathbf{A}$ (Kent, 1982). This means that (11) and (24) will define the same hyperplane equation. Furthermore, we see that the bias in (12) and that of (25) will be the same when

$$-\kappa\lambda_{1_K}^{-1} = \frac{\|\mathbf{m}\|s}{\left(\frac{\lambda_{1_G}}{\lambda_{2_G}} - 1\right)},$$

where λ_{1_K} is the eigenvalue associated to the eigenvector defining the rotation plane (as given in Theorem 10), and λ_{1_G} and λ_{2_G} are the eigenvalues associated to the eigenvectors that span the rotation matrix $\mathbf{R}$ (as shown in Theorem 2—recall that λ_{1_G} is associated to the eigenvector aligned with the mean direction).

Similarly to what happened for the vMF case, the second hyperplane may be outside S^{p-1}. When this is the case, such planes are not relevant and can be eliminated. For this to happen, the two biases need not be the same, but need be larger than one; that is,

$$|\kappa\lambda_{1_K}^{-1}| > 1 \quad \text{and} \quad \left| \frac{s\|\mathbf{m}\|}{\left(\frac{\lambda_{1_G}}{\lambda_{2_G}} - 1\right)} \right| > 1. \tag{26}$$

These two last conditions can be interpreted as follows. The second hyperplane of the Kent distribution will not intersect with the sphere when $\kappa > \lambda_{1_K}$. The second hyperplane of the Gaussian estimate will be outside the sphere when $\|m\| > 1 - \lambda_{1_G}/\lambda_{2_G}$. These two conditions hold, for example, when the data is concentrated, but not when the data is uniformly distributed. ■

Thus far, we have shown where the results obtained by modeling the true underlying spherical distributions with Gaussians do not pose a problem. We now turn to the case where both solutions may differ.

4. Spherical-Heteroscedastic Distributions

When two (or more) distributions are not spherical-homoscedastic, we will refer to them as spherical-heteroscedastic. In such a case, the classifier obtained with the Gaussian approximation needs not be the same as that computed using the original spherical distributions. To study this problem, one may want to compute the classification error that is added to the original Bayes error produced by the Bayes classifier on the two (original) spherical distributions. Following the classical notation in Bayesian theory, we will refer to this as the reducible error. This idea is illustrated in Figure 4. In Figure 4(a) we show the original Bayes error obtained when using the original vMF distributions. Figure 4(b) depicts the classifier obtained when one models these vMF using two Gaussian distributions. And, in Figure 4(c), we illustrate the reducible error added to the original Bayes error when one employs the new classifier in lieu of the original one.

In theory, we could calculate the reducible error by means of the posterior probabilities of the two spherical distributions, $P_S(w_1|\mathbf{x})$ and $P_S(w_2|\mathbf{x})$, and the posteriors of the Gaussians modeling them, $P_G(w_1|\mathbf{x})$ and $P_G(w_2|\mathbf{x})$. This is given by,

$$\begin{aligned} P(\text{reducible error}) &= \int\limits_{\frac{P_G(w_1|\mathbf{x})}{P_G(w_2|\mathbf{x})} \geq 1} P_S(w_2|\mathbf{x})\, p(\mathbf{x})\, dS^{p-1} + \int\limits_{\frac{P_G(w_1|\mathbf{x})}{P_G(w_2|\mathbf{x})} < 1} P_S(w_1|\mathbf{x})\, p(\mathbf{x})\, dS^{p-1} \\ &- \int_{S^{p-1}} \min(P_S(w_1|\mathbf{x}), P_S(w_2|\mathbf{x}))\, p(\mathbf{x})\, dS^{p-1}, \end{aligned}$$

where the first two summing terms calculate the error defined by the classifier obtained using the Gaussian approximation (as for example that shown in Figure 4(b)), and the last term is the Bayes error associated to the original spherical-heteroscedastic distributions (Figure 4(a)).

Unfortunately, in practice, this error cannot be calculated because it is given by the integral of a number of class densities over a nonlinear region on the surface of a sphere. Note that, since we are exclusively interested in knowing how the reducible error increases as the data distributions deviate from spherical-homoscedasticity, the use of error bounds would not help us solve this problem either. We are therefore left to empirically study how the reducible error increases as the original data distributions deviate from spherical-homoscedastic. This we will do next.

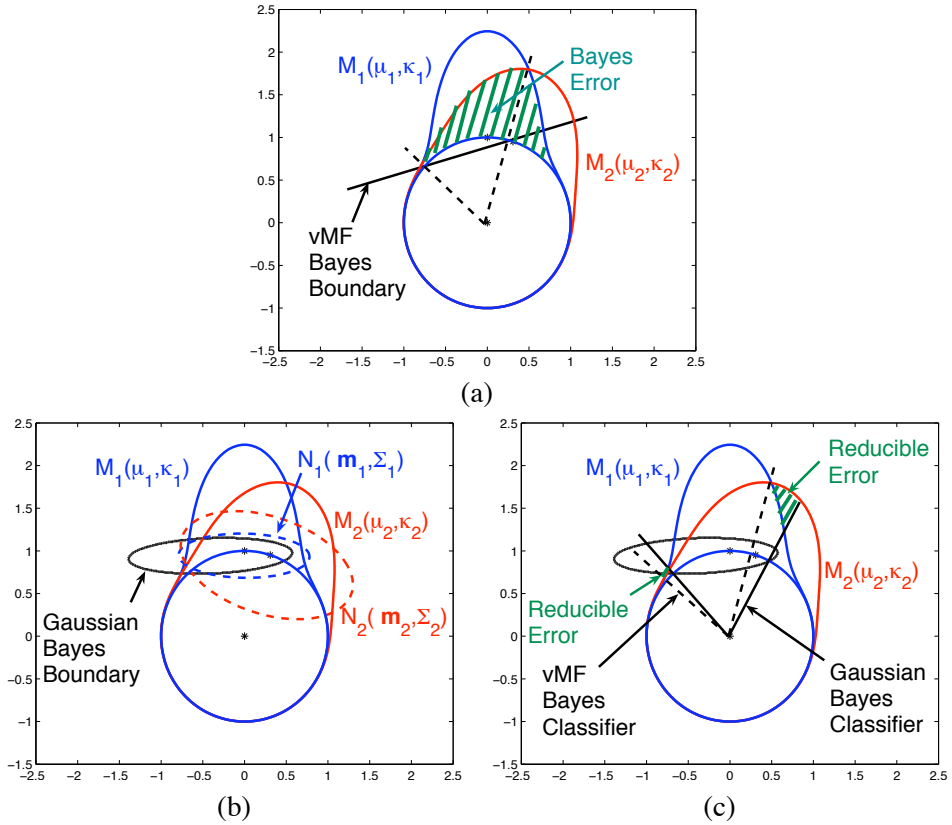


Figure 4: (a) Shown here are two spherical-heteroscedastic vMF distributions with $\kappa_1 = 10$ and $\kappa_2 = 5$. The solid line is the Bayes decision boundary, and the dashed lines are the corresponding classifiers. This classifier defines the Bayes error, which is represented by the dashed fill. In (b) we show the Bayes decision boundary obtained using the two Gaussian distributions that best model the original vMFs. (c) Provides a comparison between the Bayes classifier derived using the original vMFs (dashed lines) and that calculated from the Gaussian approximation (solid lines). The dashed area shown in (c) corresponds to the error added to the original Bayes error; that is, the reducible error.

4.1 Modeling Spherical-Heteroscedastic vMFs

We start our analysis with the case where the two original spherical distributions are given by the vMF model. Let these two vMFs be $M_1(\mu_1,\kappa_1)$ and $M_2(\mu_2,\kappa_2)$, where $\mu_2 = \mathbf{R}^T\mu_1$ and $\mathbf{R} \in SO(p)$. Recall that $\mathbf{R}$ defines the angle θ which specifies the rotation between μ_1 and μ_2. Furthermore, let $N_1(\mathbf{m}_1,\Sigma_1)$ and $N_2(\mathbf{m}_2,\Sigma_2)$ be the two Gaussian distributions that best model $M_1(\mu_1,\kappa_1)$ and $M_2(\mu_2,\kappa_2)$, respectively. From the proof of Proposition 4 we know that the means and covariance matrices of these Gaussians can be defined in terms of the mean directions and the concentration parameters of the corresponding vMF distributions. Defining the parameters of the Gaussian distributions in terms of κ_i and μ_i ($i = \{1,2\}$) allows us to estimate the reducible error with respect to different rotation angles θ, concentration parameters κ_1 and κ_2, and dimensionality p.

Since our goal is to test how the reducible error increases as the original data distributions deviate from spherical-homoscedasticity, we will plot the results for distinct values of κ_2/κ_1. Note that when $\kappa_2/\kappa_1 = 1$ the data is spherical-homoscedastic and that the larger the value of κ_2/κ_1 is, the more we deviate from spherical-homoscedasticity. In our experiments, we selected κ_1 and κ_2 so that the value of κ_2/κ_1 varied from a low of 1 to maximum of 10 at 0.5 intervals. To do this we varied κ_1 from 1 to 10 at unit steps and selected κ_2 such that the κ_2/κ_1 ratio is equal to one of the values described above.

The dimensionality of the feature space is also varied from 2 to 100 at 10-step intervals. In addition, we also vary the value of the angle θ from 10^o to 180^o at 10^o increments.

The average of the reducible error over all possible values of κ_1 and over all values of θ from 10^o to 90^o is shown in Figure 5(a). As anticipated by our theory, the reducible error is zero when $\kappa_2/\kappa_1 = 1$ (i.e., when the data is spherical-homoscedastic). We see that as the distributions start to deviate from spherical-homoscedastic, the probability of reducible error increases really fast. Nonetheless, we also see that after a short while, this error starts to decrease. This is because the data of the second distribution (M_2) becomes more concentrated. To see this, note that to make κ_2/κ_1 larger, we need to increase κ_2 (with respect to κ_1). This means that the area of possible overlap between M_1 and M_2 decreases and, hence, the reducible error will generally become smaller. In summary, the probability of reducible error increases as the data deviates from spherical-homoscedasticity and decreases as the data becomes more concentrated. This means that, in general, the more two non-highly concentrated distributions deviate from spherical-homoscedastic, the more sense it makes to take the extra effort to model the spherical data using one of the spherical models introduced in Section 2. Nevertheless, it is important to note that the reducible error remains relatively low ($\sim 3\%$).

We also observe, in Figure 5(a), that the probability of reducible error decreases with the dimensionality (which would be something unexpected had the original pdf been defined in the Cartesian space). This effect is caused by the spherical nature of the von Mises-Fisher distribution. Note that since the volume of the distributions need to remain constant, the probability of the vMF at each given point will be reduced when the dimensionality is made larger. Therefore, as the dimensionality increase, the volume of the reducible error area (i.e., the probability) will become smaller.

In Figure 5(b) we show the average probability of the reducible error over κ_1 and p, for different values of κ_2/κ_1 and θ. Here we also see that when two vMFs are spherical-homoscedastic (i.e., $\kappa_2/\kappa_1 = 1$), the average of the reducible error is zero. As we deviate from spherical-homoscedasticity, the probability of reducible error increases. Furthermore, it is interesting to note that as θ increases (in the spherical-heteroscedastic case), the probability of reducible error decreases. This is because as θ increases, the two distributions M_1 and M_2 fall farther apart and, hence, the area of possible overlap generally reduces.

4.2 Modeling Spherical-Heteroscedastic Bingham Distributions

As already mentioned earlier, the parameter estimation for Bingham is much more difficult than that of vMF and equations directly linking the parameters of any two Bingham distributions $B_1(\mathbf{A}_1)$ and $B_2(\mathbf{A_2})$ to those of the corresponding Gaussians $N_1(\mathbf{0}, \Sigma_1)$ and $N_2(\mathbf{0}, \Sigma_2)$ are not usually available. Hence, some parameters will need to be estimated from randomly chosen samples from B_i. Recall from Section 2.2 that another difficulty is the calculation of the normalizing constant $c_B(\mathbf{A})$ because

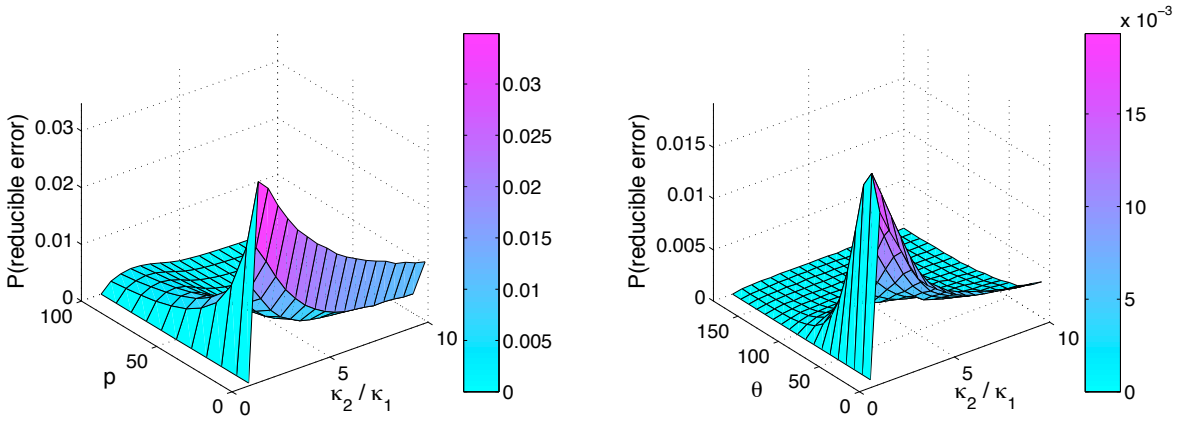


Figure 5: In (a) we show the average of the probability of reducible error over κ_1 and $\theta = \{10^o, 20^o, \ldots, 90^o\}$ for different values of κ_2/κ_1 and dimensionality p. In (b) we show the average probability of reducible error over p and κ_1 for different values of κ_2/κ_1 and $\theta = \{10^o, 20^o, \ldots, 180^o\}$.

this requires us to solve a contour integral on S^{p-1}. This hypergeometric function will be calculated with the method defined by Koev and Edelman (2006).

Moreover, if we want to calculate the reducible error on S^{p-1}, we will need to simulate each of the p variance parameters of B_1 and B_2 and the $p-1$ possible rotations between their means; that is, $3p-1$. While it would be almost impossible to simulate this for a large number of dimensions p, we can easily restrict the problem to one of our interest that is of a manageable size.

In our simulation, we are interested in testing the particular case where $p = 3$.[6] Furthermore, we constrain our analysis to the case where the parameter matrix $\mathbf{A}_1$ has a diagonal form; that is, $\mathbf{A}_1 = diag(\lambda_1, \lambda_2, \lambda_3)$. The parameter matrix $\mathbf{A}_2$ can then be defined as a rotated and scaled version of $\mathbf{A}_1$ as $\mathbf{A}_2 = \varsigma \mathbf{R}^T \mathbf{A}_1 \mathbf{R}$, where ς is the scale parameter,

$$\mathbf{R} = \mathbf{R}_1 \mathbf{R}_2 \in SO(3), \tag{27}$$

$\mathbf{R}_1$ defines a planar rotation θ in the range space given by the first two eigenvectors of $\mathbf{A}_1$, and $\mathbf{R}_2$ specifies a planar rotation ϕ in the space defined by the first and third eigenvectors of $\mathbf{A}_1$. Note that B_1 and B_2 can only be spherical-homoscedastic if $\varsigma = 1$ and the rotation is planar ($\phi = 0$ or $\theta = 0$). To generate our results, we used all possible combinations of the values given by $-1/2j$ (with j the odd numbers from 1 to 15 to represent low concentrations and $j = \{30, 60\}$ to model high concentrations) as entries for λ_1, λ_2 and λ_3 constrained to $\lambda_1 < \lambda_2 < \lambda_3$ (i.e., a total of 120 combinations). We also let $\theta = \{0^o, 10^o, \ldots, 90^o\}$, $\phi = \{0^o, 10^o, \ldots, 90^o\}$ and $\varsigma = \{1, 2, \ldots, 10\}$.

In Figure 6(a), we study the case where $\varsigma = 1$. In this case, B_1 and B_2 are spherical-homoscedastic when either ϕ or θ is zero. As seen in the figure, the probability of reducible error increases as the

6. We have also simulated the cases where p was equal to 10 and 50 and observed almost identical results to those shown in this paper.

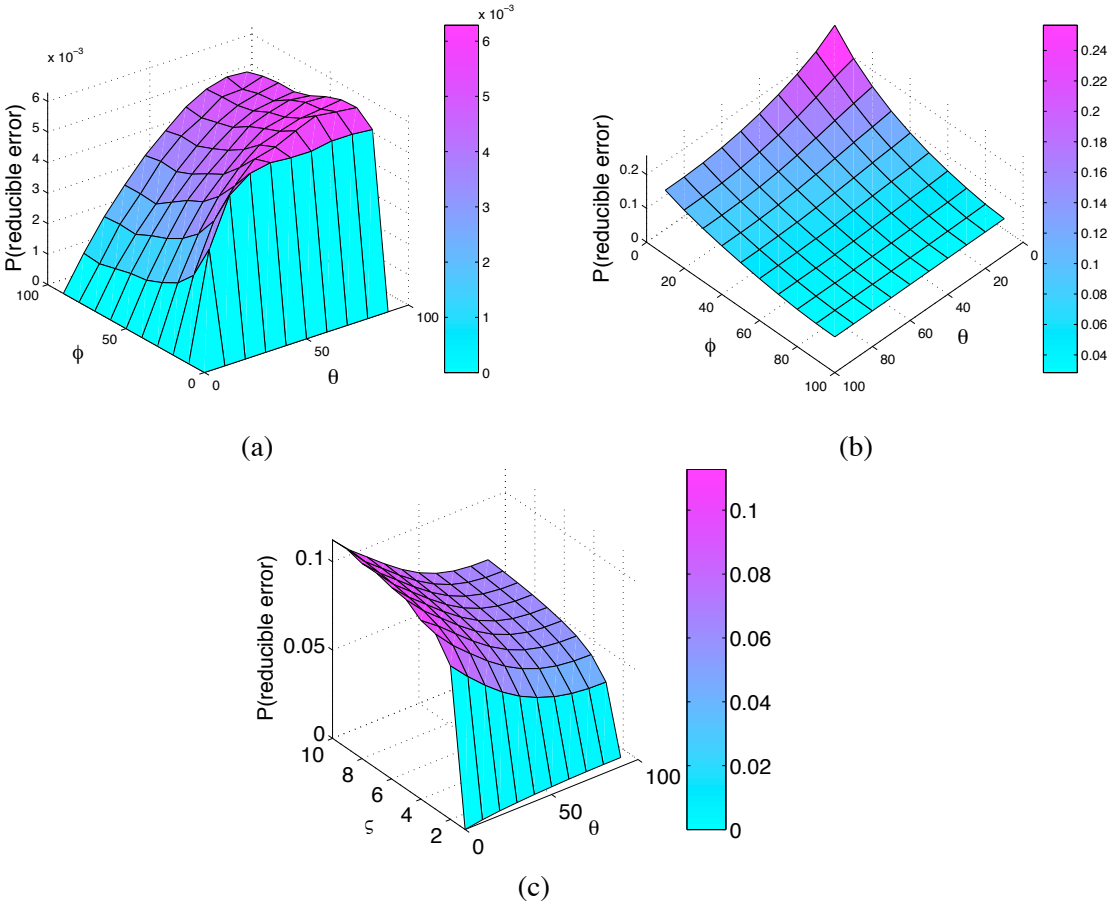


Figure 6: We show the average of the probability of reducible error over all the possible set of variance parameters when $\varsigma = 1$ in (a) and when $\varsigma \neq 1$ in (b). In (c) we show the increase of reducible error as the data deviates from spherical-homoscedastic (i.e., when ς increases). The more the data deviates from spherical-homoscedastic, the larger the reducible error is. This is independent of the closeness of the two distributions.

data starts to deviate from spherical-homoscedastic (i.e., when the rotation $\mathbf{R}$ does not define a planar rotation). Nonetheless, the probability of reducible error is still very small even for large values of ϕ and θ—approximately 0.006.[7]

When the scale parameter ς is not one, the two Bingham distributions B_1 and B_2 can never be spherical-homoscedastic. In this case, the probability of reducible error is generally expected to be larger. This is shown in Figure 6(b) where the probability of reducible error has been averaged over all possible combinations of variance parameters $(\lambda_1, \lambda_2, \lambda_3)$ and scales $\varsigma \neq 1$. Here, it is important to note that as the two original distributions get closer to each other the probability of reducible error increases quite rapidly. In fact, the error can be incremented by more than 20%. As in vMF,

7. Note that the plot shown in Figure 6(a) is not symmetric. This is due to the constraint given above ($\lambda_1 < \lambda_2 < \lambda_3$) which gives less flexibility to ϕ.

this means that if the data largely deviates from spherical-homoscedastic, extra caution needs to be taken with our results.

To further illustrate this point, we can plot the probability of reducible error over ς and θ, Figure 6(c). In this case, the larger ς is, the more different the eigenvalues of the parameter matrices ($\mathbf{A}_1$ and $\mathbf{A}_2$) will be. This means, that the larger the value of ς, the more the distributions deviate from spherical-homoscedastic. Hence, this plot shows the increase in reducible error as the two distributions deviate from spherical-homoscedastic. We note that this is in fact independent of how close the two distributions are, since the slop of the curve increases for every value of θ.

4.3 Modeling Spherical-Heteroscedastic Kent Distributions

Our final analysis involves the study of spherical-heteroscedastic Kent distributions. Here, we want to estimate the probability of reducible error when two Gaussian distributions $N_1(\mathbf{m}_1,\Sigma_1)$ and $N_2(\mathbf{m}_2,\Sigma_2)$ are used to model the data sampled from two Kent distributions $K_1(\mu_1,\kappa_1,\mathbf{A}_1)$ and $K_2(\mu_2,\kappa_2,\mathbf{A}_2)$. Recall that the shape of a Kent distribution is given by two parameters: β_i, which defines the ovalness of K_i, and κ_i, the concentration parameter. From Kent (1982) we know that if $2\beta_i/\kappa_i < 1$, then the normalizing constant $c(\kappa_i,\beta_i)$ can be approximated by $2\pi e_i^{\kappa}[(\kappa_i-2\beta_i)(\kappa_i+2\beta_i)]^{-1/2}$. Note that $2\beta_i/\kappa_i < 1$ holds regardless of the ovalness of our distribution when the data is highly concentrated, whereas in the case where the data is not concentrated (i.e., κ_i is small) the condition holds when the distribution is almost circular (i.e., β_i is very small).

To be able to use this approximation in our simulation, we have selected two sets of concentration parameters: one for the low concentration case (where $\kappa_i = \{2,3,\ldots,10\}$), and another where the data is more concentrated ($\kappa_i = \{15,20,\ldots,50\}$). The value of β_i is then given by the following set of equalities $2\beta_i/\kappa_i = \{0.1,0.3,\ldots,0.9\}$. As was done in the previous section (for the spherical-heteroscedastic Bingham), we fixed the mean direction μ_1 and then rotate μ_2 using a rotation matrix $\mathbf{R}$; that is, $\mu_2 = \mathbf{R}^T\mu_1$. To do this, we used the same rotation matrix $\mathbf{R}$ defined in (27). Now, however, $\mathbf{R}_1$ defines a planar rotation in the space spanned by μ_1 and the first eigenvector of $\mathbf{A}_1$, and $\mathbf{R}_2$ is a planar rotation defined in the space of μ_1 and the second eigenvector of $\mathbf{A}_1$. In our simulations we used $\{0^o,15^o,\ldots,90^o\}$ as values for the rotations defined by θ and ϕ.

In Figure 7(a), we show the results of our simulation for the special case where the variance parameters of K_1 and K_2 are the same (i.e., $\kappa_1 = \kappa_2$ and $\beta_1 = \beta_2$) and the data is not concentrated. Note that the criteria defined in (26) hold when either $\mathbf{R}_1$ or $\mathbf{R}_2$ is the identity matrix (or equivalently, in Figure 7(a), when θ or ϕ is zero). As anticipated in Theorem 11, in these cases the probability of reducible error is zero. Then the more these two distributions deviate from spherical-homoscedastic, the larger the probability of reducible error will become. It is worth mentioning, however, that the probability of reducible error is small over all possible values for θ and ϕ (i.e., < 0.035). We conclude (as with the analysis of the Bingham distribution comparison) that when the data only deviates from spherical-homoscedastic by a rotation (but the variances remain the same), the Gaussian approximation is a reasonable one. This means that whenever the parameters of the two distributions (Bingham or Kent) are defined up to a rotation, the results obtained using the Gaussian approximation will generally be acceptable.

The average of the probability of reducible error for all possible values for κ_i and β_i (including those where $\kappa_1 \neq \kappa_2$ and $\beta_1 \neq \beta_2$) is shown in Figure 7(b). In this case, we see that the probability of reducible error is bounded by 0.17 (i.e., 17%). Therefore, in the general case, unless the two

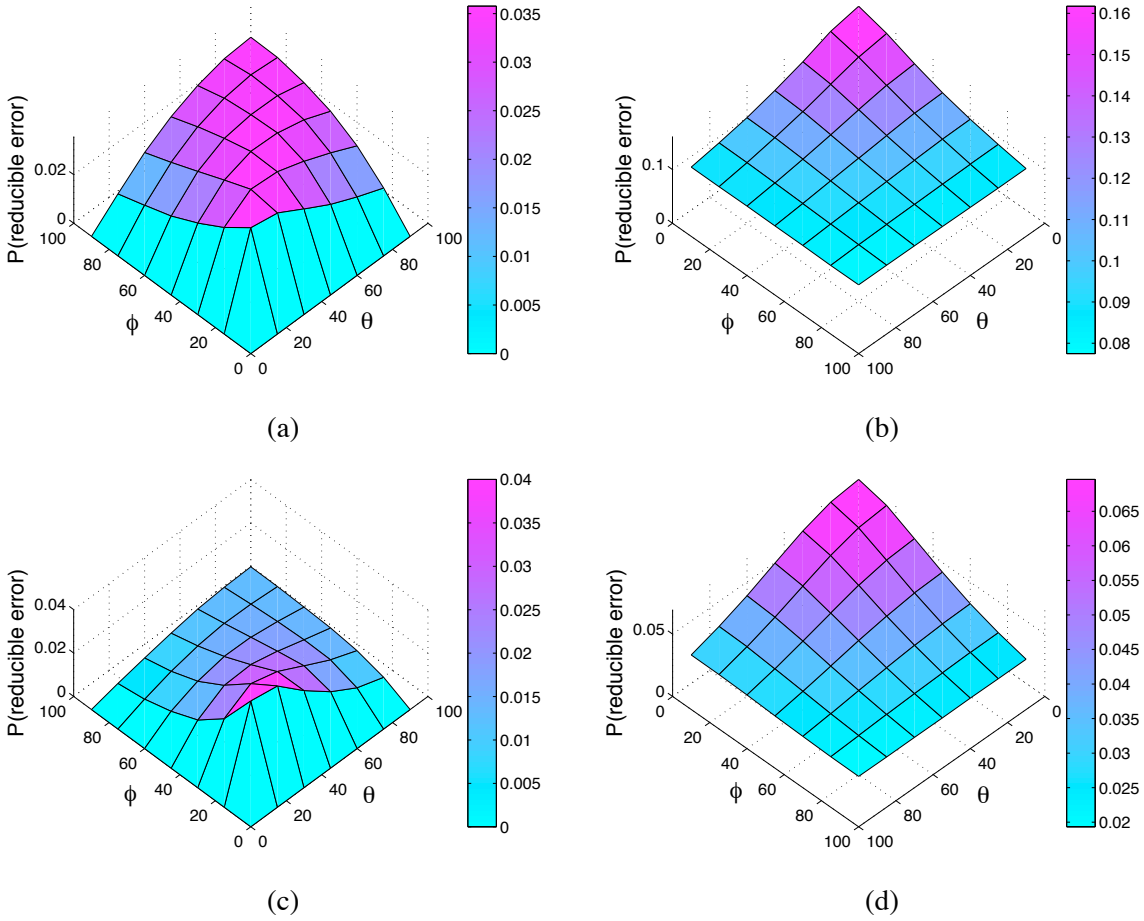


Figure 7: In (a) we show the average of the probability of reducible error when $\kappa_1 = \kappa_2$ and $\beta_1 = \beta_2$ and the data is not concentrated. (b) Shows the probability of reducible error when the parameters of the pdf are different in each distribution and the values of κ_i are small. (c-d) Do the same as (a) and (b) but for the cases where the concentration parameters are large (i.e., the data is concentrated).

original distributions are far away from each other, it is not advisable to model them using Gaussian distributions.

Figure 7(c-d) show exactly the same as (a-b) but for the case where the data is highly-concentrated. As expected, when the data is more concentrated in a small area, the probability of reducible error decreases fast as the two distributions fall far apart from each other. Similarly, since the data is concentrated, the maximum of the probability of reducible error shown in Figure 7(d) is smaller than that observed in (b).

As seen up to now, there are several conditions under which the Gaussian assumption is acceptable. In vMF, this happens when the distributions are highly concentrated, and in Bingham and Kent when the variance parameters of the distributions are the same.

Our next point relates to what can be done when neither of these assumptions hold. A powerful and generally used solution is to employ a kernel to (implicitly) map the original space to a high-dimensional one where the data can be better separated. Our next goal is thus to show that the results defined thus far are also applicable in the kernel space.

This procedure will provide us with a set of new classifiers (in the kernel space) that are based on the idea of spherical-homoscedastic distributions.

5. Kernel Spherical-Homoscedastic Classifiers

To relax the linear constraint stated in Definition 1, we will now employ the idea of the kernel trick, which will permit us to define classifiers that are nonlinear in the original space, but linear in the kernel one. This will be used to tackle the general spherical-heteroscedastic problem as if the distributions were linearly separable spherical-homoscedastic. Our goal is thus to find a kernel space where the classes adapt to this model.

We start our description for the case of vMF distributions. First, we define the sample mean direction of the first distribution $M_1(\mu,\kappa)$ as $\hat{\mu}_1$. The sample means of a set of spherical-homoscedastic distributions can be represented as rotated versions of this first one, that is, $\hat{\mu}_a = \mathbf{R}_a^T \hat{\mu}_1$, with $\mathbf{R}_a \in SO(p)$ and $a = \{2,\ldots,C\}, C$ the number of classes. Following this notation, we can derive the classification boundary between any pair of distributions from (13) as

$$\mathbf{x}^T(\hat{\mu}_a - \hat{\mu}_b) = 0, \quad \forall a \neq b.$$

The equation above directly implies that any new vector $\mathbf{x}$ will be classified to that class a for which the inner product between $\mathbf{x}$ and $\hat{\mu}_a$ is largest, that is,

$$\arg\max_a \mathbf{x}^T \hat{\mu}_a. \tag{28}$$

We are now in a position to provide a similar result in the feature space $\mathcal{F}$ obtained by the function $\phi(\mathbf{x})$, which maps the feature vector $\mathbf{x}$ from our original space S^{p-1} to a new spherical space S^d of d dimensions. In general, this can be described as a kernel $k(\mathbf{x}_i,\mathbf{x}_j)$, defined as the inner product of the two feature vectors in $\mathcal{F}$, that is, $k(\mathbf{x}_i,\mathbf{x}_j) = \phi(\mathbf{x}_i)^T\phi(\mathbf{x}_j)$.

Note that the mappings we are considering here are such that the resulting space $\mathcal{F}$ is also spherical. This is given by *all* those mappings for which the resulting norm of all vectors is a constant; that is, $\phi(\mathbf{x})^T\phi(\mathbf{x}) = h$, $h \in \mathbb{R}^+$.[8] In fact, many of the most popular kernels have such a property. This includes kernels such as the Radial Basis Function (RBF), polynomial, and Mahalanobis, when working in S^{p-1}, and several (e.g., RBF and Mahalanobis) when the original space is $\mathbb{R}^p$. This observation makes our results of more general interest yet, because even if the original space is not spherical, the use of some kernels will map the data into S^d. This will again require the use of spherical distributions or their Gaussian equivalences described in this paper.

In this new space $\mathcal{F}$, the sample mean direction of class a is given by

$$\begin{aligned}\hat{\mu}_a^\phi &= \frac{\frac{1}{n_a}\sum_{i=1}^{n_a}\phi(\mathbf{x}_i)}{\sqrt{\frac{1}{n_a}\sum_{i=1}^{n_a}\phi(\mathbf{x}_i)^T\frac{1}{n_a}\sum_{j=1}^{n_a}\phi(\mathbf{x}_j)}} \\ &= \frac{\frac{1}{n_a}\sum_{i=1}^{n_a}\phi(\mathbf{x}_i)}{\sqrt{\mathbf{1}^T\mathbf{K}\mathbf{1}}},\end{aligned}$$

8. Further, any kernel $k_1(\mathbf{x}_i,\mathbf{x}_j)$ can be defined to have this property by introducing the following simple normalization step $k(\mathbf{x}_i,\mathbf{x}_j) = k_1(\mathbf{x}_i,\mathbf{x}_j)/\sqrt{k_1(\mathbf{x}_i,\mathbf{x}_i)k_1(\mathbf{x}_j,\mathbf{x}_j)}$.

where $\mathbf{K}$ is a symmetric positive semidefinite matrix with elements $\mathbf{K}(i,j) = k(\mathbf{x}_i, \mathbf{x}_j)$, $\mathbf{1}$ is a vector with all elements equal to $1/n_a$, and n_a is the number of samples in class a.

By finding a kernel which transforms the original distributions to spherical-homoscedastic vMF, we can use the classifier defined in (28), which states that the class label of any test feature vector $\mathbf{x}$ is

$$\begin{aligned} \arg\max_a \phi(\mathbf{x})^T \hat{\mu}_a^\phi &= \frac{\frac{1}{n_a}\sum_{i=1}^{n_a} \phi(\mathbf{x})^T \phi(\mathbf{x}_i)}{\sqrt{\mathbf{1}^T\mathbf{K}\mathbf{1}}} \\ &= \frac{\frac{1}{n_a}\sum_{i=1}^{n_a} k(\mathbf{x}, \mathbf{x}_i)}{\sqrt{\mathbf{1}^T\mathbf{K}\mathbf{1}}}. \end{aligned} \tag{29}$$

Therefore, any classification problem that uses a kernel which converts the data to spherical-homoscedastic vMF distributions, can employ the solution derived in (29).

A similar result can be derived for Bingham distributions. We already know that the decision boundary for two spherical-homoscedastic Bingham distributions defined as $B_1(\mathbf{A})$ and $B_2(\mathbf{R}^T\mathbf{A}\mathbf{R})$, with $\mathbf{R}$ representing a planar rotation given by any two eigenvectors of $\mathbf{A}$, is given by the two hyperplane Equations (20) and (21). Since the rotation matrix is defined in a 2-dimensional space (i.e., planar rotation), one of the eigenvectors was described as a function of the other in the solution derived in Theorem 7. For classification purposes, this result will vary depending on which of the two eigenvectors of $\mathbf{A}$ we choose to use. To derive our solution we go back to (19) and rewrite it for classifying a new feature vector $\mathbf{x}$. That is, $\mathbf{x}$ will be classified in the first distribution, B_1, if the following holds

$$\lambda_1\left((\mathbf{x}^T\mathbf{q}_1)^2 - (\mathbf{x}^T\mathbf{R}^T\mathbf{q}_1)^2\right) + \lambda_2\left((\mathbf{x}^T\mathbf{q}_2)^2 - (\mathbf{x}^T\mathbf{R}^T\mathbf{q}_2)^2\right) > 0.$$

Using the result shown in Theorem 7, where we expressed $\mathbf{q}_2$ as a function of $\mathbf{q}_1$, we can simplify the above equation to

$$(\lambda_1 - \lambda_2)\left((\mathbf{x}^T\mathbf{q}_1)^2 - (\mathbf{x}^T\mathbf{R}^T\mathbf{q}_1)^2\right) > 0.$$

Then, if $\lambda_1 > \lambda_2$, $\mathbf{x}$ will be in B_1 when

$$(\mathbf{x}^T\mathbf{q}_1)^2 > (\mathbf{x}^T\mathbf{R}^T\mathbf{q}_1)^2,$$

which can be simplified to

$$|\mathbf{x}^T\mathbf{q}_1| > |\mathbf{x}^T\mathbf{R}^T\mathbf{q}_1|. \tag{30}$$

If this condition does not hold, $\mathbf{x}$ is classified as a sample of B_2. Also, if $\lambda_2 > \lambda_1$, the reverse applies. In the following, and without loss of generality, we will always assume $\mathbf{q}_1$ corresponds to the eigenvector of $\mathbf{A}$ defining the rotation plane of $\mathbf{R}$ that is associated to the largest of the two eigenvalues. This means that whenever (30) holds, $\mathbf{x}$ is classified in B_1.

The relevance of (30) is that (as in vMF), a test feature vector $\mathbf{x}$ is classified to that class providing the largest inner product value. We can now readily extend this result to the multi-class problem. For this, let $B_1(\mathbf{A})$ be the distribution of the first class and $B_a(\mathbf{R}_a^T\mathbf{A}\mathbf{R}_a)$ that of the a^{th} class, where now $\mathbf{R}_a$ is defined by two eigenvectors of $\mathbf{A}$, $\mathbf{q}_{a_1}$ and $\mathbf{q}_{a_2}$, with corresponding eigenvalues λ_{a_1} and λ_{a_2} and we have assumed $\lambda_{a_1} > \lambda_{a_2}$. Then, the class of a new test feature vector $\mathbf{x}$ is given by

$$\arg\max_a |\mathbf{x}^T\mathbf{q}_{a_1}|. \tag{31}$$

Following Theorem 7, we note that not all the rotations $\mathbf{R}_a$ should actually be considered, because some may result in two distributions B_a and B_b that are not spherical-homoscedastic. To see this, consider the case with three distributions, $B_1(\mathbf{A})$, $B_2(\mathbf{R}_2^T\mathbf{A}\mathbf{R}_2)$ and $B_3(\mathbf{R}_3^T\mathbf{A}\mathbf{R}_3)$. In this case, B_1 and B_2 will always be spherical-homoscedastic if $\mathbf{R}_2$ is defined in the plane spanned by two of the eigenvectors of $\mathbf{A}$. The same applies to B_1 and B_3. However, even when $\mathbf{R}_2$ and $\mathbf{R}_3$ are defined by two eigenvectors of the parameter matrix, the rotation between B_2 and B_3 may not be planar. Nonetheless, we have shown in Section 4 that if the variances of the two distributions are the same up to an arbitrary rotation, the reducible error is negligible. Therefore, and since imposing additional constraints would make it very difficult to find a kernel that can map the original distributions to spherical-homoscedastic, we will consider all rotations about every $\mathbf{q}_{a_1}$.

We see that (31) still restricts the eigenvectors $\mathbf{q}_{a_1}$ to be rotated versions of one another. This constraint comes from (30), where the eigenvector of the second distribution must be the first eigenvector rotated by the rotation matrix relating the two distributions. Since the rotation is a rigid transformation, all $\mathbf{q}_{a_1}$ will be defined by the same index i in $\hat{\mathbf{q}}_{a_i}$, where $\widehat{\mathbf{Q}}_a = \{\hat{\mathbf{q}}_{a_1}, \ldots, \hat{\mathbf{q}}_{a_p}\}$ are the eigenvectors of the autocorrelation matrix $\mathbf{S}_a$ of the a^{th} class. This equivalency comes from Section 2.2, where we saw that the eigenvectors of $\mathbf{A}_a$ are the same as those of the autocorrelation matrix. Also, since we know the data is spherical-homoscedastic, the eigenvectors of the correlation matrix will be the same as those of the covariance matrix Σ_a of the (zero-mean) Gaussian distribution as seen in Theorem 9.

Our next step is to derive the same classifier in the kernel space. From our discussion above, we require to find the eigenvectors of the covariance matrix. The covariance matrix in $\mathcal{F}$ can be computed as

$$\Sigma_a^\Phi = \Phi(\mathbf{X}_a)\Phi(\mathbf{X}_a)^T,$$

where $\mathbf{X}_a$ is a matrix whose columns are the sample feature vectors, $\mathbf{X}_a = \left(\mathbf{x}_{a_1}, \mathbf{x}_{a_2}, \ldots, \mathbf{x}_{a_{n_a}}\right)$, and $\Phi(\mathbf{X})$ is a function which maps the columns $\mathbf{x}_i$ of $\mathbf{X}$ with $\phi(\mathbf{x}_i)$.

This allows us to obtain the eigenvectors of the covariance matrix from

$$\Sigma_a^\Phi \mathbf{V}_a^\Phi = \mathbf{V}_a^\Phi \Lambda_a^\Phi.$$

Further, these d-dimensional eigenvectors $\mathbf{V}_a^\Phi = \{\mathbf{v}_{a_1}^\Phi, \ldots, \mathbf{v}_{a_d}^\Phi\}$ are not only the same as those of $\mathbf{A}_a^\Phi$, but, as shown in Theobald (1975), are also sorted in the same order.

As pointed out before though, the eigenvalue decomposition equation shown above may be defined in a very high dimensional space. A usual way to simplify the computation is to employ the kernel trick. Here, note that since we only have n_a samples in class a, $rank(\Lambda_a^\Phi) \leq n_a$. This allows us to write $\mathbf{V}_a^\Phi = \Phi(\mathbf{X}_a)\Delta_a$, where Δ_a is a $n_a \times n_a$ coefficient matrix, and thus the above eigenvalue decomposition equation can be stated as

$$\Phi(\mathbf{X}_a)\Phi(\mathbf{X}_a)^T\Phi(\mathbf{X}_a)\Delta_a = \Phi(\mathbf{X}_a)\Delta_a\Lambda_a^\Phi.$$

Multiplying both sides by $\Phi(\mathbf{X}_a)^T$ and cancelling terms, we can simplify this equation to

$$\begin{aligned} \Phi(\mathbf{X}_a)^T\Phi(\mathbf{X}_a)\Delta_a &= \Delta_a\Lambda_a^\Phi, \\ \mathbf{K}_a\Delta_a &= \Delta_a\Lambda_a^\Phi, \end{aligned}$$

where $\mathbf{K}_a$ is known as the Gram matrix.

We should now be able to obtain the eigenvectors in $\mathcal{F}$ using the equality

$$\widehat{\mathbf{V}}_a^{\Phi} = \Phi(\mathbf{X}_a)\Delta_a.$$

However, the norm of the vectors $\widehat{\mathbf{V}}_a^{\Phi}$ thus obtained is not one, but rather

$$\Lambda_a^{\Phi} = \Delta_a^T \Phi(\mathbf{X}_a)^T \Phi(\mathbf{X}_a)\Delta_a.$$

To obtain the (unit-norm) eigenvectors, we need to include a normalization coefficient into our result,

$$\mathbf{V}_a^{\Phi} = \Phi(\mathbf{X}_a)\Delta_a \Lambda_a^{\Phi^{-1/2}},$$

where $\mathbf{V}_a^{\Phi} = \{\mathbf{v}_{a_1}^{\phi}, \ldots, \mathbf{v}_{a_{na}}^{\phi}\}$, and $\mathbf{v}_{a_i}^{\phi} \in S^d$.

The classification scheme derived in (31) can now be extended to classify $\phi(\mathbf{x})$ as

$$\arg\max_a |\phi(\mathbf{x})^T \mathbf{v}_{a_i}^{\phi}|,$$

where the index $i = \{1, \ldots, p\}$ defining the eigenvector $\mathbf{v}_{a_i}^{\phi}$ must be kept constant for all a.

The result derived above, can be written using a kernel as

$$\arg\max_a \left| \sum_{l=1}^{n_a} \frac{k(\mathbf{x}, \mathbf{x}_l)\delta_{a_i}(l)}{\sqrt{\lambda_{a_i}^{\phi}}} \right|, \tag{32}$$

where $\Delta_a = \{\delta_{a_1}, \ldots, \delta_{a_{na}}\}$, $\delta_{a_i}(l)$ is the l^{th} coefficient of the vector δ_{a_i}, and again i takes a value from the set $\{1, \ldots, p\}$ but otherwise kept constant for all a.

We note that there is an important difference between the classifiers derived in this section for vMF in (29) and for Bingham in (32). While in vMF we are only required to optimize the kernel responsible to map the spherical-heteroscedastic data to one that adapts to spherical-homoscedasticity, in Bingham we will also require the optimization of the eigenvector (associated to the largest eigenvalue) $\mathbf{v}_{a_i}^{\phi}$ defining the rotation matrix $\mathbf{R}_a$. This is because there are many different solutions which can convert a set of Bingham distributions into spherical-homoscedastic.

To conclude this section, we turn to the derivations of a classifier for the Kent distribution in the kernel space. From Theorem 10, the 2-class classifier can be written as

$$\begin{aligned} \mathbf{x}^T(\mu_a - \mu_b) &> 0, \\ \mathbf{x}^T(\mu_a + \mu_b) - \frac{\kappa}{\lambda_{a_1}} &> 0. \end{aligned}$$

The first of these equations is the same as that used to derive the vMF classifier, and will therefore lead to the same classification result. Also, as seen in Theorem 11 the second equation can be eliminated when either: *i)* the second hyperplane is identical to the first, or *ii)* the second hyperplane is outside S^{p-1}. Any other case should actually not be considered, since this would not guarantee the equality of the Gaussian model. Therefore, and rather surprisingly, we conclude that the Kent classifier in the kernel space will be the same as that derived by the vMF distribution. The rational behind this is, however, quite simple, and it is due to the assumption of the concentration of the data made in the Kent distribution: Since the parameters of the Kent distributions are estimated in the tangent space of the mean direction, the resulting classifier should only use this information. This is exactly the solution derived in (29).

6. Experimental Results

In this section we show how the results reported in this paper can be used in real applications. In particular, we apply our results to the classification of text data, genomics, and object classification. Before we get to these though, we need to address the problems caused by noise and limited number of samples. We start by looking at the problem of estimating the parameters of the Gaussian fit of a spherical-homoscedastic distribution from a limited number of samples and how this may effect the equivalency results of Theorems 6, 9 and 11.

6.1 Finite Sample Set

Theorems 6, 9 and 11 showed that the classifiers separating two spherical-homoscedastic distributions are the same as those of the corresponding Gaussian fit. This assumes, however, that the parameters of the spherical distributions are known. In the applications to be presented in this section, we will need to estimate the parameters of such distributions from a limited number of sample feature vectors. The question to be addressed here is to what extent this may effect the classification results obtained with the best Gaussian fit. Following the notation used above, we are interested in finding the *reducible error* that will (on average) be added to the classification error when using the Gaussian model.

To study this problem, we ran a set of simulations. We started by drawing samples from two underlying spherical-homoscedastic distributions that are hidden (unknown) to the classifier. Then, we estimated the mean and covariance matrix defining the Gaussian distribution of the samples in each of the two classes. New test feature vectors were randomly drawn from each of the two (underlying) distributions and classified using the log-likelihood equations defined in Section 2.3. The percentage of samples misclassified by the Gaussian model (i.e., using 4) as opposed to the correct classification given by the corresponding spherical model (i.e., Equations 5-7), determines the probability of reducible error.

Our spherical-homoscedastic vMF simulation used the following concentration parameters $\kappa_1 = \kappa_2 = \{1,2,\ldots,10\}$, and the following rotations between distributions $\theta = \{10,20,\ldots,180\}$. The number of samples randomly drawn from each distribution varied from 10 times the dimensionality to 100 times that value, at increments of ten. The dimensionality was tested for the following values $p = \{2,10,20,30,40,50\}$. Our experiment was repeated 100 times for each of the possible parameters, and the average was computed to obtain the probability of reducible error.

Fig. 8(a)-(b) show these results. In (a) the probability of reducible error is shown over the dimensionality of the data p and the scalar γ, which is defined as the number of samples over their dimensionality, $\gamma = n/p$. As already noticed in Section 4, the probability of reducible error decreases with the dimensionality. This is the case since the volume of the distribution is taken over a larger number of dimensions, forcing the overlapping area to shrink. As the scalar γ increases, so does the number of samples n. In those cases, if p is small, the probability of the reducible error decreases. However, when the dimensionality is large, this probability increases at first. In short, we are compensating the loss in volume caused by the increase in dimensionality, by including additional samples. When the dimension to sample ratio is adequate, the probability of reducible error decreases as expected. The decreasing rate for larger dimensions is, however, much slower. This result calls for a reduction of dimensionality from S^{p-1} to a hypersphere where the scalar γ is not too large. This will be addressed in our next section.

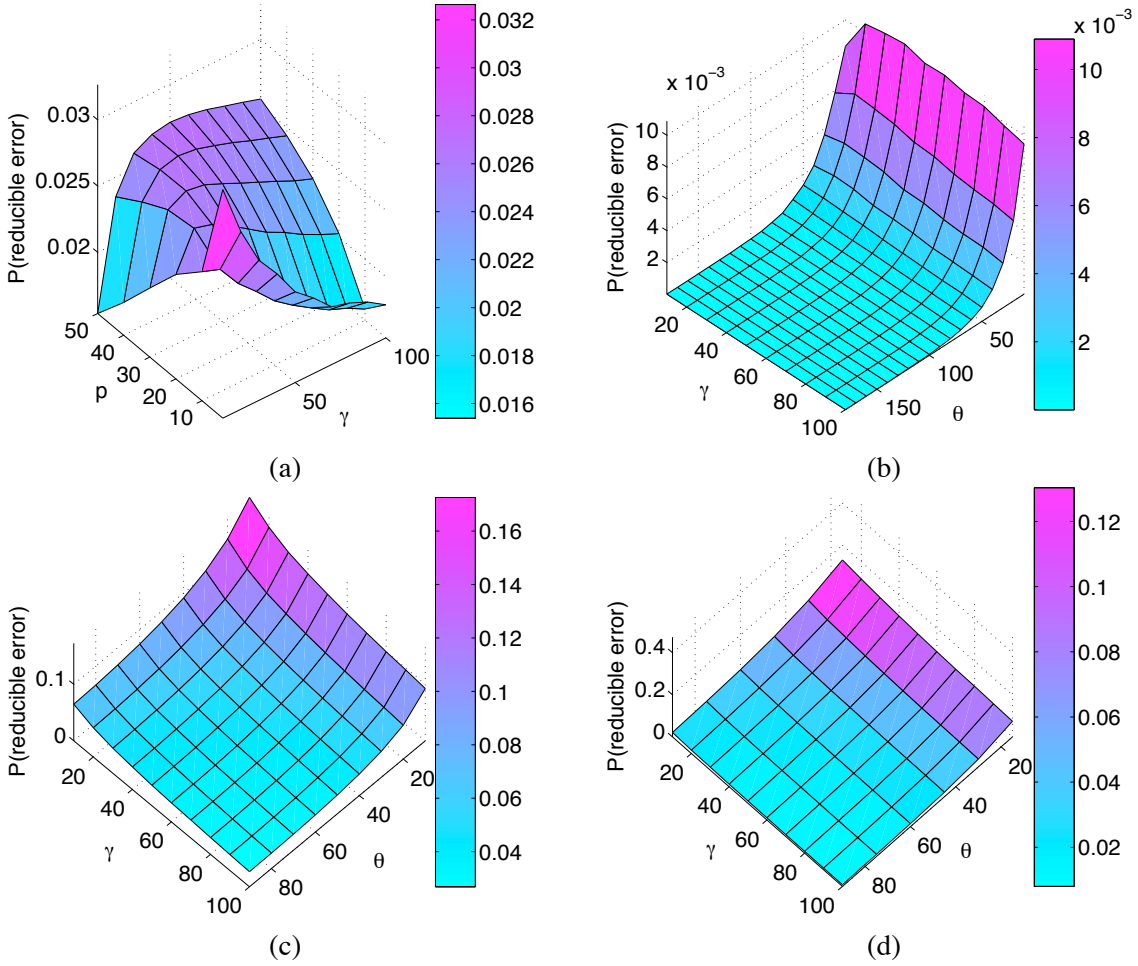


Figure 8: (a-b) Shown here are the average reducible errors obtained when fitting two Gaussian distributions to the data sampled from two spherical-homoscedastic vMFs with parameters $\kappa = \{1,2,\ldots,10\}$ and $\theta = \{10,20,\ldots,180\}$. The number of samples is $n = \gamma p$, where γ is a constant taking values in $\{10,20,\ldots,100\}$, and p is the dimensionality with values $p = \{2,10,20,\ldots,50\}$. (c) Shows the average reducible error when data sampled from spherical-homoscedastic Bingham distributions is fitted with the Gaussians estimates. In this case, $p = 3$ and the number of samples as defined above. (d) Summarizes the same simulation for spherical-homoscedastic Kent distributions, with $p = 3$.

We also note that in the worse case scenario, the probability of reducible error is very low and shall not have a dramatic effect in the classification results when working with vMF.

In Fig. 8(b), we see that the closer two vMF distributions get, the larger the reducible error can be. This is because the closer the distributions, the larger the overlap. Once more, however, this error is negligible. With this, we can conclude that the parameters of the vMF can very reliably be estimated with the corresponding Gaussian fit described in this paper even if the number of samples is limited. This is especially true when the dimensionality of the data is relatively low.

To simulate the probability of reducible error in Bingham and Kent, several of the parameters of the distributions were considered. For simplicity, we show the results obtained in the three-dimensional case, $p = 3$. Similar results were observed in larger dimensions (e.g., 10 and 50). The number of samples $n = \gamma p$, was determined as above by varying the values of the scalar, $\gamma = \{10, 20, \ldots, 100\}$. Recall that in both distributions, spherical-homoscedasticity is satisfied when $\theta_1 = 0$ and $\theta_2 = \{10, 20, \ldots, 90\}$, and when $\theta_2 = 0$ and $\theta_1 = \{10, 20, \ldots, 90\}$. The parameter matrix of the second distribution $\mathbf{A}_2$ can be defined as a rotated version of $\mathbf{A}_1$ as $\mathbf{A}_2 = \mathbf{R}^T \mathbf{A}_1 \mathbf{R}$, where $\mathbf{R}$ is given by (27). Further, in Kent, we impose the concentration and skewness parameters to be equal in both distributions, that is, $\kappa_1 = \kappa_2$ and $\beta_1 = \beta_2$.

In Fig. 8(c-d) we show the probability of reducible error as a function of γ and the rotation θ. In both cases, and as expected, when the two distributions get closer, the error increases. In Bingham, when the sample to dimensionality ratio decreases, the error slightly increases (to about 6%). But, in Kent, as in vMF, reducing the number of samples produces a negligible effect. Therefore, as in vMF, we conclude that, in the worst case scenarios, keeping a ratio of 10 samples per dimension, results in good approximations. When the distributions are not in tight proximity, this number can be smaller. These observations bring us to our next topic: how to project the data onto a feature space where the ratio sample-to-dimensionality is adequate.

6.2 Subsphere Projection

As demonstrated in Section 6.1, a major concern in fitting a distribution model to a data set is the limited amount of samples available. This problem is exacerbated when the dimensionality of the data p surpasses the number of samples n. This problem is usually referred to as the curse of dimensionality. In such circumstances, the classifier can easily overfit and lead to poor classification results on an independent set of observations. A typical technique employed to mitigate this problem, is the use of Principal Components Analysis (PCA). By projecting the sample vectors onto the subspace defined by the PCs of largest data variance, we can eliminate data noise and force independent test vectors to be classified in the subspace defined by the training data. This procedure is typically carried out in $\mathbb{R}^p$ by means of an eigendecomposition of the covariance matrix of the training data. In this section, we show how to employ PCA on the correlation matrix to do the same in S^{p-1}.

This problem can be easily stated as follows: We want to find a representation $\widetilde{\mathbf{x}}_i \in S^{r-1}$ of the original feature vectors $\mathbf{x}_i \in S^{p-1}$, $r \leq p$, with minimal loss of information. This can be done with a $p \times r$ orthogonal projection matrix $\mathbf{W}$, that is, $\mathbf{W}^T\mathbf{W} = \mathbf{I}$. The least-squares solution to this problem is then given by

$$\arg\min_{\mathbf{W}} \sum_{i=1}^{n} (\mathbf{x}_i - \mathbf{W}\widetilde{\mathbf{x}}_i)^T (\mathbf{x}_i - \mathbf{W}\widetilde{\mathbf{x}}_i), \tag{33}$$

where $\widetilde{\mathbf{x}}_i = \mathbf{W}^T\mathbf{x}_i / \|\mathbf{W}^T\mathbf{x}_i\|$, that is, the unit-length vector represented in S^{r-1}.

Note that as opposed to PCA in the Euclidean space, our solution requires the subspace vectors to have unit length, since these are also in a hypersphere. To resolve this, we can first search for that

projection which minimizes the projection error,

$$\begin{aligned}
& \arg\min_{\mathbf{W}} \sum_{i=1}^{n} (\mathbf{x}_i - \mathbf{W}\mathbf{W}^T\mathbf{x}_i)^T(\mathbf{x}_i - \mathbf{W}\mathbf{W}^T\mathbf{x}_i) \\
= & \arg\min_{\mathbf{W}} \sum_{i=1}^{n} (\mathbf{x}_i^T\mathbf{x}_i - 2\mathbf{x}_i^T\mathbf{W}\mathbf{W}^T\mathbf{x}_i + \mathbf{x}_i^T\mathbf{W}\mathbf{W}^T\mathbf{W}\mathbf{W}^T\mathbf{x}_i) \\
= & \arg\min_{\mathbf{W}} \sum_{i=1}^{n} (1 - \|\mathbf{W}^T\mathbf{x}_i\|^2) \\
= & \arg\max_{\mathbf{W}} \sum_{i=1}^{n} \|\mathbf{W}^T\mathbf{x}_i\|^2 \\
= & \arg\max_{\mathbf{W}} \mathbf{W}^T \sum_{i=1}^{n} \mathbf{x}_i\mathbf{x}_i^T\mathbf{W} \\
= & \arg\max_{\mathbf{W}} \mathbf{W}^T\mathbf{S}\mathbf{W},
\end{aligned}$$

where $\mathbf{S}$ is the sample correlation matrix. Then, the resulting data vectors need to be normalized to unit length to produce the final result.

To justify this solution, note that the direct minimization of (33) would result in

$$\begin{aligned}
\arg\min_{\mathbf{W}} \sum_{i=1}^{n} \left(\mathbf{x}_i^T\mathbf{x}_i - 2\frac{\mathbf{x}_i^T\mathbf{W}\mathbf{W}^T\mathbf{x}_i}{\|\mathbf{W}^T\mathbf{x}_i\|} + \widetilde{\mathbf{x}}_i^T\widetilde{\mathbf{x}}_i \right) &= \arg\min_{\mathbf{W}} \sum_{i=1}^{n} \left(2 - \frac{2\mathbf{x}_i^T\mathbf{W}\mathbf{W}^T\mathbf{x}_i}{\|\mathbf{W}^T\mathbf{x}_i\|} \right) \\
&= \arg\min_{\mathbf{W}} \sum_{i=1}^{n} \left(1 - \|\mathbf{W}^T\mathbf{x}_i\| \right) \\
&= \arg\max_{\mathbf{W}} \sum_{i=1}^{n} \|\mathbf{W}^T\mathbf{x}_i\|. \qquad (34)
\end{aligned}$$

Since we are only interested in eliminating those dimensions of S^{p-1} that are close to zero and $\|\mathbf{x}_i\| = 1$, it follows that $\|\mathbf{W}^T\mathbf{x}_i\| \cong \|\mathbf{W}^T\mathbf{x}_i\|^2$. Further, because (34) may not generate unit length vectors $\widetilde{\mathbf{x}}_i$, this will require a normalization step.

The projection to a lower dimensional sphere S^{r-1} minimizing the least-squares error can hence be carried out in two simple steps. First, project the data onto the subspace that keeps most of the variance as defined by the autocorrelation matrix, that is, $\check{\mathbf{x}}_i = \mathbf{W}^T\mathbf{x}_i$, where $\mathbf{S}\mathbf{W} = \mathbf{W}\Lambda$. Second, normalize the vectors to produce the following final result $\widetilde{\mathbf{x}}_i = \check{\mathbf{x}}_i / \|\check{\mathbf{x}}_i\|$.

6.3 Object Categorization

As already mentioned in the Introduction, norm normalization is a typical pre-processing step in many systems, including object recognition and categorization. In the latter problem, images of objects need to be classified according to a set of pre-defined categories, for example, cows and cars. A commonly used database for testing such systems is the ETH-80 data set of Leibe and Schiele (2003). This database includes the images of eight categories: apples, cars, cows, cups, dogs, horses, pears and tomatoes. Each of these consists of a set of images representing ten different objects (e.g., ten cars) photographed at a total of 41 orientations. This means that we have a total of 410 images per category.

In an attempt to emulate the primary visual areas of the human visual system, many feature representations of objects are based on the outputs of a set of filters corresponding to the derivatives of the Gaussian distribution at different scales. In our experiments, we consider the first derivative about the x and y axes for a total of three different scales, that is, $\upsilon = \{1,2,4\}$, where υ is the variance of the Gaussian filter. The convolution of each of these filters with the input images produces a different result, which means we will generate a total of six images. Next, we compute the histogram of each of these resulting images. This histogram representation is simplified to 32 intensity intervals. These histograms are generally assumed to be the distribution of the pixel values in the image and, therefore, the sum of the values over all 32 intervals should be one (because the integral of a density is 1). Hence, these 32 values are first represented in vector form and then normalized to have unit norm. The resulting unit-norm feature vectors are concatenated to form a single feature representation. This generates a feature space of 192 dimensions. We refer to this feature representation as *Gauss*.

As an alternative, we also experimented with the use of the magnitude of the gradient and the Laplacian operator. The latter generally used for its rotation-invariant properties, which is provided by its symmetric property. As above, the gradient and Laplacian are computed using three scales, that is, $\upsilon = \{1,2,4\}$. To reduce the number of features, we use the histogram representation described above, which again produces unit-norm feature vectors of 192 dimensions. This second representation will be referred to as *MagLap*.

These two image representations are tested using the leave-one-object-out strategy, where, at each iteration, we leave all the images of one of the objects out for testing and use the remaining for training. This is repeated (iterated) for each of the possible objects that one can leave out.

We now use the PCA procedure described in Section 6.2 to map the data of each of the two feature spaces described above onto the surface of an 18-dimensional sphere. This dimensionality was chosen such that 99.9% of the data variance was kept. Note that in this subsphere, the sample to dimensionality ratio (which is about 180) is more than adequate for classification purposes as demonstrated in Section 6.1.

To generate our results, we first use the vMF and Bingham approximations introduced in Section 2.2 and then use (5) and (6) for classification. We refer to these approaches as *vMF* and *Bingham classifiers*, respectively. These are the classifiers one can construct based on the available approximation defined to estimate the parameters of spherical distributions.

Next, we consider the Gaussian equivalency results shown in this paper, which allow us to represent the data with Gaussian distributions and then use (4) for classification. This algorithm will be labeled *Gaussian classifier* in our tables.

In Section 5, we derived three algorithms for classifying spherical-homoscedastic vMF, Bingham, and Kent. These were obtained from Theorems 3, 7 and 10. We also showed how the vMF and Kent classifiers were identical. These classifiers were given in (28) and (31), and will be labeled *SH-vMF* and *SH-Bingham*, respectively.

Table 1 shows the average of the recognition rates using the leave-one-object out with each of the algorithms just mentioned. In this table, we have also included the results we would obtained with the classical Fisher (1938) Linear Discriminant Analysis algorithm (LDA). This algorithm is typically used in classification problems in computer vision, genomics and other machine learning applications. In LDA, the data of each class is approximated with a Gaussian distribution, but only the average of these, $\bar{\Sigma}$, is subsequently used. This is combined with the scatter-matrix of the class means $\mathbf{S}_B$ to generate the LDA basis vectors $\mathbf{U}$ from $\bar{\Sigma}^{-1}\mathbf{S}_B\mathbf{U} = \mathbf{U}\Lambda_{LDA}$; where $\mathbf{S}_B$ is formally

Method	*vMF*	*Bingham*	*Gaussian*	*SH-vMF*	*SH-Bingham*	*LDA*
Gauss	13.75	73.11	73.14	45.85	46.16	62.9
MagLap	12.5	74.18	73.75	51.95	52.90	66.25

Table 1: Average classification rates for the ETH database with *Gauss* and *MagLap* feature representations.

defined as $\mathbf{S}_B = \sum_{i=1}^{C} n_i/n\,(\hat{\mu}_i - \hat{\mu})\,(\hat{\mu}_i - \hat{\mu})^T$, n_i is the number of samples in class i, $\hat{\mu}_i$ the mean of these samples, and $\hat{\mu}$ the global mean. In LDA, the nearest mean classifier is generally used in the subspace spanned by the eigenvectors of $\mathbf{U}$ associated to non-zero variance.[9]

It is clear that, in this particular case, the vMF model is too simplistic to successfully represent our data. In comparison, the Bingham model provides a sufficient degree of variability to fit the data.

Next, we turn to the results of the Gaussian fit defined in this paper, Table 1. We see that the results are comparable to those obtained with the Bingham model. A simple data analysis reveals that the data of all classes is highly concentrated. We see this by looking at the eigenvalues of each class scatter matrix $\mathbf{S}_a$, $a = \{1,\ldots,C\}$. The average of the variance ratios between the first eigenvector (defining the mean direction) and the second eigenvector (representing the largest variance of the data on S^{d-1}), which is 66.818. From the results obtained in Section 4, we expected the Gaussian fit to perform similarly to Bingham under such circumstances. This is clearly the case in the classification results shown in Table 1.

While the Gaussian fit has proven adequate to represent the data, the spherical-homosce-dastic classifiers derived in Section 5 performed worse. This is because the class distributions are far from spherical-homoscedastic. We can see that by studying the variability of the variance in each of the eigenvectors of the class distributions. To do this, we look at the variance $\hat{\lambda}_{a_i}$ about each eigenvector $\hat{\mathbf{q}}_{a_i}$. If the data were spherical-homoscedastic, the variances about the corresponding eigenvectors should be identical, that is, $\hat{\lambda}_{a_i} = \hat{\lambda}_{b_i}$, $\forall a,b$ (recall a and b are the labels of any two distributions). Seemingly, the more the class distributions deviate from spherical-homoscedastic, the larger the difference between $\hat{\lambda}_{a_i}$ and $\hat{\lambda}_{b_i}$ will be. The percentage of variability among the variances $\hat{\lambda}_{a_i}$ for different a, can be computed as

$$100\,\frac{stdv\{\hat{\lambda}_{1_i},\ldots,\hat{\lambda}_{C_i}\}}{\frac{1}{C}\sum_{j=1}^{C}\hat{\lambda}_{j_i}}, \tag{35}$$

where $stdv\{\cdot\}$ is the standard deviation of the values of the specified set. Hence, a 0% variability would correspond to perfect spherical-homoscedastic distributions. The more we deviate from this value, the more the distributions will deviate from the spherical-homoscedastic model.

Applying (35) to the resulting class distributions of the ETH data set with the *Gauss* representation and then computing the mean of all resulting differences yields 58.04%. The same can be done for the *MagLap* representation, resulting in an average variability of 76.51%. In these two cases, we clearly see that the class distributions deviate considerably from spherical-homoscedastic. As demonstrated in Figure 6(c). The effects of heteroscedasticity are further observed in the low performances of the LDA algorithm, which assumes the data is homoscedastic.

9. Note that the rank of $\mathbf{U}$ is upper-bounded by $C-1$, because the scatter-matrix matrix $\mathbf{S}_B$ is defined by the $C-1$ vectors interconnecting the C class means.

Method	*K-SH-vMF*	*K-SH-Bingham*
Gauss	79.24 ($\bar{\varsigma} = 3.2$)	78.84 ($\bar{\varsigma} = 3.96$, $\bar{\nu} = 1$)
MagLap	77.23 ($\bar{\varsigma} = 5.63$)	77.16 ($\bar{\varsigma} = 6.35$, $\bar{\nu} = 1$)

Table 2: Average classification rates for the ETH database with *Gauss* and *MagLap* feature representations using the nonlinear spherical-homoscedastic vMF and Bingham classifiers. In these results, we used the Mahalanobis kernel.

As shown in Section 5, we can improve the results of our spherical-homoscedastic classifiers by first (intrinsically) mapping the data into a space where the underlying class distributions fit to the spherical-homoscedastic model.

Since the original class distributions need to be reshaped to fit the spherical-homoscedastic model, the Mahalanobis kernel is a convenient choice. This kernel is defined as

$$k(\mathbf{x},\mathbf{y}) = \exp\left(\frac{-(\mathbf{x}-\mathbf{y})^T \bar{\Sigma}^{-1}(\mathbf{x}-\mathbf{y})}{2\varsigma^2}\right)$$

where ς is a scale parameter to be optimized using the leave-one-object-out test (looot) on the training set. This looot is not to be confused with the object left out for testing, which is used to compute the classification error. In our experimental results, we tested the following scalar values for the kernel, $\varsigma = \{1,2,\ldots,10\}$.

Table 2 shows the recognition rates for the (nonlinear) spherical-homoscedastic vMF and Bingham classifiers, using the Mahalanobis kernel just defined. These classifiers are labeled, *K-SH-vMF* and *K-SH-Bingham*, respectively. The classification rates are shown in percentages. The values in parentheses indicate the average over all scale parameters as determined by looot. This includes the average scalar kernel parameter, denoted $\bar{\varsigma}$, for vMF and Bingham, and the average value of i in $\mathbf{q}_{a_i}$, denoted $\bar{\nu}$, when optimizing the Bingham classifier defined in (32). We see that the classification results are boosted beyond those obtained with Bingham and the Gaussian fit.

6.4 Gene Expression Classification

Spherical representations can be applied to gene expression data in a variety of ways. For example, some genome sequences are circular, and the gene co-orientation as well as its expression are relevant (Audit and Ouzounis, 2003; Janssen et al., 2001). In analyzing other genomes or the expression of a set of genes from microarrays, correlation coefficients (e.g., Pearson) provide a common way to determine the similarity between samples (Banerjee et al., 2005). These are generally norm invariant and, hence, whenever density estimations are required, spherical models are necessary.

To demonstrate this application, we have used the gene expression data set A of Pomeroy et al. (2002). This data set consists of five different classes of tumors in the central nervous system: medulloblastomas, malignant gliomas, AT/RT (atypical teratoid/rhabdoid tumors), normal cerebellums and supratentorial PNETs (primitive neuroectodermal tumors). The first three classes each has a total of ten samples. The fourth class includes four samples. And, the fifth class, has a total of 8 samples. Each sample is described with $7,132$ features corresponding to the expression level from a set of genes. The goal is to classify each of these gene expression samples into their correct class.

This can be done using the leave-one-out test (loot), where we use all but one of the samples for training and determine whether our algorithm can correctly classify the sample left out.

Before we can use the data of Pomeroy et al. (2002), a set of normalization steps are required. First, it is typical to threshold the expression levels in each microarray to conform to a minimum expression value of 100 and a maximum of $1,600$. Second, the maximum and minimum of each feature, across all samples, is computed. The features with low-variance, that is, those that have $max/min < 12$ and $max - min < 1200$, do not carry significant class information and are therefore eliminated from consecutive analysis. This results in a feature representation of $3,517$. Finally, to facilitate the use of correlation-based classifiers, the variance of each of the samples is norm-normalized. This maps the feature vectors to S^{3516}.

Since the number of samples is only 42, we require to project the data onto a subsphere (Section 6.2) to a dimensionality where the sample-to-dimension ratio is appropriate. By reducing the dimensionality of our space to that of the range of the data, we get an average sample-to-dimension ratio of 1.18. Note that this ratio could not be reduced further, because the number of samples in gene expression analysis is generally very small (in this particular case 10 or less samples per class). Reducing the dimensionality further would impair our ability to analyze the data efficiently.

The average recognition rates obtained with each of the algorithms described earlier with loot are in Table 3. The small sample-to-dimension ratio, characteristic of this gene expression data sets, makes most of the algorithms perform poorly. For example, Fisher's LDA is about as bad as a random classifier which assigns the test vector to a class randomly. And, in particular, the Bingham approximation could not be completed with accuracy and its classifier is even worse than LDA's. As opposed to the results with the ETH database, when one has such a small number of samples, distributions with few parameters will usually produce better results. This is clearly the case here, with vMF and Gaussian providing a better fit than Bingham. To resolve the issues caused by the small sample-to-dimension ratio, one could regularize the covariance matrix of each class to allow the classifier to better adapt to the data (Friedman, 1989). By doing this, we were able to boost the results of our Gaussian fit to around 80%. At each iteration of the loot, the regularization parameter (also optimized with loot over the training data) selected 90% of the actual covariance matrix and 10% for the regularizing identity matrix term. We can now calculate how close to spherical-homoscedastic these regularized distributions are. As above, we can do this by estimating the average of percentage variance from the median of the eigenvalues obtained from the distribution of each class.[10] This results in $\sim 0.08\%$ deviation, which means the data can be very well represented with spherical-homoscedastic distributions. These results imply that the regularized Gaussian fit will outperform the other estimates, which is indeed the case. Since the classifiers derived in Section 5 are also based on the assumption of spherical-homoscedasticity, these should also perform well in this data set. We see in the results of Table 3 that these results were also the best. Furthermore, the spherical-homoscedastic classifiers do not require of any regularization. This means that the computation associated to these is very low, reversing the original problem associated to the (non-linear) estimation of the parameters of the distributions—making the *SH-Bingham* the best fit.

Further investigation of the spherical-homoscedastic classifiers in a higher dimensional nonlinear space by means of the Mahalanobis kernels defined above shows that these results can be further improved. In these experiments, the kernel parameters are optimized with loot over the training set.

10. We restricted the comparison to the first three eigenvalues, because the rest were many time zero due to the singularity of some of the class covariance matrices.

Method	*vMF*	*Bingham*	*Gaussian*	*SH-vMF*	*SH-Bingham*	*LDA*
Data Set A	28.57	14.29	33.33	80.95	85.71	21.43
ALL-AML	41.18	41.18	41.18	94.12	94.12	91.18

Table 3: Average classification rates on the gene expression data of Pomeroy et al. (2002) and Golub et al. (1999).

Method	*K-SH-vMF*	*K-SH-Bingham*
Data Set A	88.10 ($\bar{\varsigma} = 0.1$)	90.48 ($\bar{\varsigma} = 0.2$, $\bar{\ }v = 1.11$)
ALL-AML	85.29 ($\bar{\varsigma} = 0.7$)	85.29 ($\bar{\varsigma} = 0.7$, $\bar{\ }v = 1$)

Table 4: Average classification rates on the gene expression data using the Mahalanobis kernel. The values in parentheses correspond to the average parameters of the classifiers. These have been optimized using the leave-one-sample-out strategy on the training data.

As shown in Table 4, the performances of the *SH-vMF* and *SH-Bingham* classifiers improved to 88.10% and 90.48%, respectively.

Next, we tested our algorithms on the RNA gene expression data set of Golub et al. (1999). The training set consists of a total of 38 bone marrow samples, 27 of which correspond to Acute Lymphoblastic Leukemia (ALL) and 11 to Acute Myeloid Leukemia (AML). the testing set includes 20 ALL and 14 AML samples, respectively. Each feature vector is constructed with a total of $7,129$ probes from $6,817$ human genes, where a probe is a labeled subset of the original set of bases of a gene.

The feature vectors in this data set are norm-normalized to eliminate the variance changes across the samples. We then used the subsphere projection technique to represent the training set on a subsphere of dimensionality equal to the range space of the data, resulting in a sample-to-dimension ratio of 1.

The results obtained using the independent testing set on each of the trained classifiers are shown in Table 3. Once again, we see that the sample-to-dimension ratio is too small to allow good results in the estimate of the parameters of the spherical distributions or the Gaussian fit. As we did above, one can improve the Gaussian fit with the use of a regularization parameter, yielding $\sim 94\%$ classification rate (with a regularization of about 0.2). This is superior to the result of LDA, which in this case is much better than before. The LDA results are also surpassed by those of the spherical-homoscedastic classifiers, which again work well because the data can be estimated with spherical-homoscedastic distributions.

We note that the results obtained with the spherical-homoscedastic classifiers are already very good. Adding another optimization step to select the most appropriate kernel parameter may be too much to ask from such a small number of samples. We see in the results of Table 4 (which the use of the Mahalanobis kernel) this actually resulted in poorer classifications. This can be further studied with the use of the polynomial kernel, defined as

$$k(\mathbf{x},\mathbf{y}) = (\mathbf{x}^T\mathbf{y})^d,$$

where d defines the degree. If we substitute the Mahalanobis kernel for this one in our implementation and optimize d with loot, we find that this is given at $d = 1$. That is, the best polynomial

Method	*vMF*	*Gaussian*	*SH-vMF*	*SH-Bingham*	*LDA*
Classic	38.64	21.23	89.67	89.23 ($\bar{\nu} = 1$)	76.40
CMU	57.33	33.33	69.07	68 ($\bar{\nu} = 1$)	34.53

Table 5: Average classification rate on text data sets.

kernel is that which does not change the space, and the classification results will be the same as those shown in Table 3.

7. Text Data Set

In text recognition the documents are usually represented as a bag of words, which can be described in vector form by assigning to each feature the frequency of each of the words in the document. In this representation, the important information is that of the frequency of a word with respect to the rest (i.e., percentage of occurrence). This can be easily obtained by norm-normalizing these feature vectors, which maps the data into a spherical representation.

The first data set (labeled *Classic*) we will test is composed of three classes.[11] The first class is a collection of documents from MEDLINE (the index of medical related papers), which includes 1,033 documents. The second class consists of 1,460 documents from the CISI database. The third class includes 1,400 documents from the aeronautical system database CRANFIELD. The second data set (labeled *CMU*) we will use, is a subset of the CMU set.[12] This corresponds to 1,500 randomly selected documents, 500 from each of the following three classes: newsgroups comp.graphics, comp.os.ms-windows.misc and comp.windows.x.

The preprocessing of the data, which is common to the two data sets, includes eliminating high and low frequency words as well as those words that have less than 3 letters (examples are, "a", "and","the"). After this preprocessing, each feature vector in the *Classic* data set consists of 4,427 dimensions and those in the *CMU* data set 3,006 dimensions.

We have done a 10-fold cross-validation on these data sets using the algorithms described above. This means that we kept 10% of the data set for testing and fit the distribution models or classifiers to the remaining 90%. This was repeated 10 times and the average classification rates are shown in Table 5.

The resulting sample vectors are sparse—described in a high dimensional feature space with most of the features equal to zero. This makes the estimation problem very difficult. For instance, this sparseness did not permit computation of the normalizing constant of the Bingham distribution with the saddlepoint approximation of Kume and Wood (2005). For this reason Table 5 does not provide recognition rates for Binghams. The sparseness of the data did not allow for a good estimate of the parameters of the vMF or Gaussian modeling either, as is made evident in the poor results shown in Table 5. The Gaussian modeling result, in particular, can be improved in two ways. One, as above, would correspond to using a regularization term. A second option corresponds to calculating the average class covariance matrix of all Gaussians and use this as a common covariance matrix. This is in fact LDA's results, which is much better. Since eliminating the bases with lowest variance can reduce noise, we can further improve this results to about 98% for the Classical data set and 64% for CMU, by using the spherical projection of Section 6.2.

11. Available at ftp://ftp.cs.cornell.edu/pub/smart/.
12. Available at http://kdd.ics.uci.edu/databases/20newsgroups/20newsgroups.html.

Method	*K-SH-vMF* Polynomial	*K-SH-Bingham* Polynomial	*K-SH-vMF* RBF	*K-SH-Bingham* RBF
Classic	90.21 ($\bar{d} = 9.8$)	88.92 ($\bar{d} = 2.3$, $\bar{\nu} = 1$)	90.54 $\bar{\varsigma} = 0.3$)	88.23 ($\bar{\varsigma} = 0.7$, $\bar{\nu} = 1$)
CMU	71.13 ($\bar{d} = 7.9$)	68.27 ($\bar{d} = 5.4$, $\bar{\nu} = 1$)	75.27 $\bar{\varsigma} = 0.22$)	69.33 ($\bar{\varsigma} = 0.50$, $\bar{\nu} = 1$)

Table 6: Average classification rate on text data sets using kernel extension. The value of $\bar{d}$ specifies the average degree of the polynomial kernel optimized over the training set.

As we have shown above, these issues are generally less problematic when using the spherical-homoscedastic classifiers derived in this paper. The reason for that is given by the simplicity of these classifiers, which facilitates robustness. Yet, these results can be boosted by using the kernel classifiers derived above. To do this, we first note that we should not use the Mahalanobis kernel on these data sets, because the Gaussian fits will be biased by the sparseness. For this reason, we have used the polynomial kernel given above and the RBF kernel defined as $k(\mathbf{x},\mathbf{y}) = \exp\left(-\frac{\|\mathbf{x}-\mathbf{y}\|^2}{2\varsigma^2}\right)$. As demonstrated in Table 6, the results improved to about 90% on the Classic data set and over 75% for CMU.

8. Conclusions

In this paper, we investigated the effect of modeling von Mises-Fisher, Bingham and Kent distributions using Gaussian distributions. We first introduced the concept of spherical-homoscedasticity and showed that if two spherical distributions comply with this model, their Gaussian approximations are enough to obtain optimal classification performance in the Bayes sense. This was shown to be true for the von Mises-Fisher and Bingham distribution. For the Kent distribution the additional criteria defined in (26) must hold.

We further investigated what happens if we model the data sampled from two spherical-heteroscedastic distributions using their Gaussian approximations. In such a scenario, the Gaussian modeling will result in a decision boundary different to that produced by the original spherical distributions. We have referred to the additional error caused by this approximation as the reducible error. We have then empirically evaluated this and showed that as the two distributions start to deviate from spherical-homoscedastic, the probability of reducible error increases. We have also stated the particular cases where two spherical-heteroscedastic distributions may lead to a small error. For example, for vMFs this happens when these are highly concentrated, and for Bingham and Kent when the variance parameters are the same.

Since spherical-homoscedasticity provides an optimal model for parameter estimation, we were also able to define classifiers based on these. These classifiers are linear, since the Bayes decision boundary for two spherical-homoscedastic distributions in S^{p-1} is a hyperplane (same as for homoscedastic distributions in $\mathbb{R}^p$). When the data is spherical-heteroscedastic, we can first map the data into a space where the projected distributions adapt to the spherical-homoscedastic model. With this, we can use our spherical-homoscedastic classifiers in a large number of data sets. Finally, this can be efficiently implemented using the idea of the kernel trick as shown in Section 5. We have shown how all these results apply to a variety of problems in object recognition, gene expression classification, and text organization.

Acknowledgments

We thank the referees for their insightful comments. Thanks also go to Lee Potter for discussion. This research was partially supported by the National Institutes of Health under grant R01-DC-005241.

Appendix A. Notation

$\mathbf{x}$	feature vector
p	dimensionality of the original feature space
S^{p-1}	$(p-1)$-dimensional unit sphere in $\mathbb{R}^p$
$SO(p)$	p-dimensional Special Orthogonal Group
κ	concentration of our spherical distribution
β	ovalness of the Kent distribution
μ	mean direction vector
$\mathbf{m}$	mean feature vector
$\Gamma(\cdot)$	Gamma function
$I_\nu(\cdot)$	Bessel function of the first kind and order ν
$\mathbf{R}$	rotation matrix
Σ	covariance matrix
$\mathbf{S}$	autocorrelation (scatter) matrix
$\bar{\mathbf{S}}$	scatter matrix calculated on the null space of the mean direction
$\mathbf{A}$	parameter matrix for the Bingham and Fisher-Bingham
$\mathbf{K}$	gram matrix
$\mathbf{v}_i$	i^{th} eigenvector of the covariance matrix
$\mathbf{q}_i$	i^{th} eigenvector of the parameter matrix $\mathbf{A}$
λ_i	i^{th} eigenvalue of a symmetric matrix
$N(\mu,\Sigma)$	Gaussian (Normal) distribution
$M(\mu,\kappa)$	von Mises-Fisher distribution
$B(\mathbf{A})$	Bingham distribution
$K(\mu,\kappa,\mathbf{A})$	Kent distribution
$FB(\mu,\kappa,\mathbf{A})$	Fisher-Bingham distribution
$E(\cdot)$	expected value
ς	scale parameter
θ,ϕ,ω	rotation angles
$\mathbf{k}(.,.)$	mercer kernel
$\phi(.)$	vector mapping function
$\Phi(.)$	matrix mapping function

References

B. Audit and C.A. Ouzounis. From genes to genomes: Universal scale-invariant properties of microbial chromosome organisation. *Journal of Molecular Biology*, 332(3):617–633, 2003.

A. Banerjee, I.S. Dhillon, J. Ghosh, and S. Sra. Clustering on the unit hypersphere using von Mises-Fisher distributions. *Journal of Machine Learning Research*, 6:1345–1382, 2005.

P.N. Belhumeur and D.J. Kriegman. What is the set of images of an object under all possible lighting conditions? *International Journal of Computer Vision*, 28(3):245–260, 1998.

C. Bingham. An antipodally symmetric distribution on the sphere. *Annals of Statistics*, 2(6):1201–1225, 1974.

I.S. Dhillon and D.S. Modha. Concept decompositions for large sparse text data using clustering. *Machine Learning*, 42(1):143–175, 2001.

I.L. Dryden and K.V. Mardia. *Statistical Shape Analysis*. John Wiley & Sons, West Sussex, England, 1998.

R.A. Fisher. The statistical utilization of multiple measurements. *Annals of Eugenics*, 8:376–386, 1938.

J.H. Friedman. Regularized discriminant analysis. *Journal of The American Statistical Association*, 84(405):165–175, 1989.

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): 531–537, 1999.

P.J. Janssen, B. Audit, and C.A. Ouzounis. Strain-specific genes of Helicobacter pylori: distribution, function and dynamics. *Nucleic Acids Research*, 29(21):4395–4404, 2001.

O. Javed, M. Shah, and D. Comaniciu. A probabilistic framework for object recognition in video. In *Proceedings of International Conference on Image Processing*, pages 2713–2716, 2004.

C.M. Joshi. Some inequalities for modified Bessel functions. *Journal of the Australian Mathematics Society, Series A*, 50:333–342, 1991.

J.T. Kent. The Fisher-Bingham distribution on the sphere. *Journal of the Royal Statistical Society, Series B (Methodological)*, 44:71–80, 1982.

P. Koev and A. Edelman. The efficient evaluation of the hypergeometric function of a matrix argument. *Mathematics of Computation*, 75:833–846, 2006.

A. Kume and A.T.A. Wood. Saddlepoint approximations for the Bingham and Fisher-Bingham normalising constants. *Biometrika*, 92:465–476, 2005.

B. Leibe and B. Schiele. Analyzing appearance and contour based methods for object categorization. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2003.

K.V. Mardia and P.E. Jupp. *Directional Statistics*. John Wiley & Sons, West Sussex, England, 2000.

H. Murase and S.K. Nayar. Visual learning and recognition of 3D objects from appearance. *International Journal of Computer Vision*, 14:5–24, 1995.

S.L. Pomeroy, P. Tamayo, M. Gaasenbeek, L.M. Sturla, M. Angelo, M.E. McLaughlin, J.Y.H. 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*, pages 436–442, 2002.

D.E. Slice, editor. *Modern Morphometrics in Physical Anthropology*. Kluwer Academics, New York, NY, 2005.

C.M. Theobald. An inequality for the trace of the product of two symmetric matrices. *Proc. Cambridge Phil. Soc.*, 77:265–267, 1975.

A. Veeraraghavan, R.K. Roy-Chowdhury, and R. Chellappa. Matching shape sequences in video with applications in human movement analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(12):1896–1909, 2005.

L. Wang, T. Tan, W. Hu, and H. Ning. Automatic gait recognition based on statistical shape analysis. *IEEE Transactions on Image Processing*, 12(9):1120–1131, 2003.

Journal of Machine Learning Research 8 (2007) 1625-1657 Submitted 7/05; Revised 5/06; Published 7/07

Handling Missing Values when Applying Classification Models

Maytal Saar-Tsechansky MAYTAL@MAIL.UTEXAS.EDU
The University of Texas at Austin
1 University Station
Austin, TX 78712, USA

Foster Provost FPROVOST@STERN.NYU.EDU
New York University
44West 4th Street
New York, NY 10012, USA

Editor: Rich Caruana

Abstract

Much work has studied the effect of different treatments of missing values on model induction, but little work has analyzed treatments for the common case of missing values at prediction time. This paper first compares several different methods—predictive value imputation, the distribution-based imputation used by C4.5, and using reduced models—for applying classification trees to instances with missing values (and also shows evidence that the results generalize to bagged trees and to logistic regression). The results show that for the two most popular treatments, each is preferable under different conditions. Strikingly the reduced-models approach, seldom mentioned or used, consistently outperforms the other two methods, sometimes by a large margin. The lack of attention to reduced modeling may be due in part to its (perceived) expense in terms of computation or storage. Therefore, we then introduce and evaluate alternative, hybrid approaches that allow users to balance between more accurate but computationally expensive reduced modeling and the other, less accurate but less computationally expensive treatments. The results show that the hybrid methods can scale gracefully to the amount of investment in computation/storage, and that they outperform imputation even for small investments.

Keywords: missing data, classification, classification trees, decision trees, imputation

1. Introduction

In many predictive modeling applications, useful attribute values ("features") may be missing. For example, patient data often have missing diagnostic tests that would be helpful for estimating the likelihood of diagnoses or for predicting treatment effectiveness; consumer data often do not include values for all attributes useful for predicting buying preferences.

It is important to distinguish two contexts: features may be missing at induction time, in the historical "training"data, or at prediction time, in to-be-predicted "test"cases. This paper compares techniques for handling missing values at prediction time. Research on missing data in machine learning and statistics has been concerned primarily with induction time. Much less attention has been devoted to the development and (especially) to the evaluation of policies for dealing with missing attribute values at prediction time. Importantly for anyone wishing to apply models such as classification trees, there are almost no comparisons of existing approaches nor analyses or discussions of the conditions under which the different approaches perform well or poorly.

Although we show some evidence that our results generalize to other induction algorithms, we focus on classification trees. Classification trees are employed widely to support decision-making under uncertainty, both by practitioners (for diagnosis, for predicting customers' preferences, etc.) and by researchers constructing higher-level systems. Classification trees commonly are used as stand-alone classifiers for applications where model comprehensibility is important, as base classifiers in classifier ensembles, as components of larger intelligent systems, as the basis of more complex models such as logistic model trees (Landwehr et al., 2005), and as components of or tools for the development of graphical models such as Bayesian networks (Friedman and Goldszmidt, 1996), dependency networks (Heckerman et al., 2000), and probabilistic relational models (Getoor et al., 2002; Neville and Jensen, 2007). Furthermore, when combined into ensembles via bagging (Breiman, 1996), classification trees have been shown to produce accurate and well-calibrated probability estimates (Niculescu-Mizil and Caruana, 2005).

This paper studies the effect on prediction accuracy of several methods for dealing with missing features at prediction time. The most common approaches for dealing with missing features involve *imputation* (Hastie et al., 2001). The main idea of imputation is that if an important feature is missing for a particular instance, it can be estimated from the data that are present. There are two main families of imputation approaches: *(predictive) value imputation* and *distribution-based imputation*. Value imputation estimates a value to be used by the model in place of the missing feature. Distribution-based imputation estimates the conditional distribution of the missing value, and predictions will be based on this estimated distribution. Value imputation is more common in the statistics community; distribution-based imputation is the basis for the most popular treatment used by the (non-Bayesian) machine learning community, as exemplified by C4.5 (Quinlan, 1993).

An alternative to imputation is to construct models that employ only those features that will be known for a particular test case—so imputation is not necessary. We refer to these models as *reduced-feature* models, as they are induced using only a subset of the features that are available for the training data. Clearly, for each unique pattern of missing features, a different model would be used for prediction. We are aware of little prior research or practice using this method. It was treated to some extent in papers (discussed below) by Schuurmans and Greiner (1997) and by Friedman et al. (1996), but was not compared broadly to other approaches, and has not caught on in machine learning research or practice.

The contribution of this paper is twofold. First, it presents a comprehensive empirical comparison of these different missing-value treatments using a suite of benchmark data sets, and a follow-up theoretical discussion. The empirical evaluation clearly shows the inferiority of the two common imputation treatments, highlighting the underappreciated reduced-model method. Curiously, the predictive performance of the methods is more-or-less in inverse order of their use (at least in AI work using tree induction). Neither of the two imputation techniques dominates cleanly, and each provides considerable advantage over the other for some domains. The follow-up discussion examines the conditions under which the two imputation methods perform better or worse.

Second, since using reduced-feature models can be computationally expensive, we introduce and evaluate hybrid methods that allow one to manage the tradeoff between storage/computation cost and predictive performance, showing that even a small amount of storage/computation can result in a considerable improvement in generalization performance.

2. Treatments for Missing Values at Prediction Time

Little and Rubin (1987) identify scenarios for missing values, pertaining to dependencies between the values of attributes and the missingness of attributes. Missing Completely At Random (MCAR) refers to the scenario where missingness of feature values is independent of the feature values (observed or not). For most of this study we assume missing values occur completely at random. In discussing limitations below, we note that this scenario may not hold for practical problems (e.g., Greiner et al., 1997a); nonetheless, it is a general and commonly assumed scenario that should be understood before moving to other analyses, especially since most imputation methods rely on MCAR for their validity (Hastie et al., 2001). Furthermore, Ding and Simonoff (2006) show that the performance of missing-value treatments used when training classification trees seems unrelated to the Little and Rubin taxonomy, as long as missingness does not depend on the class value (in which case unique-value imputation should be used, as discussed below, as long as the same relationship will hold in the prediction setting).

When features are missing in test instances, there are several alternative courses of action.

1. *Discard instances*: Simply discarding instances with missing values is an approach often taken by researchers wanting to assess the performance of a learning method on data drawn from some population. For such an assessment, this strategy is appropriate if the features are missing completely at random. (It often is used anyway.) In practice, at prediction time, discarding instances with missing feature values may be appropriate when it is plausible to decline to make a prediction on some cases. In order to maximize utility it is necessary to know the cost of inaction as well as the cost of prediction error. For the purpose of this study we assume that predictions are required for all test instances.

2. *Acquire missing values*. In practice, a missing value may be obtainable by incurring a cost, such as the cost of performing a diagnostic test or the cost of acquiring consumer data from a third party. To maximize expected utility one must estimate the expected added utility from buying the value, as well as that of the most effective missing-value treatment. Buying a missing value is only appropriate when the expected net utility from acquisition exceeds that of the alternative. However, this decision requires a clear understanding of the alternatives and their relative performances—a motivation for this study.

3. *Imputation*. As introduced above, imputation is a class of methods by which an estimation of the missing value or of its distribution is used to generate predictions from a given model. In particular, either a missing value is replaced with an estimation of the value or alternatively the distribution of possible missing values is estimated and corresponding model predictions are combined probabilistically. Various imputation treatments for missing values in historical/training data are available that may also be deployed at prediction time. However, some treatments such as multiple imputation (Rubin, 1987) are particularly suitable to induction. In particular, multiple imputation (or repeated imputation) is a Monte Carlo approach that generates multiple simulated versions of a data set that each are analyzed and the results are combined to generate inference. For this paper, we consider imputation techniques that can be applied to individual test cases during inference.[1]

1. As a sanity check, we performed inference using a degenerate, single-case multiple imputation, but it performed no better and often worse than predictive value imputation.

(a) *(Predictive) Value Imputation (PVI)*: With value imputation, missing values are replaced with estimated values before applying a model. Imputation methods vary in complexity. For example, a common approach in practice is to replace a missing value with the attribute's mean or mode value (for real-valued or discrete-valued attributes, respectively) as estimated from the training data. An alternative is to impute with the average of the values of the other attributes of the test case.[2]

More rigorous estimations use predictive models that induce a relationship between the available attribute values and the missing feature. Most commercial modeling packages offer procedures for predictive value imputation. The method of *surrogate splits* for classification trees (Breiman et al., 1984) imputes based on the value of another feature, assigning the instance to a subtree based on the imputed value. As noted by Quinlan (1993), this approach is a special case of predictive value imputation.

(b) *Distribution-based Imputation (DBI)*. Given a (estimated) distribution over the values of an attribute, one may estimate the expected distribution of the target variable (weighting the possible assignments of the missing values). This strategy is common for applying classification trees in AI research and practice, because it is the basis for the missing value treatment implemented in the commonly used tree induction program, C4.5 (Quinlan, 1993). Specifically, when the C4.5 algorithm is classifying an instance, and a test regarding a missing value is encountered, the example is split into multiple pseudo-instances each with a different value for the missing feature and a weight corresponding to the estimated probability for the particular missing value (based on the frequency of values at this split in the training data). Each pseudo-instance is routed down the appropriate tree branch according to its assigned value. Upon reaching a leaf node, the class-membership probability of the pseudo-instance is assigned as the frequency of the class in the training instances associated with this leaf. The overall estimated probability of class membership is calculated as the weighted average of class membership probabilities over all pseudo-instances. If there is more than one missing value, the process recurses with the weights combining multiplicatively. This treatment is fundamentally different from value imputation because it combines the classifications across the distribution of an attribute's possible values, rather than merely making the classification based on its most likely value. In Section 3.3 we will return to this distinction when analyzing the conditions under which each technique is preferable.

(c) *Unique-value imputation*. Rather than estimating an unknown feature value it is possible to replace each missing value with an arbitrary unique value. Unique-value imputation is preferable when the following two conditions hold: the fact that a value is missing depends on the value of the class variable, and this dependence is present both in the training and in the application/test data (Ding and Simonoff, 2006).

4. *Reduced-feature Models*: Imputation is required when the model being applied employs an attribute whose value is missing in the test instance. An alternative approach is to apply a different model—one that incorporates only attributes that are known for the test instance. For

2. Imputing with the average of other features may seem strange, but in certain cases it is a reasonable choice. For example, for surveys and subjective product evaluations, there may be very little variance among a given subject's responses, and a much larger variance between subjects for any given question ("did you like the teacher?", "did you like the class?").

example, a new classification tree could be induced after removing from the training data the features corresponding to the missing test feature. This reduced-model approach may potentially employ a different model for each test instance. This can be accomplished by delaying model induction until a prediction is required, a strategy presented as "lazy" classification-tree induction by Friedman et al. (1996). Alternatively, for reduced-feature modeling one may store many models corresponding to various patterns of known and unknown test features.

With the exception of C4.5's method, dealing with missing values can be expensive in terms of storage and/or prediction-time computation. In order to apply a reduced-feature model to a test example with a particular pattern P of missing values, it is necessary either to induce a model for P on-line or to have a model for P precomputed and stored. Inducing the model on-line involves computation time[3] and storage of the training data. Using precomputed models involves storing models for each P to be addressed, which in the worst case is exponential in the number of attributes. As we discuss in detail below, one could achieve a balance of storage and computation with a hybrid method, whereby reduced-feature models are stored for the most important patterns; lazy learning or imputation could be applied for less-important patterns.

More subtly, predictive imputation carries a similar expense. In order to estimate the missing value of an attribute A for a test case, a model must be induced or precomputed to estimate the value of A based on the case's other features. If more than one feature is missing for the test case, the imputation of A is (recursively) a problem of prediction with missing values. Short of abandoning straightforward imputation, one possibility is to take a reduced-model approach for imputation itself, which begs the question: why not simply use a direct reduced-model approach?[4] Another approach is to build one predictive imputation model for each attribute, using all the other features, and then use an alternative imputation method (such as mean or mode value imputation, or C4.5's method) for any necessary secondary imputations. This approach has been taken previously (Batista and Monard, 2003; Quinlan, 1989), and is the approach we take for the results below.

3. Experimental Comparison of Prediction-time Treatments for Missing Values

The following experiments compare the predictive performance of classification trees using value imputation, distribution-based imputation, and reduced-feature modeling. For induction, we first employ the J48 algorithm, which is the Weka (Witten and Frank, 1999) implementation of C4.5 classification tree induction. Then we present results using bagged classification trees and logistic regression, in order to provide some evidence that the findings generalize beyond classification trees.

Our experimental design is based on the desire to assess the relative effectiveness of the different treatments under controlled conditions. The main experiments simulate missing values, in order to be able to know the accuracy if the values had been known, and also to control for various confounding factors, including pattern of missingness (viz., MCAR), relevance of missing values, and induction method (including missing value treatment used for training). For example, we assume that missing features are "important": that to some extent they are (marginally) predictive of the class. We avoid the trivial case where a missing value does not affect prediction, such as when a feature is not incorporated in the model or when a feature does not account for significant variance

3. Although as Friedman et al. (1996) point out, lazy tree induction need only consider the single path in the tree that matches the test case, leading to a considerable improvement in efficiency.
4. We are aware of neither theoretical nor empirical support for an advantage of predictive imputation over reduced modeling in terms of prediction accuracy.

in the target variable. In the former case, different treatments should result in the same classifications. In the latter case different treatments will not result in significantly different classifications. Such situations well may occur in practical applications; however, the purpose of this study is to assess the relative performance of the different treatments in situations when missing values will affect performance, not to assess how well they will perform in practice on any particular data set—in which case, careful analysis of the reasons for missingness must be undertaken.

Thus, we first ask: assuming the induction of a high-quality model, and assuming that the values of relevant attributes are missing, how do different treatments for missing test values compare? We then present various followup studies: using different induction algorithms, using data sets with "naturally occurring" missing values, and including increasing numbers missing values (chosen at random). We also present an analysis of the conditions under which different missing value treatments are preferable.

3.1 Experimental Setup

In order to focus on relevant features, unless stated otherwise, values of features from the top two levels of the classification tree induced with the complete feature set are removed from test instances (cf. Batista and Monard, 2003). Furthermore, in order to isolate the effect of various treatments for dealing with missing values at prediction time, we build models using training data having no missing values, except for the natural-data experiments in Section 3.6.

For distribution-based imputation we employ C4.5's method for classifying instances with missing values as described above. For value imputation we estimate missing categorical features using a J48 tree, and continuous values using Weka's linear regression. As discussed above, for value imputation with multiple missing values we use mean/mode imputation for the additional missing values. For generating a reduced model, for each test instance with missing values, we remove all the corresponding features from the training data before the model is induced so that only features that are available in the test instance are included in the model.

Each reported result is the average classification accuracy of a missing-value treatment over 10 independent experiments in which the data set is randomly partitioned into training and test sets. Except where we show learning curves, we use 70% of the data for training and the remaining 30% as test data. The experiments are conducted on fifteen data sets described in Table 1. The data sets comprise web-usage data sets (used by Padmanabhan et al., 2001) and data sets from the UCI machine learning repository (Merz et al., 1996).

To conclude that one treatment is superior to another, we apply a sign test with the null hypothesis that the average drops in accuracy using the two treatments are equal, as compared to the complete setting (described next).

3.2 Comparison of PVI, DBI and Reduced Modeling

Figure 1 shows the relative difference for each data set and each treatment, between the classification accuracy for the treatment and (as a baseline) the accuracy obtained if all features had been known both for training and for testing (the "complete" setting). The relative difference (improvement) is given by $100 \cdot \frac{AC_T - AC_K}{AC_K}$, where AC_K is the prediction accuracy obtained in the complete setting, and AC_T denotes the accuracy obtained when a test instance includes missing values and a treatment T is applied. As expected, in almost all cases the improvements are negative, indicating that missing

Data Set	Instances	Attributes	Nominal Attributes
Abalone	4177	8	1
Breast Cancer	699	9	0
BMG	2295	40	8
CalHouse	20640	8	0
Car	1728	6	6
Coding	20000	15	15
Contraceptive	1473	9	7
Credit	690	15	8
Downsize	1277	15	0
Etoys	270	40	8
Expedia	500	40	8
Move	3029	10	10
PenDigits	10992	16	0
Priceline	447	40	8
QVC	500	40	8

Table 1: Summary of Data Sets

values degrade classification, even when the treatments are used. Small negative values in Figure 1 are better, indicating that the corresponding treatment yields only a small reduction in accuracy.

Reduced-feature modeling is consistently superior. Table 2 shows the differences in the relative improvements obtained with each imputation treatment from those obtained with reduced modeling. A large negative value indicates that an imputation treatment resulted in a larger drop in accuracy than that exhibited by reduced modeling.

Reduced models yield an improvement over one of the other treatments for every data set. The reduced-model approach results in better performance compared to distribution-based imputation in 13 out of 15 data sets, and is better than value imputation in 14 data sets (both significant with $p < 0.01$).

Not only does a reduced-feature model almost always result in statistically significantly more-accurate predictions, the improvement over the imputation methods often was substantial. For example, for the Downsize data set, prediction with reduced models results in less than 1% decrease in accuracy, while value imputation and distribution-based imputation exhibit drops of 10.97% and 8.32%, respectively; the drop in accuracy resulting from imputation is more than 9 times that obtained with a reduced model. The average drop in accuracy obtained with a reduced model across all data sets is 3.76%, as compared to an average drop in accuracy of 8.73% and 12.98% for predictive imputation and distribution-based imputation, respectively. Figure 2 shows learning curves for all treatments as well as for the complete setting for the Bmg, Coding and Expedia data sets, which show three characteristic patterns of performance.

3.3 Feature Imputability and Modeling Error

Let us now consider the reasons for the observed differences in performance. The experimental results show clearly that the two most common treatments for missing values, predictive value

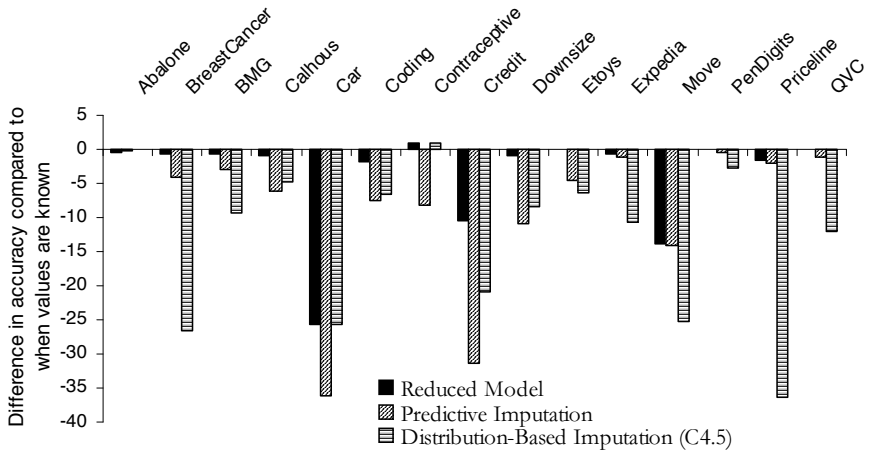


Figure 1: Relative differences in accuracy (%) between prediction with each missing data treatment and prediction when all feature values are known. Small negative values indicate that the treatment yields only a small reduction inaccuracy. Reduced modeling consistently yields the smallest reductions in accuracy—often performing nearly as well as having all the data. Each of the other techniques performs poorly on at least one data set, suggesting that one should choose between them carefully.

imputation (PVI) and C4.5's distribution-based imputation (DBI), each has a stark advantage over the other in some domains. Since to our knowledge the literature currently provides no guidance as to when each should be applied, we now examine conditions under which each technique ought to be preferable.

The different imputation treatments differ in how they take advantage of statistical dependencies between features. It is easy to develop a notion of the exact type of statistical dependency under which predictive value imputation should work, and we can formalize this notion by defining *feature imputability* as the fundamental ability to estimate one feature using others. A feature is completely imputable if it can be predicted perfectly using the other features—the feature is redundant in this sense. Feature imputability affects each of the various treatments, but in different ways. It is revealing to examine, at each end of the feature imputability spectrum, the effects of the treatments on expected error.[5] In Section 3.3.4 we consider why reduced models should perform well across the spectrum.

3.3.1 HIGH FEATURE IMPUTABILITY

First let's consider perfect feature imputability. Assume also, for the moment, that both the primary modeling and the *imputation* modeling have no intrinsic error—in the latter case, all existing feature

5. Kohavi and John (1997) focus on feature relevance and identify useful features for predictive model induction. Feature relevance pertains to the potential contribution of a given feature to prediction. Our notion of feature imputability addresses the ability to estimate a given feature's value using other feature values. In principle, these two notions are independent—a feature with low or high relevance may have high or low feature imputability.

Data Set	Predictive Imputation	Distribution-based Imputation (C4.5)
Abalone	0.12	0.36
Breast Cancer	-3.45	-26.07
BMG	-2.29	-8.67
CalHouse	-5.32	-4.06
Car	-13.94	0.00
Coding	-5.76	-4.92
Contraceptive	-9.12	-0.03
Credit	-23.24	-11.61
Downsize	-10.17	-7.49
Etoys	-4.64	-6.38
Expedia	-0.61	-10.03
Move	-0.47	-13.33
PenDigits	-0.25	-2.70
Priceline	-0.48	-35.32
QVC	-1.16	-12.05
Average	-5.38	-9.49

Table 2: Differences in relative improvement (from Figure 1 between each imputation treatment and reduced-feature modeling. Large negative values indicate that a treatment is substantially worse than reduced-feature modeling

imputability is captured. Predictive value imputation simply fills in the correct value and has no effect whatsoever on the bias and variance of the model induction.

Consider a very simple example comprising two attributes, A and B, and a class variable C with $A = B = C$. The "model" $A \rightarrow C$ is a perfect classifier. Now given a test case with A missing, predictive value imputation can use the (perfect) feature imputability directly: B can be used to infer A, and this enables the use of the learned model to predict perfectly. We defined feature imputability as a direct correlate to the effectiveness of value imputation, so this is no surprise. What is interesting is now to consider whether DBI also ought to perform well. Unfortunately, perfect feature imputability introduces a pathology that is fatal to C4.5's distribution-based imputation. When using DBI for prediction, C4.5's induction may have substantially increased bias, because it omits redundant features from the model—features that will be critical for prediction when the alternative features are missing. In our example, the tree induction does not need to include variable B because it is completely redundant. Subsequently when A is missing, the inference has no other features to fall back on and must resort to a default classification. This was an extreme case, but note that it did not rely on any errors in training.

The situation gets worse if we allow that the tree induction may not be perfect. We should expect features exhibiting high imputability—that is, that can yield only marginal improvements given the other features—to be more likely to be omitted or pruned from classification trees. Similar arguments apply beyond decision trees to other modeling approaches that use feature selection.

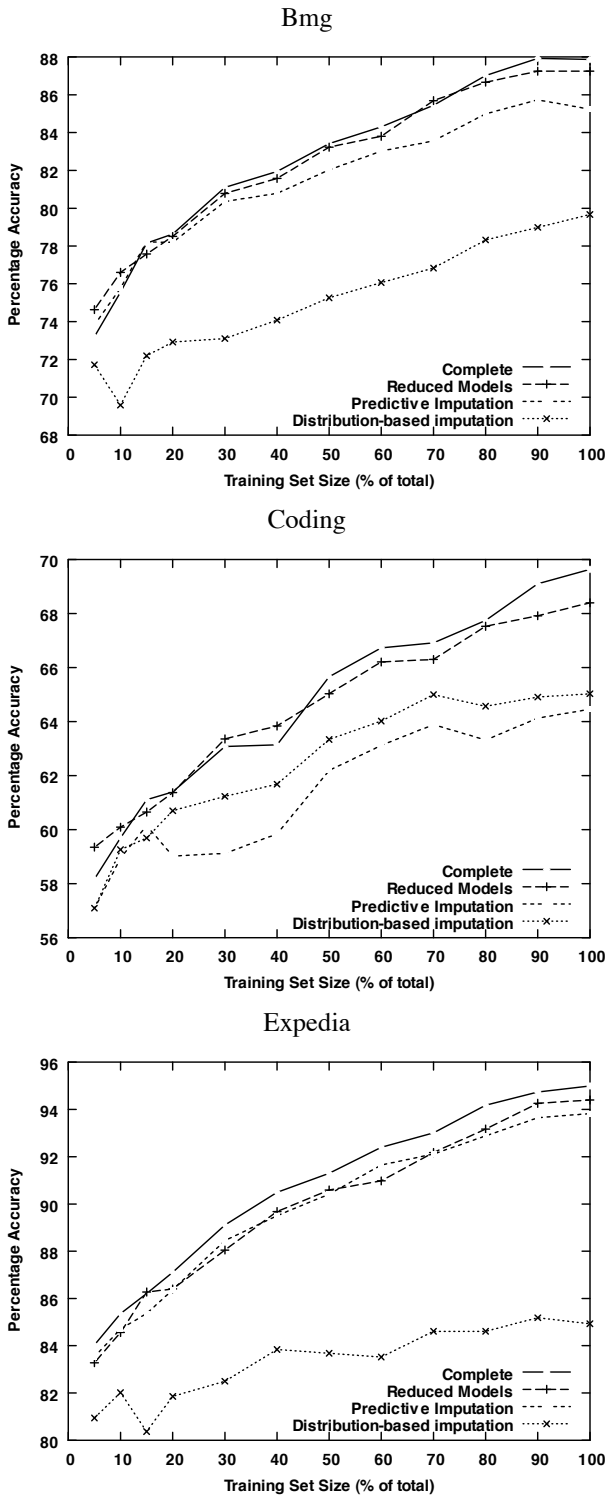


Figure 2: Learning curves for missing value treatments

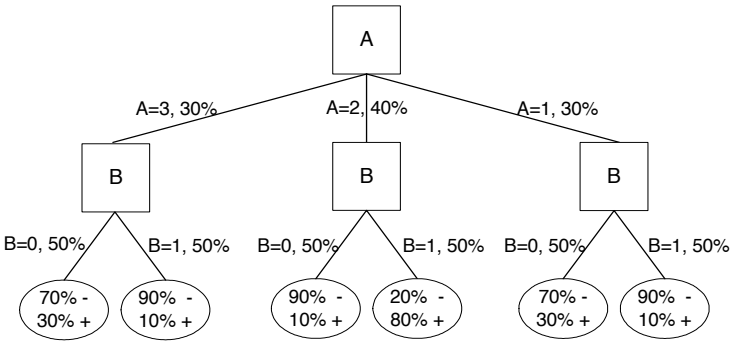


Figure 3: Classification tree example: consider an instance at prediction time for which feature A is unknown and B=1.

Finally, consider the inference procedures under high imputability. With PVI, classification trees' predictions are determined (as usual) based on the class distribution of a subset Q of training examples assigned to the same leaf node. On the other hand, DBI is equivalent to classification based on a superset S of Q. When feature imputability is high and PVI is accurate, DBI can only do as well as PVI if the weighted majority class for S is the same as that of Q. Of course, this is not always the case so DBI should be expected to have higher error when feature imputability is high.

3.3.2 LOW FEATURE IMPUTABILITY

When feature imputability is low we expect a reversal of the advantage accrued to PVI by using Q rather than S. The use of Q now is based on an uninformed guess: when feature imputability is very low PVI must guess the missing feature value as simply the most common one. The class estimate obtained with DBI is based on the larger set S and captures the expectation over the distribution of missing feature values. Being derived from a larger and unbiased sample, DBI's "smoothed" estimate should lead to better predictions on average.

As a concrete example, consider the classification tree in Figure 3. Assume that there is no feature imputability at all (note that A and B are marginally independent) and assume that A is missing at prediction time. Since there is no feature imputability, A cannot be inferred using B and the imputation model should predict the mode ($A = 2$). As a result every test example is passed to the $A = 2$ subtree. Now, consider test instances with $B = 1$. Although $(A = 2, B = 1)$ is the path chosen by PVI, it does not correspond to the majority of training examples with $B = 1$. Assuming that test instances follow the same distribution as training instances, on $B = 1$ examples PVI will have an accuracy of 38%. DBI will have an accuracy of 62%. In sum, DBI will "marginalize" across the missing feature and always will predict the plurality class. PVI sometimes will predict a minority class. Generalizing, DBI should outperform PVI for data sets with low feature imputability.

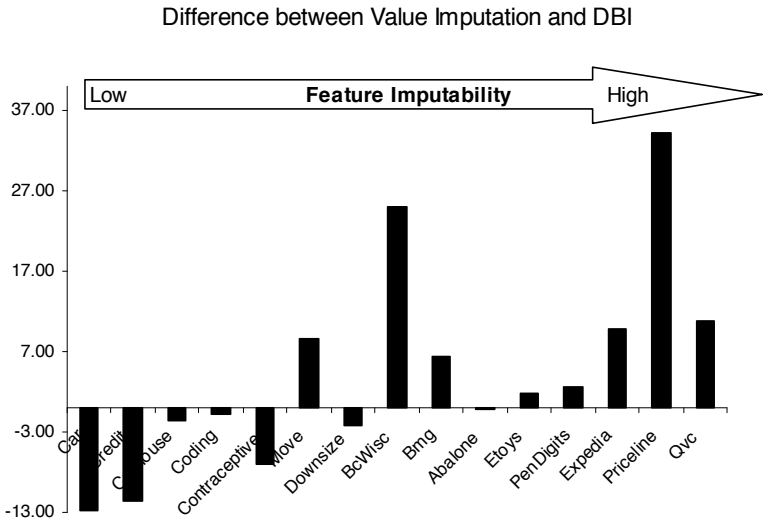


Figure 4: Differences between the relative performances of PVI and DBI. Domains are ordered left-to-right by increasing feature imputability. PVI is better for higher feature imputability, and DBI is better for lower feature imputability.

3.3.3 DEMONSTRATION

Figure 4 shows the 15 domains of the comparative study ordered left-to-right by a proxy for increasing feature imputability.[6] The bars represent the differences in the entries in Table 2, between predictive value imputation and C4.5's distribution-based imputation. A bar above the horizontal line indicates that value imputation performed better; a bar below the line indicates that DBI performed better. The relative performances follow the above argument closely, with value imputation generally preferable for high feature imputability, and C4.5's DBI generally better for low feature imputability.

3.3.4 REDUCED-FEATURE MODELING SHOULD HAVE ADVANTAGES ALL ALONG THE IMPUTABILITY SPECTRUM

Whatever the degree of imputability, reduced-feature modeling has an important advantage. Reduced modeling is a lower-dimensional learning problem than the (complete) modeling to which imputation methods are applied; it will tend to have lower variance and thereby may exhibit lower

6. Specifically, for each domain and for each missing feature we measured the ability to model the missing feature using the other features. For categorical features we measured the classification accuracy of the imputation model; for numeric features we computed the correlation coefficient of the regression. We created a rough proxy for the feature imputability in a domain, by averaging these across all the missing features in all the runs. As the actual values are semantically meaningless, we just show the trend on the figure. The proxy value ranged from 0.26 (lowest feature imputability) to 0.98 (highest feature imputability).

generalization error. To include a variable that will be missing at prediction time at best adds an irrelevant variable to the induction, increasing variance. Including an *important* variable that would be missing at prediction time may be worse, because unless the value can be replaced with a highly accurate estimate, its inclusion in the model is likely to reduce the effectiveness at capturing predictive patterns involving the other variables, as we show below.

In contrast, imputation takes on quite an ambitious task. From the same training data, it must build an accurate base classifier *and* build accurate imputation models for any possible missing values. One can argue that imputation tries implicitly to approximate the full-joint distribution, similar to a graphical model such as a dependency network (Heckerman et al., 2000). There are many opportunities for the introduction of error, and the errors will be compounded as imputation models are composed.

Revisiting the A, B, C example of Section 3.3.1, reduced-feature modeling uses the feature imputability differently from predictive imputation. The (perfect) feature imputability ensures that there will be an alternative model ($B \rightarrow C$) that will perform well. Reduced-feature modeling may have additional advantages over value imputation when the imputation is imperfect, as just discussed.

Of course, the other end of the feature imputability spectrum, when feature imputability is very low, is problematic generally when features are missing at prediction time. At the extreme, there is no statistical dependency at all between the missing feature and the other features. If the missing feature is important, predictive performance will necessarily suffer. Reduced modeling is likely to be better than the imputation methods, because of its reduced variance as described above.

Finally, consider reduced-feature modeling in the context of Figure 3, and where there is no feature imputability at all. What would happen if due to insufficient data or an inappropriate inductive bias, the complete modeling were to omit the important feature (B) entirely? Then, if A is missing at prediction time, no imputation technique will help us do better than merely guessing that the example belongs to the most common class (as with DBI) or guessing that the missing value is the most common one (as in PVI). Reduced-feature modeling may induce a partial (reduced) model (e.g., $B = 0 \rightarrow C = -, B = 1 \rightarrow C = +$) that will do better than guessing in expectation.

Figure 5 uses the same ordering of domains, but here the bars show the decreases in accuracy over the complete-data setting for reduced modeling and for value imputation. As expected, both techniques improve as feature imputability increases. However, the reduced-feature models are much more robust—with only one exception (Move) reduced-feature modeling yields excellent performance until feature imputability is very low. Value imputation does very well only for the domains with the highest feature imputability (for the highest-imputability domains, the accuracies of imputation and reduced modeling are statistically indistinguishable).

3.4 Evaluation using Ensembles of Trees

Let us now examine whether the results we have presented change substantively if we move beyond simple classification trees. Here we use bagged classification trees (Breiman, 1996), which have been shown repeatedly to outperform simple classification trees consistently in terms of generalization performance (Bauer and Kohavi, 1999; Perlich et al., 2003), albeit at the cost of computation, model storage, and interpretability. For these experiments, each bagged model comprises thirty classification trees.

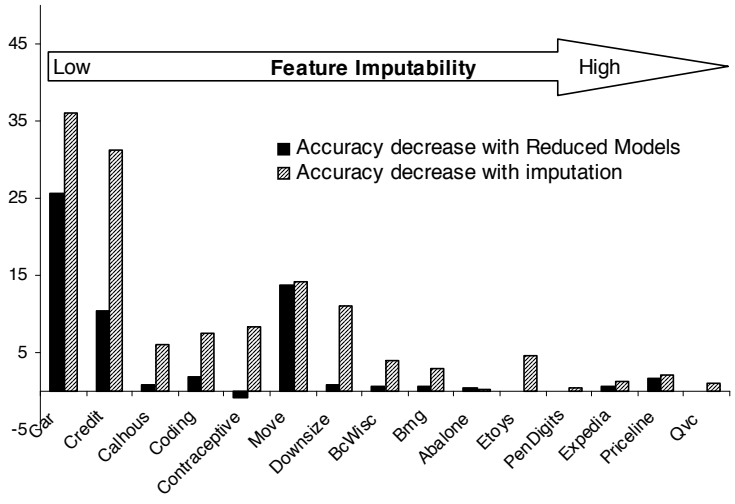


Figure 5: Decreases in accuracy for reduced-feature modeling and value imputation. Domains are ordered left-to-right by increasing feature imputability. Reduced modeling is much more robust to moderate levels of feature imputability.

Figure 6 shows the performance of the three missing-value treatments using bagged classification trees, showing (as above) the relative difference for each data set between the classification accuracy of each treatment and the accuracy of the complete setting. As with simple trees, reduced modeling is consistently superior. Table 3 shows the differences in the relative improvements of each imputation treatment from those obtained with reduced models. For bagged trees, reduced modeling is better than predictive imputation in 12 out of 15 data sets, and it performs better than distribution-based imputation in 14 out of 15 data sets (according to the sign test, these differences are significant at $p < 0.05$ and $p < 0.01$ respectively). As for simple trees, in some cases the advantage of reduced modeling is striking.

Figure 7 shows the performance of all treatments for models induced with an increasing training-set size for the Bmg, Coding and Expedia data sets. As for single classification models, the advantages obtained with reduced models tend to increase as the models are induced from larger training sets.

These results indicate that for bagging, a reduced model's relative advantage with respect to predictive imputation is comparable to its relative advantage when a single model is used. These results are particularly notable given the widespread use of classification-tree induction, and of bagging as a robust and reliable method for improving classification-tree accuracy via variance reduction.

Beyond simply demonstrating the superiority of reduced modeling, an important implication is that practitioners and researchers should not choose either C4.5-style imputation or predictive value imputation blindly. Each does extremely poorly in some domains.

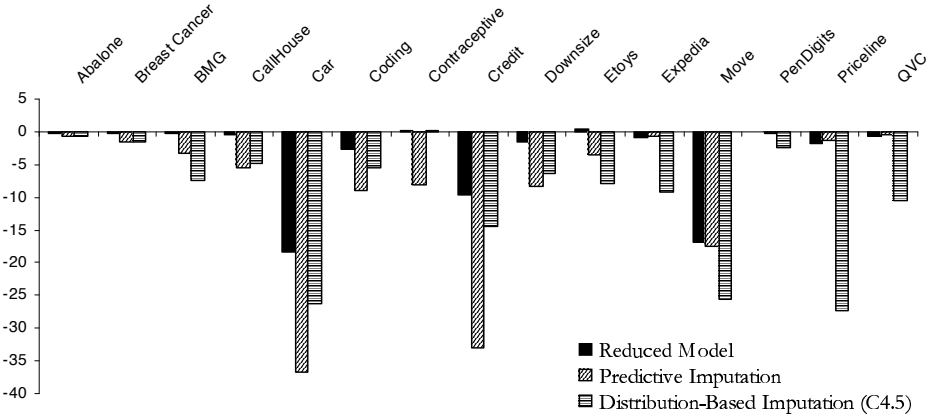


Figure 6: Relative differences in accuracy for bagged decision trees between each missing value treatment and the complete setting where all feature values are known. Reduced modeling consistently is preferable. Each of the other techniques performs poorly on at least one data set.

Data Set	Predictive Imputation	Distribution-based Imputation (C4.5)
Abalone	-0.45	-0.51
Breast Cancer	-1.36	-1.28
BMG	-3.01	-7.17
CalHouse	-5.16	-4.41
Car	-22.58	-9.72
Coding	-6.59	-2.98
Contraceptive	-8.21	0.00
Credit	-25.96	-5.36
Downsize	-6.95	-4.94
Etoys	-3.83	-8.24
Expedia	0.20	-8.48
Move	-0.92	-10.61
PenDigits	-0.11	-2.33
Priceline	0.36	-25.97
QVC	0.13	-9.99
Average	-5.47	-6.57

Table 3: Relative difference in prediction accuracy for bagged decision trees between imputation treatments and reduced-feature modeling.

3.5 Evaluation using Logistic Regression

In order to provide evidence that the relative effectiveness of reduced models is not specific to classification trees and models based on trees, let us examine logistic regression as the base classifier.

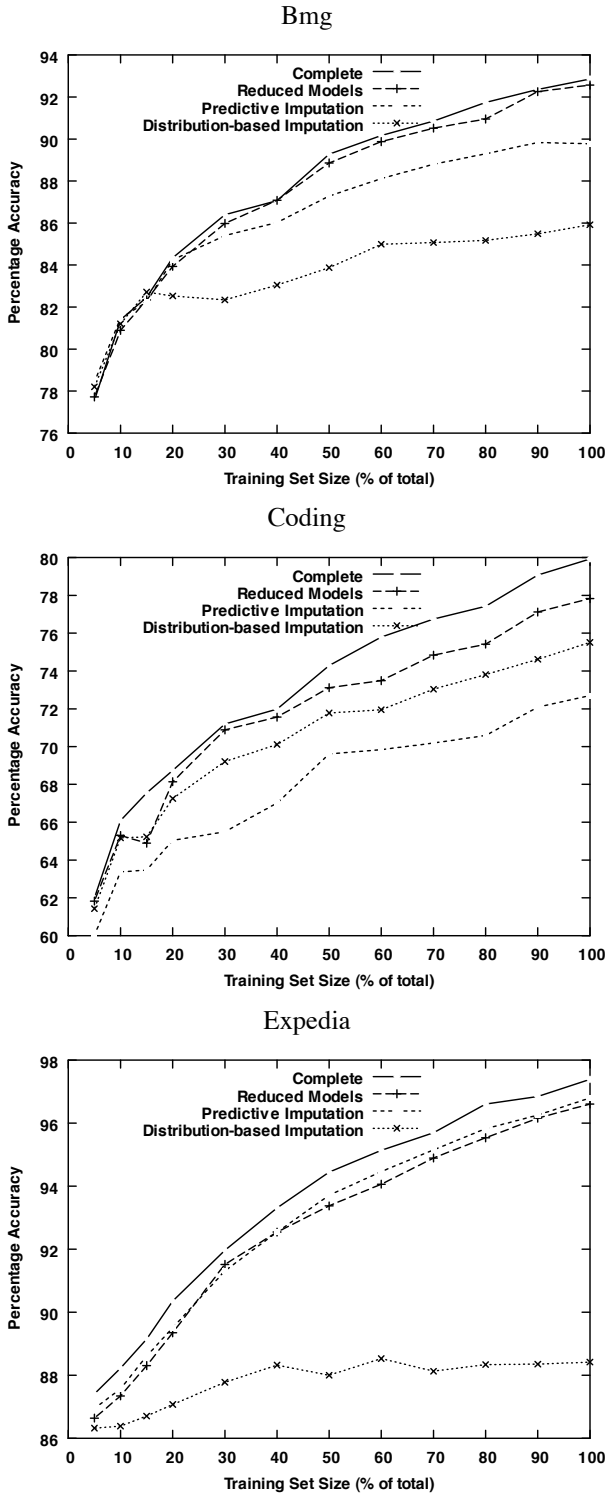


Figure 7: Learning curves for missing value treatments using bagged decision trees.

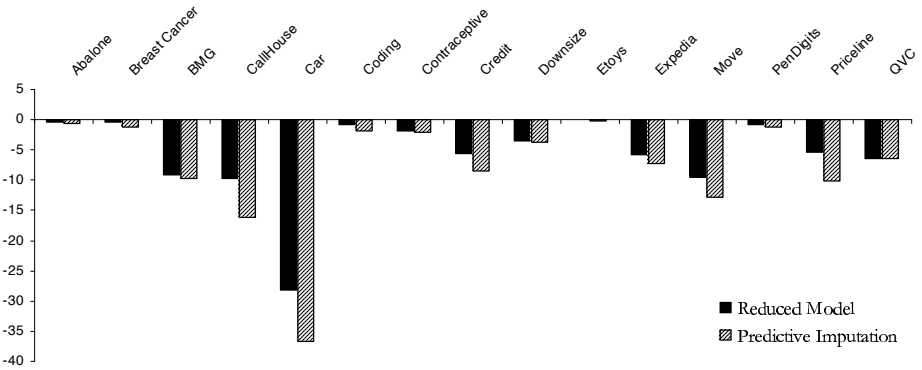


Figure 8: Relative differences in accuracies for a logistic regression model when predicitve value imputation and reduced modeling are employed, as compared to when all values are known.

Because C4.5-style distribution-based imputation is not applicable for logistic regression, we compare predictive value imputation to the reduced model approach. Figure 8 shows the difference in accuracy when predictive value imputation and reduced models are used. Table 4 shows the differences in the relative improvements of the predictive imputation treatment from those obtained with reduced models. For logistic regression, reduced modeling results in higher accuracy than predictive imputation in all 15 data sets (statistically significant with $p \ll 0.01$).

3.6 Evaluation with "Naturally Occurring" Missing Values

We now compare the treatment on four data sets with naturally occurring missing values. By "naturally occurring," we mean that these are data sets from real classification problems, where the missingness is due to processes of the domain outside of our control. We hope that the comparison will provide at least a glimpse at the generalizability of our findings to real data. Of course, the missingness probably violates our basic assumptions. Missingness is unlikely to be completely at random. In addition, missing values may have little or no impact on prediction accuracy, and the corresponding attributes may not even be used by the model. Therefore, even if the qualitative results hold, we should not expect the magnitude of the effects to be as large as in the controlled studies.

We employ four business data sets described in Table 5. Two of the data sets pertain to marketing campaigns promoting financial services to a bank's customers (Insurance and Mortgage). The Pricing data set captures consumers' responses to price increases—in response to which customers either discontinue or continue their business with the firm. The Hospitalization data set contains medical data used to predict diabetic patients' rehospitalizations. As before, we induced a model from the training data. Because instances in the training data include missing values as well, models are induced from training data using C4.5's distribution-based imputation. We applied the model to instances that had at least one missing value. Table 5 shows the average number of missing values in a test instance for each of the data sets.

Data Set	Predictive Imputation
Abalone	-0.20
Breast Cancer	-1.84
BMG	-0.75
CalHouse	-7.11
Car	-12.04
Coding	-1.09
Contraceptive	-1.49
Credit	-3.05
Downsize	-0.32
Etoys	-0.26
Expedia	-1.59
Move	-3.68
PenDigits	-0.34
Priceline	-5.07
QVC	-0.02
Average	-2.59

Table 4: Relative difference in prediction accuracy for logistic regression between imputation and reduced-feature modeling. Reduced modeling never is worse, and sometimes is substantially more accurate.

Data Set	Instances	Attributes	Nominal Attributes	Average Number of Missing Features
Hospitalization	48083	13	7	1.73
Insurance	18542	16	0	2.32
Mortgage	2950	10	1	2.76
Pricing	15531	28	8	3.56

Table 5: Summary of business data sets with "naturally occurring" missing values.

Figure 9 and Table 6 show the relative decreases in classification accuracy that result for each treatment relative to using a reduced-feature model. These results with natural missingness are consistent with those obtained in the controlled experiments discussed earlier. Reduced modeling leads to higher accuracies than both popular alternatives for all four data sets. Furthermore, predictive value imputation and distribution-based imputation each outperforms the other substantially on at least one data set—so one should not choose between them arbitrarily.

3.7 Evaluation with Multiple Missing Values

We have evaluated the impact of missing value treatments when the values of one or a few important predictors are missing from test instances. This allowed us to assess how different treatments improve performance when performance is in fact undermined by the absence of strong predictors

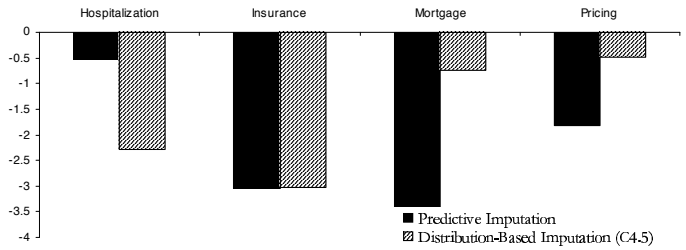


Figure 9: Relative percentage-point differences in predictive accuracy obtained with distribution-based imputation and predictive value imputation treatments compared to that obtained with reduced-feature models. The reduced models are more accurate in every case.

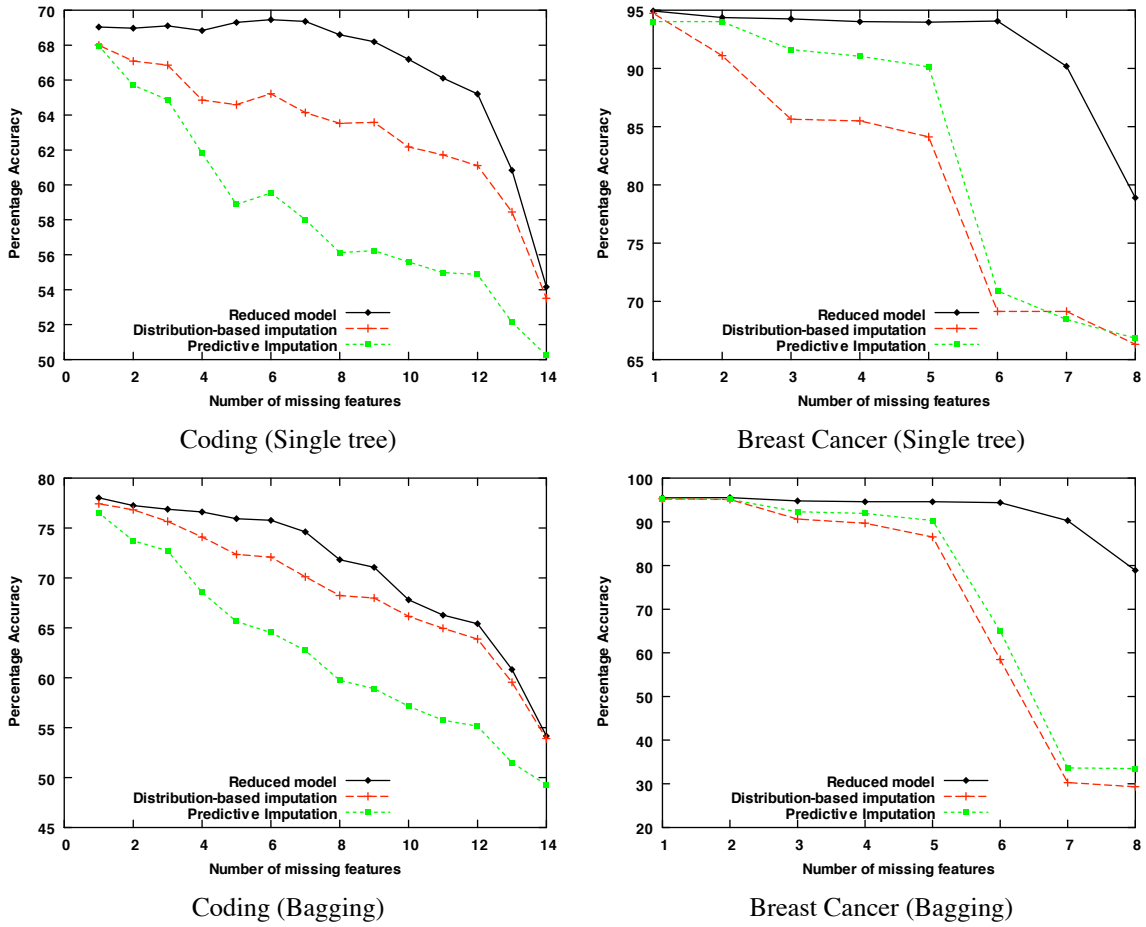
Data Set	Predictive Imputation	Distribution-based Imputation (C4.5)
Hospitalization	-0.52	-2.27
Insurance	-3.04	-3.03
Mortgage	-3.40	-0.74
Pricing	-1.82	-0.48

Table 6: Relative percentage-point difference in prediction accuracy between imputation treatments and reduced-feature modeling.

at inference time. Performance may also be undermined when a large number of feature values are missing at inference time.

Figure 10 shows the accuracies of reduced-feature modeling and predictive value imputation as the number of missing features increases, from 1 feature up to when only a single feature is left. Features are removed at random. The top graphs are for tree induction and the bottom for bagged tree induction. These results are for Breast Cancer and Coding, which have moderate-to-low feature imputability, but the general pattern is consistent across the other data sets. We see a typical pattern: the imputation methods have steeper decreases in accuracy as the number of missing values increases. Reduced modeling's decrease is convex, with considerably more robust performance even for a large number of missing values.

Finally, this discussion would be incomplete if we did not mention two particular sources of imputation-modeling error. First, as we mentioned earlier when more than one value is missing, the imputation models themselves face a missing-at-prediction-time problem, which must be addressed by a different technique. This is a fundamental limitation to predictive value imputation as it is used in practice. One could use reduced modeling for imputation, but then why not just use reduced modeling in the first place? Second, predictive value imputation might do worse than reduced modeling, if the inductive bias of the resultant imputation model is "worse" than that of the reduced model. For example, perhaps our classification-tree modeling does a much better job with numeric variables than the linear regression we use for imputation of real-value features. However, this does not seem to be the (main) reason for the results we see. If we look at the data sets comprising only



Coding (Single tree) | Breast Cancer (Single tree)

Coding (Bagging) | Breast Cancer (Bagging)

Figure 10: Accuracies of missing value treatments as the number of missing features increases

categorical features (viz., Car, Coding, and Move, for which we use C4.5 for both the base model and the imputation model), we see the same patterns of results as with the other data sets.

4. Hybrid Models for Efficient Prediction with Missing Values

The increase in accuracy of reduced modeling comes at a cost, either in terms of storage or of prediction-time computation (or both). Either a new model must be induced for every (novel) pattern of missing values encountered, or a large number of models must be stored. Storing many classification models has become standard practice, for example, for improving accuracy with classifier ensembles. Unfortunately, the storage requirements for full-blown reduced modeling become impracticably large as soon as the possible number of (simultaneous) missing values exceeds a dozen or so. The strength of reduced modeling in the empirical results presented above suggests its tactical use to improve imputation, for example by creating hybrid models that trade off efficiency for improved accuracy.

4.1 Likelihood-based Hybrid Solutions

One approach for reducing the computational cost of reduced modeling is to induce and store models for some subset of the possible patterns of missing features. When a test case is encountered, the corresponding reduced model is queried. If no corresponding model has been stored, the hybrid would call on a fall-back technique: either incurring the expense of prediction-time reduced modeling, or invoking an imputation method (and possibly incurring reduced accuracy).

Not all patterns of missing values are equally likely. If one can estimate from prior experience the likelihood for any pattern of missing values, then this information may be used to decide among different reduced models to induce and store. Even if historical data are not sufficient to support accurate estimation of full, joint likelihoods, it may be that the marginal likelihoods of different variables being missing are very different. And even if the marginals are or must be assumed to be uniform, they still may well lead to very different (inferred) likelihoods of the many patterns of multiple missing values. In the context of Bayesian network induction, Greiner et al. (1997b) note the important distinction between considering only the underlying distribution for model induction/selection and considering the querying distribution as well. Specifically, they show that when comparing different Bayesian networks one should identify the network exhibiting the best expected performance over the query distribution, that is, the distribution of tasks that the network will be used to answer, rather than the network that satisfies general measures such as maximum likelihood over the underlying event distribution. H. and F. (1992) employ a similar notion to reduce inference time with Bayesian networks. H. and F. (1992) precompute parts of the network that pertain to a subset of frequently encountered cases so as to increase the expected speed of inference.

The horizontal, dashed line in Figure 11 shows the performance of pure predictive value imputation for the CalHouse data set. The lower of the two curves in Figure 11 shows the performance of a likelihood-based reduced-models/imputation hybrid. The hybrid approach allows one to choose an appropriate space-usage/accuracy tradeoff, and the figure shows that storing even a few reduced models can result in considerable improvement. The curve was generated as follows. Given enough space to store k models, the hybrid induces and stores reduced models for the top-k most likely missing-feature patterns, and uses distribution-based imputation for the rest. The Calhouse data set has eight attributes, corresponding to 256 patterns of missing features. We assigned a random probability of occurrence for each pattern as follows. The frequency of each pattern was drawn at random from the unit uniform distribution and subsequently normalized so that the frequencies added up to one. For each test instance we sampled a pattern from the resulting distribution and removed the values of features specified by the pattern.

Notice that for the likelihood-based hybrid the marginal improvement in accuracy does not decrease monotonically with increasing model storage: the most frequent patterns are not necessarily the patterns that lead to the largest accuracy increases. Choosing the best set of models to store is a complicated optimization problem. One must consider not only the likelihood of a pattern of missing features, but also the expected improvement in accuracy that will result from including the corresponding model in the "model base." Calculating the expected improvement is complicated by the fact that the patterns of missing values form a lattice (Schuurmans and Greiner, 1997). For an optimal solution, the expected improvement for a given pattern should *not* be based on the improvement over using the default strategy (e.g., imputation), but should be based on using the next-best already-stored pattern. Determining the next-best pattern is a non-trivial estimation problem, and,

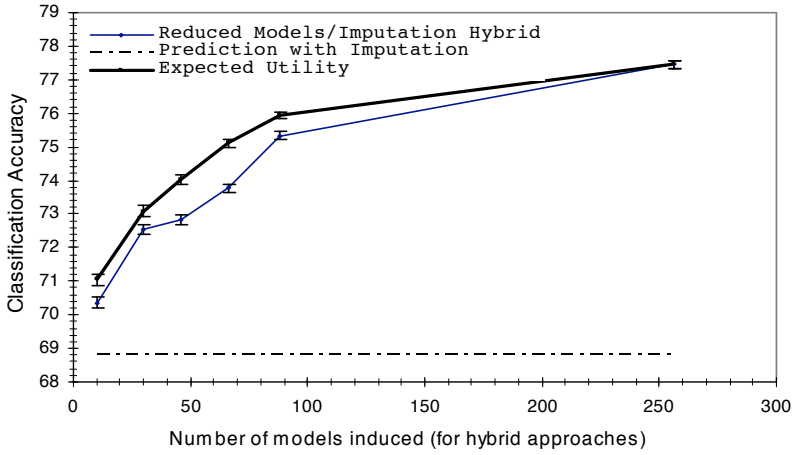


Figure 11: Accuracies of hybrid strategies for combining reduced modeling and imputation. Storing even a small fraction of the possible reduced models can improve accuracy considerably.

even if it weren't, the optimization problem is hard. Specifically, the optimal set of reduced models M corresponds to solving the following optimization task:

$$\text{argmax}_M \left(\sum_f [p(f) \cdot U(f \mid M)] \right)$$
$$s.t. \sum_{f_m \in M} t(f_m) \leq T,$$

where M is the subset of missing patterns for which reduced models are induced, $t(f)$ is the (marginal) resource usage (time or space) for reduced modeling with pattern f, T is the maximum total resource usage allocated for reduced model induction, and $U(f|M)$ denotes the utility from inference for an instance with pattern f given the set of reduced models in the subset M (when $f \in M$ the utility is derived from inference via the respective reduced model, otherwise the utility is derived from inference using the next-best already-stored pattern).

The upper curve in Figure 11 shows the performance of a heuristic approximation to a utility-maximizing hybrid. We estimate the marginal utility of adding a missing-feature pattern f as $u(f) = p(f) \cdot (\hat{a}_{rm}(f) - \hat{a}_i(f))$, where $p(f)$ is the likelihood of encountering pattern f, $\hat{a}_{rm}(f)$ is the estimated accuracy of reduced modeling for f and $\hat{a}_i(f)$ is the estimated accuracy of a predictive value imputation model for missing pattern f. We estimate $\hat{a}_{rm}(f)$ and $\hat{a}_i(f)$ based on cross-validation using the training data. The figure shows that even a heuristic expected-utility approach can improve considerably over the pure likelihood-based approach.

4.2 Reduced-Feature Ensembles

The reduced-feature approach involves either on-line computation or the storing of multiple models, and storing multiple models naturally motivates using ensemble classifiers. Consider a simple

Reduced-Feature Ensemble (ReFE), based on a set $\mathcal{R}$ of models each induced by excluding a single attribute, where the cardinality of $\mathcal{R}$ is the number of attributes. Model $i \in \mathcal{R}$ tries to capture an alternative hypothesis that can be used for prediction when the value for attribute v_i, perhaps among others, is unknown. Because the models exclude only a single attribute, a ReFE avoids the combinatorial space requirement of full-blown reduced modeling. When multiple values are missing, ReFE ensemble members rely on imputation for the additional missing values. We employ DBI.

More precisely, a ReFE classifier works as follows. For each attribute v_i a model m_i is induced with v_i removed from the training data. For a given test example in which the values for the set of attributes V are missing, for each attribute $v_i \in V$ whose value is missing, the corresponding model m_i is applied to estimate the (probability of) class membership. To generate a prediction, the predictions of all models applied to a test example are averaged. When a single feature is missing, ReFE is identical to the reduced-model approach. The application of ReFE for test instances with two or more missing features results in an ensemble. Hence, in order to achieve variance reduction as with bagging, in our experiments training data are resampled with replacement for each member of the ensemble.

Table 7 summarizes the relative improvements in accuracy as compared to a single model using predictive value imputation. For comparison we show the improvements obtained by bagging alone (with imputation), and by the full-blown reduced-model approach. For these experiments we fixed the number of missing features to be three. The accuracies of ReFE and bagging are also plotted in Figure 12 to highlight the difference in performance across domains. Bagging uses the same number of models as employed by ReFE, allowing us to separate the advantage that can be attributed to the reduced modeling and that attributable to variance reduction.

We see that ReFE consistently improves over both a single model with imputation (positive entries in the ReFE column) and over bagging with imputation. In both comparisons, ReFE results in higher accuracy on all data sets, shown in bold in Table 7, except Car; the 14-1 win-loss record is statistically significant with $p < 0.01$. The magnitudes of ReFE's improvements vary widely, but on average they split the difference between bagging with imputation and the full-blown reduced modeling. Note that although full-blown reduced modeling usually is more accurate, ReFE sometimes shows better accuracy, indicating that the variance reduction of bagging complements the (partial) reduced modeling.

The motivation for employing ReFE instead of the full-blown reduced-feature modeling is the substantially lower computational burden of ReFE as compared to that of reduced modeling. For a domain with N attributes, $(2^N - 1)$ models must be induced by reduced modeling in order to match each possible missing pattern. ReFE induces only N models—one for each attribute. For example, the Calhouse data set, which includes only 8 attributes, required more than one-half hour to produce all the 256 models for full-blown reduced modeling. It took about a minute to produce the 8 models for the ReFE.

4.3 Larger Ensembles

The previous results do not take full advantage of the variance reduction possible with large ensembles (Hastie et al., 2001). Table 8 shows the percentage improvement in accuracy over a single model with imputation, for ReFE, bagging with imputation, and bagging of reduced models, each using thirty ensemble members. The ReFE ensembles comprise 10 reduced models for each missing feature, where each individual model is generated using sampling with replacement as in bagging.

Data Sets	Bagging	ReFE	Reduced Model
Abalone	0.11	**0.26**	0.05
BreastCancer	4.35	**4.51**	4.62
Bmg	2.88	**3.51**	2.57
CalHouse	1.25	**6.06**	5.45
Car	0.10	-0.28	17.55
Coding	4.82	**6.97**	5.32
Contraceptive	0.39	**0.45**	1.16
Credit	2.58	**5.54**	8.12
Downsize	3.09	**3.78**	6.51
Etoys	0.00	**2.28**	1.07
Expedia	1.76	**2.11**	2.73
Move	3.26	**5.99**	8.97
Pendigits	0.06	**0.58**	1.57
Priceline	3.29	**4.98**	10.84
Qvc	1.83	**2.44**	2.60
Average	1.98	3.27	5.27

Table 7: Relative improvements in accuracy for bagging with imputation and ReFE, as compared to a single model with imputation. Bold entries show the cases where ReFE improves both over using a single model with imputation and over bagging with imputation. For comparison, the rightmost column shows the improvements of full-blown reduced modeling. The ReFEs are more accurate than either a single model with imputation, or bagging with imputation, while being much more efficient than reduced modeling in terms of computation and/or storage.

For control, for any given number of missing features in a test example, we evaluate the performance of bagging with the same number of individual models. Similarly, we generate a bagged version of the full-blown reduced model, with the same number of models as in the other approaches. As before, we fix the number of missing values in each test instance to three.

As expected, including a larger number of models in each ensemble results in improved performance for all treatments, for almost all data sets. The advantage exhibited by ReFE over bagging with imputation is maintained. As shown in Table 8. ReFE results in higher accuracy than bagging with imputation for all 15 data sets (statistically significant at $p \ll 0.01$).

4.4 ReFEs with Increasing Numbers of Missing Values

For the smaller ensembles, Figure 13 shows the decrease in classification accuracy that results when the number of missing values in each test instance is increased. Attributes are chosen for removal uniformly at random. For all data sets, the accuracies of all methods decrease as more attributes are missing at prediction time. The marginal reductions in accuracy with increasing missing values are similar for ReFE and for bagging with imputation, with ReFE's advantage diminishing slowly

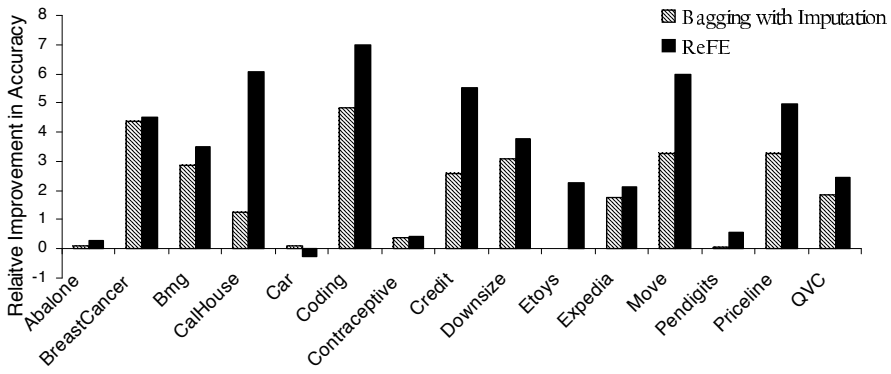


Figure 12: Relative improvement in accuracy (%) as obtained for bagging with imputation and ReFE, with respect to a single model with imputation.

Data Sets	Bagging with Imputation	ReFE	Bagging with Reduced Model
Abalone	0.34	**0.49**	0.83
BreastCancer	5.10	**5.89**	5.15
Bmg	7.22	**7.88**	8.21
CalHouse	2.66	**7.10**	8.47
Car	-0.10	-0.08	17.55
Coding	14.39	**15.28**	17.65
Contraceptive	0.64	**0.89**	1.03
Credit	4.98	**6.77**	9.35
Downsize	6.91	**7.60**	11.13
Etoys	2.95	**3.35**	3.48
Expedia	3.41	**4.19**	5.27
Move	6.48	**9.73**	13.78
PenDigits	0.44	**0.90**	1.52
Priceline	7.55	**9.42**	11.02
QVC	4.23	**5.88**	7.16
Average	4.48	5.69	8.11

Table 8: Percentage improvement in accuracy compared to a single model with imputation, for bagging with imputation, ReFE, and bagging with reduced models. All ensembles employ 30 models for prediction. Bold entries show the cases where ReFE improves both over using a single model with imputation and over bagging with imputation.

with increasing missing values. This is in stark contrast to the robust behavior of reduced models (also shown in Figure 13). This is because ReFE uses imputation to handle additional missing

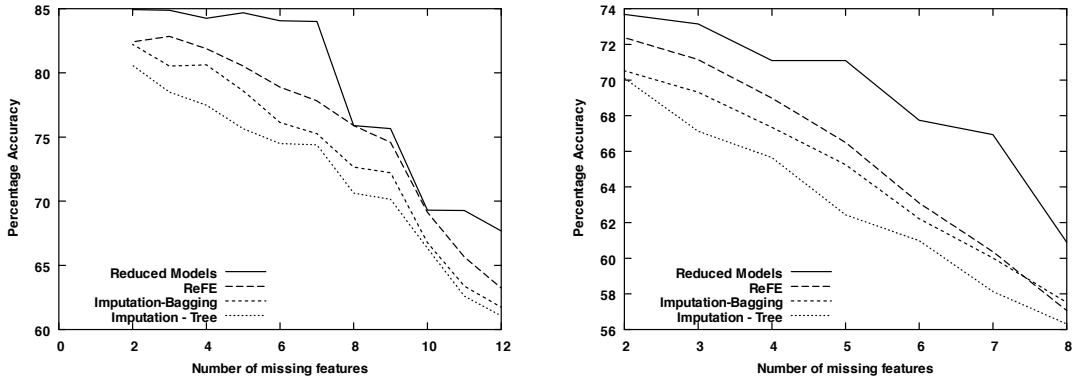


Figure 13: Performance of missing value treatments for small ensemble models as the number of missing values increases.

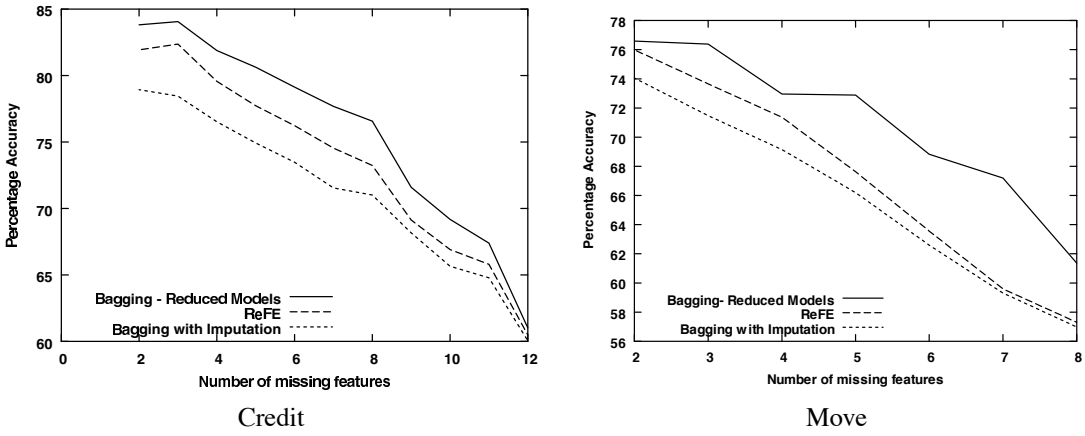


Figure 14: Performance of treatments for missing values for large ensemble models as the number of missing values increases.

values. For the larger ensembles, Figure 14 shows the classification accuracies for ReFE, bagging with imputation, and bagging with reduced models, where each ensemble includes 30 models. In general, the patterns observed for small ensembles are exhibited for larger ensembles as well.

In sum, while using no more storage space than standard bagging, ReFE offers significantly better performance than imputation and than bagging with imputation for small numbers of missing values and hence provides another alternative for domains where full-blown reduced modeling (and especially reduced modeling with bagging) is impracticably expensive. Thus, in domains in which test instances with few missing values are frequent it may be beneficial to consider the use of ReFE, resorting to reduced modeling only for (infrequent) cases with many missing values.

Finally, as desired the ReFE accuracies clearly are between the extremes, trading off accuracy and storage/computation. Clearly, ReFE models could be parameterized to allow additional points

on the tradeoff spectrum, by incorporating more reduced models. As in Section 4.1 we face a difficult optimization problem, and various heuristic approximations come to mind (e.g., somehow combining the models selected for storage in Section 4.1).

5. Related Work

Although value imputation and distribution-based imputation are common in practical applications of classification models, there is surprisingly little theoretical or empirical work analyzing the strategies. The most closely related work is the theoretical treatment by Schuurmans and Greiner's (1993) within the PAC framework (Valiant, 1984). The present paper can be seen in part as an empirical complement to their theoretical treatment. Schuurmans and Greiner consider an "attribute blocking" process in which attribute values are not available at induction time. The paper discusses instances of the three strategies we explore here: value imputation (simple default-value imputation in their paper), distribution-based prediction, and a reduced-feature "classifier lattice" of models for all possible patterns of missing values. For the missing completely at random scenario, they discuss that reduced-feature modeling is the only technique that is unconditionally consistent (i.e., is always guaranteed to converge to the optimal classifier in the large-data limit).

Our experimental results support Schuurmans and Greiner's assertion that under some conditions it is beneficial to expose the learner to the specific pattern of missing values observed in a test instance (reduced modeling), rather than to "fill in" a missing value. Our analysis gives insight into the underlying factors that lead to this advantage, particularly in terms of the statistical dependencies among the predictor variables.

Empirical work on handling missing values has primarily addressed the challenge of *induction* from incomplete training data (e.g., Rubin, 1987; Dempster et al., 1977; Schafer, 1997; Batista and Monard, 2003; Feelders, 1999; Ghahramani and Jordan, 1994, 1997). For example, Ghahramani and Jordan (1997) assume an explicit probabilistic model and a parameter estimation procedure and present a framework for handling missing values during induction when mixture models are estimated from data. Specifically for classification trees, Quinlan (1993) studies joint treatments for induction and prediction with missing nominal feature values. The study explores two forms of imputation similar to those explored here[7] and classification by simply using the first tree node for which the feature is missing (treating it as a leaf); the study does not consider reduced-feature models. Quinlan concludes that no solution dominates across domains. However, C4.5's DBI seems to perform best more often and hence the paper recommends its use.[8] Our study revealed the opposite pattern—predictive value imputation often is superior to C4.5's DBI. More importantly, however, we show that the dominance of one form of imputation versus another depends on the statistical dependencies (and lack thereof) between the features: value imputation is likely to outperform C4.5's DBI when feature imputability is particularly high, and vice versa.

Porter et al. (1990) propose a heuristic classification technique (Protos) for weak-theory domains. In contrast to the induction of an abstract generalization, Protos learns concepts by retaining exemplars, and new instances are classified by matching them with exemplars. Porter et al. apply

7. Predictive value imputation was implemented by imputing either the mode value or a prediction using a decision tree classifier.

8. In the study, some treatments for incomplete test instances are evaluated using different models that correspond to different treatments for handling incomplete training instances and therefore their relative performance cannot be compared on equal footing.

Protos, ID3, and another exemplar-based program to a medical diagnosis problem where more than 50% of the test feature values are missing, and where missingness depends on the feature values (e.g., yes/no features were always missing when the true value is "no"). They note that because of the large proportion of missing values, ID3 with various imputation techniques performed poorly. Our empirical results show a similar pattern.

To our knowledge very few studies have considered reduced-feature modeling. Friedman et al. (1996) propose the induction of lazy classification trees, an instance of run-time reduced modeling. They induce single classification-tree paths that are tailored for classifying a particular test instance, thereby not incorporating any missing features. When classifying with missing values, Friedman et al. report the performance of lazy tree induction to be superior to C4.5's technique. Explaining the results, the authors note that "avoiding any tests on unknown values is the correct thing to do probabilistically, assuming the values are truly unknown..." Our study supports this argument and complements it by showing how the statistical dependencies exhibited by relevant features are either exploited or ignored by each approach. For example, our followup analysis suggests that C4.5's technique will have particular difficulty when feature imputability is high, as it is for many benchmark data sets. Ling et al. (2004) examine strategies to reduce the total costs of feature-value acquisitions and of misclassifications; they employ lazy tree induction and show similar results to Friedman et al. Neither paper considers value imputation as an alternative, nor do they explore the domain characteristics that enable the different missing-value treatments to succeed or fail. For example, our followup analysis shows that with high feature imputability, predictive value imputation can perform just as well as lazy (reduced-feature) modeling, but reduced modeling is considerably more robust to lower levels of imputability.

We described how reduced modeling may take advantage of *alternative* predictive patterns in the training data. Prior work has noted the frequent availability of such alternative predictive patterns, and suggests that these can be exploited to induce alternative hypotheses. In particular, co-training (Blum and Mitchell, 1998) is an induction technique that relies on the assumption that the feature set comprises two disjoint subsets such that each is sufficient to induce a classifier, and that the features in each set are not highly correlated with those of the other conditional on the class. Blum and Mitchell offer web pages as an example for alternative representations, in which a page can be represented by its content or by the words occurring in hyperlinks that point to that page. Each representation can be used to induce models of comparable performance. Nigam and Ghani (2000) show that co-training is often successful because alternative representations are rather common. Specifically, Nigam and Ghani demonstrate that even for data sets for which such a natural partition does not exist, a random partition usually produces two sets that are each sufficient for accurate classification. Our empirical results for reduced models provide additional evidence that alternative feature subsets can be used effectively. Hence, accurate reduced models can frequently be induced in practice and offer an alternative that consistently is at least comparable to and usually superior to popular treatments.

6. Limitations

We consider only the MCAR setting. For practical problems there are many reasons why features may be missing, and in many cases they are not missing completely at random. To be of full practical use, analyses such as this must be extended to deal with such settings. However, as mentioned earlier, the performance of missing-value treatments for inducing classification trees seems unre-

lated to the Little and Rubin taxonomy, as long as missingness does not depend on the class value (Ding and Simonoff, 2006).

Schuurmans and Greiner (1997) consider the other end of the spectrum, missingness being a completely arbitrary function of an example's values, and conclude that none of the strategies we consider will be consistent (albeit one may perform better than another consistently in practice). However, there is a lot of ground between MCAR and completely arbitrary missingness. In the "missing at random" (MAR) scenario (Little and Rubin, 1987) missingness is conditioned only on observed values. For example, a physician may decide not to conduct one diagnostic test on the basis of the result of another. Presumably, reduced modeling would work well for MAR, since two examples with the same observed features will have the same statistical behavior on the unobserved features. If features are "missing not at random" (MNAR), there still may be useful subclasses. As a simple example, if only one particular attribute value is ever omitted (e.g., "Yes" to "Have you committed a felony?"), then unique-value imputation should work well. Practical application would benefit from a comprehensive analysis of common cases of missingness and their implications for using learned models.

Although we show some evidence of generalizability with logistic regression, our study was primarily limited to classification trees (and ensembles of trees). As noted at the outset, trees are very common both alone—especially when comprehensibility is a concern—and as components of ensembles, more sophisticated models, and larger inference systems. Some of the arguments apply to many model types, for example that reduced modeling will have lower variance. Others are specific to C4.5's DBI (which of course in the artificial intelligence and machine learning literatures is a widely used and cited missing-value treatment). C4.5's DBI is not based on an estimation of the full, joint distribution—the lack of which is the basis for the pathology presented in Section 3.3.1. However, full-joint methods also have practical drawbacks: they are very expensive computationally, can be intractable for large problems, and they are awkward for practitioners (and for researchers) in comparison to simpler classification/regression methods. Nevertheless, extending beyond classification trees, it would be well to consider DBI based on a full-joint model. (And it should be noted that naive Bayes marginalizes simply by ignoring attributes with missing values, so treatments such as these are unnecessary.)

Imputation may be more (or less) effective if we were to use other classification and regression methods. However, our arguments supporting the results are not limited a particular imputation model. In the case of multiple missing values, we have not analyzed the degree to which imputation would improve if a reduced-modeling approach were taken for the imputation itself, rather than using simple value imputation. We see no justification for doing so rather than simply using reduced modeling directly.

We avoid, as beyond the scope of this paper, the complicated question of whether there are notable interactions between the missing value treatment used at induction time and the missing value treatment used when the resultant models are applied.

Finally, we calculate feature imputability per domain, rather than per feature. Although this is sufficient for demonstrating the relationship between feature imputability and the efficacy of the various techniques, in practice it would be wise to assess imputability on a feature-by-feature basis.

7. Conclusions

Reduced-feature models are preferable both to C4.5's distribution-based imputation and to predictive value imputation. Reduced models undertake a lower-variance learning task, and do not fall prey to certain pathologies. Predictive value imputation and C4.5's DBI are easy to apply, but one almost always pays—sometimes dearly—with suboptimal accuracy.

If one must choose between C4.5's technique and predictive value imputation, the choice should be made based on the level of feature imputability, as demonstrated both by theoretical arguments and by empirical results. A lack of feature imputability is problematic for any imputation; C4.5's weighted averaging reduces estimation variance and thereby leads to more accurate estimation. High feature imputability increases the effective bias of C4.5's technique, but of course is ideal for predictive value imputation. However, even for the highest levels of feature imputability, the performance of reduced-feature modeling is indistinguishable from that of predictive value imputation. Moreover, reduced-feature modeling is substantially more robust as the level of feature imputability decreases.

Our analyses focused on suboptimalities of the imputation techniques as reasons for inferior performance. Nonetheless, these are not mistakes by the developers and users of the techniques. They are choices made to render the techniques convenient for practical use. We show the consistency and magnitude of their negative impact. In light of these results, it is clear that researchers and practitioners should choose a treatment based on a careful consideration of the relative advantages and drawbacks of the different treatments—and on the expected or estimated feature imputability.

The obvious drawback to reduced modeling is that it can be expensive either in terms of run-time computation or storage. We introduced and demonstrated several sorts of reduced-feature hybrids that allow one to manage the tradeoff between computation and storage needs or between efficiency and accuracy. Reduced-feature hybrids could be applied in various ways. Storage could be allocated to the reduced models that will see the most use or provide the most utility, and run-time computation applied for unlikely or less useful missing-data patterns. If run-time computation simply is not an option, then storage could be allocated to the most advantageous reduced models, and an imputation technique used otherwise. In the former case, the full accuracy of reduced modeling is maintained but both storage and run-time requirements are reduced from their extremes. In the latter case, accuracy is traded off for decreased storage and/or run time. The results show that even heuristic techniques for selecting the most advantageous reduced models can improve accuracy considerably. The issue of how best to choose the most advantageous reduced models is open. We also showed how ensemble methods can be modified to help deal with missing values—Reduced-Feature Ensembles—incorporating different reduced models.

Researchers and practitioners often face missing values when applying learned models. We hope this study provides a valuable step toward understanding how best to deal with them, and why.

Acknowledgments

The paper was improved by substantive comments by Haym Hirsh and the anonymous reviewers. Thanks to Eibe Frank, Tom Mitchell, and David Stork for helpful feedback, and to Jeff Simonoff and Yufeng Ding for discussions about induction with missing values. This research was supported in part by an NEC Faculty Fellowship.

References

Gustavo E. A. P. A. Batista and Maria Carolina Monard. An analysis of four missing data treatment methods for supervised learning. *Applied Artificial Intelligence*, 17(5-6):519–533, 2003.

E. Bauer and R. Kohavi. An empirical comparison of voting classification algorithms: Bagging, boosting and variants. *Machine Learning*, 36(1-2):105–139, 1999.

A. Blum and T. Mitchell. Combining labeled and unlabeled data with co-training. In *Proc. of the 11th Annual Conf. on Computational Learning Theory*, pages 92–100, Madison, WI, 1998.

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

L. Breiman, J. H. Friedman, R. Olshen, and C. Stone. *Classification and Regression Trees*. Wadsworth and Brooks, Monterey, CA, 1984.

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 B*, 39:1–38, 1977.

Y. Ding and J. Simonoff. An investigation of missing data methods for classification trees. Working paper 2006-SOR-3, Stern School of Business, New York University, 2006.

A. J. Feelders. Handling missing data in trees: Surrogate splits or statistical imputation? In *Principles of Data Mining and Knowledge Discovery*, pages 329–334, Berlin / Heidelberg, 1999. Springer. Lecture Notes in Computer Science, Vol. 1704.

J. H. Friedman, R. Kohavi, and Y. Yun. Lazy decision trees. In Howard Shrobe and Ted Senator, editors, *Proceedings of the Thirteenth National Conference on Artificial Intelligence and the Eighth Innovative Applications of Artificial Intelligence Conference*, pages 717–724, Menlo Park, California, 1996. AAAI Press.

N. Friedman and M. Goldszmidt. Learning Bayesian networks with local structure. In *Proc. of 12th Conference on Uncertainty in Artificial Intelligence (UAI-97)*, pages 252–262, 1996.

L. Getoor, N. Friedman, D. Koller, and B. Taskar. Learning probabilistic models of link structure. *Journal of Machine Learning Research*, 3:679–707, 2002.

Z. Ghahramani and M. I. Jordan. Supervised learning from incomplete data via the EM approach. In *Advances in Neural Information Processing Systems 6*, pages 120–127, 1994.

Z. Ghahramani and M. I. Jordan. Mixture models for learning from incomplete data. In R. Greiner, T. Petsche, and S.J. Hanson, editors, *Computational Learning Theory and Natural Learning Systems*, volume IV, pages 7–85. MIT Press, Cambridge, MA, 1997.

R. Greiner, A. J. Grove, and A. Kogan. Knowing what doesn't matter: Exploiting the omission of irrelevant data. *Artificial Intelligence*, 97(1-2):345–380, 1997a.

R. Greiner, A. J. Grove, and D. Schuurmans. Learning Bayesian nets that perform well. In *The Proceedings of The Thirteenth Conference on Uncertainty in Artificial Intelligence*, pages 198–207, 1997b.

Herskovits E. H. and Cooper G. F. Algorithms for Bayesian belief-network precomputation. In *Methods of Information in Medicine*, pages 362–370. 1992.

T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer Verlag, New York, August 2001.

D. Heckerman, D. M. Chickering, C. Meek, R. Rounthwaite, and C. M. Kadie. Dependency networks for inference, collaborative filtering, and data visualization. *Journal of Machine Learning Research*, 1:49–75, 2000.

R. Kohavi and G. H. John. Wrappers for feature subset selection. *Artificial Intelligence*, 97(1-2): 273–324, 1997.

N. Landwehr, M. Hall, and E. Frank. Logistic model trees. *Machine Learning*, 59(1-2):161–205, 2005.

C. X. Ling, Q. Yang, J. Wang, and S. Zhang. Decision trees with minimal costs. In *Proc. of 21st International Conference on Machine Learning (ICML-2004)*, 2004.

R. Little and D. Rubin. *Statistical Analysis with Missing Data*. John Wiley & Sons, 1987.

C. J. Merz, P. M. Murphy, and D. W. Aha. Repository of machine learning databases `http://www.ics.uci.edu/~mlearn/mlrepository.html`. Department of Information and Computer Science, University of California, Irvine, CA, 1996.

J. Neville and D. Jensen. Relational dependency networks. *Journal of Machine Learning Research*, 8:653–692, 2007.

A. Niculescu-Mizil and R. Caruana. Predicting good probabilities with supervised learning. In *Proc. of 22nd International Conference on Machine Learning (ICML-2005)*, pages 625–632, New York, NY, USA, 2005. ACM Press. ISBN 1-59593-180-5.

K. Nigam and R. Ghani. Understanding the behavior of co-training. In *Proc. of 6th Intl. Conf. on Knowledge Discovery and Data Mining (KDD-2000)*, 2000.

B. Padmanabhan, Z. Zheng, and S. O. Kimbrough. Personalization from incomplete data: what you don't know can hurt. In *Proc. of 7th Intl. Conf. on Knowledge Discovery and Data Mining (KDD-2001)*, pages 154–163, 2001.

C. Perlich, F. Provost, and J. S. Simonoff. Tree induction vs. logistic regression: a learning-curve analysis. *Journal of Machine Learning Research*, 4:211–255, 2003. ISSN 1533-7928.

B. W. Porter, R. Bareiss, and R. C. Holte. Concept learning and heuristic classification in weak-theory domains. *Artificial Intelligence*, 45:229–263, 1990.

J. R. Quinlan. Unknown attribute values in induction. In *Proc. of 6th International Workshop on Machine Learning*, pages 164–168, Ithaca, NY, June 1989.

J. R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, San Mateo, CA, 1993.

D. B. Rubin. *Multiple imputation for nonresponse in surveys*. John Wiley & Sons, New York, 1987.

J.L. Schafer. *Analysis of Incomplete Multivariate Data*. Chapman & Hall, London, 1997.

D. Schuurmans and R. Greiner. Learning to classify incomplete examples. In *Computational Learning Theory and Natural Learning Systems IV: Making Learning Systems Practical*, pages 87–105. MIT Press, Cambridge MA, 1997.

L. G. Valiant. A theory of the learnable. *Communications of the Association for Computing Machinery*, 27(11):1134–1142, 1984.

I. H. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*. Morgan Kaufmann, San Francisco, 1999.

Journal of Machine Learning Research 8 (2007) 1659-1685 Submitted 1/07; Published 7/07

Compression-Based Averaging of Selective Naive Bayes Classifiers

Marc Boullé MARC.BOULLE@ORANGE-FTGROUP.COM
France Telecom R&D
2, avenue Pierre Marzin
22300 Lannion, France

Editors: Isabelle Guyon and Amir Saffari

Abstract

The naive Bayes classifier has proved to be very effective on many real data applications. Its performance usually benefits from an accurate estimation of univariate conditional probabilities and from variable selection. However, although variable selection is a desirable feature, it is prone to overfitting. In this paper, we introduce a Bayesian regularization technique to select the most probable subset of variables compliant with the naive Bayes assumption. We also study the limits of Bayesian model averaging in the case of the naive Bayes assumption and introduce a new weighting scheme based on the ability of the models to conditionally compress the class labels. The weighting scheme on the models reduces to a weighting scheme on the variables, and finally results in a naive Bayes classifier with "soft variable selection". Extensive experiments show that the compression-based averaged classifier outperforms the Bayesian model averaging scheme.

Keywords: naive Bayes, Bayesian, model selection, model averaging

1. Introduction

The naive Bayes classification approach (see Langley et al., 1992; Mitchell, 1997; Domingos and Pazzani, 1997; Hand and Yu, 2001) is based on the assumption that the variables are independent within each output label, and simply relies on the estimation of univariate conditional probabilities. The evaluation of the probabilities for numeric variables has already been discussed in the literature (see Dougherty et al., 1995; Liu et al., 2002; Yang and Webb, 2002). Experiments demonstrate that even a simple equal width discretization brings superior performance compared to the assumption using a Gaussian distribution.

The naive independence assumption can harm the performance when violated. In order to better deal with highly correlated variables, the selective naive Bayes approach of Langley and Sage (1994) exploits a wrapper approach (see Kohavi and John, 1997) to select the subset of variables which optimizes the classification accuracy. In the method of Boullé (2006a), the area under the receiver operating characteristic (ROC) curve (see Fawcett, 2003) is used as a selection criterion and exhibits a better predictive performance than the accuracy criterion.

Although the selective naive Bayes approach performs quite well on data sets with a reasonable number of variables, it does not scale on very large data sets with hundreds of thousands of instances and thousands of variables, such as in marketing applications. The problem comes both from the search algorithm, whose complexity is quadratic in the number of the variables, and from the selection process which is prone to overfitting.

In this paper, we present a new regularization technique to compromise between the number of selected variables and the performance of the classifier. The resulting variable selection criterion is optimized owing to an efficient search heuristic whose computational complexity is $O(KN\log(KN))$, where N is the number of instances and K the number of variables. We also apply the Bayesian model averaging approach of Hoeting et al. (1999) and extend it with a compression-based averaging scheme, which better accounts for the distribution of the models. We show that averaging the models turns into averaging the contribution of the variables in the case of the selective naive Bayes classifier. Finally we proceed with extensive experiments to evaluate our method.

The remainder of the paper[1] is organized as follows. Section 2 introduces the assumptions and recalls the principles of the naive Bayes and selective naive Bayes classifiers. Section 3 presents the regularization technique for variable selection based on Bayesian model selection and Section 4 applies the Bayesian model averaging method to selective naive Bayes classifiers. In Section 5, the new selective naive Bayes classifiers are evaluated on an illustrative example. Section 6 analyzes the limits of Bayesian model averaging and proposes a new model averaging technique based on model compression coefficients. Section 7 proceeds with extensive experimental evaluations and Section 8 reports the results obtained in the performance prediction challenge organized by Guyon et al. (2006c). Finally, Section 9 concludes this paper and outlines research directions.

2. Selective Naive Bayes Classifier

This section formally states the assumptions and notations and recalls the naive Bayes and selective naive Bayes approaches.

2.1 Assumptions and Notation

Let $X = (X_1, X_2, \ldots X_K)$ be the vector of the K explanatory variables and Y the class variable. Let $\lambda_1, \lambda_2, \ldots \lambda_J$ be the J class labels of Y.

Let N be the number of instances and $D = \{D_1, D_2, \ldots, D_N\}$ the labeled database containing the instances $D_n = (x^{(n)}, y^{(n)})$.

Let $\mathcal{M} = \{M_m\}$ be the set of all the potential selective naive Bayes models. Each model M_m is described by K parameter values a_{mk}, where a_{mk} is 1 if variable k is selected in model M_m and 0 otherwise.

Let us denote by $P(\lambda_j)$ the prior probabilities $P(Y = \lambda_j)$ of the class values, and by $P(X_k|\lambda_j)$ the conditional probability distributions $P(X_k|Y = \lambda_j)$ of the explanatory variables given the class values.

We assume that the prior probabilities $P(\lambda_j)$ and the conditional probability distributions $P(X_k|\lambda_j)$ are known, once the preprocessing is performed.

In the paper, the class conditional probabilities are estimated using the MODL discretization method of Boullé (2006c) for the numeric variables and the MODL grouping method of Boullé (2005a,b) for the categorical variables. MODL stands for minimum optimized description length and refers to the principle of minimum description length (MDL) of Rissanen (1978) as a model selection technique. More specifically, the MODL preprocessing methods exploit a maximum a posteriori (MAP) technique (see Robert, 1997) to select the most probable model of discretization

1. This paper is an extended version of the 2006 IJCNN conference paper (Boullé, 2006b).

(resp. value grouping) given the input data. The choice of the prior distribution of the models is *optimized* for the task of data preparation, and the search algorithms are deeply *optimized*.

Using the Bayes optimal MODL preprocessing methods to estimate the conditional probabilities has proved to be very efficient in detecting irrelevant variables (see Boullé, 2006a). In the experimental section, the $P(\lambda_j)$ are estimated by counting and the $P(X_k|\lambda_j)$ are computed using the contingency tables, resulting from the preprocessing of the explanatory variables. The conditional probabilities are estimated using a m-estimate $(support + mp)/(coverage + m)$ with $m = J/N$ and $p = 1/J$, in order to avoid zero probabilities.

2.2 Naive Bayes Classifier

The naive Bayes classifier assigns to each instance the class value having the highest conditional probability

$$P(\lambda_j|X) = \frac{P(\lambda_j)P(X|\lambda_j)}{P(X)}.$$

Using the assumption that the explanatory variables are independent conditionally to the class variable, we get

$$P(\lambda_j|X) = \frac{P(\lambda_j)\prod_{k=1}^{K} P(X_k|\lambda_j)}{P(X)}. \quad (1)$$

In classification problems, Equation (1) is sufficient to predict the most probable class given the input data, since $P(X)$ is constant. In problems where a prediction score is needed, the class conditional probability can be estimated using

$$P(\lambda_j|X) = \frac{P(\lambda_j)\prod_{k=1}^{K} P(X_k|\lambda_j)}{\sum_{i=1}^{J} P(\lambda_i)\prod_{k=1}^{K} P(X_k|\lambda_i)}. \quad (2)$$

The naive Bayes classifier is poor at predicting the true class conditional probabilities, since the independence assumption is usually violated in real data applications. However, Hand and Yu (2001) show that the prediction score given by Equation (2) often provides an effective ranking of the instances for each class value.

2.3 Selective Naive Bayes Classifier

The selective naive Bayes classifier reduces the strong bias of the naive independence assumption, owing to variable selection. The objective is to search among all the subsets of variables, in order to find the best possible classifier, compliant with the naive Bayes assumption.

Langley and Sage (1994) propose to evaluate the selection process with the accuracy criterion, estimated on the train data set. However, this criterion suffers from some limits, even when the predictive performance is the only concern. In case of a skewed distribution of class labels for example, the accuracy may never be better than the majority accuracy, so that the selection process ends with an empty set of variables. This problem also arises when several consecutive selected variables are necessary to improve the accuracy. In the method proposed by Langley and Sage (1994), the selection process is iterated as long as there is no decay in the accuracy. This solution raises new problems, such as the selection of irrelevant variables with no effect on accuracy, or even the selection of redundant variables with either insignificant effect or no effect on accuracy.

Provost et al. (1998) propose to use receiver operating characteristic (ROC) analysis rather than the accuracy to evaluate induction models. This ROC criterion, estimated on the train data set (as in Langley and Sage, 1994), is used by Boullé (2006a) to assess the quality of variable selection for naive Bayes classifier. The method exploits the forward selection algorithm to select the variables, starting from an empty subset of variables. At each step of the algorithm, the variable which brings the best increase of the area under the ROC curve (AUC) is chosen and the selection process stops as soon as this area does not rise anymore. This allows capturing slight enhancements in the learning process and helps avoiding the selection of redundant variables or probes that have no effect on the ROC curve.

Altogether, the variable selection method can be implemented in $O(K^2 N \log N)$ time. The preprocessing step needs $O(KN \log N)$ to discretize or group the values of all the variables. The forward selection process requires $O(K^2 N \log N)$ time, owing to the decomposability of the naive Bayes formula on the variables. The $O(N \log N)$ term in the complexity is due to the evaluation of the area under the ROC curve, based on the sort of the training instances.

3. MAP Approach for Variable Selection

After introducing the aim of regularization, this section applies the Bayesian approach to derive a new evaluation criterion for variable selection and presents the search algorithm used to optimize this criterion.

3.1 Introduction

The naive Bayes classifier is a very robust algorithm. It can hardly overfit the data, since no hypothesis space is explored during the learning process. On the opposite, the selective naive Bayes classifier explores the space of all subsets of variables to reduce the strong bias of the naive independence assumption. The size of the searched hypothesis space grows exponentially with the number of variables, which might cause overfitting. Experiments show that during the variable selection process, the last added variables raise the "complexity" of the classifier while having an insignificant impact on the evaluation criterion (AUC for example). These slight improvements during the training step, which have an insignificant impact on the test performance, are detrimental to the ease of deployment of the models and to their understandability.

We propose to tackle this overfitting problem by relying on a Bayesian approach, where the MAP model is found by maximizing the probability $P(Model|Data)$ of the model given the data. In the following, we describe how we compute the likelihood of the models $P(Data|Model)$ and propose a prior distribution $P(Model)$ for variable selection.

3.2 Likelihood of Models

For a given model M_m parameterized by the set of selected variable indicators $\{a_{mk}\}$, the estimation of the class conditional probability $P_m(\lambda_j|X)$ turns into

$$\begin{aligned} P_m(\lambda_j|X) &= \frac{P(\lambda_j)\prod_{k=1}^{K} P(X_k|\lambda_j)^{a_{mk}}}{P(X)} \\ &= \frac{P(\lambda_j)\prod_{k=1}^{K} P(X_k|\lambda_j)^{a_{mk}}}{\sum_{i=1}^{J} P(\lambda_i)\prod_{k=1}^{K} P(X_k|\lambda_i)^{a_{mk}}}. \end{aligned} \quad (3)$$

Equation (3) provides the class conditional probability distribution for each model M_m on the basis of the parameter values a_{mk} of the model. For a given instance D_n, the probability of observing the class value $y^{(n)}$ given the explanatory values $x^{(n)}$ and given the model M_m is $P_m(Y = y^{(n)}|X = x^{(n)})$. The likelihood of the model is obtained by computing the product of these quantities on the whole data set. The negative log-likelihood of the model is given by

$$-\log P(D|M_m) = \sum_{n=1}^{N} -\log P_m(Y = y^{(n)}|X = x^{(n)}).$$

3.3 Prior for Variable Selection

The parameters of a variable selection model M_m are the Boolean values a_{mk}. We propose a hierarchic prior, by first choosing the number of selected variables and second choosing the subset of selected variables.

For the number K_m of variables, we propose to use a uniform prior between 0 and K variables, representing $(K+1)$ equiprobable alternatives.

For the choice of the K_m variables, we assign the same probability to every subset of K_m variables. The number of combinations $\binom{K}{K_m}$ seems the natural way to compute this prior, but it has the disadvantage of being symmetric. Beyond $K/2$ variables, every new variable makes the selection more probable. Thus, adding irrelevant variables is favored, provided that this has an insignificant impact on the likelihood of the model. As we prefer simpler models, we propose to use the number of combinations with replacement $\binom{K+K_m-1}{K_m}$.

Taking the negative log of this prior, we get the following code length $l(M_m)$ for the variable selection models

$$l(M_m) = \log(K+1) + \log \binom{K+K_m-1}{K_m}.$$

Using this prior, the "informational cost" of the first selected variables is about $\log K$ and about $\log 2$ for the last variables.

To summarize our prior, each number of K_m variable is equiprobable, and for a given K_m, each subset of K_m variables randomly chosen with replacement is equiprobable. This means that each specific small subset of variables has a greater probability than each specific large subset of variables, since the number of variable subsets of given size grows with K_m.

3.4 Posterior Distribution of the Models

The posterior probability of a model M_m is evaluated as the product of the prior and the likelihood. This is equivalent to the MDL approach of Rissanen (1978), where the code length of the model plus the data given the model has to be minimized:

$$l(M_m) + l(D|M_m) = \log(K+1) + \log \binom{K+K_m-1}{K_m} - \sum_{n=1}^{N} \log P_m(y^{(n)}|X = x^{(n)}). \tag{4}$$

The first two terms encode the complexity of the model and the last one the fit of the data. The compromise is found by minimizing this criterion.

We can notice a trend of increasing attention to the predicted probabilities in the evaluation criteria proposed for variable selection. Whereas the accuracy criterion focuses only on the majority class and the area under the ROC curve evaluates the correct ordering of the predicted probabilities,

our regularized criterion evaluates the correctness of all the predicted probabilities (not only their rank) and introduces a regularization term to balance the complexity of the models.

3.5 An Efficient Search Heuristic

Many heuristics have been used for variable selection (see Guyon et al., 2006b). The greedy forward selection heuristic evaluates all the variables, starting from an empty set of variables. The best variable is added to the current selection, and the process is iterated until no new variable improves the evaluation criterion. This heuristic may fall in local optima and has a quadratic time complexity with respect to the number of variables. The forward backward selection heuristic allows to add or drop one variable at each step, in order to avoid local optima. The fast forward selection heuristic evaluates each variable one at a time, and adds it to the selection as soon as this improves the criterion. This last heuristic is time effective, but its results exhibit a large variance caused by the dependence over the order of the variables.

Algorithm 1 Algorithm MS(FFWBW)

Require: $X \leftarrow (X_1, X_2, \ldots X_K)$ {Set of input variables}
Ensure: B {Best subset of variables}
$B \leftarrow \emptyset$ {Start with an empty subset of variables}
for Step=1 to $\log_2 KN$ **do**
  {Fast forward backward selection}
  $S \leftarrow \emptyset$ {Initialize an empty subset of variables}
  $Iter \leftarrow 0$
  repeat
    $Iter \leftarrow Iter + 1$
    $X' \leftarrow$ Shuffle(X) {Randomly reorder the variables to add}
    {Fast forward selection}
    for $X_k \in X'$ **do**
      if $cost(S \cup \{X_k\}) < cost(S)$ **then**
        $S \leftarrow S \cup \{X_k\}$
      end if
    end for
    $X' \leftarrow$ Shuffle(X) {Randomly reorder the variables to remove}
    {Fast backward selection}
    for $X_k \in X'$ **do**
      if $cost(S - \{X_k\}) < cost(S)$ **then**
        $S \leftarrow S - \{X_k\}$
      end if
    end for
  until no improvement or $Iter \geq MaxIter$
  {Update best subset of variables}
  if $cost(S) < cost(B)$ **then**
    $B \leftarrow S$
  end if
end for

We introduce a new search heuristic called fast forward backward selection (FFWBW), based on a mix of the preceding approaches. It consists in a sequence of fast forward selection and fast backward selection steps. The variables are randomly reordered between each step, and evaluated only once during each forward or backward search. This process is iterated as long as two successive (forward and backward) search steps bring at least one improvement of the criterion or when the iteration number exceeds a given parameter $MaxIter$. In practice, the whole process converges very quickly, in one or two steps in most of the cases. Setting $MaxIter = 5$ for example is sufficient to bound the worst case complexity without decreasing the quality of the search algorithm.

Evaluating a selective naive Bayes model requires $O(KN)$ computation time, mainly to evaluate all the class conditional probabilities. According to Equation (3), these class conditional probabilities can be updated in $O(1)$ per instance and $O(N)$ for the whole data set when one variable is added or removed from the current subset of selected variables. Each fast forward selection or fast backward selection step considers $O(K)$ additions or removals of variables and requires $O(KN)$ computation time. The total time complexity of the FFWBW heuristic is $O(KN)$, since the number of search steps is bounded by the constant parameter $MaxIter$.

In order to further reduce both the possibility of local optima and the variance of the results, this FFWBW heuristic is embedded into a multi-start (MS) algorithm, by repeating the search heuristic starting from several random orderings of the variables. The number of repetitions is set to $\log_2 KN$, which offers a reasonable compromise between time complexity and quality of the optimization. Overall, the time complexity of the MS(FFWBW) heuristic is $O(KN \log KN)$. The heuristic is detailed in Algorithm 1.

4. Bayesian Model Averaging of Selective Naive Bayes Classifiers

Model averaging consists in combining the prediction of an ensemble of classifiers in order to reduce the prediction error. This section reminds the principles of Bayesian model averaging and applies this averaging scheme to the selective naive Bayes classifier.

4.1 Bayesian Model Averaging

The Bayesian model averaging (BMA) method (Hoeting et al., 1999) aims at accounting for the model uncertainty. Whereas the MAP approach retrieves the most probable model given the data, the BMA approach exploits every model in the model space, weighted by their posterior probability. This approach relies on the definition of a prior distribution on the models, on an efficient computation technique to estimate the model posterior probabilities and on an effective method to sample the posterior distribution. Apart from these technical difficulties, the BMA approach is an appealing technique, with strong theoretical results concerning the optimality of its long-run performance, as shown by Raftery and Zheng (2003).

The BMA approach has been applied to the naive Bayes classifier by Dash and Cooper (2002). Apart from the differences in the weighting scheme, their method (DC) differs from ours mainly on the initial assumptions. The DC method does not manage the numeric variables and assumes multinomial distributions with Dirichlet priors for the categorical variables, which requires the choice of hyper-parameters for each variable. Structure modularity of the Bayesian network is also assumed: each selection of a variable is independent from the others. The DC approach estimates the full data distribution (explanatory and class variables), whereas we focus on the class conditional probabilities. Once the prior hyper-parameters are fixed, the DC method allows to compute an exact model

averaging, whereas we rely on an heuristic to estimate the averaged model. Compared to the DC method, our method is not restricted to categorical attributes and does not need any hyper-parameter.

4.2 From Bayesian Model Averaging to Expectation

For a given variable of interest Δ, the BMA approach averages the predictions of all the models weighted by their posterior probability.

$$P(\Delta|D) = \sum_m P(\Delta|M_m, D)P(M_m|D).$$

This formula can be written, using only the prior probabilities and the likelihood of the models.

$$P(\Delta|D) = \frac{\sum_m P(\Delta|M_m, D)P(M_m)P(D|M_m)}{\sum_m P(M_m)P(D|M_m)}.$$

Let $f(M_m, D) = P(\Delta|M_m, D)$ and $f(D) = P(\Delta|D)$. Using these notations, the BMA formula can be interpreted as the expectation of function f for the posterior distribution of the models

$$E(f) = \sum_m f(M_m, D)P(M_m|D).$$

We propose to extend the BMA approach in the case where f is not restricted to be a probability function.

4.3 Expectation of the Class Conditional Information

The selective naive Bayes classifier provides an estimation of the class conditional probabilities. These estimated probabilities are the natural candidates for averaging. For a given model M_m defined by the variable selection $\{a_{mk}\}$, we have

$$f(M_m, D) = \frac{P(Y)\prod_{k=1}^{K} P(X_k|Y)^{a_{mk}}}{P(X)}. \tag{5}$$

Let $I(M_m, D) = -\log f(M_m, D)$ be the class conditional information. Whereas the expectation of f relates to a (weighted) arithmetic mean of the class conditional probabilities, the expectation of I relates to a (weighted) geometric mean of these probabilities. This puts more emphasis on the magnitude of the estimated probabilities. Taking the negative log of (5), we obtain

$$I(M_m, D) = I(Y) - I(X) + \sum_{k=1}^{K} a_{mk} I(X_k|Y). \tag{6}$$

We are looking for the expectation of this conditional information

$$\begin{aligned} E(I) &= \frac{\sum_m I(M_n, D)P(M_m|D)}{\sum_m P(M_m|D)} \\ &= I(Y) - I(X) + \frac{\sum_m P(M_m|D)\sum_{k=1}^{K} a_{mk} I(X_k|Y)}{\sum_m P(M_m|D)} \\ &= I(Y) - I(X) + \sum_{k=1}^{K} I(X_k|Y)\frac{\sum_m a_{mk} P(M_m|D)}{\sum_m P(M_m|D)}. \end{aligned}$$

Let $b_k = \frac{\sum_m a_{mk} P(M_m|D)}{\sum_m P(M_m|D)}$. We have $b_k \in [0,1]$.

The b_k coefficients are computed using (4), on the basis of the prior probabilities and of the likelihood of the models. Using these coefficients, the expectation of the conditional information is

$$E(I) = I(Y) - I(X) + \sum_{k=1}^{K} b_k I(X_k|Y). \tag{7}$$

The averaged model thus provides the following estimation for the class conditional probabilities:

$$P(Y|X) = \frac{P(Y)\prod_{k=1}^{K} P(X_k|Y)^{b_k}}{P(X)}.$$

It is noteworthy that the expectation of the conditional information in (7) is similar to the conditional information estimated by each individual model in (6). The weighting scheme on the models reduces to a weighting scheme on the variables. When the MAP model is selected, the variables have a weight of 1 when selected and 0 otherwise: this is a "hard selection" of the variables. When the above averaging scheme is applied, each variable has a $[0,1]$ weight, which can be interpreted as a "soft selection".

4.4 An Efficient Algorithm for Model Averaging

We have previously introduced a model averaging method which relies on the expectation of the class conditional information. The calculation of this expectation requires the evaluation of all the variable selection models, which is not computationally feasible as soon as the number of variables goes beyond about 20. This expectation can heuristically be evaluated by sampling the posterior distribution of the models and accounting only for the sampled models in the weighting scheme.

We propose to reuse the MS(FFWBW) search heuristic to perform this sampling. This heuristic is effective for finding high probability models and searching in their neighborhood. The repetition of the search from several random starting points (in the multi-start meta-heuristics) brings diversity and allows to escape local optima. We use the whole set of models evaluated during the search to estimate the expectation.

Although this sampling strategy is biased by the search heuristic, it has the advantage of being simple and computationally tractable. The overhead in the time complexity of the learning algorithm is negligible, since the only need is to collect the posterior probability of the models and to compute the weights in the averaging formula. Concerning the deployment of the averaged model, the overhead is also negligible, since the initial naive Bayes estimation of the class conditional probabilities is just extended with variable weights.

5. Evaluation on an Illustrative Example

This section describes the waveform data set, introduces three evaluation criteria and illustrates the behavior of each variant of the selective naive Bayes classifier.

5.1 The Waveform Data Set

The waveform data set introduced by Breiman et al. (1984) contains 5000 instances, 21 continuous variables and 3 equidistributed class values. Each instance is defined as a linear combination of two

out of the three triangular waveforms pictured in Figure 1, with randomly generated coefficients and Gaussian noise. Figure 2 plots 10 random instances from each class.

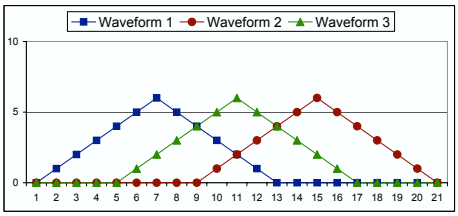


Figure 1: Basic waveforms used to generated the 21 input variables of the waveform data set.

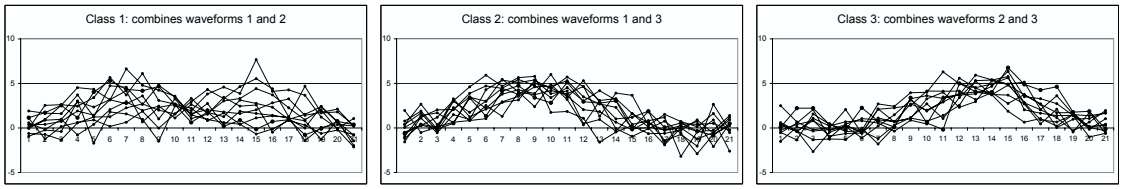


Figure 2: Waveform data.

Learning on the waveform data set is generally considered a difficult task in pattern recognition, with reported accuracy of 86.8% using a Bayes optimal classifier. The input variables are correlated, which violates the naive Bayes assumption. Selecting the best subset of variables compliant with the naive Bayes assumption is a challenging problem.

5.2 The Evaluation Criteria

We evaluate three criteria of increasing complexity: accuracy (ACC), area under the ROC curve (AUC) and informational loss function (ILF).

The ACC criterion evaluates the accuracy of the prediction, no matter whether its conditional probability is 51% or 100%.

The AUC criterion (see Fawcett, 2003) evaluates the ranking of the class conditional probabilities. In a two-classes problem, the AUC is equivalent to the probability that the classifier will rank a randomly chosen positive instance higher than a randomly chosen negative instance. Extending the AUC criterion to multi-class problems is not a trivial task and has lead to computationally intensive methods (see for example Deng et al., 2006). In our experiments, we use the approach of Provost and Domingos (2001) to calculate the multi-class AUC, by computing each one-against-the-others two-classes AUC and weighting them by the class prior probabilities $P(\lambda_j)$. Although this version of multi-class AUC is sensitive to the class distribution, it is easy to compute, which motivates our choice.

The ILF criterion (see Witten and Frank, 2000) evaluates the probabilistic prediction owing to the negative log of the predicted class conditional probabilities

$$-\log P_m(Y = y^{(n)}|X = x^{(n)}).$$

The empirical mean of the ILF criterion is equal to

$$\overline{ILF(M_m)} = \frac{1}{N}\sum_{n=1}^{N} -\log P_m(Y = y^{(n)}|X = x^{(n)}).$$

The predicted class conditional probabilities in the ILF criterion are given by Equation (2) for the naive Bayes classifier and by Equation (3) for the selective naive Bayes classifier.

Let $M_\emptyset$ be the "null" model, with no variable selected. The null model estimates the class conditional probabilities by their prior probabilities, ignoring all the explanatory variables. For the null model $M_\emptyset$, we obtain

$$\begin{aligned}\overline{ILF(M_\emptyset)} &= \frac{1}{N}\sum_{n=1}^{N} -\log P(Y = y^{(n)}) \\ &= -\sum_{j=1}^{J} P(\lambda_j)\log P(\lambda_j) \\ &= H(Y),\end{aligned}$$

where $H(Y)$ is the entropy of Shannon (1948) of the class variable.

We introduce a compression rate to normalize the ILF criterion using

$$\begin{aligned}CR(M_m) &= 1 - \overline{ILF(M_m)} \,/\, \overline{ILF(M_\emptyset)} \\ &= 1 - \overline{ILF(M_m)} \,/\, H(Y).\end{aligned}$$

The normalized CR criterion is mainly ranged between 0 (prediction not better than the basic prediction of the class priors) and 1 (prediction of the true class probabilities in case of perfectly separable classes). It can be negative when the predicted probabilities are worse than the basic prior predictions.

5.3 Evaluation on the Waveform Data Set

We use 70% of the waveform data set to train the classifiers and 30% to test them. These evaluations are merely illustrative; extensive experiments are reported in Section 7.

In the case of the waveform data set, the MODL preprocessing method determines that 2 variables (1^{st} and 21^{st}) are irrelevant, and the naive Bayes classifier uses all the 19 remaining variables. We evaluate four variants of selective naive Bayes methods. The SNB(ACC) method of Langley and Sage (1994) optimizes the train accuracy and the SNB(AUC) method of Boullé (2006a) optimizes the area under the ROC curve on the train data set. The SNB(MAP) method introduced in Section 3 selects the most probable subset of variables compliant with the naive Bayes assumption, and the SNB(BMA) method described in Section 4 averages the selective naive Bayes classifiers weighted by their posterior probability. In this experiment, we evaluate exhaustively the half a million models related to the 2^{19} possible variable subsets. This allows us to focus on the variable selection criterion and to avoid the potential bias of the optimization algorithms.

The selected subsets of variables are pictured in Figure 3. The SNB(ACC) method selects 12 variables and the SNB(AUC) 18 variables. The SNB(MAP) which focuses on a subset of variables compliant with the naive Bayes assumption selects only 8 variables, which turns out to be a subset of the variables selected by the alternative methods.

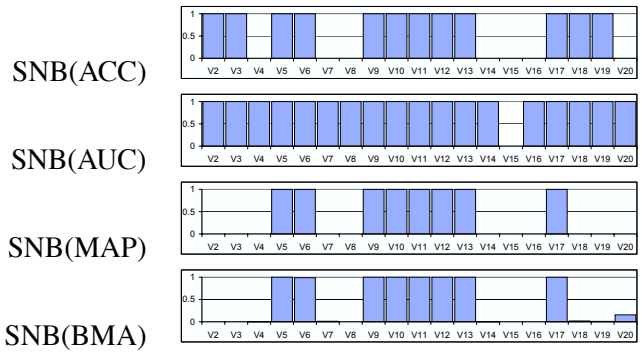


Figure 3: Variables selected by the selective naive Bayes classifiers for the waveform data set.

The predictive performance for the ACC, AUC and ILF criteria are reported in Figure 4. In multi-criteria analysis, a solution *dominates* (or is *non-inferior* to) another one if it is better for all criteria. A solution that cannot be dominated is *Pareto* optimal: any improvement of one of the criteria causes a deterioration on another criterion (see Pareto, 1906). The *Pareto surface* is the set of all the Pareto optimal solutions.

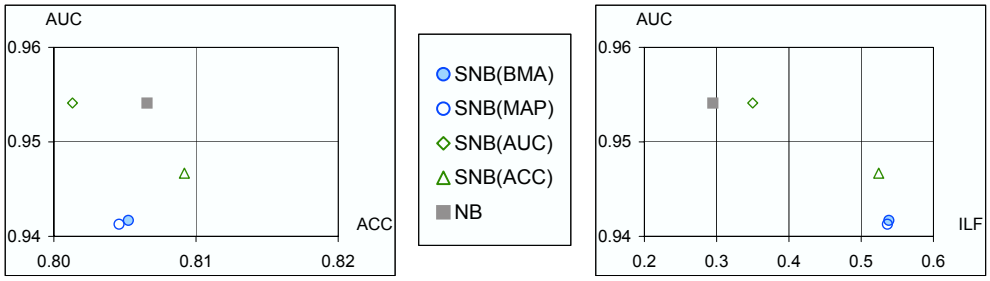


Figure 4: Evaluation of the predictive performance of selective naive Bayes classifiers on the waveform data set.

The SNB(ACC) method is slightly better than the NB method on the ACC criterion. Owing to its small subset of selected variables, it manages to reduce the redundancy between the variables and to significantly improve the estimation of the class conditional probabilities, as reported by its ILF evaluation. The SNB(AUC) method gets the same AUC performance as the NB method with one variable less. The SNB(MAP) and SNB(BMA) methods almost directly optimizes the ILF criterion on the train data set, with a regularization term related to the model prior. They get the best ILF evaluation on the test set, but are dominated by the NB and SNB(ACC) methods on the two other criteria, as shown in Figure 4. Almost all the methods are Pareto optimal: none of them is the best on the three evaluated criteria.

Compared to the other variable selection methods, the SNB(MAP) truly focuses on complying with the naive independence assumption. This results in a much smaller subset of variables and a better prediction of the class conditional probabilities, at the expense of a decay on the other criteria.

The SNB(BMA) method exploits a model averaging approach which results in soft variable selection. Figure 3 shows the weights of each variable. Surprisingly, the selected variables are almost the same as the 8 variables selected by the SNB(MAP) method. Compared to the hard variable selection scheme, the soft variable selection exhibits mainly one minor change: a new variable (V20) is selected with a small weight of 0.15. The other modifications of the variable weights are insignificant: two variables (V6 and V17) decrease their weight from 1.0 to 0.99 and three variables (V7, V18 and V19) appear with a tiny weight of 0.01.

Since the variable selection is almost the same as in the MAP approach, the model averaging approach does not bring any significant improvement in the evaluation results, as shown in Figure 4.

6. Compression Model Averaging of Selective Naive Bayes Classifiers

This section analyzes the limits of Bayesian model averaging and proposes a new weighting scheme that better exploits the posterior distributions of the models.

6.1 Understanding the Limits of Bayesian Model Averaging

We use again the waveform data set to explain why the Bayesian model averaging method fails to outperform the MAP method.

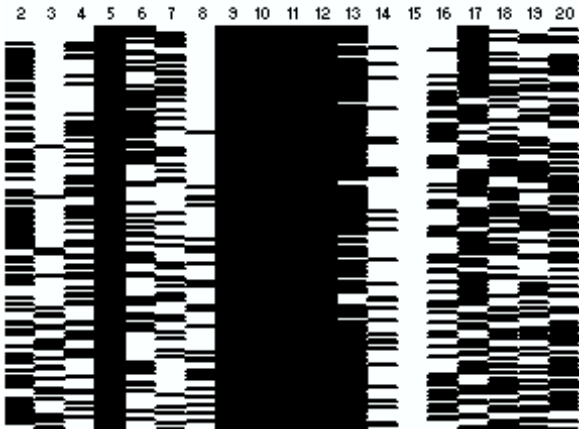


Figure 5: Index of the selected variables in the 200 most probable selective naive Bayes models for the waveform data set. Each line represents a model, where the variables are in black color when selected.

The variable selection problem consists in finding the most probable subset of variables compliant with the naive Bayes assumption among about half a million (2^{19}) potential subsets. In order to study the posterior distribution of the models, all these subsets are evaluated. The MAP model selects 8 variables (V5, V6, V9, V10, V11, V12, V13, V17). A close look at the posterior distribution shows that most of the good models (in the top 50%) contain around 10 variables. Figure 5 displays the selected variables in the top 200 models (0.05%). Five variables (V5, V9, V10, V11,

V12) among the 8 MAP variables are always selected, and the other models exploit a diversity of subsets of variables. The potential benefit of model averaging is to account for all these models, with higher weights for the most probable models.

However, the posterior distribution is very sharp everywhere, not only around the MAP. Variable V18 is first selected in the 3^{rd} model, which is about 40 times less probable than the MAP model. Variable V4 is first selected in the 10^{th} model, about 4000 times less probable than the MAP model. Figure 6 displays the repartition function of the posterior probabilities, using a log scale. Using this logarithmic transformation, the posterior distribution is flattened and can be visualized. The MAP model is 10^{1033} times more probable than the minimum a posteriori model, which selects no variable.

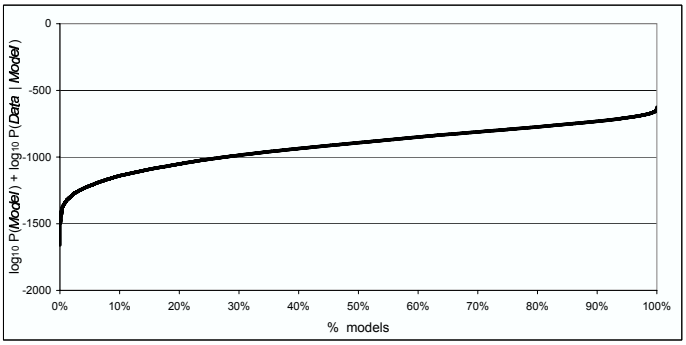


Figure 6: Repartition function of the posterior probabilities of half a million variable selection models evaluated for the waveform data set, sorted by increasing posterior probability. For example, the 10% models on the left represent the models having the lowest posterior probability.

In the waveform example, averaging using the posterior probabilities to weight the models is almost the same as selecting the MAP model (which itself is hard to find with a heuristic search) and model averaging is almost useless. In theory, BMA is optimal (see Raftery and Zheng, 2003), but this optimality result assumes that the true distribution of the data belongs to the space of models. In the case of the selective naive Bayes classifier, this assumption is violated on most real data sets and BMA fails to build effective model averaging.

6.2 Model Averaging with Compression Coefficients

When the posterior distribution is sharply peaked around the MAP, averaging is almost the same as selecting the MAP model. These peaked posterior distributions are more and more likely to happen when the number of instances rises, since a few tens of instances better classified by a model are sufficient to increase its likelihood by several orders of magnitude. Therefore, the algorithmic overhead is not valuable if averaging turns out to be the same as selecting the MAP.

In order to have a theoretical insight on the relation between MAP and BMA, let us analyze again the model selection criterion (4). It is closely related to the ILF criterion described in Section 5.2, according to

$$l(M_m) + l(D|M_m) = -\log P(M_m) + N\overline{ILF(M_m)}.$$

For the the "null" model $M_\emptyset$, with no variable selected, we have:

$$l(M_\emptyset) + l(D|M_\emptyset) = -\log P(M_\emptyset) + NH(Y).$$

The posterior probability $P(M_m|D)$ of a model M_m relative to that of the null model is

$$\frac{P(M_m|D)}{P(M_\emptyset|D)} = \frac{P(M_m)}{P(M_\emptyset)} \left(\frac{H(Y)}{\overline{ILF(M_m)}} \right)^N. \tag{8}$$

Equation (8) shows that the posterior probability of the models is exponentially peaked when N goes to infinity. Small improvements in the estimation of the conditional entropy brings very large differences in the posterior probability of the models, which explains why Bayesian model averaging is asymptotically equivalent to selecting the MAP model.

We propose an alternative weighting scheme, whose objective is to better account for the set of all models. Let us precisely define the compression coefficient $c(M_m, D)$ of a model. The model selection criterion $l(M_m) + l(D|M_m)$ defined in Equation (4) represents the quantity of information required to encode the model plus the class values given the model. The code length of the null model $M_\emptyset$ can be interpreted as the quantity of information necessary to describe the classes, when no explanatory data is used to induce the model.

Each model M_m can potentially exploit the explanatory data to better "compress" the class conditional information. The ratio of the code length of a model to that of the null model stands for a relative gain in compression efficiency. We define the compression coefficient $c(M_m, D)$ of a model as follows:

$$c(M_m, D) = 1 - \frac{l(M_m) + l(D|M_m)}{l(M_\emptyset) + l(D|M_\emptyset)}.$$

The compression coefficient is 0 for the null model, is maximal when the true class conditional probabilities are correctly estimated and tends to 1 in case of separable classes. This coefficient can be negative for models which provide an estimation worse than that of the null model.

In our heuristic attempt to better account for all the models, we replace the posterior probabilities by their related compression coefficient in the weighting scheme.

Let us focus again on the variable weights b_k introduced in Section 4 in our first model averaging method. Dividing the posterior probabilities by those of the null model, we get

$$b_k = \frac{\sum_m a_{mk} \frac{P(M_m|D)}{P(M_\emptyset|D)}}{\sum_m \frac{P(M_m|D)}{P(M_\emptyset|D)}}.$$

We introduce new c_k coefficients by taking the log of the probability ratios and normalizing by the code length of the null model. We obtain

$$c_k = \frac{\sum_m a_{mk} c(M_m, D)}{\sum_m c(M_m, D)}.$$

Mainly, the principle of this new heuristic weighting scheme consists in smoothing the exponentially peaked posterior probability distribution of Equation (8) with the log function.

In the implementation, we ignore the "bad" models and consider the positive compression coefficients only. We evaluate the compression based model averaging (CMA) model using the model averaging algorithm introduced in Section 4.4.

6.3 Evaluation on the Waveform Data Set

We use the protocol introduced in Section 5 to evaluate the SNB(CMA) compression model averaging method on the waveform data set, with an exhaustive evaluation of all the models to avoid the potential bias of the optimization algorithms.

Figure 7 shows the weights of each variable resulting from the soft variable selection of the SNB(CMA) compression model averaging method. Contrary to the SNB(BMA) method, the averaging has a significant impact on variable selection.

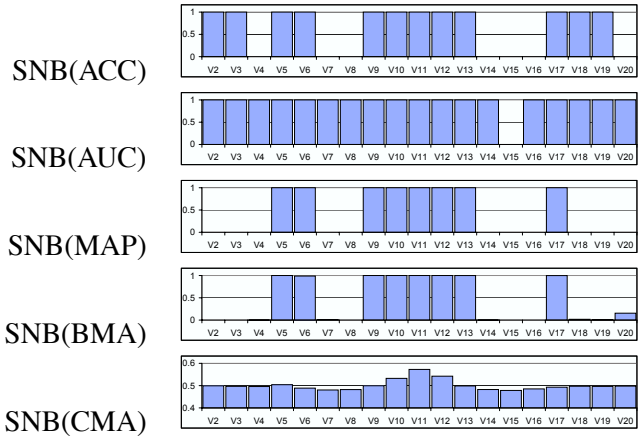


Figure 7: Variables selected by the SNB(CMA) method and the alternative selective naive Bayes classifiers for the waveform data set.

Instead of "hard selecting" about half of the variables as in the SNB(MAP) method, the SNB(CMA) method selects all the variable with weights around 0.5. Interestingly, the variable selection pattern is similar to that of the alternative variable selection methods, in a smoothed version. A central group of variables is emphasized around variable V11, between two less important groups of variables around variables V5 and V17.

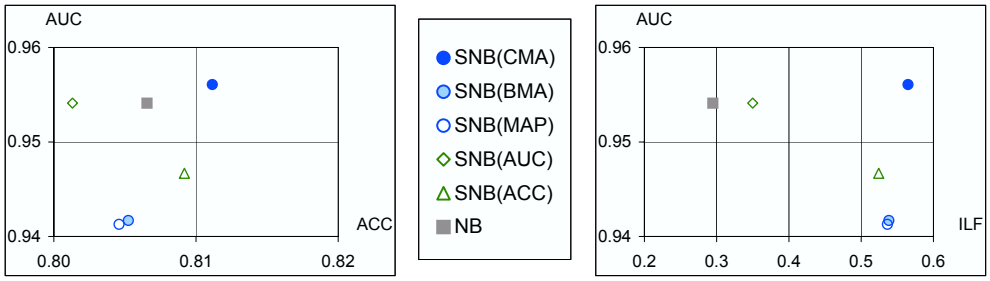


Figure 8: Evaluation of the predictive performance of selective naive Bayes classifiers on the waveform data set.

In the waveform data set, all the variables are informative, but the most probable subsets of variables compliant with the naive Bayes assumption select only half of the variables. In other words, whereas many good SNB classifiers are available, none of them is able to account for all the information contained in the variables. Since the BMA model is almost the same as the MAP model, it fails to perform better than one single classifier . Our CMA approach averages complementary subsets of variables and exploits more information than the BMA approach. This smoothed variable selection results in improved performance, as shown in Figure 8. The SNB(CMA) method is the best one: it dominates all the other methods on the three evaluated criteria.

7. Experiments

This section presents an experimental evaluation of the performance of the selective naive Bayes methods described in the previous sections.

7.1 Experimental Setup

The experiments aim at comparing the performance of model averaging methods versus the MAP method, the standard selective naive Bayes (SNB) and naive Bayes (NB) methods. All the classifiers except the last one exploit the same MODL preprocessing, allowing a fair comparison. The evaluated methods are:

- No variable selection
  - NB(EF): NB with 10 bins equal frequency discretization and no value grouping,
  - NB: NB with MODL preprocessing,
- Variable selection
  - SNB(ACC): optimization of the accuracy,
  - SNB(AUC): optimization of the area under the ROC curve,
  - SNB(MAP): MAP SNB model,
- Variable selection and model averaging
  - SNB(BMA): Bayesian model averaging,
  - SNB(CMA)[2] : compression-based model averaging.

The three last SNB classifiers (SNB(MAP), SNB(BMA) and SNB(CMA)) represent our contribution in this paper. All the SNB classifiers are optimized with the same MS(FFWBW) search heuristic, except the SNB(ACC), based on the forward selection greedy heuristic. The DC method (Dash and Cooper, 2002), similar to the SNB(BMA) approach, was not evaluated since it is restricted to categorical attributes.

The evaluated criteria are the same as for the waveform data set: accuracy (ACC), area under the ROC curve (AUC) and informational loss function (ILF) (with its compression rate (CR) normalization).

2. The method is implemented in a tool available as a shareware at http://www.francetelecom.com/en/group/rd/offer/software/technologies/middlewares/khiops.html.

Name	Instances	Numerical variables	Categorical variables	Classes	Majority accuracy
Abalone	4177	7	1	28	16.5
Adult	48842	7	8	2	76.1
Australian	690	6	8	2	55.5
Breast	699	10	0	2	65.5
Crx	690	6	9	2	55.5
German	1000	24	0	2	70.0
Glass	214	9	0	6	35.5
Heart	270	10	3	2	55.6
Hepatitis	155	6	13	2	79.4
HorseColic	368	7	20	2	63.0
Hypothyroid	3163	7	18	2	95.2
Ionosphere	351	34	0	2	64.1
Iris	150	4	0	3	33.3
LED	1000	7	0	10	11.4
LED17	10000	24	0	10	10.7
Letter	20000	16	0	26	04.1
Mushroom	8416	0	22	2	53.3
PenDigits	7494	16	0	10	10.4
Pima	768	8	0	2	65.1
Satimage	6435	36	0	6	23.8
Segmentation	2310	19	0	7	14.3
SickEuthyroid	3163	7	18	2	90.7
Sonar	208	60	0	2	53.4
Spam	4307	57	0	2	64.7
Thyroid	7200	21	0	3	92.6
TicTacToe	958	0	9	2	65.3
Vehicle	846	18	0	4	25.8
Waveform	5000	21	0	3	33.9
Wine	178	13	0	3	39.9
Yeast	1484	8	1	10	31.2

Table 1: UCI Data Sets

We conduct the experiments on two collections of data sets: 30 data sets from the repository at University of California at Irvine (Blake and Merz, 1996) and 10 data sets from the NIPS 2003 feature selection challenge (Guyon et al., 2006a) and the IJCNN 2006 performance prediction challenge (Guyon et al., 2006c). A summary of some properties of these data sets is given in Table 1 for the UCI data sets and in Table 2 for the challenge data sets. We use stratified 10-fold cross validation to evaluate the criteria. A two-tailed Student test at the 5% confidence level is performed in order to evaluate the significant wins or losses of the SNB(CMA) method versus each other method.

7.2 Results

We collect and average the three criteria owing to the stratified 10-fold cross validation, for the seven evaluated methods on the forty data sets. The results are presented in Table 3 for the UCI data

Name	Instances	Numerical variables	Categorical variables	Classes	Majority accuracy
Arcene	200	10000	0	2	56.0
Dexter	600	20000	0	2	50.0
Dorothea	1150	100000	0	2	90.3
Gisette	7000	5000	0	2	50.0
Madelon	2600	500	0	2	50.0
Ada	4147	48	0	2	75.2
Gina	3153	970	0	2	50.8
Hiva	3845	1617	0	2	96.5
Nova	1754	16969	0	2	71.6
Sylva	13086	216	0	2	93.8

Table 2: Challenge Data Sets

sets and in Table 4 for the challenge data sets. They are summarized across the data sets using the mean, the number of wins and losses (W/L) for the SNB(CMA) method and the average rank, for each of the three evaluation criteria.

Method	ACC			AUC			CR		
	Mean	W/L	Rank	Mean	W/L	Rank	Mean	W/L	Rank
SNB(CMA)	0.824		2.2	0.920		1.9	0.577		2.2
SNB(BMA)	0.817	9/2	3.7	0.916	11/0	3.3	0.559	12/6	2.6
SNB(MAP)	0.813	11/1	4.5	0.913	17/1	4.4	0.549	15/6	3.6
SNB(AUC)	0.820	8/0	3.3	0.918	10/2	3.1	0.532	17/4	4.4
SNB(ACC)	0.817	5/1	3.5	0.910	14/0	4.5	0.536	13/2	4.5
NB	0.814	11/0	4.0	0.913	16/1	4.6	0.476	19/2	5.3
NB(EF)	0.796	15/1	4.6	0.911	13/3	4.8	0.401	15/2	5.3

Table 3: Evaluation of the methods on the UCI data sets

The three ways of aggregating the results (mean, W/L and rank) are consistent, and we choose to display the mean of each criterion to ease the interpretation. Figure 9 summarizes the results for the UCI data sets and Figure 10 for the challenge data sets.

The results of the two NB methods are reported mainly as a sanity check. The MODL preprocessing in the NB classifier exhibits better performance than the equal frequency discretization method in the NB(EF) classifier.

The experiments confirm the benefit of selecting the variables, using the standard selection methods SNB(ACC) and SNB(AUC). These two methods achieve comparable results, with an emphasis on their respective optimized criterion. They significantly improve the results of the NB methods, especially for the estimation of class conditional probabilities measured by the CR criterion. It is noteworthy that the NB and NB(EF) classifiers obtain poor CR results. Their mean CR

Method	ACC			AUC			CR		
	Mean	W/L	Rank	Mean	W/L	Rank	Mean	W/L	Rank
SNB(CMA)	0.883		1.9	0.904		1.0	0.510		1.0
SNB(BMA)	0.872	3/0	3.5	0.882	6/0	2.9	0.446	9/0	2.3
SNB(MAP)	0.865	4/0	4.5	0.863	6/0	5.1	0.425	9/0	3.7
SNB(AUC)	0.872	6/0	3.6	0.888	7/0	2.7	0.331	10/0	4.1
SNB(ACC)	0.875	3/2	2.9	0.869	8/0	4.8	0.365	9/0	4.3
NB	0.841	7/0	4.9	0.846	9/0	5.3	-0.321	9/0	5.9
NB(EF)	0.823	9/0	6.6	0.833	9/0	6.2	-0.423	10/0	6.7

Table 4: Evaluation of the methods on the challenge data sets

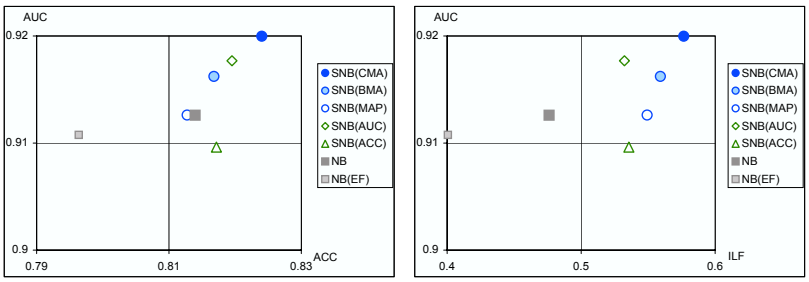


Figure 9: Mean of the ACC, AUC and CR evaluation criteria on the 30 UCI data sets.

result is less than 0 in the case of the challenge data sets, which means that their estimation of the class conditional probabilities is worse than that of the null model (which selects no variable).

The three regularized methods SNB(MAP), SNB(BMA) and SNB(CMA) focus on the estimation of the class conditional probabilities, which are evaluated using the compression rate criterion. They clearly outperform the other methods on this criterion, especially for the challenge data sets where the improvement amounts to about 50%. However, the SNB(MAP) method is not better than the two standard SNB methods for the accuracy and AUC criteria. The MAP method increases the bias of the models by penalizing the complex models, leading to a decayed fit of the data.

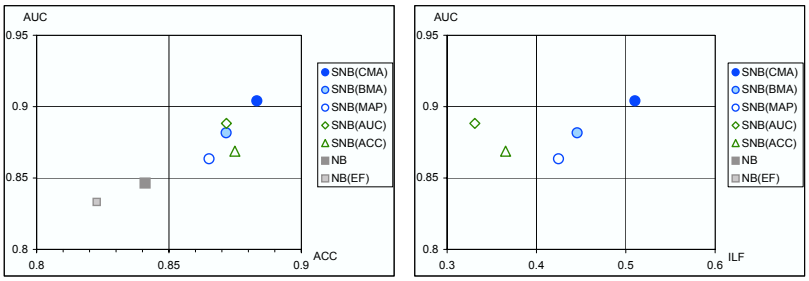


Figure 10: Mean of the ACC, AUC and CR evaluation criteria on the 10 challenge data sets.

The model averaging approach exploited in SNB(BMA) method offers only slight enhancements compared to the SNB(MAP) method. This confirms the analysis drawn from the waveform case study.

The compression-based averaging method SNB(CMA) clearly dominates all the other methods on all the criteria. On average, the number of significant wins is about 10 times the number of significant losses, and amounts to more than half of the 40 data sets. On the 10 challenge data sets, having very large numbers of variables, the SNB(CMA) method always gets the best results on the AUC and CR criteria, and almost always on the accuracy criterion. The domination of the SNB(CMA) method increases with the complexity of the criteria: it is noteworthy for accuracy (ACC), important for the ordering of the class conditional probabilities (AUC) and very large for the prediction of the class conditional probabilities (CR). This shows that the regularized and averaged naive Bayes classifier becomes effective for conditional probability estimation, whereas the standard naive Bayes classifier is usually considered to be poor at estimating these probabilities.

To summarize, this experiment demonstrates that variable selection is useful to improve the classification accuracy of the naive Bayes classifier. The MAP selection approach presented in Section 3 allows to find a selective naive Bayes classifier which is as compliant as possible with the naive Bayes assumption. Although this has little impact on the classification accuracy, this greatly improves the estimation of the class conditional probabilities.

Model averaging aims at improving the predictive performance at the expense of models which are more difficult to understand and to deploy. From this point of view, the experiment indicates that Bayesian model averaging is not much useful, since it does not significantly outperform the MAP model. On the opposite, our compression model averaging scheme introduced in Section 6 takes benefit from the full posterior distribution of the models and obtains superior results for all the evaluated criteria.

8. Evaluation on the Performance Prediction Challenge

This section reports the results obtained by our compression-based averaging method on the performance prediction challenge of Guyon et al. (2006c).

8.1 The Performance Prediction Challenge

The purpose of the performance prediction challenge is "to stimulate research and reveal the state-of-the art in model selection". Each method is evaluated according to its predictive performance and to its ability to guess its performance. The performance is assessed using the balanced error rate (BER) criterion to account for skewed distributions. The BER guess error is evaluated as the absolute value of the difference between the test BER and the predicted BER. A test score is computed as a combination of the test BER and the BER guess error to rank the participants.

Five data sets are used in the challenge (the five last data sets in Table 2). The ada data set comes from the marketing domain, the gina data set from handwriting recognition, the hiva data set from drug discovery, the nova data set from text classification and the sylva data set from ecology. The test sets used to assess the performance are 10 times larger than the train data sets.

8.2 Details of the Submissions

All our entries are based on the compression-based averaging of the selective naive Bayes classifier SNB(CMA).

The method computes the posterior probabilities of the classes, which is convenient when the accuracy criterion or the area under the ROC curve is evaluated. In a two-classes problem, any instance whose class posterior probability is beyond a threshold $\tau = 0.5$ is classified as positive, and otherwise as negative. For the challenge, the BER criterion is the main criterion, and it is no longer optimal to predict the most probable class. In order to improve the BER criterion, we adjust the decision threshold τ in a post-optimization step. We sort the train instances by decreasing class posterior probabilities, which determines N possible values of the threshold τ. We then loop on the instances, and for each τ, we compute the confusion matrix between the prediction outcome and the actual class value. We keep the threshold that maximizes the BER of the related confusion matrix. Post-optimizing the BER criterion requires $0(N \log N)$ computation time to sort the instances and $O(N)$ to compute the N possible confusion matrices, since evaluating each successive τ involves the move of only one instance in the confusion matrix.

For the challenge, we perform several trials of feature construction in order to evaluate the computational and statistical scalability of the method, and to leverage the naive Bayes assumption:

- 10k F(2D): 10 000 features constructed for each data set, each one is the sum of two randomly selected initial features,
- 100k F(2D): 100 000 features constructed (sums of two features),
- 10k F(3D): 10 000 features constructed (sums of three features).

The performance prediction guess is computed using a stratified tenfold cross-validation on the train data set.

8.3 Results

The challenge started Friday September 30, 2005, and ended Monday March 1, 2006. About 145 entrants participated to the challenge and submitted more than 4000 "development entries". A total of 28 participants competed for the final ranking by providing valid challenge entries (results on train, validation, and test sets for all five tasks of the challenge). Each participant was allowed to submit at most 5 entries.

In the challenge, we rank 7^{th} out of 28, according to the average rank computed by the organizers. On 2 of the 5 five data sets (ada and sylva), our best entry ranks 1^{st}, as shown in Table 5. The AUC criterion, which evaluates the ranking of the class posterior probabilities, indicates high performance for our method, which ranks 3^{rd} on this criterion.

The detailed results of our entries are presented in Figure 11, together with all the final entries of the 28 finalists. The analysis of Guyon et al. (2006c) reveals that the top ranking entries exploit a variety of methods: ensembles of decision trees, support vector machines (SVM) kernel methods, Bayesian neural networks, ensembles of linear methods. On three out of the five data sets (gina, hiva and nova), the data set winner exploits a SVM kernel method. This type of the method is the most frequently used by the challenge participants, but their performance shows a lot of variance, so they need human expertise to adjust their parameters. On the contrary, ensembles of decision trees, like the method of the challenge winner, perform consistently well on all the data sets. Overall, the

Data Set	Our best entry				The challenge best entry			
	Test BER	BER Guess	Guess Error	Test Score	Test BER	BER Guess	Guess Error	Test Score
Ada	0.1723	0.1650	0.0073	0.1793	0.1723	0.1650	0.0073	0.1793
Gina	0.0733	0.0770	0.0037	0.0767	0.0288	0.0305	0.0017	0.0302
Hiva	0.3080	0.3170	0.0090	0.3146	0.2757	0.2692	0.0065	0.2797
Nova	0.0776	0.0860	0.0084	0.0858	0.0445	0.0436	0.0009	0.0448
Sylva	0.0061	0.0060	0.0001	0.0062	0.0061	0.0060	0.0001	0.0062
Overall	0.1307	0.1306	0.0096	0.1399	0.1090	0.1040	0.0079	0.1165

Table 5: Results of our best entry on the performance prediction challenge data sets.

top five ranked methods get an average test BER of 11%. Our method gets an average test BER of 13% and is ranked only 11^{th} on the BER criterion, even though it obtains very good results on the ada and sylva data sets.

The main limitation of our SNB(CMA) method comes from the naive Bayes assumption. Our method fails to correctly approximate the true class conditional distribution when the representation space of the data set does not contain any competitive subset of variables compliant with the naive Bayes assumption. For example, the gina data set consists of a set of image pixels, where the classification problem is to predict the parity of a number. On the initial representation of the gina data set, our test BER is only 13%, far from the best result which is about 3%. However, when the constructed features allow to "partially" circumvent the naive Bayes assumption, the method succeeds in significantly improving its performance, from 13% down to 7%. According to the challenge organizers, the hiva and nova data sets are also highly non-linear, which explains our poor BER results. For example, the nova data set is a text classification problem with approximately 17000 variables in a bag-of-words representation. In the case of this sparse data set, adding randomly constructed features is useless and results mainly in duplicating the variables. This explains why all our nova entries obtained the same BER results of 8%, far from the best result which is about 4%.

It is noteworthy that our method is very robust and ranks 4^{th} on average on the guess error criterion. Although we use all the train data to select our model without reserving validation data, our method is not prone to overfitting. In our feature construction schemes, we expand the size of the initial representation space of the data sets by a one hundred factor, which turns variable selection into a challenging problem. However, adding many variables never decreases the performance of our method. This means that our method correctly account for many useless and redundant variables, and is able to benefit from the potentially informative constructed variables, like in the gina data set for example.

Our method is evaluated with data sets having almost one billion values (up to 100 000 constructed features). Figure 12 reports the training time for all our submissions in the challenge. Our method is highly scalable and resistant to noisy or redundant features: it is able to quickly process about 100 000 constructed features without decreasing the predictive performance.

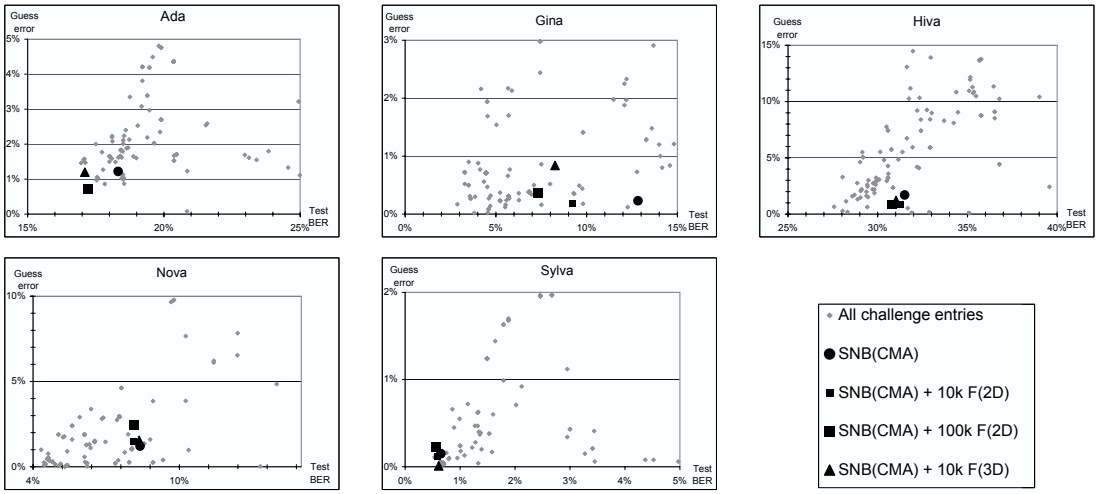


Figure 11: Detailed results of all of our entries on the performance prediction challenge data sets.

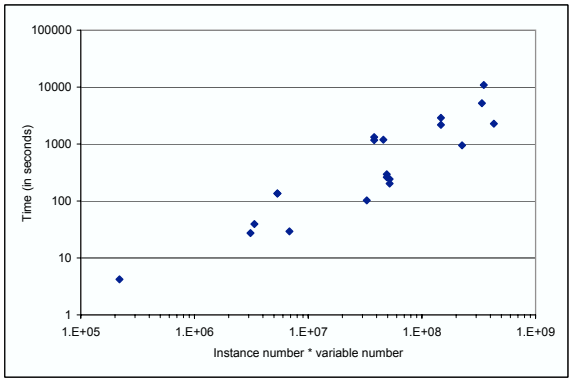


Figure 12: Training times for the SNB(CMA) classifier for all our entries in the performance prediction challenge.

9. Conclusion

The naive Bayes classifier is a popular method that is often highly effective on real data sets and is competitive with or even sometimes outperforms much more sophisticated classifiers. This paper confirms the potential benefit of variable selection to obtain still better performance.

We have proposed a MAP approach to select the best subset of variables compliant with the naive Bayes assumption and introduced an efficient search algorithm which time complexity is $O(KN\log(KN))$, where K is the number of variables and N the number of instances. We have also showed empirically that Bayesian model averaging is not much useful, since it does not perform significantly better that the MAP model.

On the basis of experimental and theoretical evidence that indicates that the posterior distribution of the models is exponentially peaked, we have shown that choosing a logarithmic smoothing of the posterior distribution makes sense. We have empirically demonstrated that the resulting compression-based model averaging scheme clearly outperforms the Bayesian model averaging scheme. This is encouraging and suggests that further research could be done to design a still more effective averaging scheme with more grounded foundations.

Our method consistently improves the performance of the naive Bayes classifier, but is outperformed by more sophisticated methods when the naive Bayes assumption is too harmful. In future work, we plan to exploit multivariate preprocessing methods in order to circumvent the naive Bayes assumption. On the basis of a set of univariate and multivariate conditional density estimators, our goal is to build a classifier that better approximates the true conditional density. In this setting, we think that compression-based model averaging might still be superior to Bayesian model averaging to account for the whole posterior distribution of the models.

Acknowledgments

I am grateful to the editor and the anonymous reviewers for their useful comments.

References

C.L. Blake and C.J. Merz. UCI repository of machine learning databases, 1996. http://www.ics.uci.edu/mlearn/MLRepository.html.

M. Boullé. *Feature Extraction: Foundations And Applications*, chapter 25, pages 499–507. Springer, 2006a.

M. Boullé. Regularization and averaging of the selective naive Bayes classifier. In *International Joint Conference on Neural Networks*, pages 2989–2997, 2006b.

M. Boullé. A Bayes optimal approach for partitioning the values of categorical attributes. *Journal of Machine Learning Research*, 6:1431–1452, 2005a.

M. Boullé. MODL: a Bayes optimal discretization method for continuous attributes. *Machine Learning*, 65(1):131–165, 2006c.

M. Boullé. A grouping method for categorical attributes having very large number of values. In P. Perner and A. Imiya, editors, *Proceedings of the Fourth International Conference on Machine Learning and Data Mining in Pattern Recognition*, volume 3587 of *LNAI*, pages 228–242. Springer verlag, 2005b.

L. Breiman, J.H. Friedman, R.A. Olshen, and C.J. Stone. *Classification and Regression Trees*. California: Wadsworth International, 1984.

D. Dash and G.F. Cooper. Exact model averaging with naive Bayesian classifiers. In *Proceedings of the Nineteenth International Conference on Machine Learning*, pages 91–98, 2002.

K. Deng, C. Bourke, S. Scott, and N.V. Vinodchandran. New algorithms for optimizing multi-class classifiers via ROC surfaces. In *Proceedings of the ICML 2006 Workshop on ROC Analysis in Machine Learning*, pages 17–24, 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 *Proceedings of the 12th International Conference on Machine Learning*, pages 194–202. Morgan Kaufmann, San Francisco, CA, 1995.

T. Fawcett. ROC graphs: Notes and practical considerations for researchers. Technical Report HPL-2003-4, HP Laboratories, 2003.

I. Guyon, S. Gunn, A. Ben Hur, and G. Dror. *Feature Extraction: Foundations And Applications*, chapter 9, pages 237–263. Springer, 2006a. Design and Analysis of the NIPS2003 Challenge.

I. Guyon, S. Gunn, M. Nikravesh, and L. Zadeh, editors. *Feature Extraction: Foundations And Applications*. Springer, 2006b.

I. Guyon, A.R. Saffari, G. Dror, and J.M. Bumann. Performance prediction challenge. In *International Joint Conference on Neural Networks*, pages 2958–2965, 2006c.

D.J. Hand and K. Yu. Idiot bayes ? not so stupid after all? *International Statistical Review*, 69(3): 385–399, 2001.

J.A. Hoeting, D. Madigan, A.E. Raftery, and C.T. Volinsky. Bayesian model averaging: A tutorial. *Statistical Science*, 14(4):382–417, 1999.

R. Kohavi and G. John. Wrappers for feature selection. *Artificial Intelligence*, 97(1-2):273–324, 1997.

P. Langley and S. Sage. Induction of selective Bayesian classifiers. In *Proceedings of the 10th Conference on Uncertainty in Artificial Intelligence*, pages 399–406. Morgan Kaufmann, 1994.

P. Langley, W. Iba, and K. Thompson. An analysis of Bayesian classifiers. In *10th national conference on Artificial Intelligence*, pages 223–228. AAAI Press, 1992.

H. Liu, F. Hussain, C.L. Tan, and M. Dash. Discretization: An enabling technique. *Data Mining and Knowledge Discovery*, 4(6):393–423, 2002.

T.M. Mitchell. *Machine Learning*. McGraw-Hill, New York, 1997.

V. Pareto. *Manuale di Economia Politica*. Piccola Biblioteca Scientifica, Milan, 1906. Translated into English by Ann S. Schwier (1971), Manual of Political Economy, MacMillan, London.

F. Provost and P. Domingos. Well-trained pets: Improving probability estimation trees. Technical Report CeDER #IS-00-04, New York University, 2001.

F. Provost, T. Fawcett, and R. Kohavi. The case against accuracy estimation for comparing induction algorithms. In *Proceedings of the Fifteenth International Conference on Machine Learning*, pages 445–553, 1998.

A.E. Raftery and Y. Zheng. Long-run performance of Bayesian model averaging. Technical Report 433, Department of Statistics, University of Washington, 2003.

J. Rissanen. Modeling by shortest data description. *Automatica*, 14:465–471, 1978.

C.P. Robert. *The Bayesian Choice: A Decision-Theoretic Motivation*. Springer-Verlag, New York, 1997.

C.E. Shannon. A mathematical theory of communication. Technical report, Bell systems technical journal, 1948.

I.H. Witten and E. Frank. *Data Mining*. Morgan Kaufmann, 2000.

Y. Yang and G. Webb. A comparative study of discretization methods for naive-Bayes classifiers. In *Proceedings of the Pacific Rim Knowledge Acquisition Workshop*, pages 159–173, 2002.

Journal of Machine Learning Research 8 (2007) 1687-1723 Submitted 1/07; Revised 4/07; Published 8/07

A Nonparametric Statistical Approach to Clustering via Mode Identification

Jia Li JIALI@STAT.PSU.EDU
Department of Statistics
The Pennsylvania State University
University Park, PA 16802, USA

Surajit Ray SRAY@MATH.BU.EDU
Department of Mathematics and Statistics
Boston University
Boston, MA 02215, USA

Bruce G. Lindsay BGL@STAT.PSU.EDU
Department of Statistics
The Pennsylvania State University
University Park, PA 16802, USA

Editor: Charles Elkan

Abstract

A new clustering approach based on mode identification is developed by applying new optimization techniques to a nonparametric density estimator. A cluster is formed by those sample points that ascend to the same local maximum (mode) of the density function. The path from a point to its associated mode is efficiently solved by an EM-style algorithm, namely, the Modal EM (MEM). This method is then extended for hierarchical clustering by recursively locating modes of kernel density estimators with increasing bandwidths. Without model fitting, the mode-based clustering yields a density description for every cluster, a major advantage of mixture-model-based clustering. Moreover, it ensures that every cluster corresponds to a bump of the density. The issue of diagnosing clustering results is also investigated. Specifically, a pairwise separability measure for clusters is defined using the ridgeline between the density bumps of two clusters. The ridgeline is solved for by the Ridgeline EM (REM) algorithm, an extension of MEM. Based upon this new measure, a cluster merging procedure is created to enforce strong separation. Experiments on simulated and real data demonstrate that the mode-based clustering approach tends to combine the strengths of linkage and mixture-model-based clustering. In addition, the approach is robust in high dimensions and when clusters deviate substantially from Gaussian distributions. Both of these cases pose difficulty for parametric mixture modeling. A C package on the new algorithms is developed for public access at http://www.stat.psu.edu/∼jiali/hmac.

Keywords: modal clustering, mode-based clustering, mixture modeling, modal EM, ridgeline EM, nonparametric density

1. Introduction

Clustering is a technology employed in tremendously diverse areas for a multitude of purposes. It simplifies massive data by extracting essential information, based on which many subsequent analysis or processes become feasible or more efficient. For instance, in information systems, clustering

is applied to text documents or images to speed up indexing and retrieval (Li, 2005a). Clustering can be a stand-alone process. For example, microarray gene expression data are often clustered in order to find genes with similar functions. Clustering is also the technical core of several prototype-based supervised learning algorithms (Hastie et al., 2001) and has been extended to non-vector data in this regard (Li and Wang, 2006). Recent surveys (Kettenring, 2006; Jain et al., 1999) discuss the methodologies, practices, and applications of clustering.

1.1 Background

Clustering methods fall roughly into three types. The first type uses only pairwise distances between objects to be clustered. These methods enjoy wide applicability since a tractable mathematical representation for objects is not required. However, they do not scale well with large data sets due to the quadratic computational complexity of calculating all the pairwise distances. Examples include linkage clustering (Gower and Ross, 1969) and spectral graph partitioning (Pothen et al., 1990). The second type targets on optimizing a given merit function. The merit function reflects the general belief about good clustering, that is, objects in the same cluster should be similar to each other while those in different clusters be as distinct as possible. Different algorithms vary in the similarity measure and the criterion for assessing the global quality of clustering. K-means and k-center clustering (Gonzalez, 1985) belong to this type.

The third type relies on statistical modeling (Fraley and Raftery, 2002). In particular, each cluster is characterized by a basic parametric distribution (referred to as a component), for instance, the multivariate Gaussian for continuous data and the Poisson distribution for discrete data. The overall probability density function (pdf) is a mixture of the parametric distributions (McLachlan and Peel, 2000). The clustering procedure involves first fitting a mixture model, usually by the EM algorithm, and then computing the posterior probability of each mixture component given a data point. The component possessing the largest posterior probability is chosen for that point. Points associated with the same component form one cluster. Moreover, the component posterior probabilities evaluated in mixture modeling can be readily used as a soft clustering scheme. In addition to partitioning data, a probability distribution is obtained for each cluster, which can be helpful for gaining insight into the data. Another advantage of mixture modeling is its flexibility in treating data of different characteristics. For particular applications, mixtures of distributions other than Gaussian have been explored for clustering (Banfield and Raftery, 1993; Li and Zha, 2006). Banerjee et al. (2005) have also used the mixture of Mises-Fisher distributions to cluster data on a unit sphere.

The advantages of mixture modeling naturally result from its statistical basis. However, the parametric assumptions about cluster distributions are found restrictive in some applications. Li (2005b) addresses the problem by assuming each cluster itself is a mixture of Gaussians, providing greater flexibility for modeling a single cluster. This method involves selecting the number of components for each cluster and is sensitive to initialization. Although some success has been shown using the Bayesian Information Criterion (BIC), choosing the right number of components for a mixture model is known to be difficult, especially for high dimensional data.

Another limitation for mixture modeling comes from the sometimes serious disparity between a component and a cluster complying to geometric heuristics. If a probability density is estimated from the data, preferably, every cluster corresponds to a unique "bump" in the density resulting from a tight mass of data. We refer to a bump as a "hill" and the local maximum associated with it the

"hilltop", that is, the mode. The rational for clustering by a mixture model is that if the component distributions each possess a single hilltop, by fitting a mixture of them, every component captures one separate hilltop in the data. However, this is often not true. Without careful placement and control of their shapes, the mixture components may not align with the hills of the density, especially when clusters are poorly separated or the assumed parametric component distribution is violated. It is known that two Gaussians located sufficiently close result in a single mode. (On the other hand, a two component multivariate Gaussian mixture can have more than two modes, as shown by Ray and Lindsay, 2005). In this case, equating a component with a cluster is questionable. This profound limitation of mixture modeling has not been adequately investigated. In fact, even to quantify the separation between components is not easy.

Here, we develop a new nonparametric clustering approach, still under a statistical framework. This approach inherits the aforementioned advantages of mixture modeling. Furthermore, data are allowed to reveal a nonparametric distribution for each cluster as part of the clustering procedure. It is also guaranteed that every cluster accounts for a distinct hill of the probability density.

1.2 Clustering via Mode Identification

To avoid restrictions imposed by parametric assumptions, we model data using kernel density functions. By their nature, such densities have a mixture structure. Given a density estimate in the form of a mixture, a new algorithm, aptly called the Modal EM (MEM), enables us to find an increasing path from any point to a local maximum of the density, that is, a hilltop. Our new clustering algorithm groups data points into one cluster if they are associated with the same hilltop. We call this approach modal clustering. A new algorithm, namely the Ridgeline EM (REM), is also developed to find the ridgeline linking two hilltops, which is proven to pass through all the critical points of the mixture density of the two hills.

The MEM and REM algorithms allow us to exploit the geometry of a probability density function in a nontrivial manner. As a result, clustering can be conducted in accurate accordance with our geometric heuristics. Specifically, every cluster is ensured to be associated with a hill, and every sample point in the cluster can be moved to the corresponding hilltop along an ascending path without crossing any "valley" that separates two hills. Moreover, by finding the ridgeline between two hilltops, the way two hills separate from each other can be adequately measured and exhibited, enabling diagnosis of clustering results and application-dependent adjustment of clusters. Modal clustering using MEM also has practical advantages such as the irrelevance of initialization and the ease of implementing required optimization techniques.

Our modal clustering algorithm is not restricted to kernel density estimators. In fact, it can be used to find the modes of any density in the form of a mixture distribution. It is known that when the number of components in a mixture increases, as long as there are sufficiently many components, the overall fitted density of the mixture is not sensitive to that number. On the other hand, the resulting partition of data can change dramatically if we identify each mixture component with a cluster, as normally practiced in mixture-model-based clustering. In modal clustering, there is no such identification, and mixture components only play the role of approximating a density. We thus have much more flexibility at choosing mixture distributions. Specifically, we adopt the fully nonparametric kernel density estimation, using Gaussian kernels for continuous data.

We summarize the main contributions of this paper as follows:

- A new nonparametric statistical clustering algorithm and its hierarchical extension are developed by associating data points to their modes identified by MEM. Approaches to improve computational efficiency and to visualize cluster structures for high dimensional data are presented.

- The REM algorithm is developed to find the ridgeline between the modes of any two clusters. Measures for the pairwise separability between clusters are proposed using ridgelines. A cluster merging algorithm to enhance modal clustering is developed.

- Experiments are conducted using both simulated and real data sets. Comparisons are made with several other clustering algorithms such as linkage clustering, k-means, and Gaussian mixture modeling.

1.3 Related Work

Clustering is an extensively studied research topic with vast existing literature (see Jain et al., 1999; Everitt et al., 2001, for general coverage). Works most related to ours are mode-based clustering methods independently developed in the communities of pattern recognition (Leung et al., 2000) and statistics (Cheng et al., 2004). The MEM and REM algorithms, the cluster diagnosis tool, and cluster merging procedure built upon REM are unique to our work.

In pattern recognition, mode-based clustering is studied under the name of scale-space method, inspired by the smoothing effect of the human visual system. The scale-space clustering method is pioneered by Wilson and Spann (1990), and furthered studied by Roberts (1997) using density estimation and by Chakravarthy and Ghosh (1996) using the radial basis function neural network. Leung et al. (2000) forms a function called "space scale image" for a data set. This function is essentially a Gaussian kernel density estimate (differing from a true density by an ignorable constant). The modes of the density function are solved by numerical methods. To associate a data point with a mode, a gradient dynamic system starting from the point is defined and solved by the Euler difference method. A hierarchical clustering algorithm is proposed by gradually increasing the Gaussian kernel bandwidth. The authors also note the non-nested nature of clustering results obtained from increasing bandwidths.

It can be shown that the iteration equation derived from the Euler difference method is identical to that from MEM. However, MEM applies generally to any mixture of Gaussians as well as mixtures of other parametric distributions. Its ascending property is proved rather than based on approximation. Under the framework of scale-space clustering, general mixtures do not arise as a concern, and naturally, only the case of Gaussian kernel density is discussed (Leung et al., 2000). It is not clear whether the gradient dynamic system can be efficiently solved for general mixture models. Our hierarchical clustering algorithm differs slightly from that of Leung et al. (2000) by enforcing nested clustering. This difference only reflects an algorithmic preference and is not intrinsic to the key ideas of modal clustering.

Cheng et al. (2004) defined a gradient tree to be the set of steepest ascent curves of a kernel density estimate, treating each sample point as the starting position of a curve. The gradient curves are similar to the paths solved by the gradient dynamic system of Leung et al. (2000), but are computed by discrete approximation. Minnotte and Scott (1993) developed the mode tree as a visualization tool for nonparametric density estimate. The emphasis is on graphically illustrating the

relationship between kernel bandwidths and the modes of uni- or bivariate kernel density functions. Minnotte et al. (1998) also extended the mode tree to the mode forest.

Because clustering of independent vectors has been a widely used method for image segmentation, especially in the early days (Jain and Dubes, 1988), we test the efficiency of our algorithm on segmentation, a good example of computationally intensive applications. We have no intention to present our clustering algorithm as a state-of-the-art segmentation method although it may well be applied as a fast method. We note that much advance has been achieved in image segmentation using approaches beyond the framework of clustering independent vectors (Pal and Pal, 1993; Shi and Malik, 2000; Joshi et al., 2006).

The rest of the paper is organized as follows. Notations and the MEM algorithm are introduced in Section 2. In Section 3, a new clustering algorithm and its hierarchical extension are developed by associating data points with the modes of a kernel density estimate. In Section 4, the REM algorithm for finding ridgelines between modes is presented. In addition, several measures of pairwise separability between clusters are defined, which lead to the derivation of a new cluster merging algorithm. This merging method strengthens the framework of modal clustering. In Section 5, we present a method to visualize high dimensional data so that the discrimination between clusters is well preserved. Experimental results on both simulated and real data sets and comparisons with other clustering approaches are provided in Section 6. Finally, we conclude and discuss future work in Section 7.

2. Preliminaries

We introduce in this section the *Modal EM (MEM)* algorithm that solves a local maximum of a mixture density by ascending iterations starting from any initial point. The algorithm is named Modal EM because it comprises two iterative steps similar to the expectation and the maximization steps in EM (Dempster et al., 1977). The objective of the MEM algorithm is different from the EM algorithm. The EM algorithm aims at maximizing the likelihood of data over the parameters of an assumed distribution. The goal of MEM is to find the local maxima, that is, modes, of a given distribution.

Let a mixture density be $f(x) = \sum_{k=1}^{K} \pi_k f_k(x)$, where $x \in \mathcal{R}^d$, π_k is the prior probability of mixture component k, and $f_k(x)$ is the density of component k. Given any initial value $x^{(0)}$, MEM solves a local maximum of the mixture by alternating the following two steps until a stopping criterion is met. Start with $r = 0$.

1. Let
$$p_k = \frac{\pi_k f_k(x^{(r)})}{f(x^{(r)})}, \; k = 1, ..., K.$$
2. Update
$$x^{(r+1)} = \operatorname*{argmax}_x \sum_{k=1}^{K} p_k \log f_k(x).$$

The first step is the "Expectation" step where the posterior probability of each mixture component k, $1 \le k \le K$, at the current point $x^{(r)}$ is computed. The second step is the "Maximization" step. We assume that $\sum_{k=1}^{K} p_k \log f_k(x)$ has a unique maximum, which is true when the $f_k(x)$ are normal densities. In the special case of a mixture of Gaussians with common covariance matrix, that is, $f_k(x) = \phi(x \mid \mu_k, \Sigma)$, where $\phi(\cdot)$ is the pdf of a Gaussian distribution, we simply have

$x^{(r+1)} = \sum_{k=1}^{K} p_k \mu_k$. For other parametric densities $f_k(x)$, the solution to the maximization in the second step can be more complicated and sometimes requires numerical procedures. On the other hand, similarly as in the EM algorithm, it is usually much easier to maximize $\sum_{k=1}^{K} p_k \log f_k(x)$ than the original objective function $\log \sum_{k=1}^{K} \pi_k f_k(x)$.

The proof of the ascending property of the MEM algorithm is provided in Appendix A. We omit a rigorous discussion regarding the convergence of $x^{(r)}$ here. By Theorem 1 of Wu (1983), if $f(x)$ is a mixture of normal densities, all the limit points of $\{x^{(r)}\}$ are stationary points of $f(x)$. It is possible that $\{x^{(r)}\}$ converges to a stationary, but not locally maximal, point, although we have not observed this in our experiments. We refer to (Wu, 1983) for a detailed treatment of the convergence properties of EM style algorithms.

3. Clustering by Mode Identification

We focus on clustering continuous vector data although the framework extends readily to discrete data. Given a data set $\{x_1, x_2, ..., x_n\}, x_i \in \mathcal{R}^d$, a probability density function for the data is estimated nonparametrically using Gaussian kernels. As the kernel density estimate is in the form of a mixture distribution, MEM is applied to find a mode using every sample point x_i, $i = 1, ..., n$, as the initial value for the iteration. Two points x_i and x_j are grouped into one cluster if the same mode is obtained from both. When the variances of Gaussian kernels increase, the density estimate becomes smoother and tends to group more points into one cluster. A hierarchy of clusters can thus be constructed by gradually increasing the variances of Gaussian kernels. Next, we elaborate upon the clustering algorithm, illustrate it with an example, and discuss approaches to speed up computation.

3.1 The Algorithm

Let the set of data to be clustered be $S = \{x_1, x_2, ..., x_n\}$, $x_i \in \mathcal{R}^d$. The Gaussian kernel density estimate is formed:

$$f(x) = \sum_{i=1}^{n} \frac{1}{n} \phi(x \mid x_i, \Sigma),$$

where the Gaussian density function

$$\phi(x \mid x_i, \Sigma) = \frac{1}{(2\pi)^{d/2} |\Sigma|^{1/2}} \exp(-\frac{1}{2}(x - x_i)^t \Sigma^{-1} (x - x_i)) .$$

We use a spherical covariance matrix $\Sigma = diag(\sigma^2, \sigma^2, ..., \sigma^2)$. The standard deviation σ is also referred to as the *bandwidth* of the Gaussian kernel. We use notation $D(\sigma^2) = diag(\sigma^2, \sigma^2, ..., \sigma^2)$ for brevity.

With a given Gaussian kernel covariance matrix $D(\sigma^2)$, data are clustered as follows:

1. Form kernel density

$$f(x \mid S, \sigma^2) = \sum_{i=1}^{n} \frac{1}{n} \phi(x \mid x_i, D(\sigma^2)), \tag{1}$$

2. Use $f(x|S, \sigma^2)$ as the density function. Use each x_i, $i = 1, 2, ..., n$, as the initial value in the MEM algorithm to find a mode of $f(x|S, \sigma^2)$. Let the mode identified by starting from x_i be $\mathcal{M}_\sigma(x_i)$.

3. Extract distinctive values from the set $\{\mathcal{M}_\sigma(x_i), i = 1,2,...,n\}$ to form a set G. Label the elements in G from 1 to $|G|$. In practice, due to finite precision, two modes are regarded equal if their distance is below a threshold.

4. If $\mathcal{M}_\sigma(x_i)$ equals the kth element in G, x_i is put in the kth cluster.

In the basic version of the algorithm, the density $f(x|S,\sigma^2)$ is a sum of Gaussian kernels centered at every data point. However, the algorithm can be carried out with any density estimate in the form of a mixture. The key step in the clustering algorithm is the identification of a mode starting from any x_i. MEM moves from x_i via an ascending path, or, figuratively, via hill climbing, to a mode. Points that climb to the same mode are located on the same hill and hence grouped into one cluster. We call this the *Mode Association Clustering (MAC)* algorithm.

Although the density of each cluster is not explicitly modeled by MAC, this nonparametric method retains a major advantage of mixture-model-based clustering, that is, a pdf is obtained for each cluster. These density functions facilitate soft clustering as well as cluster assignment of samples outside the data set. Denote the set of points in cluster k, $1 \le k \le |G|$, by C_k. The density estimate for cluster k is

$$g_k(x) = \sum_{x_i : x_i \in C_k} \frac{1}{|C_k|} \phi(x \mid x_i, D(\sigma^2)) \,. \tag{2}$$

Because we do not assume a parametric form for the densities of individual clusters, our method tends to be more robust and characterizes clusters more accurately when the attempted parametric assumptions are violated.

It is known in the literature of mixture modeling that if the density of a cluster is estimated using only points assigned to this cluster, the variance tends to be under estimated, although the effect on clustering may be small (Celeux and Govaert, 1993). The under estimation of variance becomes more severe for poorly separated clusters, which often decay towards zero too quickly on leaving the cluster. We will see a similar phenomenon here with $g_k(x)$ having over fast decaying tails. A correction to this problem in mixture modeling is to use soft instead of hard clustering. Every point is allowed to contribute to every cluster by a weight computed from the posterior probability of the cluster.

Under this spirit, we can make an ad-hoc modification on the density estimation. With $g_k(x)$ in (2) as the initial cluster density, compute the posterior of cluster k given each x_i by $p_{i,k} \propto \frac{|C_k|}{n} g_k(x)$, $k = 1, ..., |G|$, subject to $\sum_{k'=1}^{|G|} p_{i,k'} = 1$. Form the updated density of cluster k by

$$\tilde{g}_k(x) = \frac{\sum_{i=1}^n p_{i,k} \phi(x \mid x_i, D(\sigma^2))}{\sum_{i=1}^n p_{i,k}} \,.$$

With the cluster density modified, it is natural to question if $p_{i,k}$ should be updated again, which in turn will lead to another update of $\tilde{g}_k(x)$. This iterative procedure can be carried out infinitely. Whether it converges is not clear and may be worthy of investigation. On the other hand, if the maximum a posteriori clustering based on the final $\tilde{g}_k(x)$ differs significantly from the result of modal clustering, this procedure may have defeated the very purpose of modal clustering and turned it into merely an initialization scheme. We thus do not recommend many iterative updates on $\tilde{g}_k(x)$. One round of adjustment from $g_k(x)$ may be sufficient. Or one can take $g_k(x)$ cautiously as a smooth signature of a cluster, likely tighter than an accurate density estimate.

When the bandwidth σ increases, the kernel density estimate $f(x|S,\sigma^2)$ in (1) becomes smoother and more points tend to climb to the same mode. This suggests a natural approach for hierarchical clustering. Given a sequence of bandwidths $\sigma_1 < \sigma_2 < \cdots < \sigma_\eta$, hierarchical clustering is performed in a bottom-up manner. We start with every point x_i being a cluster by itself. The set of cluster representatives is thus $G_0 = S = \{x_1, ..., x_n\}$. This extreme case corresponds to the limit when σ approaches 0. At any bandwidth σ_l, the cluster representatives in G_{l-1} obtained from the preceding bandwidth are input into MAC using the density $f(x|S,\sigma_l^2)$. Note that the kernel centers remain at all the original data points although modes are identified only for cluster representatives when $l > 1$. The modes identified at this level form a new set of cluster representatives G_l. This procedure is repeated across all σ_l's. We refer to this hierarchical clustering algorithm as *Hierarchical MAC (HMAC)*. It corresponds to the mappings $x_i \to \mathcal{M}_{\sigma_1}(x_i) \to \mathcal{M}_{\sigma_2}(\mathcal{M}_{\sigma_1}(x_i)) \to \cdots$.

Denote the partition of points obtained at bandwidth σ_l by $\mathcal{P}_l$, a function mapping x_i's to cluster labels. If K clusters labeled 1, 2, ..., K, are formed at bandwidth σ_l, $\mathcal{P}_l(x_i) \in \{1, 2, ..., K\}$. HMAC ensures that $\mathcal{P}_l$'s are nested, that is, if $\mathcal{P}_l(x_i) = \mathcal{P}_l(x_j)$, then $\mathcal{P}_{l+1}(x_i) = \mathcal{P}_{l+1}(x_j)$. Recall that the set of cluster representatives at level l is G_l. HMAC starts with $G_0 = \{x_1, ..., x_n\}$ and solves G_l, $l = 1, 2, ..., \eta$, sequentially by the following procedure:

1. Form kernel density
$$f(x|S,\sigma_l^2) = \sum_{i=1}^{n} \frac{1}{n} \phi(x \mid x_i, D(\sigma_l^2)).$$

2. Cluster G_{l-1} by MAC using density $f(x|S,\sigma_l^2)$. Let the set of distinct modes obtained be G_l.

3. If $\mathcal{P}_{l-1}(x_i) = k$ and the kth element in G_{l-1} is clustered to the k'th mode in G_l, then $\mathcal{P}_l(x_i) = k'$. That is, the cluster of x_i at level l is determined by its cluster representative in G_{l-1}.

It is worthy to note that HMAC differs profoundly from linkage clustering, which also builds a hierarchy of clusters. In linkage clustering, at every level, only the two clusters with the minimum pairwise distance are merged. The hierarchy is constructed by a sequence of such small greedy merges. The lack of overall consideration tends to result in skewed clusters. In HMAC, however, at any level, the merging of clusters is conducted globally and the effect of every original data point on clustering is retained through the density $f(x|S,\sigma_l^2)$.

3.2 An Example

We now illustrate the HMAC algorithm using a real data set. This is the glass identification data provided by the UCI machine learning database repository (Blake et al., 1998). The original data were contributed by Spiehler and German at the Diagnostic Products Corporation. For clarity of demonstration, we take 105 sample points from two types of glass in this data set. Moreover, we only use the first two principal components of the original 9 dimensions.

HMAC is applied to the data using a sequence of kernel bandwidths $\sigma_1 < \sigma_2 < \cdots < \sigma_\eta$, $\eta = 20$, chosen equally spaced from $[0.1\hat{\sigma}, 2\hat{\sigma}] = [0.225, 4.492]$, where $\hat{\sigma}$ is the larger one of the sample standard deviations of the two dimensions. Among the 20 different σ_l's, only 6 of them result in clustering different from σ_{l-1}, reflecting the fact that the bandwidth needs to increase by a sufficient amount to drive the merging of some existing cluster representatives. The number of clusters at the 6 levels is sequentially $21, 11, 5, 3, 2, 1$.

We demonstrate the clustering results at the 2nd and 3rd level in Figure 1. At the 2nd level, 11 clusters are formed, as shown by different symbols in Figure 1(a). The 11 modes identified at level 2 are merged into 5 modes at level 3 when the bandwidth increases from 0.449 to 0.674. Figure 1(b) shows the ascending paths generated by MEM from the 11 modes (crosses) at level 2 to the 5 modes (squares) at level 3. The contour lines of the density function with the corresponding bandwidth of level 3 are plotted in the background for better illustration. The 5 clusters at level 3 are shown in Figure 1(c). These 5 modes are again merged into 3 modes at level 4, as shown in Figure 1(d).

3.3 Measures for Enhancing Speed

Because the nonparametric density estimate in (1) is a sum of kernels centered at every data point, the amount of computation to identify the mode associated with a single point grows linearly with n, the size of the data set. The computational complexity of clustering all the data by MAC is thus quadratic in n. In practice, however, it is often unnecessary to use the basic kernel estimate. A preliminary clustering can be first applied to $\{x_1,...,x_n\}$ to yield m clusters, where m is significantly smaller than n, but still much larger than the desired number of clusters. Suppose the m cluster centroids are $\overline{S} = \{\overline{x}_1,...,\overline{x}_m\}$ and the number of points in cluster $\overline{S}_j$ is n_j, $j = 1,2,...,m$. We use the density estimate

$$f(x \mid \overline{S}, D(\sigma^2)) = \sum_{j=1}^{m} \frac{n_j}{n} \phi(x \mid \overline{x}_j, D(\sigma^2))$$

in MAC to cluster the x_i's. Since MEM applies to general mixture models, the modified density function causes no essential changes to the clustering procedure.

The purpose of the preliminary clustering is more of quantizing the data than clustering. Computation is reduced by not discerning points in the same quantization region when formulating the density estimate. If m is sufficiently large, $\overline{S}$ is adequate to retain the topological structures in the nonparametric density estimate. In this fast version of MAC, we search for a mode for every $\overline{x}_i$. Examples exploiting the fast MAC are given in Section 6.

4. Analysis of Cluster Separability via Ridgelines

A measure for the separability between clusters is useful for gaining deeper understanding of clustering structures in data. With this measure, the difference between clusters is quantified, rather than being simply categorical. This quantification can be useful in certain situations. For instance, in taxonomy study, after grouping instances into species, scientists may need to numerically assess the disparity between species, often taken as an indicator for evolutionary proximity. A separability measure between the clusters of species can effectively reflect such disparity. Such a measure is also useful for diagnosing clustering results and for the mere interest of designing clustering algorithms. Based upon it, we derive a mechanism to merge weakly separated clusters. Although the separability measure is a diagnostic tool and the cluster merging method can adjust the number of clusters, in this paper, we do not pursue the problem of choosing the number of clusters fully automatically. It is well known that determining this number is a deep problem, and domain knowledge often needs to be called upon for a final decision in various applications.

The separability measure we define here exploits the geometry of the density functions of two clusters in a comprehensive manner. We only require the cluster pdf to be a mixture distribution, for example, a Gaussian kernel density estimate. Before formulating the separability measure, we

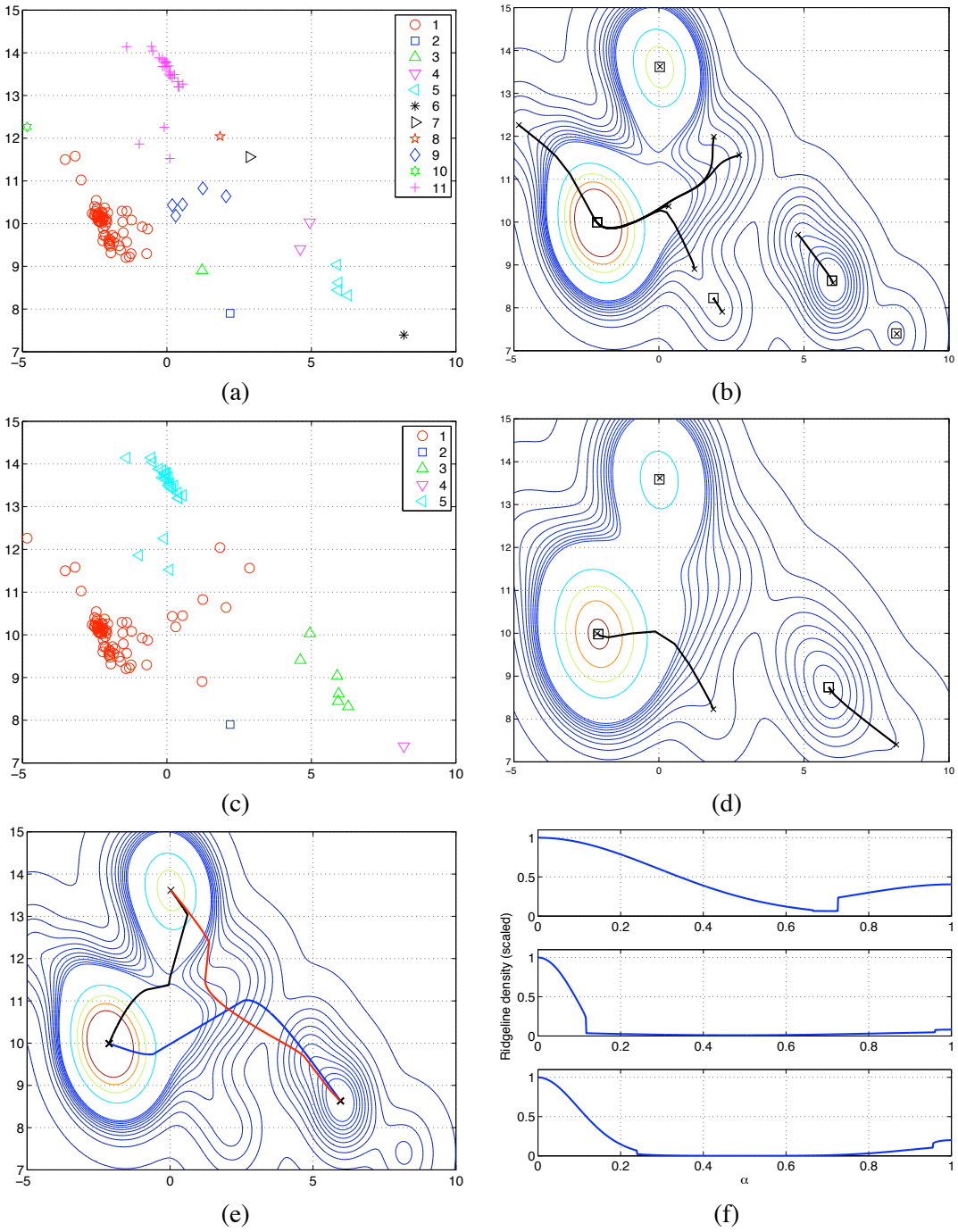


Figure 1: Clustering results for the glass data set obtained from HMAC. (a) The 11 clusters at level 2. (b) The MEM ascending paths from the modes at level 2 (crosses) to the modes at level 3 (squares), and the contours of the density estimate at level 3. (c) The 5 clusters at level 3. (d) The ascending paths from the modes at level 3 (crosses) to those at level 4 (squares) and the contours of the density estimate at level 4. (e) Ridgelines between the 3 major clusters at level 3. (f) The density function along the 3 ridgelines.

define a ridgeline between two unimodal clusters. The REM algorithm is developed to solve the ridgeline.

4.1 Ridgeline

The ridgeline between two clusters with density $g_1(x)$ and $g_2(x)$ is

$$\mathcal{L} = \{x(\alpha) : (1-\alpha)\nabla \log g_1(x) + \alpha \nabla \log g_2(x) = 0, 0 \le \alpha \le 1\} . \tag{3}$$

For a mixture density of the two clusters, $\tilde{g}(x) = \pi_1 g_1(x) + \pi_2 g_2(x)$, $\pi_1 + \pi_2 = 1$, if $\tilde{g}(x) > 0$ for any x, the modes, antimodes (local minimums), and saddle points of $\tilde{g}(x)$ must occur in $\mathcal{L}$ for any prior probability π_1. The locations of these critical points, however, depend on π_1. This fact is referred to as the *critical value theorem* and is proved by Ray and Lindsay (2005).

Remarks:

1. Eq. (3) is precisely the critical point equation for the *exponential tilt* density $g(x|\alpha) = c(\alpha) g_1(x)^{1-\alpha} g_2(x)^{\alpha}$, where $c(\alpha)$ is a normalizing constant. This density family is an exponential family, with sufficient statistic $\log(g_2(x)/g_1(x))$.

2. The set of solutions in $\mathcal{L}$ is, in general, a 1-dimensional manifold; that is, a curve. When both g_1 and g_2 are normal densities, the solution is explicit (see Ray and Lindsay, 2005), and the solutions form a unique one-dimensional curve. More generally, the solutions are possibly a set of curves that pass through the modes of the $g_1(x)$ and $g_2(x)$.

3. If both g_1 and g_2 are unimodal and have convex upper contour sets, it can be proved that the solutions form a unique curve between the modes of g_1 and g_2 respectively. In our discussion, we assume unimodal g_1 and g_2.

Since the local maxima of the exponential tilt function $g(x;\alpha)$ satisfy Eq. (3), we solve (3) by maximizing $\log(g(x|\alpha)) = (1-\alpha)\log g_1(x) + \alpha \log g_2(x)$. In the case when the two cluster densities g_1 and g_2 are themselves mixtures of basic parametric distributions, for example, normal, we develop an ascending algorithm to maximize the function, referred to as the *Ridgeline EM (REM)* Algorithm. For notational brevity, assume that both g_1 and g_2 are mixtures of T parametric distributions:

$$g_i(x) = \sum_{\kappa=1}^{T} \pi_{i,\kappa} h_{i,\kappa}(x) , \quad i = 1,2 .$$

Starting from an initial value $x^{(0)}$, REM updates x by iterating the two steps:

1. Compute

$$p_{i,\kappa} = \pi_{i,\kappa} h_{i,\kappa}(x^{(r)}) / \sum_{j=1}^{T} \pi_{i,j} h_{i,j}(x^{(r)}) , \ \kappa = 1,...,T , \ i = 1,2 .$$

2. Update $x^{(r+1)}$:

$$x^{(r+1)} = \underset{x}{\operatorname{argmax}} (1-\alpha) \sum_{\kappa=1}^{T} p_{1,\kappa} \log h_{1,\kappa}(x) + \alpha \sum_{\kappa=1}^{T} p_{2,\kappa} \log h_{2,\kappa}(x) .$$

REM ensures that $g(x^{(r+1)} \mid \alpha) \geq g(x^{(r)} \mid \alpha)$. Proof is given in Appendix B. As with MEM, we will not rigorously study the convergence properties of the sequence $\{x^{(r)}\}$. In the special case $h_{i,\kappa}(x) = \phi(x \mid \mu_{i,\kappa}, \Sigma)$, the update equation for $x^{(r+1)}$ becomes

$$x^{(r+1)} = (1-\alpha) \sum_{\kappa=1}^{T} p_{1,\kappa}\mu_{1,\kappa} + \alpha \sum_{\kappa=1}^{T} p_{2,\kappa}\mu_{2,\kappa} \, .$$

The Gaussian kernel density estimate belongs to this case.

At the two extreme values $\alpha = 0, 1$, the solutions are the modes of $g_1(x)$ and $g_2(x)$ respectively. We solve $x(\alpha)$ sequentially on a set of grid points $0 = \alpha_0 < \alpha_1 < \cdots < \alpha_\xi = 1$. First, $x(0) = \text{argmax}_x g_1(x)$ is solved by MEM. For every α_l, $x(\alpha_{l-1})$, previously calculated, is used as initial value to start the iterations in REM.

Suppose two clusters, denoted by z_1 and z_2, have densities g_1 and g_2, and prior probabilities π_1 and π_2 respectively. We define a pairwise *separability* as

$$S(z_1, z_2) = 1 - \frac{\min_\alpha \pi_1 g_1(x(\alpha)) + \pi_2 g_2(x(\alpha))}{\pi_1 g_1(x(0)) + \pi_2 g_2(x(0))} = 1 - \frac{\pi_1 g_1(x(\alpha^*)) + \pi_2 g_2(x(\alpha^*))}{\pi_1 g_1(x(0)) + \pi_2 g_2(x(0))} \tag{4}$$

where $\alpha^* = \text{argmin}_\alpha \pi_1 g_1(x(\alpha)) + \pi_2 g_2(x(\alpha))$. Usually, the prior probabilities π_1 or π_2 are proportional to the cluster sizes. It is obvious that $S(z_1, z_2) \in [0, 1]$. To symmetrize the measure, we define the pairwise *symmetric separability* as $\tilde{S}(z_1, z_2) = \min[S(z_1, z_2), S(z_2, z_1)]$.

By finding $x(\alpha^*)$, we can evaluate the amount of "dip" along the ridgeline. By the critical value theorem, the minimum of $\pi_1 g_1(x) + \pi_2 g_2(x)$, if exists, lies on the ridgeline and therefore must be $x(\alpha^*)$. Hence, if there is a "dip" in the mixture of the two clusters, it will be captured by the ridgeline. According to definition (4), a deeper "dip" leads to higher separability. In our implementation, we approximate α^* by

$$\alpha^* \approx \underset{\alpha_l, 0 \leq l \leq \zeta}{\text{argmin}} \, \pi_1 g_1(x(\alpha_l)) + \pi_2 g_2(x(\alpha_l)) \, ,$$

where $\alpha_l, 0 \leq l \leq \zeta$, are the grid points.

We also define the separability for a single cluster to quantify its overall distinctness from other clusters. Specifically, suppose there are m clusters denoted by z_i, $i = 1, ..., m$. The separability of cluster z_i, denoted by $s(z_i)$ is defined by

$$s(z_i) = \min_{j: 1 \leq j \leq m, j \neq i} S(z_i, z_j) \, .$$

We call a cluster "insignificant" if its separability is below a given threshold ε. In our discussion, $\varepsilon = 0.5$.

Take the glass data set as an example. As shown in Figure 1(c), the points are divided into 5 groups at that level of the clustering hierarchy. Two of the clusters each contain a single sample point, which is far from the other points and forms a mode alone. The separability of these two clusters are weak, respectively 0.00 and 0.30. The three other clusters are highly separable, with separability values 0.94, 0.84 and 0.81. Figure 1(e) shows the ridgelines between any two of the three significant clusters, and Figure 1(f) shows the density function along the ridgelines, normalized to one at the ridgeline end point $x(0)$ or $x(1)$ (depending on whichever is larger).

The task of identifying insignificant clusters in Figure 1(c) is not particularly challenging because the two smallest clusters are "singletons" (containing only one sample). However, cluster size

is not the sole factor affecting separability. Take the clusters in Figure 1(a) as an example. The separability of the 11 clusters and their corresponding sizes (number of points contained) are listed in Table 1. It is shown that although cluster 9 is the third largest cluster, its separability is low. At threshold $\varepsilon = 0.5$, this cluster is insignificant. In contrast, by being far from all the other clusters, cluster 6, a singleton, has separability 0.87. The low separability of cluster 9 is caused by its proximity to cluster 1, the largest cluster which accounts for 60% of the data. Clusters that contain a large portion of data tend to "absorb" surrounding smaller clusters. The attraction of a small cluster to a bigger one depends on its relative size, tightness, distance to the bigger cluster, as well as the orientation of the data masses in the two clusters.

Cluster	1	2	3	4	5	6	7	8	9	10	11
Size	63	1	1	2	4	1	1	1	5	1	25
Separability	0.94	0.41	0.41	0.31	0.68	0.87	0.13	0.13	0.18	0.46	0.92

Table 1: Separability between clusters in the glass data set

4.2 Merging Clusters Based on Separability

To elaborate on the relationships between clusters, we compute the matrix of separability between any pair of clusters. We can potentially use this matrix to decide which clusters can be merged due to weak separation. As discussed previously, one way to merge clusters is to increase the bandwidth of the kernel function used by HMAC. However, an enlarged bandwidth may cause prominent clusters to be clumped while leaving undesirable small "noisy" clusters unchanged. Merging clusters according to the separability measure is one possible approach to eliminate "noisy" clusters without losing important clustering structures found at a small bandwidth. We will show by the glass data that the merging method makes clustering results less sensitive to bandwidth.

Let the clusters in consideration be $\{z_1, z_2, ..., z_m\}$. We denote the pairwise separability between cluster z_i and z_j in short by $S_{i,j}$, where $S_{i,j} = S(z_i, z_j)$. Note in general $S_{i,j} \neq S_{j,i}$. Let a threshold for separability be ε, $0 < \varepsilon < 1$. Let the density function of cluster z_i be $g_i(\cdot)$ and the prior probability be π_i. Denote the weighted mode of each cluster density function by $\delta(z_i) = \pi_i \max g_i(x)$. Since in MEM the mode $\max g_i(x)$ for each cluster z_i is computed when z_i is formed, $\delta(z_i)$ requires no extra computation after clustering. We refer to $\delta(z_i)$ as the *significance index* of cluster z_i.

The main idea of the merging algorithm is to have clusters with a higher significance index absorb other clusters that are not well separated from them and are less dominant (lower significance index). Several issues need to be resolved to avoid arbitrariness in merging. First, a cluster z_i may be weakly separated from several other clusters with higher significance indices. Among those clusters, we let the one from which z_i is worst separated to absorb z_i. Second, two weakly separated clusters z_i and z_j may have the same significance indices, that is, $\delta(z_i) = \delta(z_j)$; and hence it is ambiguous which cluster should be treated as the dominant one. We solve this problem by introducing the concept of *cliques*. The clusters are first grouped into cliques which contain weakly separated z_i's with the same value of $\delta(z_i)$. Clusters in different cliques are ensured to be either well separated or have different significance indices. We then apply the absorbing process to the cliques, without the possibility of encountering the aforementioned ambiguity.

Next, we describe how to form cliques and extend the definition of pairwise separability to cliques. In order to carry out the merging of cliques, a directed graph is constructed upon the cliques based on pairwise separability and the comparison of significance indices.

1. *Tied*: Cluster z_i and z_j are defined to be *tied* if $S(z_i, z_j) < \varepsilon$ and $\delta(z_i) = \delta(z_j)$.

2. *Clique*: Cluster z_i and z_j are in the same *clique* if (a) z_i and z_j are tied, or (b) there is a cluster z_k such that z_i and z_k are in the same clique, and z_j and z_k are in the same clique.

   Remark: the relationship "tied" results in a partition of z_i's. Each group formed by the partition is a clique. Because being tied requires $\delta(z_i) = \delta(z_j)$, in practice, we only observe clusters being tied when they are all singletons.

We now define the *separability between cliques*. Without loss of generality, let clique $c_1 = \{z_1, z_2, ..., z_{m_1}\}$ and $c_2 = \{z_{m_1+1}, ..., z_{m_1+m_2}\}$. Denote the clique-wise separability by $S_c(c_1, c_2)$:

$$S_c(c_1, c_2) \triangleq \min_{1 \le i \le m_1} \min_{m_1+1 \le j \le m_1+m_2} S(z_i, z_j) \, .$$

Since in general, $S(z_i, z_j) \neq S(z_j, z_i)$, the asymmetry carries over to $S_c(c_1, c_2) \neq S_c(c_2, c_1)$. We also denote the significance index of a clique c_i as $\delta(c_i)$. Since all the clusters in c_i have equal significance indices, we let $\delta(c_i) = \delta(z_{i'})$, where $z_{i'}$ is any cluster included in c_i.

Regard each clique as a node in a graph. Suppose there are $\tilde{m}$ cliques $\{c_1, ..., c_{\tilde{m}}\}$. A directed graph is built as follows. For two arbitrary cliques c_i and c_j, a link from c_i to c_j is made if

1. $S_c(c_i, c_j) < \varepsilon$.

2. $S_c(c_i, c_j) = \min_{k \neq i} S_c(c_i, c_k)$ and j is the smallest index among all those j''s that achieve $S_c(c_i, c_{j'}) = \min_{k \neq i} S_c(c_i, c_k)$.

3. $\delta(c_i) < \delta(c_j)$.

A clique c_i is said to be *linked* to c_j if there is a directed edge from c_i to c_j. It is obvious by the second requirement in the link construction that every clique is linked to at most another clique. In Appendix C, it is proved that a graph built by the above rules has no loops. An example graph is illustrated in Figure 2(b). If we disregard the directions of the links, the graph is partitioned into connected subgraphs. In the given example in Figure 2(b), there are four connected subgraphs. The basic idea of the merging algorithm is to let the most dominant clique in one subgraph absorb all the others in the same subgraph.

We call clique c_j the *parent clique* of c_i if there is a directed link from c_i to c_j. In this case, we also say c_i is *directly absorbed* by c_j. By construction, $\delta(c_i) < \delta(c_j)$. More generally, if there is a directed path from c_i to c_j, then c_j is called an *ancestor* of c_i, and c_i is *absorbed* by c_j. Again, we have $\delta(c_i) < \delta(c_j)$ by transitivity. In each connected subgraph containing k nodes, because there is no loop, there are exactly $k-1$ links. Since every node has at most one link originating from it, the $k-1$ links have to originate from $k-1$ different nodes. Therefore, there is precisely 1 node in each connected subgraph that has no link originating from it. This node is called the *head* node (clique) of the connected nodes. It is not difficult to see that the head node is an ancestor for all the other nodes in the subgraph. As a result, the significance index of the head clique is strictly larger

than that of any other clique in the subgraph. In this sense, the head clique dominates all the other connected cliques.

Combining clusters by the above method is the first and main step in our merging algorithm. To account for outliers, the second step in the algorithm employs the notation of *coverage rate*. Outlier points far from all the essential clusters tend to yield high separability and hence will not be merged. In HMAC, to "grab" those outliers, the kernel bandwidth needs to grow so large that under such a bandwidth, many significant clusters are undesirably merged. To address this issue, we find the smallest clusters and mark them as outliers if their total size proportional to the entire data set is below a threshold. For instance, if the *coverage rate* allowed is 95%, this threshold is then 5%.

We now summarize our cluster merging algorithm as follows. We call the merging procedure conducted based on the separability measure stage I merging and that based on coverage rate stage II merging. The two stages do not always have to be concatenated. We can apply each alone. Applying only stage I is equivalent to applying two stages and setting the coverage rate to 100%; applying only stage II is equivalent to setting the threshold of separability to 0.0. Assume the starting clusters are $\{z_1, z_2, ..., z_m\}$.

1. Stage I: merging based on separability.
   (a) Compute the separability matrix $[S_{i,j}]$, $i, j = 1, ..., m$, and the significant index $\delta(z_i)$, $i = 1, ..., m$.
   (b) Form cliques $\{c_1, c_2, ..., c_{\tilde{m}}\}$ based on $[S_{i,j}]$ and $\delta(z_i)$'s, where $\tilde{m} \leq m$. Record the z_i's contained in each clique.
   (c) Construct the directed graph.
   (d) Merge cliques that are in the same connected subgraph. z_i's contained in merged cliques are grouped into one cluster. Denote those merged clusters by $\{\hat{z}_1, \hat{z}_2, ..., \hat{z}_{\hat{m}}\}$, where $\hat{m} \leq \tilde{m}$.
2. Stage II: merging based on coverage rate. Denote the coverage rate by ρ.
   (a) Calculate the sizes of clusters $\{\hat{z}_1, \hat{z}_2, ..., \hat{z}_{\hat{m}}\}$ and denote them by $\hat{n}_i$, $i = 1, ..., \hat{m}$. The size of the whole data set is $n = \sum_{i=1}^{\hat{m}} \hat{n}_i$.
   (b) Sort $\hat{n}_i$, $i = 1, ..., \hat{m}$, in ascending order. Let the sorted sequence be $\hat{n}_{(1)}$, $\hat{n}_{(2)}$, ..., $\hat{n}_{(\hat{m})}$. Let $\hat{n}_{(0)} = 0$. Let k be the largest integer such that $\sum_{i=0}^{k} \hat{n}_{(i)} \leq (1-\rho)n$.
   (c) If $k > 0$, go to the next step. Otherwise, stop and the final clusters are $\{\hat{z}_1, \hat{z}_2, ..., \hat{z}_{\hat{m}}\}$.
   (d) For each $\hat{z}_{(i)}, i = 1, ..., k$, find all the original clusters z_j's that are merged into $\hat{z}_{(i)}$. Denote the index set of the z_j's by $H_{(i)}$. Let $H' = \cup_{i=1}^{k} H_{(i)}$ and $H'' = \cup_{i=k+1}^{\hat{m}} H_{(i)}$.
   (e) For each $\hat{z}_{(i)}, i = 1, ..., k$, find $j^* = \text{argmin}_{j \in H''} \min_{l \in H_{(i)}} S(z_l, z_j)$. Find the cluster $\hat{z}_{j'}$ that contains z_{j^*}. Merge $\hat{z}_{(i)}$ with $\hat{z}_{j'}$. The new clusters obtained are $\{\overline{z}_1, \overline{z}_2, ..., \overline{z}_{\overline{m}}\}$, where $\overline{m} = \hat{m} - k < \hat{m}$.
   (f) Reset $\overline{m} \to \hat{m}$ and $\overline{z}_i \to \hat{z}_i$, $i = 1, ..., \overline{m}$. Go back to step (a).

Unless there is a definite need to assign every data point to one of the major clusters, in certain applications, it may be more preferable to keep the outlier status of some points rather than allocating them to distant clusters.

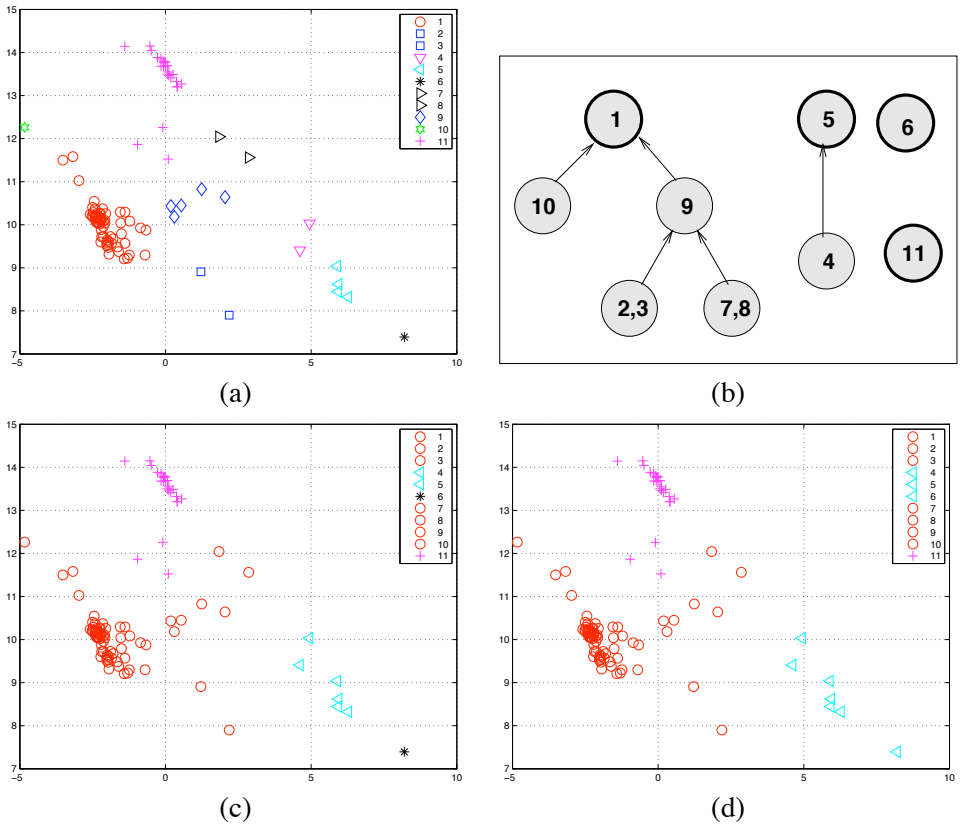


Figure 2: The process of merging clusters for the glass data set. (a) The clustering result after merging clusters in the same cliques. (b) The cliques and directed graph constructed based on separability. (c), (d) The clustering results after stage I and stage II merging respectively.

The first stage merging based on separability is intrinsically connected with linkage-based agglomerative clustering. For details on linkage clustering, see Jain et al. (1999). In a nutshell, linkage clustering forms clusters by progressively merging a pair of current clusters. Initially, every data point is a cluster. The two clusters with the minimum pairwise distance are chosen to merge at each step. The procedure is greedy in nature since minimization is conducted sequentially through every merge. Linkage clustering methods differ by the way between-cluster distance is updated when a new cluster is combined from two smaller ones. For instance, in single linkage, if cluster ξ_2 and ξ_3 are merged into ξ_4, the distance between ξ_1 and ξ_4 is calculated as $d(\xi_1,\xi_4) = \min(d(\xi_1,\xi_2),d(\xi_1,\xi_3))$. If complete linkage, the distance becomes $d(\xi_1,\xi_4) = \max(d(\xi_1,\xi_2),d(\xi_1,\xi_3))$.

Our merging algorithm is a particular kind of linkage clustering where the elements to be clustered are cliques and the distance between the cliques is the separability. The update of this distance for merged groups of cliques is however different from commonly used versions in linkage clustering. The update is the same as single linkage under a certain scenario, but differs in general because of the directed links and the notion of head cliques. We may call this linkage clustering

algorithm *directional single linkage* for reasons that will be self-evident shortly. Consider the above example where ξ_2 and ξ_3 merge into ξ_4. We call ξ more dominant than ξ' if the head clique in ξ has a higher significance index than that in ξ', and denote $\delta(\xi) > \delta(\xi')$. Without loss of generality, assume $\delta(\xi_2) > \delta(\xi_3)$. Then the update of $d(\xi_1, \xi_4)$ and the ordering of $\delta(\xi_1)$ and $\delta(\xi_4)$ (needed for updating the distance) follows three cases:

$$\begin{cases} d(\xi_1,\xi_4) = d(\xi_1,\xi_2), \text{ and } \delta(\xi_1) > \delta(\xi_4), & \text{if } \delta(\xi_1) > \delta(\xi_2) \\ d(\xi_1,\xi_4) = d(\xi_1,\xi_2), \text{ and } \delta(\xi_1) < \delta(\xi_4), & \text{if } \delta(\xi_3) < \delta(\xi_1) < \delta(\xi_2) \\ d(\xi_1,\xi_4) = \min(d(\xi_1,\xi_2), d(\xi_1,\xi_3)), \text{ and } \delta(\xi_1) < \delta(\xi_4), & \text{if } \delta(\xi_1) < \delta(\xi_3) \ . \end{cases}$$

In our proposed merging procedure, we essentially employ a threshold to stop merging when all the between-cluster distances exceed this value. An alternative is to apply the directional single linkage clustering and stop merging when a desired number of clusters is achieved.

We use the glass data set in the previous section to illustrate the merging algorithm. The threshold for separability is set to $\varepsilon = 0.5$ and the coverage rate is $\rho = 95\%$. Apply the algorithm to the 11 clusters formed at the 2nd level of the hierarchical clustering, shown in Figure 1(a). We refer to the clusters as z_1, ..., z_{11}, where the label assignment follows the indication in the figure. The 11 clusters form 9 cliques. Figure 2(a) shows the clustering after merging clusters in the same clique. The square (triangle) symbol used for z_2 (z_7) is now used for both z_2 (z_7) and z_3 (z_8) to indicate that they have been merged. To highlight the relationship between the merged clusters and the original clusters, the list of updated symbols for each original cluster is given in every scatter plot. Figure 2(b) demonstrates the directed graph constructed for the cliques. The z_i's contained in each clique are indicated in the node. Figure 2(c) shows the clustering result after stage I merging. The symbol of the head clique in each connected subgraph is adopted for all the clusters it absorbs. The 4 clusters generated at stage I contain 73, 25, 6, 1 points respectively. At $\rho = 95\%$, only the cluster of size 1 is marked as an outlier cluster, and is merged with the cluster of size 6.

Because clusters with low separability are apt to be grouped together when the kernel bandwidth increases, it is not surprising for the clustering result obtained by the merging algorithm to agree with clustering at a higher level of HMAC. On the other hand, examining separability and identifying outlier clusters enhance the robustness of clustering results, a valuable trait especially in high dimensions. Examples will be shown in Section 6. For practical interest, when equipped with the merging algorithm, we do not need to generate all the hierarchical levels in HMAC until reaching a targeted number of clusters. Instead, we can apply the merging algorithm to a relatively large number of clusters obtained at an early level and reduce the number of clusters to the desired value.

5. Visualization

For clarity, we have used 2-D data to illustrate our new clustering methods, although these methods are not limited by data dimensions. Projection into lower dimensions is needed to visualize clustering results for higher dimensional data. PCA (principal component analysis) is a widely used linear projection method, but it is not designed to reveal clustering structures. We will describe in this section a linear projection method that aims at effectively showing clustering structures. The method is employed in our experiments. We note that visualization is a rich research area in its own right. However, because this topic is beyond the focus of the current paper, we will not discuss it in great depth, nor make thorough comparisons with other methods.

Modal clustering provides us an estimated density function and a prior probability for each cluster. Suppose K clusters are generated. Let the cluster density function of x, $x \in \mathcal{R}^d$, be $g_k(x)$, and the prior probability be $\pi_k, k = 1,2,...,K$. For any $x \in \mathcal{R}^d$, its extent of association with each cluster k is indicated by the posterior probability $p_k(x) \propto \pi_k g_k(x)$. To determine the posterior probabilities $p_k(x)$, under a given set of priors, it suffices to specify the discriminant functions $\log \frac{g_1(x)}{g_K(x)}$, ..., $\log \frac{g_{K-1}(x)}{g_K(x)}$. Without loss of generality, we use $g_K(x)$ as the basis for computing the ratios. Our projection method attempts to find a plane such that $\log \frac{g_k(x)}{g_K(x)}, k = 1,...,K-1$ can be well approximated if only the projection of data into the plane is specified. By preserving the discriminant functions, the posterior probabilities of clusters will remain accurate.

Let the data set be $\{x_1, x_2, ..., x_n\}$, $x_i \in \mathcal{R}^d$. Denote a particular dimension of the data set by $x_{\cdot,l} = (x_{1,l}, x_{2,l}, ..., x_{n,l})^t, l = 1,...,d$. For each $k, k = 1,...,K-1$, the pairs $(x_i, \log \frac{g_k(x_i)}{g_K(x_i)}), i = 1,...,n$, are computed. Let $y_{i,k} = \log \frac{g_i(x_i)}{g_K(x_i)}$. Linear regression is performed based on the pairs $(x_i, y_{i,k}), i = 1,...,n$, to acquire a linear approximation for each discriminant function. Let $\beta_{k,0}, \beta_{k,1}, \beta_{k,2}, ..., \beta_{k,d}$ be the regression coefficients for the kth discriminant function. Denote $\beta_k = (\beta_{k,1}, \beta_{k,2}, ..., \beta_{k,d})^t$ and the fitted values for $\log \frac{g_k(x_i)}{g_K(x_i)}$ by $\hat{y}_{i,k} = \beta_{k,0} + \beta_k^t x_i$. Also denote $\tilde{y}_{i,k} = \beta_k^t x_i = \hat{y}_{i,k} - \beta_{k,0}$. For mathematical tractability, we convert the approximation of the discriminant functions to the approximation of their linearly regressed values $(\hat{y}_{i,1}, \hat{y}_{i,2}, ..., \hat{y}_{i,K-1}), i = 1,...,n$, which is equivalent to approximate $(\tilde{y}_{i,1}, \tilde{y}_{i,2}, ..., \tilde{y}_{i,K-1})$ since the two only differ by a constant. To precisely specify $(\tilde{y}_{i,1}, \tilde{y}_{i,2}, ..., \tilde{y}_{i,K-1})$, we need the projection of x_i onto the $K-1$ directions, $\beta_1, \beta_2, ..., \beta_{K-1}$. If we are restricted to showing the data in a plane and $K-1 > 2$, further projection of $(\tilde{y}_{i,1}, \tilde{y}_{i,2}, ..., \tilde{y}_{i,K-1})$ is needed. At this stage, we employ PCA on the vectors $(\tilde{y}_{i,1}, \tilde{y}_{i,2}, ..., \tilde{y}_{i,K-1})$ (referred to as the discriminant vectors), $i = 1,...,n$, to yield a two-dimensional projection. Suppose the two principal component directions for the discriminant vectors are $\gamma_j = (\gamma_{j,1}, ..., \gamma_{j,K-1})^t$, $j = 1,2$. The two principal components v_j, $j = 1,2$, are

$$\begin{pmatrix} v_{1,j} \\ v_{2,j} \\ \vdots \\ v_{n,j} \end{pmatrix} = \gamma_{j,1} \begin{pmatrix} \tilde{y}_{1,1} \\ \tilde{y}_{2,1} \\ \vdots \\ \tilde{y}_{n,1} \end{pmatrix} + \cdots + \gamma_{j,K-1} \begin{pmatrix} \tilde{y}_{1,K-1} \\ \tilde{y}_{2,K-1} \\ \vdots \\ \tilde{y}_{n,K-1} \end{pmatrix} = \sum_{l=1}^{d} \left[\sum_{k=1}^{K-1} \gamma_{j,k} \beta_{k,l} \right] x_{\cdot,l} \, .$$

To summarize, the two projection directions for x_i are

$$\tau_j = \left(\sum_{k=1}^{K-1} \gamma_{j,k} \beta_{k,1}, \sum_{k=1}^{K-1} \gamma_{j,k} \beta_{k,2}, ..., \sum_{k=1}^{K-1} \gamma_{j,k} \beta_{k,d} \right)^t, \quad j = 1,2 \, . \tag{5}$$

In practice, it may be unnecessary to preserve all the $K-1$ discriminant functions. We can apply the above method to a subset of discriminant functions corresponding to major clusters. The two projection directions in (5) are not guaranteed to be orthogonal, but it is easy to find two orthonormal directions spanning the same plane.

If we use a basis function other than $g_K(x)$, say $g_{k'}(x)$, to form the discriminant functions, the new set of vectors β_k's will span the same linear space as the β_k's obtained with $g_K(x)$. The reason is that the new discriminant functions $\log \frac{g_k(x_i)}{g_{k'}(x_i)}, 1 \le k \le K, k \ne k'$, can be linearly transformed from the previously defined $(y_{i,1}, y_{i,2}, ..., y_{i,K-1})$ by $\log \frac{g_k(x_i)}{g_{k'}(x_i)} = y_{i,k} - y_{i,k'}$, for $k \ne k', K$, $\log \frac{g_K(x_i)}{g_{k'}(x_i)} = -y_{i,k'}$, and linear regression is used on the $y_{i,k}$'s. On the other hand, as the linear transform is not orthonormal, the PCA result is not invariant under the transform and the projection directions τ_j can change.

6. Experiments

We present in this section experimental results of the proposed modal clustering methods on simulated and real data. We also discuss measures for enhancing computational efficiency and describe the application to image segmentation, which may involve clustering millions of data points.

6.1 Simulated Data

We experiment with three simulated examples to illustrate the effectiveness of modal clustering. We start by comparing linkage clustering with mixture modeling using two data sets. This will allow us to illustrate the strengths and weaknesses of these two methods and therefore better demonstrate the tendency of modal clustering to combine their strengths. We then present a study to assess the stability of HMAC over multiple implementations and its performance under increasing dimensions. In this study, comparisons are made with the `Mclust` function in `R`, a state-of-the-art mixture-model-based clustering tool (Fraley and Raftery, 2002, 2006).

For our two data sets, single linkage, complete linkage, and average linkage yield similar results. For brevity, we only show results of average linkage. In average linkage, if cluster z_2 and z_3 are merged into z_4, the distance between z_1 and z_4 is calculated as $d(z_1,z_4) = \frac{n_2}{n_2+n_3} d(z_1,z_2) + \frac{n_3}{n_2+n_3} d(z_1,z_3)$, where n_2 and n_3 are the cluster sizes of z_2 and z_3 respectively. Details on clustering by mixture modeling are referred to Section 1.1. We will also show results of k-means clustering.

The first data set, referred to as the noisy curve data set, contains a half circle and a straight line (or bar) imposed with noise, as shown in Figure 3(a). The circle centers at the origin and has radius 7. The line is a vertical segment between $(13,-8)$ and $(13,0)$. Roughly $\frac{2}{3}$ of the 200 points are uniformly sampled from the half circle and $\frac{1}{3}$ of them uniformly from the bar. Then, independent Gaussian noise with standard deviation 0.5 is added to both the horizontal and vertical directions of each point.

Consider clustering into two groups. The results of average linkage, mixture modeling, and k-means are shown in Figure 3(a), (b), (c). For this example, average linkage partitions the data perfectly into a noisy half circle and bar. Results of mixture modeling and k-means are close. In both cases, nearly one side of the half circle is grouped with the bar. In this example, the mixture model is initialized by the clustering obtained from k-means; and the covariance matrices of the two clusters are not restricted.

The second data set contains 200 samples generated as follows. The data are sampled from two clusters with prior probability $1/3$ and $2/3$ respectively. The first cluster follows a single Gaussian distribution with mean $(6,0)$ and covariance matrix $diag(1.5^2,1.5^2)$. The second cluster is generated by a mixture of two Gaussian components with prior probability $1/5$ and $4/5$, means $(-3,0)$ and $(0,0)$, and covariance matrices $diag(3^2,3^2)$ and $diag(1,1)$ respectively. The two clusters are shown in Figure 4(a). Again, we compare results of average linkage, mixture modeling, and k-means, shown in Figure 4(b), (c), (d). For mixture modeling, we use Mclust with three components and optimally selected covariance structures by BIC. Two of the three clusters generated by Mclust are combined to show the binary grouping. For this example, mixture modeling and k-means yield a partition close to the two original clusters, while average linkage gives highly skewed clusters, one of which contains a very small number of points on the outskirt of the mass of data.

We apply HMAC to both data sets and show the clustering results obtained at the level where two clusters are formed. For the noisy curve data set, HMAC perfectly separates the noisy half circle and the bar, as shown in Figure 3(a). For the second example, shown in Figure 4(e), HMAC yields

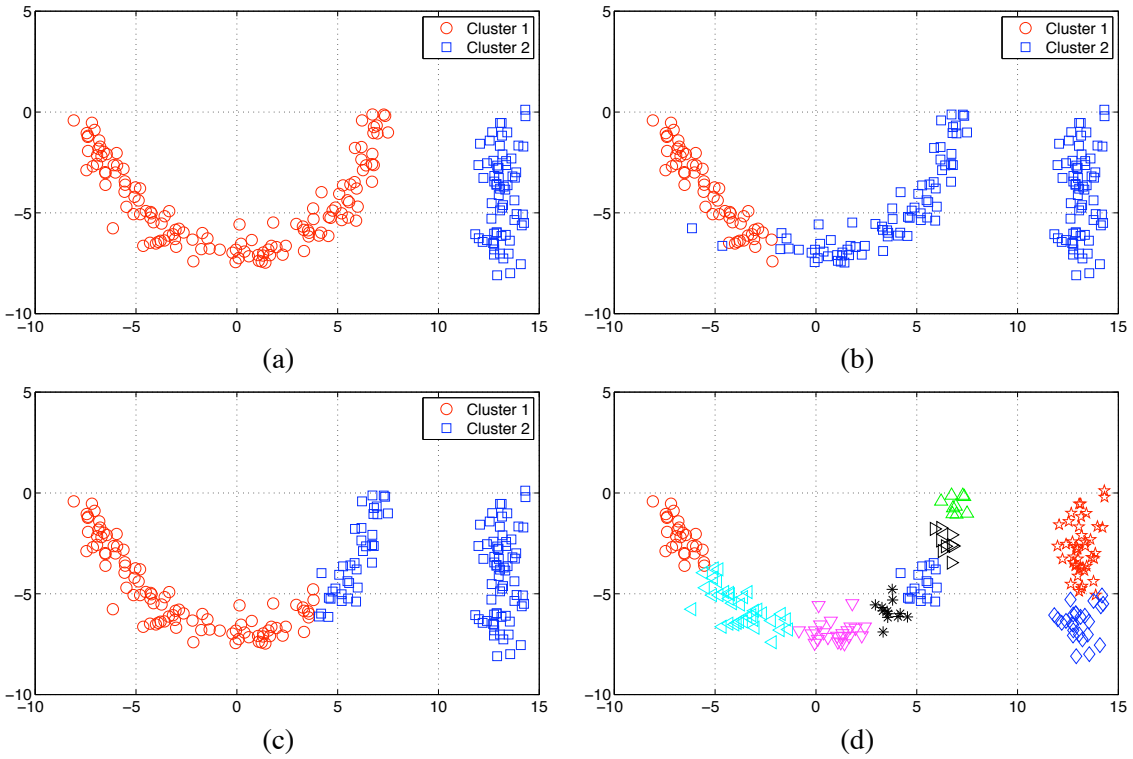


Figure 3: Clustering results for the noisy curve data set. (a) The two original clusters. The two clusters obtained by average linkage clustering and HMAC are identical to the original ones. (b) Clustering by mixture modeling with two Gaussian components. (c) Clustering by k-means. (d) Clustering by HMAC at the first level of the hierarchy.

clusters closest to the original ones among all the methods. Comparing with mixture modeling, HMAC is more robust to the non-Gaussian cluster distributions.

The above two data sets exemplify situations in which either the average linkage clustering or mixture modeling (or k-means) performs well but not both. In the first data set, the two clusters are well separated but seriously violate the assumption of Gaussian distributions. By greedy pairwise merging, average linkage successfully divides the two clusters. In contrast, both mixture modeling and k-means attempt to optimize an overall clustering criterion. Mixture modeling favors elliptical clusters because of the Gaussian assumption, and k-means favors spherical clusters due to extra restrictions on Gaussian components. As a result, one side of the noisy half circle is grouped with the bar to achieve better fitting of Gaussian distributions. On the other hand, mixture modeling and k-means perform significantly better than average linkage in the second example. The greedy pairwise merging in average linkage becomes rather arbitrary when clusters are not well separated.

HMAC demonstrates a blend of traits from average linkage and mixture modeling. When the kernel bandwidth is small, the cluster to which a point is assigned is largely affected by its neighbors. Points close to each other tend to be grouped together, as shown by Figure 3(d) and Figure 4(f). This strong inclination of putting neighboring points in the same cluster is also a feature of average linkage. However, a difference between HMAC and average linkage is that decisions to merge in

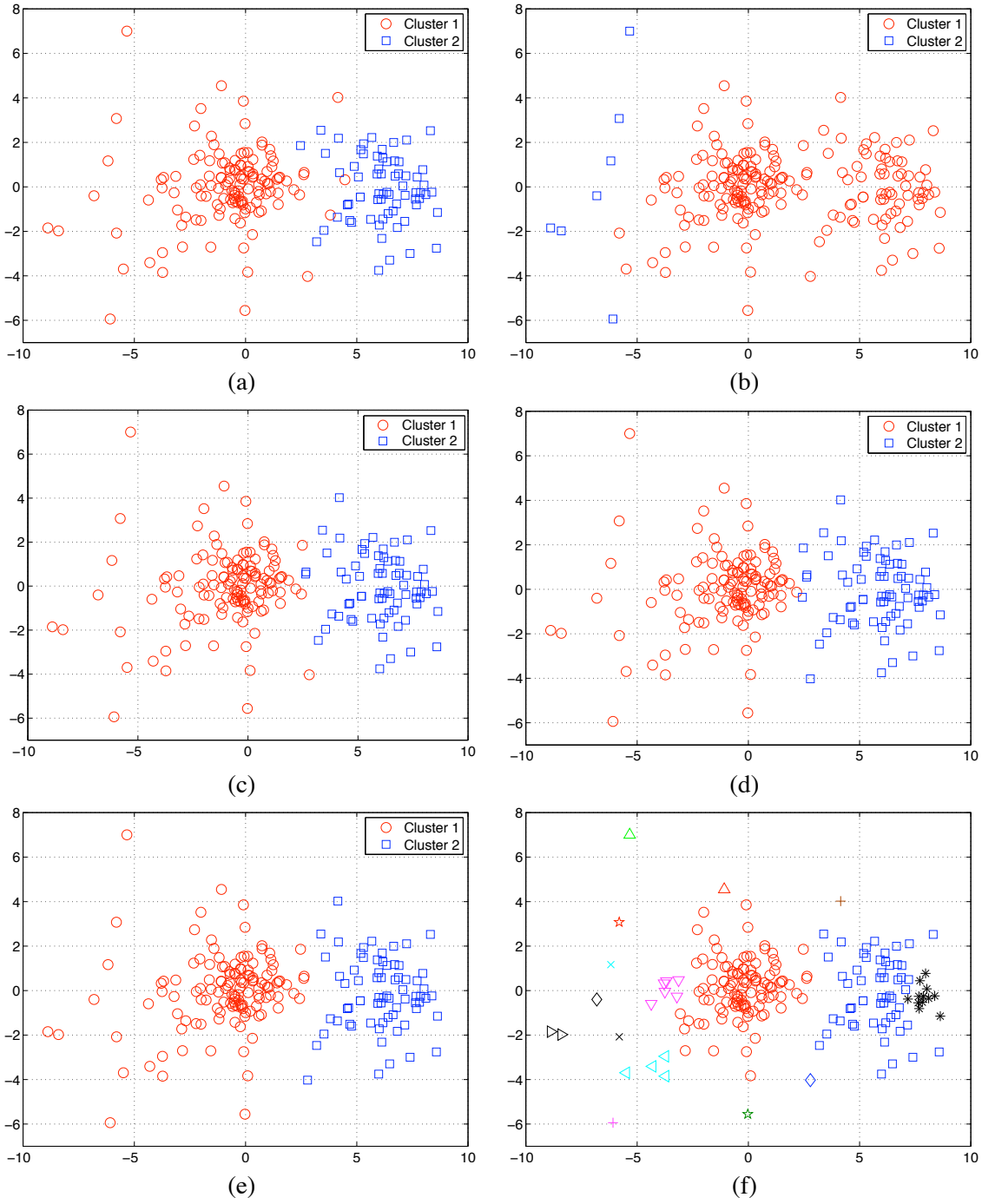


Figure 4: Clustering results for the second simulated data set. (a) The original two clusters. (b) Clustering by average linkage. (c) Clustering by mixture modeling using Mclust with three Gaussian components. Two clusters are merged to show the binary grouping. (d) Clustering by k-means. (e) Clustering by HMAC. (f) Clustering by HMAC at the first level of the hierarchy.

the latter are always local. While in HMAC, when the bandwidth increases, global characteristics of the data become highly influential on clustering results. Hence the clustering result tends to resemble that of mixture modeling (k-means) which enforces a certain kind of global optimality. In spite of this similarity, HMAC is more robust for clusters deviating from Gaussian distributions. In practice, it is usually preferable to ensure very close points are grouped together and in the mean time to generate clusters with global optimality. These two traits, however, are often at odds with each other, a phenomenon discussed in depth by Chipman and Tibshirani (2006).

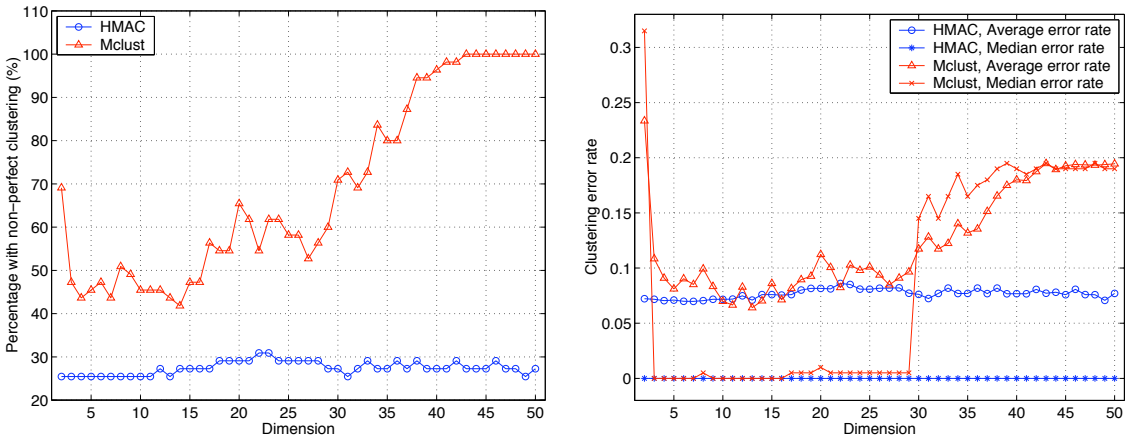
Chipman and Tibshirani (2006) noted that bottom-up agglomerative clustering methods, such as average linkage, tend to generate good small clusters but suffer at extracting a few large clusters. The strengths and weaknesses of top-down clustering methods, such as k-means, are the opposite. A hybrid approach is proposed in that paper to combine the advantages of bottom-up and top-down clustering, which first seeks mutual clusters by bottom-up linkage clustering and then applies top-down clustering to the mutual clusters. HMAC also integrates the strengths of both types of clustering, in a way not as explicit as the hybrid method but more automatically.

To systematically study the performance of HMAC for high dimensional data and its stability over multiple implementations, we conduct the following experiment. We generate 55 random data sets each of dimension 50 and size 200. The first two dimensions of the data follow the distribution of the noisy curve data in the first example described. The other 48 dimensions are independent Gaussian noise all following the normal distribution with mean zero and standard deviation 0.5, same as the noise added to the half circle and line segment in the first two dimensions. As gold standard, we regard data generated by adding noise to the half circle as one cluster and those to the line segment as the other.

Clustering results are obtained for each of the 55 data sets using HMAC and Mclust respectively. For HMAC, the level of the dendrogram yielding two clusters is chosen to create the partition of the data. In another word, the basic version of HMAC is used without the separability based merging of clusters. For Mclust, we set the number of clusters to 2, but allow the algorithm to optimally select the structure of the covariance matrices using BIC. All the structural variations of the covariance matrices provided by Mclust are searched over. In Mclust, the mixture model is initialized using the suggested default option, that is, to initialize the partition of data by an agglomerative hierarchical clustering approach, an extension of linkage clustering based on Gaussian mixtures (Fraley and Raftery, 2006). This initialization may be of advantage especially to data in this study because as shown previously, linkage clustering generates perfect clustering for the noisy curve data set in the first example.

Denote each data set k, $k = 1,2,...,55$, by $A_k = \{x_i^{(k)}, i = 1,...,200\}$, where $x_i^{(k)} = (x_{i,1}^{(k)}, x_{i,2}^{(k)}, ..., x_{i,50}^{(k)})^t$. For each $x_i^{(k)}$, we form a sequence of its lower dimensional vectors: $x_i^{(k,l)} = (x_{i,1}^{(k)}, x_{i,2}^{(k)}, ..., x_{i,l}^{(k)})^t$, $l = 2,3,...,50$. Let $A_{k,l} = \{x_i^{(k,l)}, i = 1,...,200\}$. HMAC and Mclust are applied to every $A_{k,l}$, $l = 2,...,50$, $k = 1,...,55$. Let the clustering error rate obtained by HMAC for $A_{k,l}$ be $r_{k,l}^{(H)}$ and that by Mclust be $r_{k,l}^{(M)}$. We summarize the clustering results in Figure 5.

To assess the variation of clustering performance over multiple implementations, we compute the percentage of the 55 data sets that are not perfectly clustered by the two methods at each dimension $l = 2,...,50$. Figure 5(a) shows the result. For HMAC, this percentage stays between 25% and 32% over all the dimensions. While, for Mclust, the percentage is consistently and substantially higher, and varies greatly across the dimensions. For $l \geq 43$, none of the data sets can be perfectly clustered by Mclust.



(a) (b)

Figure 5: Clustering results obtained by HMAC and Mclust for the 55 high dimensional noisy curve data sets. (a) The percentage of data sets that are not perfectly clustered with increasing dimensions. (b) The average and median of clustering error rates.

To examine clustering accuracy with respect to increasing dimensions, for each $l = 2, ..., 50$, the mean $\bar{r}_{\cdot,l}^{(H)}$, $\bar{r}_{\cdot,l}^{(M)}$, and the median $\tilde{r}_{\cdot,l}^{(H)}$, $\tilde{r}_{\cdot,l}^{(M)}$ over the data sets are computed and shown in Figure 5(b). For HMAC, the mean $\bar{r}_{\cdot,l}^{(H)}$ varies only slightly around 7.5% when the dimension increases, and the median $\tilde{r}_{\cdot,l}^{(H)}$ stays at zero. The error rates obtained from Mclust, on the other hand, change dramatically with the dimension. And for $l = 2$ and $l \geq 30$, both $\bar{r}_{\cdot,l}^{(M)}$ and $\tilde{r}_{\cdot,l}^{(M)}$ are significantly higher than $\bar{r}_{\cdot,l}^{(H)}$. Interestingly, the worst error rate from Mclust is achieved at $l = 2$ with $\bar{r}_{\cdot,2}^{(M)} > 23\%$ rather than at the high end of the range of dimensions. This seems to suggest that the extra noise dimensions help to mollify the effect of non-Gaussian shaped clusters. When $l \geq 30$, $\bar{r}_{\cdot,l}^{(M)}$ and $\tilde{r}_{\cdot,l}^{(M)}$ increase steadily with the dimension and eventually reaches nearly 20%.

Because it is expected that Gaussian components in the mixture model cannot well capture the half circle structure in the data, we have also tested Mclust with 3 components so that the half circle can possibly be characterized by two components. In this case, two of the three clusters need to be combined in order to compute the accuracy with respect to the original two classes. The selection of the two clusters to combine is not trivial. If we use a simple rule of combining the two clusters with the closest pair of centers, the average classification accuracies (over different dimensions) are mostly inferior to those by Mclust with 2 components. Note that the ridgeline based merging procedure in Section 4 may be invoked instead. A detailed examination in this direction is out of the scope of this paper. If we search through all the possible combinations and always choose the one that yields the best classification accuracy, the average accuracies are better than those achieved by HMAC. However, this comparison inherently favors Mclust because the true class labels are used to decide the binary grouping of the 3 components, while HMAC is purely unsupervised.

6.2 Real Data Sets

In this section, we apply HMAC to real data sets, compare our method of visualization described in Section 5 with PCA, and demonstrate the usage of the cluster merging algorithm described in Section 4.2.

6.2.1 GLASS DATA

We first examine the aforementioned glass data set using the entire set of 214 samples and all the original 9 features. Applying HMAC, a dendrogram of 10 levels is obtained, as shown in Figure 6(a). The number of clusters at each level is listed in Table 2. The sizes of the largest 4 clusters at each level are also given in this table. The dendrogram and the table show that 3 clusters containing more than 5 points are formed at the first level. These 3 prominent clusters are retained or augmented up to level 3. At level 4, two of the 3 prominent clusters are merged, leaving 2 prominent clusters which are further merged at level 7. At this level, both the dendrogram and the table suggest the main clustering structure in the data is annihilated by the very large kernel bandwidth. However, the number of clusters generated at level 7 is 7 instead of 1. Except the largest cluster, the other 5 clusters each contain no more than 3 points and in total only 9. They may be considered more appropriately as "outliers" than clusters. At even higher levels, these tiny clusters are merged gradually into the main mass of data.

Level	1	2	3	4	5	6	7	8	9	10
# clusters	29	25	18	15	13	11	7	6	4	3
Size of 1st cluster	160	163	176	177	179	180	205	208	210	211
Size of 2nd cluster	12	12	12	21	21	22	3	2	2	2
Size of 3rd cluster	9	9	9	3	3	3	2	1	1	1
Size of 4th cluster	5	6	2	2	2	2	1	1	1	not exist

Table 2: The clustering results for the full glass data set.

The above discussion suggests that to obtain a given number of clusters, it is not always a good practice to choose a level in the hierarchy that yields the desired number of clusters. We may select a level at which major clusters are merged and outliers are mistaken as plausible clusters. One remedy is to apply the cluster merging algorithm to a larger number of clusters formed at a lower level. We will present results of this approach shortly.

To compare our visualization method with PCA, the clustering results at level 3 are shown in Figure 6(b) and (c) using the two projection methods respectively. Both projections are orthonormal. In our visualization method, the projection only attempts to approximate the discriminant functions between the 3 major clusters. It is obvious that the new visualization method shows the clustering structure better than PCA. The two projection directions derived by our method are used when presenting other clustering results for a clear correspondence between points in different plots.

If we apply our cluster merging algorithm at level 3, we obtain results shown in Figure 6(d) and (e). In Figure 6(d), three clusters are formed by applying stage II merging based on coverage rate. The parameter $\rho = 95\%$. Merging based on separability is not conducted because we attempt to get 3 clusters and the 3 largest clusters at this level already account for close to 95% of the data. This clustering result is much more preferable than the 3 clusters directly generated by HMAC at level

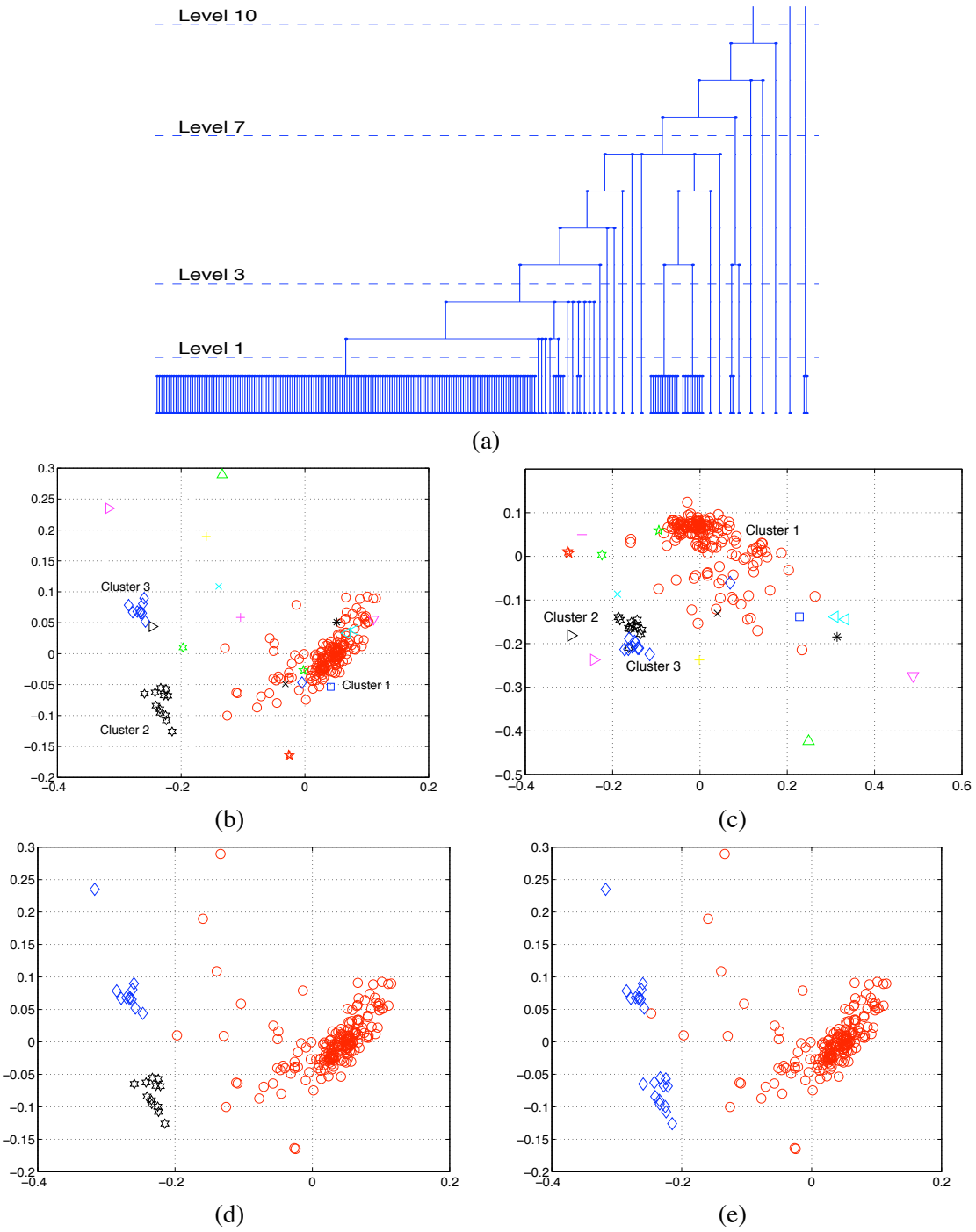


Figure 6: Results for the full glass data set. (a) The dendrogram created by HMAC. (b) Visualization by our method using regression on discriminant functions. (c) Visualization by PCA. (d), (e) Three (two) clusters obtained by applying the merging algorithm to clusters generated by HMAC at level 3.

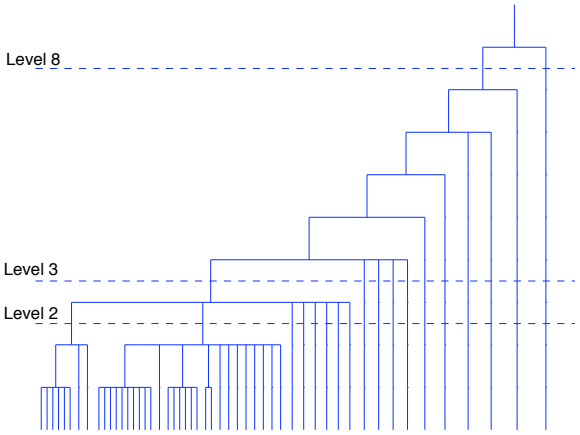


Figure 7: The dendrogram generated by HMAC for the infant data set.

10, containing 211, 2, and 1 points respectively. If we apply merging based on separability with threshold $\varepsilon = 0.4$ (stage I) and merging based on coverage rate with $\rho = 95\%$ (stage II), we obtain two clusters shown in Figure 6(e).

6.2.2 INFANT ATTENTION TIME

The second real data set is provided by Hoben Thomas, funded by the German Research Foundation Grant (Az. Lo 337/19-2), in the Department of Psychology at Penn State. In a study of infants' behavior, 51 infants were tested in two occasions apart by several months. In each occasion, a visual stimulus was given to the infants repeatedly for 11 times, with a fixed amount of time separating the stimuli. An infant's attention time in every stimulus was recorded. We thus have a data set of 51 samples with dimension 22. It is of interest to examine whether the infant data possess clustering structure and the behavior patterns of different groups. It is challenging to cluster this data set because of the high dimensionality and relatively small sample size.

Applying HMAC to the data, we obtain a dendrogram shown in Figure 7. At level 2, two prominent clusters emerge, containing 8 and 27 samples respectively. All the other clusters are singletons. At the next higher level, the two main clusters are merged. Because all the other clusters are singletons, we essentially partition the data into a main group and several outliers. There is no clear clustering structure preserved after level 2. Figure 8(a) and (b) show the clustering results obtained at level 2 using projection directions derived by our visualization method and PCA respectively. Specifically, in our method, the density of the largest cluster is used as the basis to form the discriminant functions of the other clusters. The projection directions are derived to best preserve the discriminant functions of the second and third largest clusters. The separation of the 2 main clusters is reflected better in (a) than (b). The ridgeline between the two major clusters at level 2 is computed, and the density function along this ridgeline is plotted in Figure 8(c). The plot shows that the "valley" between the two clusters is not deep comparing with the peak of the less prominent cluster, indicating weak separation between the clusters.

As the dendrogram suggests, if we need to cluster the infants into two groups, specifically, to assign the singletons into the two main clusters, we cannot simply cut the dendrogram at a level that

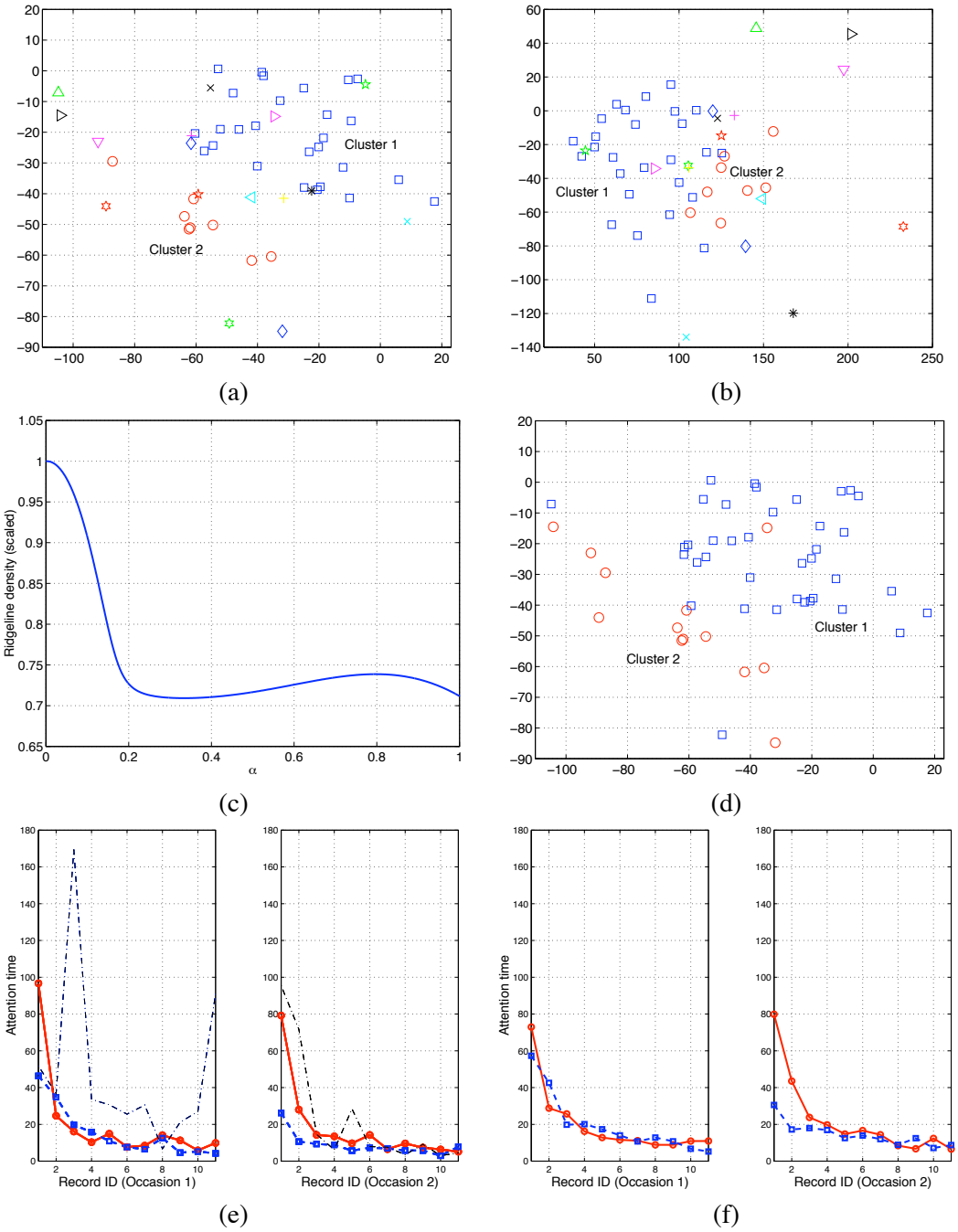


Figure 8: Results for the infant data set. (a) Visualization by our method using regression on discriminant functions. (b) Visualization by PCA. (c) Density function along the ridgeline between the two major clusters at level 2. (d) Two clusters obtained by applying the merging algorithm to clusters generated by HMAC. (e) The modal curves of the two main clusters obtained by HMAC at level 2. Dashed line: cluster 1. Solid line: cluster 2. Dash-dot line: the observation on the 10th infant, who deviates enormously from the cluster modes. (f) The mean curves of the two clusters obtained by fitting a mixture of 2 Gaussian components.

yields two clusters. For this purpose, we use the stage II merging procedure with a coverage rate of 85%. The clustering result is shown in Figure 8(d).

To compare the infants in the two main clusters identified by HMAC, we plot the "modal curves" for both test occasions in Figure 8(e). By modal curve, we refer to the mode of a cluster displayed with the dimension index as the horizontal axis and the value in that dimension as the vertical axis. It is meaningful to view a mode as a curve in this study because the experiments were conducted sequentially and dimension i corresponds to the ith measurement in the sequence. According to Figure 8(e), the main difference between the two groups of infants is their attention time for the first stimulus in each test occasion. The circle cluster exhibits a significantly longer attention time for the initial stimulus in both occasions. According to HMAC, 8 infants belong to the long initial attention time group and 27 infants belong to the short time group. The other 16 infants' data are distinct from both groups. In Figure 8(e), the attention time of the 10th infant is shown by the dash-dot lines. This infant is the most "extreme" outlier which corresponds to the right most branch in the dendrogram of Figure 8(c). The plot shows that this infant behaved very differently from the average in the first test. Instead of gradually decreasing, his attention time jumped to a very high value for the third stimulus and again for the last stimulus. If a clear-cut two groups are desired, Figure 8(d) shows that 13 infants are partitioned into the long time group and 38 into the short time group.

We also perform clustering on this data set by fitting a mixture of two Gaussian components. The EM algorithm is used to estimate the mixture model using k-means clustering to initialize, and the maximum a posterior criterion is used to partition the data. For the two clusters obtained, we display the two Gaussian mean vectors as curves in Figure 8(f). Just as in the clustering by HMAC, one cluster had longer attention time to the initial stimulus in both occasions. However, the difference between the attention time in the first occasion for the two clusters was not substantial. Using mixture modeling, the long time group included 26 infants while the short time group included 25.

6.2.3 NEWSGROUP DATA

In the previous two examples, we cannot quantitatively examine the clustering performance because there are no given class labels for the data points. In the third example, we use a data set containing documents labeled by different topics. This data set is taken from the newsgroup data (Lang, 1995). Each instance corresponds to a document in one of the two computer related topics: `comp.os.ms-windows.misc` (class 1) and `comp.windows.x` (class 2). The numbers of instances in the two classes are close: 975 (class 1) and 997 (class 2). The raw data for each document simply contain the words appeared in this document and their counts of occurrences. We process the data by stemming the words and employing two stages of dimension reduction. For details on the dimension reduction procedure, which relies mainly on two-way mixture modeling, we refer to (Li and Zha, 2006). In summary, we represent every document by a 10-dimensional vector, where each dimension is the total number of occurrences for a chosen collection of words. We then normalize the vector such that each dimension becomes the frequency of the corresponding set of words.

Due to the large data size, it is unrealistic to show the dendrogram generated by HMAC directly. Instead, we provide a summarized version of the dendrogram emphasizing prominent clusters in Figure 9(a). In order to retain as much information as possible in the summary, a cluster will be regarded "prominent" and shown individually if it contains at least 3% of the total data. This requirement for showing a cluster is rather mild. The number in each node box in the tree indicates

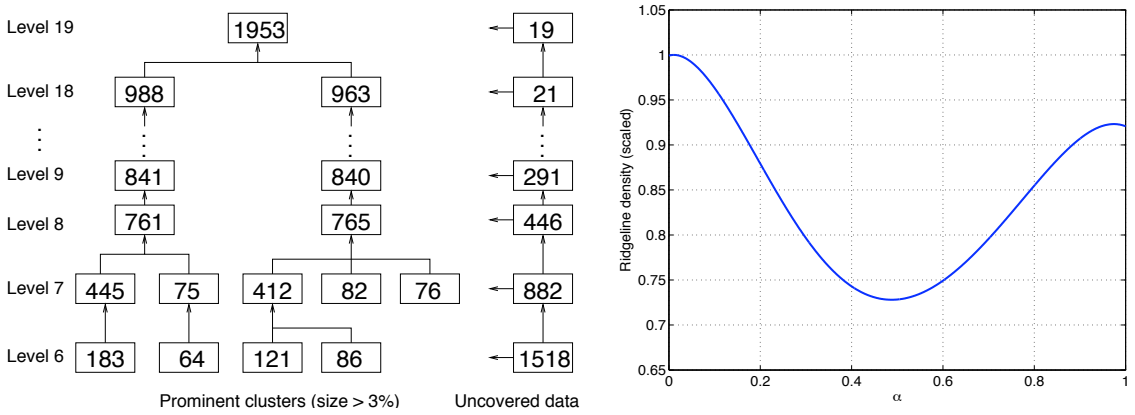


Figure 9: Results for the document data set. (a) The summarized dendrogram showing the prominent clusters. (b) The density function along the ridgeline between the two major clusters at level 12.

the cluster size. The number of data points not covered by the prominent clusters at any level is shown in the box to the right of the dendrogram. As shown in the figure, prominent clusters do not appear until level 6. At level 6, the four prominent clusters are very small. About 77% of the data are scattered in tiny clusters. At level 8, two major clusters emerge, each accounting for nearly 39% of the data. The rest data do not form any prominent clusters. The two major clusters remain through level 8 to level 18, and each absorbs more data points at every increased level. For brevity, we omit showing the sizes of these two clusters between level 9 and 18. At level 19, the two major clusters are merged, and there are 19 outlier points not absorbed into the main mass of data. This dendrogram strongly suggests there are two major clusters in this data set because before level 8, the clusters formed are too small and the percentage of data not covered by the clusters is too high. If we allow 5% points to lie outside prominent clusters, we can choose level 12 in the dendrogram. At level 12, the two clusters are of size 950 and 932. To examine the separation of the two clusters at this level, we calculate the ridgeline between them and plot the density function along the ridgeline in Figure 9(b). The mode heights of the two clusters are close, and the cluster separation is strong.

Level	8	9	10	11	12	13	14	15	16	17	18
Correct (%)	73.5	80.4	85.4	87.7	89.0	90.2	91.0	91.2	91.5	91.6	91.7
Incorrect (%)	3.9	4.9	5.7	6.1	6.4	6.5	6.8	6.8	6.9	7.1	7.2
Uncovered (%)	22.6	14.8	8.9	6.2	4.6	3.2	2.2	2.0	1.6	1.2	1.0

Table 3: The clustering accuracy of HMAC for the document data set. The percentages of points that are correctly clustered, incorrectly clustered, and not covered by the two major clusters are listed.

From level 8 to 18, we compute the percentages of points that are correctly clustered, incorrectly clustered, and not covered by the two major clusters. Table 3 provides the result. At level 18, 91.7% data points are correctly clustered. For comparison, we apply Mclust to the same data set. If we specify the range for the number of clusters as 2 to 10 and let Mclust choose the best number and the best covariance structure using BIC, the number of clusters chosen is 3. The sizes of the 3 clusters are 763, 668, and 541. The first two clusters are highly pure in the sense of containing points from the same class. The third cluster however contains about 40% class 1 points and 60% class 2. If we label the third cluster as class 2, the overall clustering accuracy is 87.6%. On the other hand, if we fix the number of clusters at 2 and run Mclust, the clustering accuracy becomes 94%.

6.2.4 DISCUSSION ON PARAMETER SELECTION

As shown by the above examples of real data clustering, it is often not obvious which level in the dendrogram produced by HMAC should be used to yield the final clustering. This is a general issue faced by agglomerative clustering approaches. Prior information about data or our implicit assumptions about good clusters can play an important role in this decision. One common assumption we make is that a valid cluster ought not be too small. Under this principle, we declare a level of the dendrogram too low (small bandwidth) if all the clusters are small and a level too high (large bandwidth) if a very large portion of the data (e.g., 95%) belong to a single cluster. For the intermediate acceptable levels in the dendrogram, we have designed the coverage rate mechanism to ensure that very small clusters are excluded. The chosen coverage rate reflects the amount of outliers one believes to exist. Despite the inevitable subjectiveness of this value, the coverage rate affects only the grouping of a small percentage of outlier points.

A more profound effect on the clustering result comes from the merging procedure based on separability which involves choosing a separability threshold. As we have discussed in Section 4.2, the merging process based on separability is essentially the directional single linkage clustering which stops when all the between-cluster separability measures are above the threshold. The threshold directly determines the final number of clusters, but is irrelevant to the full clustering hierarchy generated by the directional single linkage. Hence in situations where we have a targeted number of clusters to create, we can avoid choosing the threshold and simply stop the directional single linkage when the given number is reached. Of course, if at a certain level of the dendrogram, there exist precisely the targeted number of valid clusters, we can use that level directly rather than applying merging on clusters at a lower level. Whether the HMAC dendrogram clearly suggests the right number of clusters can be highly data dependent. For instance, in the document clustering example, the dendrogram in Figure 9(a) strongly suggests two clusters because at all the acceptable levels there are two reasonably large clusters. On the other hand, for the glass data set with results shown in Figure 6, it is not so clear-cut whether there are three or two clusters.

Another choice we need to make in HMAC is the sequence of kernel bandwidths, $\sigma_1 < \sigma_2 < \cdots < \sigma_\eta$. It is found empirically that as long as the grid of bandwidths is sufficiently fine, the prominent clusters created are not sensitive to the exact sequence. Major clustering structures often remain over a wide range of bandwidths. We have always used a uniformly spaced sequence of bandwidths in our experiments. If precisely two clusters merge at every increased level of a dendrogram, that is, the number of clusters decreases exactly by one, the dendrogram will have n levels, where n is the data size. We call such a dendrogram *all-size* since the clustering into any number of groups smaller than n appears at a certain level. We rarely observe an all-size dendrogram generated

by HMAC. On the other hand, by using extremely refined bandwidths, we indeed obtained all-size dendrograms for the example in Section 3.2 and the infant data set.

We note that linkage clustering by construction generates the all-size dendrogram. However, it is not necessarily a good practice to simply choose a level in the dendrogram that yields the desired number of clusters, as we have discussed previously for the dendrogram of HMAC. Hence, the fact that the dendrogram generated by HMAC is usually not all-size raises little concern. In practice, since the targeted number of clusters is normally much smaller than the data size, it is easy to find a relatively low level at which the number of clusters exceeds the target. We can then apply the separability based directional single linkage clustering at that level, and achieve any number of clusters smaller than the starting value.

6.3 Image Segmentation

To demonstrate the applicability of HMAC to computationally intensive tasks, we develop an image segmentation algorithm based on HMAC. A basic approach to image segmentation is to cluster the pixel color components and label pixels in the same cluster as one region (Li and Gray, 2000). This approach partitions images into regions that are relatively homogeneous. Examples of segmentation via clustering are shown in Figure 10. We employed the speeding up method described in Section 3.3. Our image segmentation method comprises the following steps: (a) Apply k-center algorithm to cluster image pixels into a given number of groups. This number is significantly larger than the desired number of regions. In particular, we set it to 100. (b) Form a data set $\{x_1,...,x_n\}$, $n = 100$, where x_i is the mean of the vectors assigned to the ith group by k-center clustering. For each x_i, assign weight w_i, where w_i is the percentage of pixels assigned to x_i. (c) Apply the weighted version of HMAC to the data set. The only difference lies in the formula for the kernel density estimator. In the weighted version, $f(x) = \sum_{i=1}^{n} w_i \phi(x \mid x_i, D(\sigma^2))$. (d) Starting from the first level of the dendrogram formed by HMAC, apply the cluster merging algorithm described in Section 4.2. If the number of clusters after merging is smaller than or equal to the given targeted number of segments, stop and output the clustering results at this level. Otherwise, repeat the merging process at the next higher level of the dendrogram. For brevity, we simply refer to this segmentation algorithm as HMAC.

To assess the computational efficiency of HMAC, we experiment with 100 digital photo images randomly selected from the Corel image database (Wang et al., 2001). Every image is of size 256×384 or 384×256. The experiments were conducted on a 1.2GHz Sun UltraSparc processor. We compare the segmentation time of HMAC and k-means. On average, it takes 4.41 seconds to segment an image using HMAC. The average number of segmented regions is 6.0. For k-means clustering, we experimented with both dynamically determining and fixing the number of segmented regions. In the dynamic case, the average number of regions generated per image by thresholding is 5.5. The average segmentation time for each image is 4.43 seconds, roughly equal to the time of HMAC. However, the computation time of k-means increases if more regions are formed for an image. If we fix the number of segmented regions to 6.0, the average segmentation time is 4.87 seconds per image.

In terms of segmentation results, whether HMAC or k-means is preferred is application dependent. K-means clusters the pixels and computes the centroid vector for each cluster according to the criterion of minimizing the mean squared distance between the original vectors and the centroid vectors. HMAC, however, finds the modal vectors, at which the kernel density estimator achieves

Figure 10: Segmentation results. First row: Original images. Second row: mode colors of the clusters generated by HMAC. Third row: mean colors of the clusters generated by k-means.

a local maxima. These vectors are peaks of density bumps. They are significant in the sense of possessing locally maximum density, but may not be the best approximation to the original vectors in an average sense. Figure 10 shows the representative colors extracted by HMAC and k-means for several impressionism paintings. For HMAC, the mode color vector of each cluster is shown as a color bar; and for K-means, the mean vector of each cluster is shown. The representative colors generated by k-means tend to be "muddier" due to averaging. Those by HMAC retain the true colors better, for instance, the white color of the stars in the first picture and the purplish pink of the vase in the second. On the other hand, HMAC may ignore certain colors that either are not distinct enough from others or do not contain enough pixels. For the purpose of finding the main palette of a painting, HMAC may be more preferable.

7. Conclusion

In this paper, we have introduced an EM-style algorithm, namely, Modal EM (MEM), for finding local maxima of mixture densities. For a given data set, we model the density of the data non-parametrically using kernel functions. Clustering is performed by associating each point to a mode identified by MEM with initialization at this point. A hierarchical clustering algorithm, HMAC, is developed by gradually increasing the bandwidth of the kernel functions and by recursively treating modes acquired at a smaller bandwidth as points to be clustered when a larger bandwidth is used.

The Ridgeline EM (REM) algorithm is developed to find the ridgeline between the density bumps of two clusters. A separability measure between two clusters is defined based on the ridgeline, which takes comprehensive consideration of the exact densities of the clusters. A cluster merging method based on pairwise separability is developed, which addresses the competing factors of using a small bandwidth to retain major clustering structures and using a large one to achieve a low number of clusters.

The HMAC clustering algorithm and its combination with the cluster merging algorithm are tested using both simulated and real data sets. Experiments show that our algorithm tends to unite merits of linkage clustering and mixture-model-based clustering. Applications to both simulated and real data also show that the algorithm works robustly with high dimensional data or clusters deviating substantially from Gaussian distributions. Both of these cases pose difficulty for parametric mixture modeling.

A C package at `http://www.stat.psu.edu/~jiali/hmac` is developed, which includes the implementation of the HMAC algorithm, REM for computing ridgelines, and the separability/ coverage rate based merging algorithm.

There are several directions that can be pursued in the future to strengthen the framework of modal clustering. In the current work, we use a fixed bandwidth for the kernel density functions. We can explore ways to make the bandwidth vary with the location of the data because there may not exist a single bandwidth suitable for the entire data set. Moreover, in this paper, we have discussed an approach to best visualize clusters in lower dimensions. A related and interesting question is to find a lower dimensional subspace in which the data form well separated modal clusters. We expect that the optimization of the subspace needs to be conducted as an integrated part of the modal clustering procedure.

Acknowledgments

We would like to thank the reviewers and the associate editor for insightful comments and constructive suggestions. We also thank Hongyuan Zha at Georgia Institute of Technology for pointing out some useful references, and Hoben Thomas at The Pennsylvania State University for providing us a real data set. Jia Li and Bruce Lindsay's research is supported by the National Science Foundation.

Appendix A.

We prove the ascending property of the MEM algorithm. Let the mixture density be $f(x) = \sum_{k=1}^{K} \pi_k f_k(x)$. Denote the value of x at the rth iteration of MEM by $x^{(r)}$. We need to show $f(x^{(r+1)}) \geq f(x^{(r)})$, or equivalently, $\log f(x^{(r+1)} \geq \log f(x^{(r)})$.

Let us introduce the latent discrete random variable $J \in \{1,2,...,K\}$ with prior probabilities $P(J=k) = \pi_k$. Assume that the conditional density of X given $J=k$ is $f_k(x)$. Then the marginal

distribution of X is $f(x)$, as specified above. Define the following functions:

$$\begin{aligned} L(x) &= \log f(x), \\ Q(x' \mid x) &= \sum_{k=1}^{K} p_k(x) \log \pi_k f_k(x'), \\ H(x' \mid x) &= Q(x' \mid x) - L(x') = \sum_{k=1}^{K} p_k(x) \log p_k(x') \end{aligned}$$

where $p_k(x) = P(J = k \mid X = x) = \frac{\pi_k f_k(x)}{f(x)}$ is the posterior probability of J being k given x. Denote the posterior probability mass function (pmf) given x by $\mathbf{p}(x) = (p_1(x), p_2(x), ..., p_K(x))$.

Because $H(x \mid x) - H(x' \mid x) = D(\mathbf{p}(x) \parallel \mathbf{p}(x'))$ and relative entropy $D(\cdot \parallel \cdot)$ is always nonnegative (Cover and Thomas, 1991), $H(x' \mid x) \leq H(x \mid x)$ for any pair of x and x'. On the other hand, according to the MEM algorithm,

$$\begin{aligned} x^{(r+1)} &= \operatorname*{argmax}_x \sum_{k=1}^{K} p_k(x^{(r)}) \log f_k(x) \\ &= \operatorname*{argmax}_x \left(\sum_{k=1}^{K} p_k(x^{(r)}) \log \pi_k + \sum_{k=1}^{K} p_k(x^{(r)}) \log f_k(x) \right) \\ &= \operatorname*{argmax}_x Q(x \mid x^{(r)}). \end{aligned}$$

Hence, $Q(x^{(r+1)} \mid x^{(r)}) \geq Q(x^{(r)} \mid x^{(r)})$. Finally, we prove the ascending property:

$$L(x^{(r+1)}) = Q(x^{(r+1)} \mid x^{(r)}) - H(x^{(r+1)} \mid x^{(r)}) \geq Q(x^{(r)} \mid x^{(r)}) - H(x^{(r)} \mid x^{(r)}) = L(x^{(r)}) \, .$$

Appendix B.

Recall that the Ridgeline EM algorithm aims at maximizing $\log g(x \mid \alpha) = (1-\alpha) \log g_1(x) + \alpha \log g_2(x)$, where $0 \leq \alpha \leq 1$ and $g_1(x)$ and $g_2(x)$ are two mixture densities $g_i(x) = \sum_{\kappa=1}^{T} \pi_{i,\kappa} h_{i,\kappa}(x)$, $i = 1, 2$. We prove here the ascending property of this algorithm, as described in Section 4.1.

Following the definitions in Appendix A, we form functions $L_i(x)$, $Q_i(x' \mid x)$, and $H_i(x'|x)$ for densities $g_i(x)$, $i = 1, 2$, respectively. Specifically, $L_i(x) = \log g_i(x)$, $Q_i(x' \mid x) = \sum_{\kappa=1}^{T} p_{i,\kappa}(x) \log \pi_{i,\kappa} h_{i,\kappa}(x')$, and $H_i(x' \mid x) = \sum_{\kappa=1}^{T} p_{i,\kappa}(x) \log p_{i,\kappa}(x')$, where $p_{i,\kappa}(x) = \pi_{i,\kappa} h_{i,\kappa}(x) / g_i(x)$. Note that, based on the proof in Appendix A, we have $L_i(x') = Q_i(x' \mid x) - H_i(x' \mid x)$ and $H_i(x \mid x) \geq H_i(x' \mid x)$. We now define

$$\begin{aligned} \tilde{L}(x) &= (1-\alpha) L_1(x) + \alpha L_2(x) \, , \\ \tilde{Q}(x' \mid x) &= (1-\alpha) Q_1(x' \mid x) + \alpha Q_2(x' \mid x) \, , \\ \tilde{H}(x' \mid x) &= (1-\alpha) H_1(x' \mid x) + \alpha H_2(x' \mid x) \, . \end{aligned}$$

According to the Ridgeline EM algorithm, $x^{(r+1)} = \operatorname{argmax}_{x'} \tilde{Q}(x' \mid x^{(r)})$. Hence $\tilde{Q}(x^{(r+1)} \mid x^{(r)}) \geq \tilde{Q}(x^{(r)} \mid x^{(r)})$. Also, it is obvious that $\tilde{H}(x^{(r)} \mid x^{(r)}) \geq \tilde{H}(x^{(r+1)} \mid x^{(r)})$. Finally, we prove the ascending property:

$$\tilde{L}(x^{(r+1)}) = \tilde{Q}(x^{(r+1)} \mid x^{(r)}) - \tilde{H}(x^{(r+1)} \mid x^{(r)}) \geq \tilde{Q}(x^{(r)} \mid x^{(r)}) - \tilde{H}(x^{(r)} \mid x^{(r)}) = \tilde{L}(x^{(r)}) \, .$$

Appendix C.

We prove here the graph constructed in Section 4.2, where every clique corresponds to a node, has no loop. We denote a node (i.e., a clique) by c_i. By construction, a directed edge from c_i to c_j exists if the following conditions are satisfied:

1. $S_c(c_i, c_j) < \varepsilon$.
2. $S_c(c_i, c_j) = \min_{k \neq i} S_c(c_i, c_k)$ and j is the smallest index among all those j''s that achieve $S_c(c_i, c_{j'}) = \min_{k \neq i} S_c(c_i, c_k)$.
3. $\delta(c_i) < \delta(c_j)$.

We refer to the three conditions as Condition 1, 2, 3.

We prove the non-existence of loops by contradiction. We will first show that if there is a loop in the graph, this loop is directed. Then, we will prove that a directed loop cannot exist. Without loss of generality, assume that there is a loop connecting nodes $\{c_1, c_2, ..., c_k\}$ sequentially. The edges in the loop are $\{e_{1,2}, e_{2,3}, ..., e_{k-1,k}, e_{k,1}\}$, where $e_{i,j}$ connects c_i and c_j. Let $head(e_{i,i+1})$ indicate the node from which edge $e_{i,i+1}$ starts. Obviously, $head(e_{i,i+1}) = c_i$ or c_{i+1}.

Without loss of generality, let $head(e_{k,1}) = c_k$. By Condition 2, every node can have almost one edge starting from it. Hence $head(e_{k-1,k}) = c_{k-1}$. For an arbitrary $j > 1$, assume that for $i = j, j+1, ..., k-1$, we have $head(e_{i,i+1}) = c_i$. Since $head(e_{j,j+1}) = c_j$, again by Condition 2, $head(e_{j-1,j}) = c_{j-1}$. Thus, for $i = j-1, j, ..., k-1$, we have $head(e_{i,i+1}) = c_i$. By induction, for any $i = 1, ..., k-1$, we have $head(e_{i,i+1}) = c_i$. Therefore, the loop connecting $\{c_1, c_2, ..., c_k\}$ is directed.

By Condition 3 of the graph construction procedure, if $head(e_{i,j}) = c_i$, then $\delta_i < \delta_j$. Thus, if there is a directed loop connecting nodes $\{c_1, c_2, ..., c_k\}$ and $head(e_{i,i+1}) = c_i$, we get the contradiction: $\delta_1 < \delta_2 < \cdots < \delta_k < \delta_1$. This proves that there is no loop (regardless of directed or not) in the graph.

References

A. Banerjee, I. S. Dhillon, J. Ghosh, and S. Sra. Clustering on the unit hypersphere using von Mises-Fisher distributions. *Journal of Machine Learning Research*, 6:1345-1382, 2005.

J. D. Banfield and A. E. Raftery. Model-based Gaussian and non-Gaussian clustering. *Biometrics*, 49:803-821, 1993.

C. Blake, E. Keogh, and C. J. Merz. *UCI repository of machine learning databases*. *http://www.ics.uci.edu/mlearn/MLRepository.html*, Dept. of Information and Computer Science, UCI, 1998.

G. Celeux and G. Govaert. Comparison of the mixture and the classification maximum likelihood in cluster analysis. *Journal of Statistical Computation & Simulation*, 47:127-146, 1993.

S. V. Chakravarthy and J. Ghosh. Scale-based clustering using the radial basis function network. *IEEE Trans. Neural Networks*, 7(5):1250-1261, 1996.

M.-Y. Cheng, P. Hall, and J. A. Hartigan. Estimating gradient trees. *A Festschrift for Herman Rubin, IMS Lecture Notes Monogr. Ser.*, 45:237-49, Inst. Math. Statist., Beachwood, OH, 2004.

H. Chipman and R. Tibshirani. Hybrid hierarchical clustering with applications to microarray data. *Biostatistics*, 7(2):286-301, 2006.

T. M. Cover and J. A. Thomas. *Elements of Information Theory*. John Wiley & Sons, Inc., 1991.

A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal Royal Statistics Society*, 39(1):1-21, 1977.

B. S. Everitt, S. Landau, and M. Leese. *Cluster Analysis*. Oxford University Press US, 2001.

C. Fraley and A. E. Raftery. Model-based clustering, discriminant analysis, and density estimation. *Journal of the American Statistical Association*, 97:611-631, 2002.

C. Fraley and A. E. Raftery. MCLUST Version 3 for R: Normal mixture modeling and model-based clustering. *Technical Report*, no. 504, Department of Statistics, University of Washington, 2006.

T. F. Gonzalez. Clustering to minimize the maximum intercluster distance. *Theoret. Comp. Sci.*, 38(22):293-306, 1985.

J. C. Gower and G. J. S. Ross. Minimum spanning trees and single linkage cluster analysis. *Applied Statistics*, 18(1):54-64, 1969.

T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer-Verlag, 2001.

A. K. Jain and R. C. Dubes. *Algorithms for Clustering Data*. Prentice-Hall, Inc., NJ, USA, 1988.

A. K. Jain , M. N. Murty, and P. J. Flynn. Data clustering: A review. *ACM Computing Surveys*, 31(3):264-323, 1999.

D. Joshi, J. Li, and J. Z. Wang. A computationally efficient approach to the estimation of two- and three-dimensional hidden Markov models. *IEEE Transactions on Image Processing*, 15(7):1871-1886, 2006.

J. R. Kettenring. The practice of cluster analysis. *Journal of Classification*, 23(1):3-30, 2006.

K. Lang. NewsWeeder: Learning to filter netnews. In *Proceeding of the International Conference on Machine Learning*, pages 331-339, 1995.

Y. Leung, J.-S. Zhang, and Z.-B. Xu. Clustering by scale-space filtering. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 22(12):1396-1410, 2000.

J. Li. Two-scale image retrieval with significant meta-information feedback. In *Proc. ACM Multimedia*, pages 499-502, Singapore, November 2005.

J. Li. Clustering based on a multi-layer mixture model. *Journal of Computational and Graphical Statistics*, 14(3):547-568, 2005.

J. Li and R. M. Gray. *Image Segmentation and Compression Using Hidden Markov Models*. Springer, 2000.

J. Li and J. Z. Wang. Real-time computerized annotation of pictures. In *Proc. ACM Multimedia Conference*, pages 911-920, ACM, Santa Barbara, CA, October 2006.

J. Li and H. Zha. Two-way Poisson mixture models for simultaneous document classification and word clustering. *Computational Statistics and Data Analysis*, 50(1):163-180, 2006.

G. J. McLachlan and D. Peel. *Finite Mixture Models*. New York: Wiley, 2000.

M. C. Minnotte and D. W. Scott. The mode tree: A tool for visualization of nonparametric density features. *Journal of Computational and Graphical Statistics*, 2(1):51-68, 1993.

M. C. Minnotte, D. J. Marchette, and E. J. Wegman. The bumpy road to the mode forest. *Journal of Computational and Graphical Statistics*, 7(2):239-51, 1998.

N. R. Pal and S. K. Pal. A review on image segmentation techniques. *Pattern Recognition*, 26:1277-94, 1993.

A. Pothen, H. D. Simon, and K. Liou. Partitioning sparse matrices with eigenvectors of graphs. *SIAM Journal on Matrix Analysis and Applications*, 11(3):430-452, 1990.

S. Ray and B. G. Lindsay. The topography of multivariate normal mixtures. *Annals of Statistics*, 33(5):2042-2065, 2005.

S. J. Roberts. Parametric and nonparametric unsupervised clustering analysis. *Pattern Recognition*, 30(2):261-272, 1997.

J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 22(8):888-905, 2000.

J. Z. Wang, J. Li, and G. Wiederhold. SIMPLIcity: Semantics-sensitive integrated matching for picture libraries. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(9):947-963, 2001.

R. Wilson and M. Spann. A new approach to clustering. *Pattern Recognition*, 23(12):1413-25, 1990.

C. F. J. Wu. On the convergence properties of the EM algorithm. *The Annals of Statistics*, 11(1):95-103, 1983.

Journal of Machine Learning Research 8 (2007) 1725-1745 Submitted 3/07; Revised 5/07; Published 8/07

Polynomial Identification in the Limit of Substitutable 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
Laboratoire d'Informatique Fondamentale
Centre de Mathmatiques et Informatique
39 rue Joliot-Curie
Marseille Cedex 13
FRANCE

Editor: Rocco Servedio

Abstract

This paper formalises the idea of substitutability introduced by Zellig Harris in the 1950s and makes it the basis for a learning algorithm from positive data only for a subclass of context-free languages. We show that there is a polynomial characteristic set, and thus prove polynomial identification in the limit of this class. We discuss the relationship of this class of languages to other common classes discussed in grammatical inference. It transpires that it is not necessary to identify constituents in order to learn a context-free language—it is sufficient to identify the syntactic congruence, and the operations of the syntactic monoid can be converted into a context-free grammar. We also discuss modifications to the algorithm that produces a reduction system rather than a context-free grammar, that will be much more compact. We discuss the relationship to Angluin's notion of reversibility for regular languages. We also demonstrate that an implementation of this algorithm is capable of learning a classic example of structure dependent syntax in English: this constitutes a refutation of an argument that has been used in support of nativist theories of language.

Keywords: grammatical inference, context-free languages, positive data only, reduction system, natural languages

1. Introduction

Current techniques for grammatical inference have for a long time been focused to a great extent on learnable subclasses of regular languages. For many application domains though, there are structural dependencies in the data that are more naturally modelled by context-free grammars of various types. One of the oldest ideas for a grammatical inference algorithm, and one geared towards context-free inference, is Harris's use of substitutability (Chomsky, 1953; Harris, 1954). Though this has formed the intuitive motivation for a number of grammatical inference algorithms before, it has never been adequately formalized. In this paper we present an explicit mathematical formalization of this idea of substitutability and use it to define a subclass of context-free languages that we call the *substitutable* languages, that can be learned according to the polynomial identification

in the limit paradigm (de la Higuera, 1997). These languages are not comparable to the very simple languages, but seem better suited to be the basis for algorithms that can learn natural languages.

In this paper we use a polynomial variant of Gold's identification in the limit (IIL) paradigm, working from positive data only. We hope in the future to be able to extend this to a more practical PAC-learning result, but in the meantime work in this paradigm allows some foundational issues to be addressed. The contribution of the work presented in this paper lies in two main directions. First we demonstrate that it is not necessary to learn constituent structure in order to learn context-free grammars. Indeed, it is sufficient to be able to learn the syntactic congruence: the syntactic monoid can be converted into a context-free grammar in Chomsky normal form in a very natural way. Secondly, we define a simple, purely language theoretic criterion, which allows the syntactic congruence to be identified very easily. This criterion is sufficient to guarantee polynomial identification in the limit.

Additionally, we point out the relationship to NTS grammars, which in our opinion are an unarticulated assumption underlying many algorithms in GI and unsupervised learning of natural language (see for instance the work of Adriaans et al., 2000; van Zaanen, 2002; Solan et al., 2004).

The key to the Harris approach for learning a language L, is to look at pairs of strings u and v and to see whether they occur in the same contexts; that is to say, to look for pairs of strings of the form lur and lvr that are both in L. This can be taken as evidence that there is a non-terminal symbol that generates both strings. In the informal descriptions of this, there is an ambiguity between two ideas. The first is that they should appear in *all* the same contexts; and the second is that they should appear in *some* of the same contexts. We can write the first criterion as follows: (we define our notation more formally in the next section, but we hope the reader will bear with us for the moment)

$$\forall l, r \; lur \in L \text{ if and only if } lvr \in L. \tag{1}$$

The second, weaker, criterion is

$$\exists l, r \; lur \in L \text{ and } lvr \in L. \tag{2}$$

The problem is then that to draw conclusions about the structure of the language, one needs the former; but all one can hope for by observation of given data is the latter. In general, the class of context-free grammars will be unlearnable: certainly according to the Gold style approach we take in this paper since it is a superfinite class. Therefore to obtain learnability results we must define subclasses of the languages that sufficiently restrict the class so that learning can take place. The restriction we consider here is that whenever two strings have one context in common, then they have all contexts in common: Equation 2 implies Equation 1. We call these the *substitutable* languages.

Our main formal result is that this simple, but powerful constraint on languages—and note that it is expressed in purely language theoretic terms—sufficiently restricts the class of context-free languages to the extent that it can be learned using a simple polynomial algorithm. In this case, we can learn according to the IIL criterion, and the algorithm will be polynomial in the amount of data it needs (the characteristic set) and in computation.

More generally, this work shows how one can move from the syntactic congruence of a context-free language to a grammar for that language under certain assumptions. This can be done through a remarkably simple construction, and finally provides a solid theoretical grounding for the numerous empirical algorithms based on different heuristics that have relied on learning the syntactic congruence.

2. Definitions

We start by defining some standard notation.

An *alphabet* Σ is a finite nonempty set of symbols called *letters*. A *string* w over Σ is a finite sequence $w = a_1 a_2 \dots a_n$ of letters. Let $|w|$ denote the length of w. In the following, letters will be indicated by $a, b, c, \dots$, strings by $u, v, \dots, z$, and the empty string by λ. We shall write $|u|_a$ for the number of occurrences of the letter a in the string u. Let Σ^* be the set of all strings, the free monoid generated by Σ. By a language we mean any subset $L \subseteq \Sigma^*$. The set of all substrings of a language L is denoted $Sub(L) = \{u \in \Sigma^+ : \exists l, r \in \Sigma^* \text{ such that } lur \in L\}$ (notice that the empty word does not belong to $Sub(L)$). We shall assume an order $\prec$ or $\preceq$ on Σ which we shall extend to Σ^* in the normal way by saying that $u \prec v$ if $|u| < |v|$ or $|u| = |v|$ and u is lexicographically before v.

In general, and though it is not the case of the main class studied in this paper, the definition of a class of languages $\mathbb{L}$ relies on a class $\mathbb{R}$ of abstract machines, here called *representations*, together with a function $\mathcal{L}$ from representations to languages, that characterize all and only the languages of $\mathbb{L}$: (1) $\forall R \in \mathbb{R}, \mathcal{L}(R) \in \mathbb{L}$ and (2) $\forall L \in \mathbb{L}, \exists R \in \mathbb{R}$ such that $\mathcal{L}(R) = L$. Two representations R_1 and R_2 are *equivalent iff* $\mathcal{L}(R_1) = \mathcal{L}(R_2)$.

Definition 1 (Grammar) *A grammar is a quadruple $G = \langle \Sigma, V, P, S \rangle$ where Σ is a finite alphabet of terminal symbols, V is a (distinct) finite alphabet of variables or non-terminals, P is a finite set of production rules, and $S \in V$ is a start symbol.*

If $P \subseteq V \times (\Sigma \cup V)^+$ then the grammar is said to be context-free (CF), and we will write the productions as $T \to \alpha$.

Note that we do not allow empty right hand sides to the productions and thus these grammars will not generate the empty string.

We will write $uTv \Rightarrow u\alpha v$ when $T \to \alpha \in P$. $\overset{*}{\Rightarrow}$ is the reflexive and transitive closure of $\Rightarrow$.

We denote by $\mathcal{L}(G) = \{w \in \Sigma^* : S \overset{*}{\Rightarrow}_G w\}$ the language defined by the grammar. Since we do not allow rules with an empty right hand side this language cannot contain λ.

Definition 2 (Syntactic congruence) *We say that two words u and v are syntactically congruent* `w.r.t.` *a language L, written $u \equiv_L v$, if and only if $\forall l, r \in \Sigma^*$ $lur \in L$ iff $lvr \in L$.*

We can think of this syntactic congruence as the strong notion of substitutability. Note two things: first this is clearly an equivalence relation, and secondly, it is a congruence of the monoid Σ^*, that is,

$$u \equiv_L v \text{ implies } \forall l, r \; lur \equiv_L lvr.$$

The syntactic monoid of the language L is just the quotient of Σ^* by this relation. It is a standard result that this will be finite if and only if L is regular.

We shall write $[u]_L$ for the equivalence class of the string u under $\equiv_L$.

Example 1 *Consider the language $L = \{u \in \{a, b\}^* : |u|_a = |u|_b\}$. We can see that $u \equiv_L v$ iff $(|u|_a - |u|_b) = (|v|_a - |v|_b)$; the congruence classes will thus correspond to particular values of $(|u|_a - |u|_b)$ and the syntactic monoid will be isomorphic to $\mathbb{Z}$.*

Another way of looking at this relation is to define the set of contexts of a string:

Definition 3 (Set of contexts) *The set of contexts of a string u in a language L is written $C_L(u)$ and defined as $C_L(u) = \{(l,r) : lur \in L\}$.*

Using this definition we can say that $u \equiv_L v$ if and only if $C_L(u) = C_L(v)$.

We define the weaker idea of substitutability that we will use in the following way.

Definition 4 (Weak substitutability) *Given a language L, we say that two words u and v are weakly substitutable* **w.r.t.** *L, written $u \doteq_L v$, if there exist $l,r \in \Sigma^*$ such that $lur \in L$ and $lvr \in L$.*

Note that this is in general not a congruence or even a transitive relation. Normally we will have a finite sample S of the language L: clearly $u \doteq_S v$ implies $u \doteq_L v$.

We now can define the class of languages that we are concerned with:

Definition 5 (Substitutable language) *A language L is substitutable if and only if for every pair of strings u,v, $u \doteq_L v$ implies $u \equiv_L v$.*

In terms of contexts we can say that a language is substitutable, if whenever the sets of contexts of two strings have non-empty intersection, they are identical. The substitutable context-free languages are just those languages that are both substitutable and context-free. A number of examples can be found in Section 6.

Lemma 6 *There are languages which are substitutable but not context-free.*

Proof Let $\Sigma = \{a,b,c,d\}$ be the alphabet. The language $L = \{u \in \Sigma^* : |u|_a = |u|_b \wedge |u|_c = |u|_d\}$ is substitutable, as can easily be verified: the syntactic monoid here is isomorphic to $\mathbb{Z} \times \mathbb{Z}$. The intersection of L with the regular language $\{a^*c^*b^*d^*\}$ gives the language $\{a^n c^m b^n d^m : n,m > 0\}$ which is not context-free; therefore L is not context-free. ■

2.1 Learning

We now define our learning criterion. This is identification in the limit from positive text (Gold, 1967), with polynomial bounds on data and computation (de la Higuera, 1997), but not on errors of prediction (Pitt, 1989).

A learning algorithm A for a class of representations $\mathbb{R}$, is an algorithm that computes a function from a finite sequence of strings $s_1,\ldots,s_n$ to $\mathbb{R}$. We define a presentation of a language L to be an infinite sequence of elements of L such that every element of L occurs at least once. Given a presentation, we can consider the sequence of hypotheses that the algorithm produces, writing $R_n = A(s_1,\ldots s_n)$ for the nth such hypothesis.

The algorithm A is said to identify the class $\mathbb{R}$ in the limit if for every $R \in \mathbb{R}$, for every presentation of $\mathcal{L}(R)$, there is an N such that for all $n > N$, $R_n = R_N$ and $\mathcal{L}(R) = \mathcal{L}(R_N)$.

We further require that the algorithm needs only polynomially bounded amounts of data and computation. We use the slightly weaker notion defined by de la Higuera (1997); note that the size of a set of strings S is defined as $\sum_{w \in S} |w|$.

Definition 7 (Polynomial identification in the limit) *A representation class $\mathbb{R}$ is identifiable in the limit from positive data with polynomial time and data iff there exist two polynomials $p(),q()$ and an algorithm A such that*

1. *Given a positive sample S of size m A returns a representation $R \in \mathbb{R}$ in time $p(m)$*

2. *For each representation R of size n there exists a characteristic set CS of size less than $q(n)$ such that if $CS \subseteq S$, A returns a representation R' such that $\mathcal{L}(R) = \mathcal{L}(R')$.*

However, this definition initially designed for the learning of regular languages, is somehow unsuitable as a model for learning context-free grammars.

Example 2 *A context-free grammar G_n with the one letter alphabet $\{a\}$ is defined as follows: it contains a set of non-terminals $N_1, \ldots N_n$. The productions consist of $N_i \rightarrow N_{i-1}N_{i-1}$ for $i = 2, \ldots n$ and $N_1 \rightarrow aa$. The sentence symbol is N_n. It can easily be seen that $L(G_n)$ consists of the single string a^{2^n}, yet the size of the representation of G_n is linear in n.*

According to the de la Higuera definition, no non-trivial language class that contains the grammars $G_1, G_2 \ldots$ can be learnable, since any characteristic set for G_n must contain the string a^{2^n}, and will therefore have size exponential in the size of the representation. Yet it seems absurd to insist on a polynomial size characteristic set, when reading a single example requires exponential time.

There is not at present a consensus on the most appropriate modification of this criterion for the learning of context-free grammars. Clearly, relaxing the constraint for polynomial *size* of the characteristic set, to merely requiring polynomial *cardinality*, is unsatisfactory, since it would allow algorithms to use exponential amounts of computation, by specifying an exponentially long string in the characteristic set. Several ideas have been formulated to tackle this problem, such as to focus on shallow languages (Adriaans, 2002) or to require a characteristic sample that is polynomial in a parameter other than the size of the target (Wakatsuki and Tomita, 1993), but none of them represents a consensus.[1] Nonetheless, it is beyond the scope of this paper to attempt to resolve this difficulty, and we shall thus adopt this approach in this paper.

3. Algorithm

We now define an algorithm *SGL* (Substitution Graph Learner), that will learn a context-free grammar from a sample of positive strings of a language.

The primary data structure of our algorithm can be conceived of as a graph, where each node of the graph corresponds to a substring of a string in the sample, and where there is an edge between any two nodes corresponding to substrings u,v if and only if $u \doteq_S v$ where S is the set of positive examples.

In the next section we will define a characteristic set of examples for a context-free grammar, and show that whenever the corresponding context-free language is substitutable, and the sample contains the characteristic set, then *SGL* will produce a grammar that generates the same language as the target.

Definition 8 (Substitution graph) *Given a finite set of words S, we define the substitution graph $SG(S) = (V,E)$ as follow:*

$V = \{u \in \Sigma^+ : \exists l, r \in \Sigma^*, lur \in S\}$,

$E = \{(u,v) \in \Sigma^+ \times \Sigma^+ : u \doteq_S v\}$.

1. One can check that our main result holds in both of the frameworks cited above.

This graph will consist of a number of components, in the usual graph theoretic sense. If the language is substitutable, then every member of the same component will be syntactically congruent, and can thus be freely swapped with each other without altering language membership. In general, there may be more than one component corresponding to the same congruence class, since we are deriving the graph from a small finite sample.

First, note that since syntactic congruence is transitive, and we are interested in substitutable languages, we can compute the transitive closure of the graph, by adding any edges (u,w) when we have edges $(u,v),(v,w)$. We will write $\cong_S$ for the transitive closure of $\doteq_S$. If S is a subset of a substitutable language L then $u \cong_S v$ implies $u \equiv_L v$.

We can write $SG/\cong_S$ for the set of components of the substitution graph and $\lceil u \rceil_S$ for the component that contains the string u. We will normally omit the subscript where there is no risk of confusion. Recall that we write $[u]_L$ for the congruence class of u with respect to the syntactic congruence of L.

3.1 Constructing the Grammar

Algorithm 1: Algorithm to generate a grammar from a substitution graph.

Data: A substitution graph $SG = (V,E)$
Result: A context-free grammar $\hat{G}$
Let Σ be the set of all the letters used in the nodes of SG ;
Compute $\hat{V}$ the set of components of SG ;
Store map $V \to \hat{V}$ defined by $u \mapsto \lceil u \rceil$;
Let $\hat{S}$ be the unique element of V corresponding to the context (λ,λ). ;
$\hat{P} = \{\}$;
for $u \in V$ **do**
  if $|u| > 1$ **then**
    for *v,w such that* $u = vw$ **do**
      $\hat{P} \leftarrow \hat{P} \cup (\lceil u \rceil \to \lceil v \rceil \lceil w \rceil)$;
    end
  else
    $\hat{P} \leftarrow \hat{P} \cup (\lceil u \rceil \to u)$
  end
end
output $\hat{G} = \langle \Sigma, \hat{V}, \hat{S}, \hat{P} \rangle$;

Given the SG we now construct a grammar $\hat{G} = \langle \Sigma, \hat{V}, \hat{P}, \hat{S} \rangle$.

We define the set of non-terminals to be the set of components of the substitution graph, $\hat{V} = SG/\cong_S$. First note that there will be precisely one component of the substitution graph that will contain all the strings in the sample S. This is because they will all appear in the empty context (λ,λ). This component is defined to be $\hat{S}$.

We now define the set of productions for the grammar. These consist of two types. First for every letter in the alphabet, that occurs as a substring in the language, we have a production.

$$\lceil a \rceil \to a$$

Note that if we have two letters such that $a \doteq b$, then $\lceil a \rceil = \lceil b \rceil$ and the same non-terminal will have two productions rewriting it.

The second set of productions is defined for every substring of length greater than 1. For every node in the substitution graph u, if $|u| > 1$, for every pair of non-empty strings v, w such that $u = vw$, we add a production $\lceil u \rceil \to \lceil v \rceil \lceil w \rceil$. Again note that if the component has more than one node in it, then all of the productions will have the same left hand side.

This is the most important step in the algorithm. It is worth pausing here to consider this construction informally: we shall prove these results formally below. Here we are taking an operation in the syntactic monoid; and constructing a production directly from it. Given some substrings such that $[u] = [v][w]$, we can consider this as saying, that any element of the congruence class $[v]$ concatenated with any element of the congruence class $[w]$ will give an element of the congruence class $[u]$. Phrased like this, it is clear that this can be directly considered as a production rule: if we wish to generate an element of $[u]$ one way of doing it is to generate a string from $[v]$ and then a string from $[w]$.

We can define the set of productions formally as:

$$\hat{P} = \{\lceil u \rceil \to \lceil v \rceil \lceil w \rceil : u = vw, u \in V\ of\ SG, |v| > 0, |w| > 0\} \cup \{\lceil a \rceil \to a : a \in \Sigma\}.$$

Algorithm 1 defines the process of generating a grammar from a substitution graph. To be fully explicit about the global algorithm, we show it in Algorithm 2, rather than relying on the characteristic set.

Algorithm 2: SGL algorithm

```
Data: A sequence of strings s_1, s_2, ...
Result: A sequence of CFGs G_1, G_2, ...
G = Grammar generating the empty language ;
while true do
    read next string s_n;
    if s_n ∉ L(G) then
        set SG to be the substitution graph generated from {s_1, ... s_n};
        set G to be the grammar generated from SG;
    end
    output G;
end
```

3.2 Examples

To clarify the behaviour of the algorithm, we shall now give two examples of its execution. The first one is a step-by-step description on a (very) small sample. The second one allows a discussion about the induced grammar for a particular language.

Example 3 *Suppose the sample consists of the two strings $S = \{a, aa\}$. $Sub(S) = \{a, aa\}$. It is clear that $a \doteq_S aa$. Therefore there is only one component in the substitution graph, associated with the non-terminal $\hat{S}$. The grammar will thus have productions $\lceil aa \rceil \to \lceil a \rceil \lceil a \rceil$ which is $\hat{S} \to \hat{S}\hat{S}$ and $\lceil a \rceil \to a$ which is $\hat{S} \to a$. Thus the learned grammar will be $\langle \{a\}, \{\hat{S}\}, \{\hat{S} \to \hat{S}\hat{S}, \hat{S} \to a\}, \hat{S} \rangle$ which generates the language a^+.*

Example 4 *Consider the language $L = \{a^n cb^n : n \geq 0\}$ that can be represented, for instance, by the set of productions $\{S \to aSb, S \to c\}$. Suppose we have a large sample of strings from this language. The substitution graph will have components $C_i \subset \{a^m cb^{m+i} : m, m+i \geq 0\}$ for both positive and negative values of i, with $C_0 = \hat{S} \subset \{a^m cb^m : m \geq 0\}$ being the sentence symbol. We also have components made of a unique string: $A_i = \{a^i\}$ and $B_i = \{b^i\}$, for positive values of i. The grammar generated from this sample will then have rules of the form*

$$
\begin{aligned}
C_{j+k} &\to C_j B_k \text{ for all } k > 0,\ j \in \mathbb{Z}\,, \\
C_{j-k} &\to A_k C_j \text{ for all } k > 0,\ j \in \mathbb{Z}, \\
C_0 &\to c, \\
A_{i+j} &\to A_i A_j \text{ for all } i, j > 0, \\
A_1 &\to a, \\
B_{i+j} &\to B_i B_j \text{ for all } i, j > 0, \\
B_1 &\to b.
\end{aligned}
$$

This grammar defines the language L, but as can be seen the set of non-terminals can be substantially larger than that of the original grammar.

3.3 Polynomial Time

We now show that SGL runs in a time bounded by a polynomial in the total size of the sample. Suppose the sample is $S = \{w_1, \ldots, w_n\}$. We can define $N = \sum |w_i|$, and $L = \max |w_i|$. Clearly $L \leq N$, and $n \leq N$. The total number of substrings, and thus nodes in the graph, is less than N^2. The cost of computing, for a given pair of strings u,v, all of the substrings u', v' such that $u' \doteq_S v'$ can be done in time less than L^2, and thus assuming a constant time map from substrings to nodes in the graph, we can compute all the edges in the graph in time less than $L^2 n^2$. Computing the transitive closure of $\doteq$ or equivalently identifying the components of the substitution graph, can be done in time linear in the sum of the number of nodes and edges which are both polynomially bounded. When constructing the grammar, the number of rules defined by each component/non-terminal is clearly bounded by the number of different ways of splitting the strings in the component, and thus the total number of rules must be bounded by LN^2, and each rule can be constructed in constant time.

There are clearly much more efficient algorithms that could be used: hashing from contexts to components and using a union-find algorithm to identify the components, for example.

4. Proof

Our main result is as follows:

Theorem 9 *SGL polynomially identifies in the limit the class of substitutable (context-free) languages.*

We shall proceed in two steps. First, we shall show that for any sample, the language defined by the grammar inferred from the sample will be a subset of the target language; another way of saying this is that the learner will never make a false positive error. The second step is to show that there

is a characteristic set; so that whenever the sample contains this characteristic set, the hypothesis will be correct. This set will have a cardinality which is linear in the size of the representation; however the size of the characteristic set (the sum of the lengths of all the strings in the set) will not necessarily be polynomially bounded. Indeed this is inevitable as the problem class contains examples where *every string* is exponentially large.

4.1 Proof that Pypothesis is Not Too Large

First of all we shall show that, for any finite sample of a substitutable language, the grammar derived from a finite sample does not define a language that is too large.

We start by proving a basic lemma, which is intuitively clear from the definitions.

Lemma 10 *For any sample C, not necessarily including a characteristic set, if $w \in Sub(C)$ then $\lceil w \rceil \stackrel{*}{\Rightarrow}_{\hat{G}} w$.*

Proof The proof can be done by induction on the length of the string. If $|w| = 1$, then by the construction of the grammar there is a unary rule $\lceil w \rceil \rightarrow w$. If $|w| = k+1$, we can write $w = ua$, where $|u| = k, a \in \Sigma$. By the inductive hypothesis, $\lceil u \rceil \stackrel{*}{\Rightarrow}_{\hat{G}} u$. By the construction of the grammar, there will be a production $\lceil w \rceil \rightarrow \lceil u \rceil \lceil a \rceil$, which establishes the lemma. Therefore the generated grammar will always include the training sample. ■

The next lemma states that derivation with respect to $\hat{G}$ maintains syntactic congruence.

Lemma 11 *For any sample C, for all $x \in \Sigma^*$, for all $u \in Sub(C)$, $\lceil u \rceil \stackrel{*}{\Rightarrow}_{\hat{G}} x$ implies $u \equiv_L x$*

Proof As $u \equiv_L u$ is trivial, we can restrict ourselves to the case $u \neq x$.

By induction on the length of the derivations k. Base step: $k = 1$. This means the derivation must be a single production of the form $\lceil u \rceil \rightarrow x$. This will only be the case if $|x| = 1$ and x is in the same component as u; therefore $u \equiv_L x$.

Inductive step: suppose this is true for all derivations of length less than k. Suppose we have a derivation of length $k > 1$. By construction of the grammar, there exist $\lceil v \rceil$ and $\lceil w \rceil$ such that $\lceil u \rceil \Rightarrow \lceil v \rceil \lceil w \rceil \Rightarrow^{k-1} x$. There must be strings x_1, x_2 such that $x = x_1 x_2$ and $\lceil v \rceil \stackrel{*}{\Rightarrow}_{\hat{G}} x_1$ and $\lceil w \rceil \stackrel{*}{\Rightarrow}_{\hat{G}} x_2$ with derivations of length less than k. Therefore by the inductive hypothesis, $v \equiv_L x_1$ and $w \equiv_L x_2$. Since we have a production $\lceil u \rceil \rightarrow \lceil v \rceil \lceil w \rceil$ in $\hat{P}$, there must be strings v', w' such that $v'w'$ is a string in the same component as u, and $v' \equiv_L v$ and $w' \equiv_L w$ and $u \equiv_L v'w'$. Since $\equiv_L$ is a monoid congruence, we have $u \equiv_L v'w' \equiv_L vw' \equiv_L vw \equiv_L x_1 w \equiv_L x_1 x_2 = x$. ■

Theorem 12 *For any positive sample of a substitutable context-free language L, if $\hat{G}$ is the result of applying the algorithm to it then $\mathcal{L}(\hat{G}) \subseteq L$.*

Proof Let $C \subset L$ be the sample. If $u \in L(\hat{G})$ then there must be some $v \in C$, such that $\hat{S} = \lceil v \rceil \stackrel{*}{\Rightarrow}_{\hat{G}} u$. Therefore $u \equiv_L v$, which implies $u \in L$. ■

4.2 Proof that Hypothesis is Large Enough

To prove that the hypothesis is large enough, we first need to define a characteristic set, that is to say a subset of a target language L_* which will ensure that the algorithm will output a grammar G such that $\mathcal{L}(G) = L_*$.

4.2.1 CONSTRUCTION OF THE CHARACTERISTIC SAMPLE

Let $G_* = \langle V, \Sigma, P, S \rangle$ be a target grammar. We will assume without loss of generality, that G_* is reduced, that is to say for every non-terminal $N \in V$, there are strings $l, u, r \in \Sigma^*$ such that $S \stackrel{*}{\Rightarrow} lNr \stackrel{*}{\Rightarrow} lur$. We are going to define a set CS of words of L_*, such that the algorithm *SGL* will identify L_* from any superset of CS.

We define $w(\alpha) \in \Sigma^+$ to be the smallest word, according to $\prec$, generated by $\alpha \in (\Sigma \cup V)^+$. Thus in particular for any word $u \in \Sigma^+$, $w(u) = u$. For each non-terminal $N \in V$ define $c(N)$ to be the smallest pair of terminal strings (l, r) (extending $\prec$ from Σ^* to $\Sigma^* \times \Sigma^*$, in some way), such that $S \stackrel{*}{\Rightarrow} lNr$.

We can now define the characteristic set $CS(G_*) = \{lwr : (N \to \alpha) \in P, (l, r) = c(N), w = w(\alpha)\}$. The cardinality of this set is at most $|P|$ which is clearly polynomially bounded. In general the cardinality of the set will not polynomially bound the size of the sample, for reasons discussed above in Section 2.1. However, notice that if there exists a polynomial-sized *structurally complete sample* (that is to say a sample where there is at least a string that uses each production rule Dupont et al., 1994) then the size of that characteristic set is polynomial.

Example 5 *Let $G = \langle \{a, b, c\}, \{S\}, \{S \to aSb, S \to c\}, S \rangle$ be the target grammar. It generates the language $\{a^n c b^n : n \geq 0\}$. For the rule $S \to c$, the construction gives $C(S) = (\lambda, \lambda)$ and $w(c) = c$, so the string c belongs to the characteristic set. Concerning the rule $S \to aSb$, we have $C(S) = (\lambda, \lambda)$ and $w(aSb) = acb$. The characteristic set is then $CS(G) = \{c, acb\}$.*

4.2.2 CONVERGENCE

We now must show that for any substitutable context-free grammar G, if the sample C contains the characteristic set $CS(G)$, that is, $CS(G) \subseteq C \subseteq \mathcal{L}(G)$, and if $\hat{G}$ is the output *SGL* produces on the sample C, then $\mathcal{L}(\hat{G}) = \mathcal{L}(G)$.

Lemma 13 *For any sample C containing the characteristic set, if $(N \to \alpha) \in P$, where $\alpha = V_1 \ldots V_l$, then for any $0 < i < j \leq l$, $\lceil w(V_i) \ldots w(V_j) \rceil \stackrel{*}{\Rightarrow}_{\hat{G}} \lceil w(V_i) \rceil \ldots \lceil w(V_j) \rceil$.*

Proof By construction of the characteristic set, if $(N \to \alpha) \in P$ then $w(\alpha) = w(V_1) \ldots w(V_l)$ is going to appear as a substring in the characteristic sample. Each production in the derivation $\lceil w(V_i) \ldots w(V_j) \rceil \Rightarrow_{\hat{G}} \lceil w(V_i) \ldots w(V_{j-1}) \rceil \lceil w(V_j) \rceil \cdots \Rightarrow_{\hat{G}} \lceil w(V_i) \rceil \ldots \lceil w(V_j) \rceil$ will be in the set of productions by the construction of the grammar. ■

Lemma 14 *For all $N \in V$ if $N \stackrel{*}{\Rightarrow}_G u$ then $\lceil w(N) \rceil \stackrel{*}{\Rightarrow}_{\hat{G}} u$.*

Proof By induction on the length of the derivation. Suppose the derivation is of length 1. Then $N \Rightarrow_G u$ and we must have $N \to u \in P$. Therefore by the construction of the characteristic set,

$w(N) \in Sub(C)$, $u \in Sub(C)$, and $\lceil u \rceil = \lceil w(N) \rceil$. By Lemma 10, $\lceil w(N) \rceil = \lceil u \rceil \stackrel{*}{\Rightarrow}_{\hat{G}} u$. This establishes the base case. Suppose this is true for all derivations of length at most k. Suppose $N \Rightarrow_G^{k+1} u$. So we take the first production $N \Rightarrow_G \alpha \Rightarrow_G^k u$. Suppose $|\alpha| = l$. Then we write $\alpha = V_1 \dots V_l \stackrel{*}{\Rightarrow}_G u_1 \dots u_l = u$, where $V_i \in V \cup \Sigma$ and $V_i \stackrel{*}{\Rightarrow}_G u_i$. if $V_i \in V$, then by the induction hypothesis $\lceil w(V_i) \rceil \stackrel{*}{\Rightarrow}_{\hat{G}} u_i$. If $V_i \in \Sigma$, then we have $\lceil w(V_i) \rceil \stackrel{*}{\Rightarrow}_{\hat{G}} u_i$ thanks to Lemma 10. By the construction of the characteristic set, we have $\lceil w(N) \rceil = \lceil w(\alpha) \rceil$. By Lemma 13 we have $\lceil w(N) \rceil \stackrel{*}{\Rightarrow}_{\hat{G}} \lceil w(V_1) \rceil \dots \lceil w(V_l) \rceil \stackrel{*}{\Rightarrow}_{\hat{G}} u_1 \dots u_l = u$. ■

Theorem 15 *For any substitutable CFG G, when the sample contains the characteristic set,* $L \subseteq \mathcal{L}(\hat{G})$.

This follows immediately by applying the previous theorem to S: if $S \stackrel{*}{\Rightarrow}_G u$ then $\lceil w(S) \rceil \stackrel{*}{\Rightarrow}_{\hat{G}} u$. Combining this with Theorem 12 establishes that $\mathcal{L}(\hat{G}) = \mathcal{L}(G)$, and therefore Theorem 9 holds.

5. Reduction System

As described up to now, our focus has been on producing an algorithm for which it is easy to prove some correctness results. However, the algorithm is not practical, since the number of non-terminals will often become very large, since in the worst case the algorithm will have one non-terminal for each substring in the training data. Many of these substrings will of course be redundant, and can be removed without reducing the language. In this section we will consider some algorithms for doing this; the languages defined will be unchanged, but the algorithms will be more efficient.

There are a number of algorithms for reducing the number of non-terminals. Clearly one can recursively remove all non-terminals that only have one production by replacing the non-terminal on the left hand side of the production with the right hand side, wherever it occurs. Secondly, one can remove non-terminals, one by one, and test whether the grammar continues to accept all of the sample, and thus arrive at a minimal CFG. Although the cost of that last operation is considerable, it obviously is polynomial.

In this section we describe a variant algorithm that is efficient and practical for large data sets, but that produces a reduction system, rather than a grammar. The idea is that if two substrings appear in the same component of the substitution graph, then we can rewrite every occurrence of the bigger one into the smaller.

The key point here is to reduce the substitution graph, by removing strings that are potentially redundant. In particular if we have one component that contains the strings u and v, where $v \prec u$, and another that contains the strings lur and lvr, we can reduce the graph by removing the string lur. This is equivalent to reducing the reduction system associated with the graph. Indeed, if we know we can rewrite u into v, then there is no need to add a rule that rewrites lur into lvr.

5.1 Definitions

We will briefly describe semi-Thue systems or reduction systems (Book and Otto, 1993).

Definition 16 (Reduction system) *A reduction system T, over an alphabet Σ is a finite set of pairs of strings $T \subset \Sigma^* \times \Sigma^*$, where each pair (u,v) is normally written $u \vdash_T v$, is called a reduction rule and satisfies $v \prec u$.*[2]

By extension, we will denote $lur \vdash lvr$ when $u \vdash v \in T$. $\vdash^*$ is the reflexive and transitive closure of $\vdash$.

Definition 17 (Confluent and weakly confluent reduction system) *A reduction system T is*

- *confluent if and only if for all $w, w1, w2 \in \Sigma^*$ such that $w \vdash w1$ and $w \vdash w2$, there exists $e \in \Sigma^*$ such that $w1 \vdash e$ and $w2 \vdash e$.*
- *weakly confluent on a set S if and only if for all $w, w1, w2 \in S$ such that $w \vdash w1$ and $w \vdash w2$, there exists $e \in S$ such that $w1 \vdash^* e$ and $w2 \vdash^* e$.*

Note that as defined these reduction systems are *Noetherian* which means that there is no infinite sequence of reductions. This defines a congruence relation where u and v are congruent if and only if they can be reduced to the same element. Being confluent and Noetherian means that there is a simple algorithm to determine this congruence: each string belong to only one congruence class. If we have the strict requirement that the reductions must be length reducing ($|v| < |u|$), then the maximum number of reductions is the length of the string you start with. Since we have a looser definition ($v \prec u$), this number can be exponential.

Given a reduction system one can define a language as the union of finitely many congruence classes. Thus given a reduction system T and a set of irreducible strings A on an alphabet Σ, we can define a language $L(T,A) = \{v \in \Sigma^* : \exists a \in A \ \wedge \ v \vdash_T^* a\}$. These are the congruential languages. In some cases, this is a more natural way of defining the structure of a language than systems from the traditional Chomsky hierarchy.

For example consider the reduction system $T = \{(aca, c), (bcb, c)\}$, and the axiom c (i.e., we are looking at the congruence class of c). The language defined by $L(T, \{c\})$ is exactly the palindrome language over a, b with center marker c.

The next subsection deals with the modifications of the algorithm SGL allowed by the use of reduction systems instead of grammars.

5.2 Reduction of a Substitution Graph

Given a substitution graph $SG = \langle V, E \rangle$, we say that SG reduces to $SG' = \langle V', E' \rangle$ if and only if there exists $(u,v) \in E : v \prec u$, and (l,r), $|l| + |r| > 0$, such that $lur \in V$, $V' = (V \setminus \{lur\}) \cup \{lvr\}$, $E' = \{(x,y) \in V' \times V' : (x,y) \in E \vee ((lur, y) \in E \wedge x = lvr)\}$.

We say that a substitution graph SG is *irreducible* if there exists no other substitution graph SG' such that SG reduces to SG'.

Given this reduced graph, we define a reduction system directly from the graph.

In this case we will define the set of reductions T to be exactly the set of all pairs $u \vdash v$, where $v \prec u$ and u, v are nodes in the same component of the substitution graph. We can also limit v to be the unique least node (with respect to $\prec$) in each component.

Assuming that we have a set of examples generated from a substitutable CFG that contains the characteristic set, it is easy to prove the following lemmas.

2. This differs slightly from the standard definition which requires $|v| < |u|$.

Lemma 18 *If $N \in V$ and $N \stackrel{*}{\Rightarrow} u$ for $u \in \Sigma^*$, then $u \vdash^* w(N)$.*

Proof Suppose $N = \alpha_0 \Rightarrow \alpha_1 \Rightarrow \cdots \Rightarrow \alpha_n = u$ is a derivation of u. Map this to a sequence $(w(N), w(\alpha_1), \ldots, w(\alpha_n), u)$ of strings from Σ^*. Consider a single step $\alpha_i = lMr$ and $\alpha_{i+1} = l\beta r$ and there is a production $M \to \beta$ in P. We have $w(\alpha_i) = w(l)w(M)w(r)$ and $w(\alpha_{i+1}) = w(l)w(\beta)w(r)$, which implies $w(\alpha_{i+1}) \vdash_T^* w(\alpha_i)$ by construction. Therefore $u \vdash^* w(N)$. ■

Lemma 19 *If $u \vdash v$ then $u \in L$ iff $v \in L$.*

Proof $u \vdash v$ implies $\exists (x,y) \in T$ and $l, r \in \Sigma^*$ such that $u = lxr$ and $v = lyr$. $x \doteq_S y$ implies $x \doteq_L y$ implies $x \equiv_L y$ implies $lxr \in L$ iff $lyr \in L$. ■

The reduction system will be weakly confluent on L, and it is Noetherian, since the number of strings smaller (`w.r.t.` $\prec$) than a given string is clearly finite. Unfortunately in general we will not be able to compute an irreducible string for any given word u in a polynomial (in the size of u) number of reductions (see Lohrey, 2000, for a formal proof). Thus though the reduction system itself may be much smaller, in some cases the "parsing" algorithm, determining whether a word is in the language, may be exponential. Subject to this caveat, we can define a small reduction system that represents the same language, namely the set of all strings that reduces to the least string $w(S)$ (`w.r.t.` $\prec$) in the language, but without the large number of redundant rules that the simple grammar construction produces. Without positing further restrictions on the sample set, it is not possible to give precise bounds on the relative sizes of the reduction system and the grammar.

6. Substitutable Languages

This section contains some examples of substitutable CFLs, as well as some simple CFLs that are not substitutable, and a discussion on the relationship of this class of languages to other standard classes. This is without a doubt a restricted class of languages but contains some interesting examples. They are not closed under any standard operation except reversal.

Since we are learning under a Gold style paradigm, we cannot hope to learn all finite languages (Gold, 1967). Indeed, the more complex the languages we hope to learn, the smaller the set of finite languages we will be able to learn.

6.1 Examples

We will now give some examples of some simple languages that are substitutable as well as some simple languages that are not. In general it will not be possible to decide for any given context-free language if it is substitutable. However, it can be checked "by hand" on particular examples.

Given an alphabet Σ, the following languages are substitutable:

- The set Σ^* of all the strings on the alphabet is substitutable. Indeed, all substrings appear in all possible contexts.
- Any language consisting of only one string (namely a singleton language) is substitutable, since in that case $u \doteq v$ implies $u = v$.

- The languages $\{a^n : n > 0\}$, for all $a \in \Sigma$, are substitutable.
- A more complex language, $\{wcw^R : w \in (a,b)^*\}$ (the language of palindromes with center marker) is substitutable.

We now turn our attention to simple languages that are not substitutable.

First, the finite language $L = \{a, aa\}$ is *not* substitutable. Indeed, $a \doteq_L aa$, since they appear in the context (λ, λ), but they are not congruent since (a, λ) is a context of a but not aa. This (counter) example shows that the class is not superfinite and is thus part of the explanation of the positive result proven above. The algorithm presented here would return the hypothesis $\{a^n : n > 0\}$.

The well-known context-free language $\{a^n b^n : n > 0\}$ is also *not* substitutable. Indeed, $a \doteq aab$, because they share the context (a, bb), but they are clearly not syntactically congruent, since $(a, abbb)$ is a context of a but not of aab. However, the language $\{a^n c b^n : n > 0\}$ is substitutable. Here the addition of a center marker removes the problem.

6.2 Relation to Other Language Classes

The state of the art concerning polynomial identification of context-free languages from positive example only is quite limited. In addition, as all studied subclasses are defined with respect to a particular class of representations, the comparison with our purely syntactically defined class is a hard task.

The fact that substitutable context-free languages can be represented by reduction systems proves that they are properly included within the class of congruential languages (Book and Otto, 1993). The examples previously presented show that they are incomparable with the classes of finite languages and regular languages.

The most important subclass of context-free languages that is polynomially identifiable is that of very simple grammars (Yokomori, 2003). It consists of CFGs in Greibach normal form such that no terminal symbol is used in more than one production. Some very simple grammars are not substitutable: an example is the regular grammar with productions $S \to aNP, S \to bN, N \to n, P \to rP, P \to p$. This generates the language $anr^*p \cup bn = \{bn, anp, anrp, anrrp, \ldots\}$. We can see that $n \doteq nr$ but it is not the case that $n \equiv nr$, since bn is in the language but bnr is not.

On the other hand, some substitutable languages, as for instance $\{wcw^R : w \in (a,b)^*\}$, are not very simple.

Finally, we also note the relationship to NTS grammars (Sénizergues, 1985); which can be seen to be relevant in Section 8. NTS grammars have the property that if $N \stackrel{*}{\Rightarrow} v$ and $M \stackrel{*}{\Rightarrow} uvw$ then $M \stackrel{*}{\Rightarrow} uNw$. We conjecture that all substitutable context free languages are NTS languages.

7. Practical Experiment

We will now present a simple experiment that demonstrates the applicability of this algorithm to the learning of natural languages.

For some years, a particular set of examples has been used to provide support for nativist theories of first language acquisition (FLA). These examples, which hinge around auxiliary inversion in the formation of questions in English, have been considered to provide a strong argument in favour of the nativist claim: that FLA proceeds primarily through innately specified domain specific mechanisms or knowledge, rather than through the operation of general-purpose cognitive mechanisms.

A key point of empirical debate is the frequency of occurrence of the forms in question. If these are vanishingly rare, or non-existent in the primary linguistic data, and yet children acquire the construction in question, then the hypothesis that they have innate knowledge would be supported. But this rests on the assumption that examples of that specific construction are necessary for learning to proceed. In this paper we show that this assumption is false: that this particular construction can be learned without the learner being exposed to any examples of that particular type. Our demonstration is primarily mathematical/computational: we present a simple experiment that demonstrates the applicability of this approach to this particular problem neatly, but the data we use is not intended to be a realistic representation of the primary linguistic data, nor is the particular algorithm we use suitable for large scale grammar induction. We will present the dispute in traditional terms, though later we shall analyse some of the assumptions implicit in this description. In English, polar interrogatives (yes/no questions) are formed by fronting an auxiliary, and adding a dummy auxiliary "do" if the main verb is not an auxiliary. For example,

Example 1a The man is hungry.

Example 1b Is the man hungry?

When the subject NP has a relative clause that also contains an auxiliary, the auxiliary that is moved is not the auxiliary in the relative clause, but the one in the main (matrix) clause.

Example 2a The man who is eating is hungry.

Example 2b Is the man who is eating hungry?

An alternative rule would be to move the first occurring auxiliary, that is, the one in the relative clause, which would produce the form

Example 2c Is the man who eating is hungry?

In some sense, there is no reason that children should favour the correct rule, rather than the incorrect one, since they are both of similar complexity and so on. Yet children do in fact, when provided with the appropriate context, produce sentences of the form of Example 2b, and rarely if ever produce errors of the form Example 2c (Crain and Nakayama, 1987). The problem is how to account for this phenomenon.

Chomsky claimed first, that sentences of the type in Example 2b are vanishingly rare in the linguistic environment that children are exposed to, yet when tested they unfailingly produce the correct form rather than the incorrect Example 2c. This is put forward as strong evidence in favour of innately specified language specific knowledge: we shall refer to this view as linguistic nativism.

In a special volume of the Linguistic Review, Pullum and Scholz (2002) showed that in fact sentences of this type are not rare at all. Much discussion ensued on this *empirical* question and the consequences of this in the context of arguments for linguistic nativism. These debates revolved around both the methodology employed in the study, and also the consequences of such claims for nativist theories. It is fair to say that in spite of the strength of Pullum and Scholz's arguments, nativists remained completely unconvinced by the overall argument.

Reali and Christiansen (2004) present a possible solution to this problem. They claim that local statistics, effectively n-grams, can be sufficient to indicate to the learner which alternative should be preferred. However this argument has been carefully rebutted by Kam et al. (2005), who show that this argument relies purely on a phonological coincidence in English. This is unsurprising since it is implausible that a flat, finite-state model should be powerful enough to model a phenomenon that is clearly structure dependent in this way.

7.1 Implementation

We have implemented the algorithm described above. There are a number of algorithmic issues that were addressed. First, in order to find which pairs of strings are substitutable, the naive approach would be to compare strings pairwise which would be quadratic in the number of sentences. A more efficient approach maintains a hashtable mapping from contexts to congruence classes. Caching hashcodes, and using a union-find algorithm for merging classes allows an algorithm that is effectively linear in the number of sentences. However, since it is necessary to consider every substring in the sentence, there appears to be a quadratic dependence on sentence length that seems to be hard to remove. Nonetheless this compares favourably to the complexity of context-free parsing.

In order to handle large data sets with thousands of sentences, it was necessary to modify the algorithm in various ways which slightly altered its formal properties. Primarily these involved pruning non-terminals that appeared to be trivial: since in the worst case the number of non-terminals approaches the number of distinct substrings in the corpus this is an enormous improvement. However for the experiments reported here we used a version which performs exactly in line with the mathematical description above.

7.2 Data

For clarity of exposition, we have used extremely small artificial data-sets, consisting only of sentences of types that would indubitably occur in the linguistic experience of a child. We do not attempt in this paper to demonstrate that, with naturally occurring distributions of data, this algorithm will correctly learn this distinction.

Our first experiments were intended to determine whether the algorithm could determine the correct form of a polar question when the noun phrase had a relative clause, even when the algorithm was not exposed to any examples of that sort of sentence. We accordingly prepared a small data set shown in Table 1. Above the line is the training data that the algorithm was trained on. It was then tested on all of the sentences, including the ones below the line. By construction the algorithm would generate all sentences it has already seen, so it scores correctly on those. The learned grammar also correctly generated the correct form and did not generate the final form.

We can see how this happens quite easily since the simple nature of the algorithm allows a straightforward analysis. We can see that in the learned grammar "the man" will be congruent to "the man who is hungry", since there is a pair of sentences which differ only by this. Similarly, "hungry" will be congruent to "ordering dinner". Thus the sentence "is the man hungry ?" which is in the language, will be congruent to the correct sentence.

One of the derivations for this sentence would be: [is the man hungry ?] $\rightarrow$ [is the man hungry] [?] $\rightarrow$ [is the man] [hungry] [?] $\rightarrow$ [is] [the man] [hungry] [?] $\rightarrow$ [is] [the man] [who is hungry] [hungry] [?] $\rightarrow$ [is] [the man] [who is hungry] [ordering dinner] [?].

Our second data set is shown in Table 2, and is a fragment of the English auxiliary system. This has also been claimed to be evidence in favour of nativism. This was discussed in some detail by Pilato and Berwick (1985). Again the algorithm correctly learns: the learned language includes the correct form but not the incorrect form.

the man who is hungry died .
the man ordered dinner .
the man died .
the man is hungry .
is the man hungry ?
the man is ordering dinner .
is the man who is hungry ordering dinner ?
∗is the man who hungry is ordering dinner ?

Table 1: Auxiliary fronting data set. Examples above the line were presented to the algorithm during the training phase, and it was tested on examples below the line.

it rains.
it may rain.
it may have rained.
it may be raining.
it has rained.
it has been raining.
it is raining.
it may have been raining.
∗it may have been rained.
∗it may been have rain.
∗it may have been rain.

Table 2: English auxiliary data. Training data above the line, and testing data below.

7.3 Conclusion of the Experiment

Chomsky was among the first to point out the limitations of Harris's approach, and it is certainly true that the grammars produced from these toy examples overgenerate radically. On more realistic language samples SGL would eventually start to generate even the incorrect forms of polar questions.

Given the solution we propose it is worth looking again and examining why nativists have felt that auxiliary fronting was such an important issue. It appears that there are several different areas. First, the debate has always focused on how to construct the interrogative from the declarative form. The problem has been cast as finding which auxiliary should be "moved". Implicit in this is the assumption that the interrogative structure must be defined with reference to the declarative, one of the central assumptions of traditional transformational grammar. Now, of course, given our knowledge of many different formalisms which can correctly generate these forms without movement we can see that this assumption is false. There is of course a relation between these two sentences, a semantic one, but this does not imply that there need be any particular syntactic relation, and certainly not a "generative" relation.

Secondly, the view of learning algorithms is very narrow. It is considered that only sentences of that exact type could be relevant. We have demonstrated, if nothing else, that that view is false. The distinction can be learned from a set of data that does not include any example of the exact piece of data required: as long as the various parts can be learned separately, the combination will function in the natural way.

8. Discussion

The important step in the algorithm is the realisation that it is not necessary to learn constituent structure in order to learn context-free languages. The rule $\lceil uv \rceil \rightarrow \lceil u \rceil \lceil v \rceil$ appears at first sight vacuous, yet it is sufficient to define correct languages for an interesting class of grammars.

8.1 Related Work

This work is related to two other strands of work. First work that proves polynomial IIL of other subclasses of context-free grammars. Yokomori (2003) shows that the class of very simple languages can be polynomially identified in the limit. Unfortunately the time complexity is $N^{|\Sigma|+1}$ and the alphabet size is equal to the number of productions in a very simple grammar, so this algorithm is not practical for large scale problems. Secondly, we can relate it to the work of Adriaans et al. (2000), who uses a similar heuristic to identify languages. Finally, we can mention the similar work of Starkie (2004) who shows an identification in the limit result of a class of grammars called "left-aligned R grammars". This work defines a rather complicated family of grammars, and shows how constituents can be identified. We also note Laxminarayana and Nagaraja (2003) who show a learnable subclass of CFGs.

We can compare substitutability with reversibility (Angluin, 1982; Makinen, 2000). Recall that a regular language is reversible if whenever uw and vw are in the language then ux is in the language if and only if vx is in the language. Thus reversibility is the exact analogue of substitutability for regular languages. Note that reversibility is a weaker criterion than substitutability. Substitutability implies reversibility, but not vice versa, as can be seen from the finite language $\{ab, bb\}$ which is reversible but not substitutable. Nonetheless, following the work on reversible regular languages,

one can think of a definition of k-substitutable languages and may obtain positive learning results on that hierarchy of languages.

We can also compare the substitutability to μ-distinguishability for inference of regular languages (Ron et al., 1998). Ron uses a measure of similarity of residual languages, rather than of contexts as we use here. Considered in this way, our measure is very crude, and brittle—contexts are equal if they have non empty intersection. Nonetheless the techniques of Ron et al., suggest a way that this technique could be extended to a PAC-learning result, using a bound on a statistical property of the distribution. There are some technical problems to be overcome, since the number of syntactic congruence classes will be infinite for non regular languages, and thus the distinguishability will not in general be bounded from below. A more serious problem is that the worst case sample complexity, if the data is drawn randomly, is clearly exponential, since the chance of getting two strings that differ only in a single point is in general exponential in the derivational entropy of the grammar.

Algorithms for learning regular languages focus on identifying the states of a deterministic automaton. When trying to move to learning context-free languages, the obvious way is to try to identify configurations (i.e., pairs of states and strings of stack symbols) of a deterministic push down automaton. A problem here is that the structure of this set depends on the representation, the automaton. One way of viewing the work presented in this paper is to say that a better approach is to try to identify the elements of the syntactic monoid. This monoid represents in the barest form the combinatorial structure of the language. From a learnability point of view this is interesting because it is purely syntactic—it is not semantic as it does not depend on the representation of the language but only on the language itself. Since we are interested in algorithms that learn from unstructured data—strings from the language that are not annotated with structural information—this seems a more natural approach. Importantly, our algorithm does not rely on identifying constituents: that is to say on identifying which substrings have been generated by the non-terminals of the target *grammar*. This has up to now been considered the central problem in context-free grammatical inference, though it is in some sense an ill-posed problem since there may be many different grammars with different constituent structure that are nonetheless equivalent, in the sense that they define the same language.

One of the weaknesses in the work is the fact that we do not yet have a grammatical characterisation of substitutability, nor an algorithm for determining whether a grammar defines a substitutable language. It is clear from standard results in the field that this property will be undecidable in general, but it might be possible to decide it for NTS grammars (Sénizergues, 1985).

Looking at our approach more generally, it is based on identifying syntactically congruent substrings. Substitutable languages have a property that allows a trivial procedure for determining when two substrings are congruent, but it is easy to think of much more robust methods of determining this that rely on more complex properties of the context distributions. Thus in principle we can use any property of the sample from the context distribution: average length, substring counts, marginal distributions at certain offsets and so on.

To conclude, we have shown how a simple formalization of Harris's substitutability criterion can be used to polynomially learn an interesting subclass of context-free languages.

Acknowledgments

This work has benefited from the support of the EU funded PASCAL Network of Excellence on Pattern Analysis, Statistical Modelling and Computational Learning. We would like to thank Colin de la Higuera, Jean-Christophe Janodet, Géraud Sénizergues, Brad Starkie and the two anonymous reviewers for helpful comments and discussions.

References

P. Adriaans. Learning shallow context-free languages under simple distributions. In K. Vermeulen and A. Copestake, editors, *Algebras, Diagrams and Decisions in Language, Logic and Computation*, volume 127. CSLI Publications, 2002.

P. Adriaans, M. Trautwein, and M. Vervoort. Towards high speed grammar induction on large text corpora. In *SOFSEM 2000*, pages 173–186. Springer Verlag, 2000.

D. Angluin. Inference of reversible languages. *Communications of the ACM*, 29:741–765, 1982.

R. Book and F. Otto. *String Rewriting Systems*. Springer Verlag, 1993.

N. Chomsky. Systems of syntactic analysis. *Journal of Symbolic Logic*, 18(3):242–256, 1953.

S. Crain and M. Nakayama. Structure dependence in grammar formation. *Language*, 63(3):522–543, 1987.

C. de la Higuera. Characteristic sets for polynomial grammatical inference. *Machine Learning*, 27 (2):125–138, 1997.

P. Dupont, L. Miclet, and E. Vidal. What is the search space of the regular inference? In R. C. Carrasco and J. Oncina, editors, *Grammatical Inference and Applications, Second International Colloquium, ICGI-94*, pages 25–37, 1994.

E. M. Gold. Language identification in the limit. *Information and Control*, 10(5):447 – 474, 1967.

Z. Harris. Distributional structure. *Word*, 10(2-3):146–62, 1954.

X. N. C. Kam, I. Stoyneshka, L. Tornyova, J. D. Fodor, and W. G. Sakas. Non-robustness of syntax acquisition from n-grams: A cross-linguistic perspective. In *The 18th Annual CUNY Sentence Processing Conference*, April 2005.

J. A. Laxminarayana and G. Nagaraja. Inference of a subclass of context free grammars using positive samples. In *Proceedings of the Workshop on Learning Context-Free Grammars at ECML/PKDD 2003*, 2003.

M. Lohrey. Word problems and confluence problems for restricted semi-thue systems. In *RTA*, volume 1833 of *LNCS*, pages 172–186. Springer, 2000.

E. Makinen. On inferring zero-reversible languages. *Acta Cybernetica*, 14:479–484, 2000.

S. Pilato and R. Berwick. Reversible automata and induction of the english auxiliary system. In *Proceedings of the ACL*, pages 70–75, 1985.

L. Pitt. Inductive inference, DFA's, and computational complexity. In *AII '89: Proceedings of the International Workshop on Analogical and Inductive Inference*, pages 18–44. Springer-Verlag, 1989.

G. Pullum and B. Scholz. Empirical assessment of stimulus poverty arguments. *The Linguistic Review*, 19(1-2):9–50, 2002.

Florencia Reali and Morten H. Christiansen. Structure dependence in language acquisition: Uncovering the statistical richness of the stimulus. In *Proceedings of the 26th Annual Conference of the Cognitive Science Society*, Mahwah, NJ, 2004. Lawrence Erlbaum.

D. Ron, Y. Singer, and N. Tishby. On the learnability and usage of acyclic probabilistic finite automata. *J. Comput. Syst. Sci.*, 56(2):133–152, 1998.

G. Sénizergues. The equivalence and inclusion problems for NTS languages. *J. Comput. Syst. Sci.*, 31(3):303–331, 1985.

Z. Solan, D. Horn, E. Ruppin, and S. Edelman. Unsupervised context sensitive language acquisition from a large corpus. In *Proceedings of NIPS 2003*, 2004.

B. Starkie. *Identifying Languages in the Limit using Alignment Based Learning*. PhD thesis, University of Newcastle, Australia, 2004.

M. van Zaanen. Implementing alignment-based learning. In *Grammatical Inference: Algorithms and Applications, ICGI*, volume 2484 of *LNCS*, pages 312–314. Springer, 2002.

M. Wakatsuki and E. Tomita. A fast algorithm for checking the inclusion for very simple deterministic pushdown automata. *IEICE Trans. on Information and Systems*, E76-D(10):1224–1233, 1993.

T. Yokomori. Polynomial-time identification of very simple grammars from positive data. *Theoretical Computer Science*, 298(1):179–206, 2003.

Journal of Machine Learning Research 8 (2007) 1747-1768 Submitted 7/06; Revised 3/07; Published 8/07

Structure and Majority Classes in Decision Tree Learning

Ray J. Hickey RJ.HICKEY@ULSTER.AC.UK
School of Computing and Informating Engineering
University of Ulster
Coleraine
Co. Londonderry
N. Ireland, UK, BT52 1SA

Editor: Greg Ridgeway

Abstract

To provide good classification accuracy on unseen examples, a decision tree, learned by an algorithm such as ID3, must have sufficient structure and also identify the correct majority class in each of its leaves. If there are inadequacies in respect of either of these, the tree will have a percentage classification rate below that of the maximum possible for the domain, namely (100 - Bayes error rate). An error decomposition is introduced which enables the relative contributions of deficiencies in structure and in incorrect determination of majority class to be isolated and quantified. A sub-decomposition of majority class error permits separation of the sampling error at the leaves from the possible bias introduced by the attribute selection method of the induction algorithm. It is shown that sampling error can extend to 25% when there are more than two classes. Decompositions are obtained from experiments on several data sets. For ID3, the effect of selection bias is shown to vary from being statistically non-significant to being quite substantial, with the latter appearing to be associated with a simple underlying model.

Keywords: decision tree learning, error decomposition, majority classes, sampling error, attribute selection bias

1 Introduction

The ID3 algorithm (Quinlan, 1986) learns classification rules by inducing a decision tree from classified training examples expressed in an attribute-value description language. A rule is extracted from the tree by associating a path from the root to a leaf (the rule condition) with the majority class at the leaf (the rule conclusion). The majority class is simply that having the greatest frequency in the class distribution of training examples reaching the leaf. The set of such rules, one for each path, is the *induced classifier* and can be used to classify unseen examples.
Many different trees may adequately fit a training set. The bias of ID3 is that, through use of an information gain heuristic (expected entropy) to select attributes for tree expansion, it will tend to produce small, that is, shallower, trees (Mitchell, 1997).

For good generalization accuracy, the induced tree must have sufficient structure, that is, depth, to fully extract the conditions of each rule and, in addition, must identify the correct majority class in each leaf. Yet, as is well-known, a major weakness of decision tree induction lies in its progressive sub-division of the training set as the tree develops (divide and conquer).

This causes the two requirements work to against each other: deepening the tree to create the necessary structure reduces the sample sizes in the leaves upon which inferences about majority classes are based. In a real-world domain there may be hundreds of attributes and it would require a massive training set to build a full tree having an adequate number of examples reaching each leaf.

In the literature, building trees has received the most attention. There has been comparatively little investigation into whether the class designated as the majority using the leaf sample distribution will be the true majority class. Frank (2000) provided some analysis, for two classes, of the error in classification arising from a random sample. Weiss and Hirsh (2000) noted that small disjuncts (rules with low coverage extracted from the tree) contribute disproportionately to classification error and that this behaviour is related to noise level. A sister problem, that of estimating probability distributions in the leaves of the grown tree, has been discussed by Provost and Domingos (2003) but this has little direct bearing on the problem faced here.

The classification rate of an induced tree on unseen examples is limited by the Bayes rate, *BCR* = (100 - Bayes error rate), which is the probability (expressed as a percentage) that a correct classification would be obtained if the underlying rules in the domain were used as the classifier. This is 100% in a noise-free domain but decreases accordingly with increasing noise. It is an asymptote in the learning curve of accuracy against training set size.

If the classification rate of an induced tree is *CR*, then the shortfall in accuracy compared to the maximum that can be achieved is *BCR* - *CR*. Throughout the paper this shortfall will be called the (*total) inductive error* of the induced tree. Thus here *error* is relative to the best performance possible, which differs from the usual practice that defines classification error as complementary to classification rate, that is, 100 - *CR*. The intention is to assign blame for inductive error partially to inadequacies in tree structure and partially to inadequacies in majority class identification.

In this paper, a decomposition of inductive error for decision trees will be introduced. Initially this will separate inductive error into the two components mentioned above. The component for majority class determination will then be further broken down to allow the sampling behaviour at the leaves and the bias introduced by the induction algorithm's attribute selection competition to be isolated and quantified.

Such a decomposition is reminiscent of the bias-variance decomposition of induced classifier performance that has received considerable attention recently. In the latter, the intention is to account for expected mean square loss for a given loss function defined on the classification process. In part this deviation is due to the classifier being 'off target' (bias) and in part to its variability over learning trials (variance). A major feature of the work of the authors involved has been the pursuit of an appropriate definition of the loss function for the classification problem. James (2003) provides a general framework for bias-variance decomposition and compares the different approaches that have been proposed.

The decomposition of inductive *error* that will be discussed below differs from bias-variance decomposition in that there is no term representing variability. Instead, the focus will be solely on average performance as assessed by classification rate. The analysis will require complete knowledge of the probabilistic model of the domain although it is estimable from a sufficiently large data set.

Nevertheless, it may be that aspects of these two different types of decomposition are indirectly related in some way but this will not be investigated further here.

In Section 2, the notion of a classification model and its decision tree representation are discussed. The fundamental notion of a core tree is defined. In Section 3, the main decomposition for inductive error and for a sub-decomposition are introduced. In Section 4, an analysis of the probability of selecting the correct majority class from a random sample is presented and this is

applied to the decomposition. In Section 5, experiments are carried out on data from the well-known LED domain to show the behaviour of the error decompositions along the learning curve. In Section 6, an automatic classification model generator is described and is used to obtain several models. Error decompositions are then obtained from experiments on data generated from these models. The results show how the decomposition is influenced by the major factors in induction, that is, training set size, complexity of the underlying rules, noise level and numbers of irrelevant attributes. In Section 7, the decomposition is applied to a large real data set.

2 The Class Model and its Representation by a Decision Tree

A model for a set of attributes, consisting of description attributes and a class attribute, can be specified by the joint probability distribution of all the attributes. This will be called a domain here. From the domain may be derived the *class model*, that is, a set of rules specifying the mapping associating a description attribute vector with a probability distribution over classes (Hickey, 1996). The class model is analogous to a regression model in statistics with the class attribute as the dependent variable and the description attributes as the independent variables. Noise in the relationship is then explicated by the class distributions (analogous to the Normal error distribution is regression). As discussed in Hickey (1996), these distributions account for all physical sources of uncertainty in the relationship between example descriptions and class, namely attribute noise, class noise and inadequacy of attributes. The model may contain *pure noise* attributes.[1] These are irrelevant to the determination of class. Their presence, however, usually makes learning more difficult.

A class model may be represented in a number of different ways. If all attributes are finite discrete (to which case we limit ourselves here) then a fully extensional representation is a table relating fully instantiated description vectors to class distributions. At the other end of the scale, it may be possible to represent the model using a small number of very general rules. It is an obvious but important point that altering the representation does not alter the model. Finding a representation to satisfy some requirement, for example, that with the smallest number of rules, will usually require a search.

A decision tree[2] can be used to represent a class model (Hartmann et al., 1982; Hickey, 1992). Each leaf would contain the class probability distribution conditional on the path to the leaf. Such a distribution is the theoretical analogue of the class frequency distribution in a leaf of a tree induced from training examples. Using the tree as a classifier, where the assigned class is the majority class in the appropriate leaf, will achieve the Bayes classification rate.

Often, only the mapping of description attribute vector to majority class is of interest. This is typically the case in ID3 induction. Recently there has been work on estimating the full class model, that is, including the class distributions, by inducing probability estimation trees (Provost and Domingos, 2003).

To fully represent a class model a tree must have sufficient depth. The notion of the core of a tree is central to the development below.

Definition 1. With regard to the representation of a class model, a decision tree is said to be a *core* tree if un-expansion of any set of sibling leaves would result in a reduction in expected

[1]As a property of an attribute, the notion of *pure noise* as used by Breiman et al. (1984) and Hickey (1996) corresponds to that of *irrelevant* as defined by Kohavi and John (1997). Also, the latter's the notion of *weakly relevant* corresponds to *redundant* in Hickey (1996).

[2] The discussion here is limited to trees in which node expansion is based on the values of a single attribute.

information[3] about class. If, in addition, there is no expansion of the leaves of the core to any depth that would increase the expected information then the tree is said to be a *complete* core; else it is *incomplete*. The leaf nodes of a core are referred to collectively as its *edge*. □

A complete core, together with the appropriate class distributions in its leaves, adequately represents the class model (and any further expansion is superfluous) whereas an incomplete core under represents it.

Any given tree has a unique core and can be reduced to this by recursively un-expanding its leaves until sibling nodes having different distributions are first encountered. This is analogous to post-pruning of an induced tree but, of course, does not involve statistical inference because all distributions are known. Expansions thus removed may involve attributes, which, while being locally uninformative, are globally informative, and hence appear elsewhere in the tree. Pure noise attributes will also have been removed as they are always locally uninformative.

In addition, the core may also contain internal 'locked-in' pure noise nodes (Liu and White, 1994) and is said to be *inflated* by them. A core that does not contain internal splits on pure noise attributes is said to be *deflated*. A deflated complete core offers an economical tree representation of a class model: it has no wasteful expansion on pure noise attributes either internal to the core or beyond its edge.

2.1 Deterministic Classifiers and the Reduced Core

Replacing each class distribution in the leaf of a tree with a majority class for that distribution will produce a *deterministic classification tree*. To achieve the Bayes rate in classification, this tree must have sufficient structure. Sufficient structure will normally mean a complete core (whether inflated or not); the tree may extend beyond the core. The only exception to this occurs when, near the edge of the core, there is a final internal node, N, all of whose children (leaves of the core) possess the same true majority class. In this case, it is possible to have a sub-complete tree, with N as a leaf node, which achieves the Bayes rate. Cutting back the core in such a situation will be called *same majority class* pruning. A tree thus obtained will be referred to as a *reduced core*. This lossless pruning applies only to the building of a deterministic classification tree, not to a probability estimation tree. A reduced core deterministic classification tree which achieves the Bayes rate is called *complete*.

Since the concern here is with inducing trees for deterministic classification, it will be assumed, in what follows, that all core trees are reduced.

3 A Decomposition of Inductive Error

Insufficient tree structure and inaccurate majority class identification both contribute to inductive error in trees. It is possible to break down the overall inductive error into components that are attributable to these separate sources.

3.1 Tree Structure and Majority Class Errors

Let the classification rate of an induced tree, T, be $CR(T)$. The correct majority classes for any tree can be determined from the class model. Altering an induced tree to label each leaf with the true majority class, as distinct from the leaf sample estimate of this, produces the *corrected majority class* version of the tree, $T(maj)$. The classification rate of this tree is called the *corrected majority class classification* rate. For any tree it follows that

[3] This is the usual entropy-based definition applied to domain probabilities; however any strong information measure can be used. See Hickey (1996) for a general discussion on information measures.

$$CR(T) \leq CR(T(maj)) \leq BCR.$$

Recall that inductive error is *BCR* - *CR*. Correcting majority classes as indicated above removes majority class determination as a source of inductive error. Thus, the amount by which $CR(T(maj))$ falls short of *BCR* is solely a measure of inadequacy of the tree structure. This component of inductive error will be called *(tree) structure error* so that

$$structure\ error = BCR - CR(T(maj)) .$$

The amount by which *CR* falls short of $CR(T(maj))$ is then attributable to incorrect determination of majority classes in the fully-grown tree. This is called *majority class error* so that

$$majority\ class\ error = CR(T(maj)) - CR .$$

This gives the initial decomposition:

$$BCR - CR = (BCR - CR(T(maj))) + (CR(T(maj)) - CR) . \quad (1)$$

That is:

$$inductive\ error = structure\ error + majority\ class\ error .$$

Let T_{core} be the reduced core of *T*. This core can also be majority class corrected. From the definition of a core, it is easy to see that the corrected core and the corrected full tree must have the same classification rate, that is:

$$CR(T(maj)) = CR\ (T_{core}(maj)) .$$

Structure error can then be re-expressed as:

$$structure\ error = BCR - CR(T_{core}(maj)) .$$

Since the core is the essential structural element of the tree, this reinforces the notion of structure error. The completeness of a core can be expressed in terms of structure error: the reduced core of a tree is complete if and only if structure error is zero.

3.2 A Sub-decomposition of Majority Class Error

As noted by Frank (2000), the majority class as determined from the leaf of an induced tree may be the wrong one because it is based on a small sample and also because that sample is obtained as a result of competitions taking place, as the tree is grown, to select which attribute to use to expand the tree. The latter is an example of a multiple comparison problem (MCP) as discussed by Jensen and Cohen (2000). In theory, though, the effect of this could be to improve majority class estimation: the intelligence in the selection procedure might increase the chance that the majority class in the leaf is the correct one.

It is possible to decompose majority class error, as defined above, into two terms that reflect the contribution of each of these factors, namely sampling and (attribute) selection bias.

Ideally, the sample arriving at a leaf should be a random sample from the probability distribution at the leaf as derived from the class model. In the induced tree, the sample in each leaf can be replaced by a new random sample of the same size generated from this distribution. This new tree will be called the *corrected sample* tree, $T(ran)$.

The classification rate of this tree, $CR(T(ran))$, depends on the particular random samples obtained at each of its leaves. Let $E(CR(T(ran)))$ be the expectation of $CR(T(ran))$ over all possible random samples of the appropriate size at each leaf of *T* and then over all leaves. If there is a difference between $E(CR(T(ran)))$ and $CR(T)$ then this indicates that the samples reaching the leaves of *T* are not random. The *selection bias error* can thus be defined as

$$selection\ bias\ error = E(\ CR(T(ran))\ ) - CR(T)$$

and can be positive, negative or zero.

The complementary component of majority class error is then

$$CR(T(maj)) - E(\ CR(T(ran))\ ) .$$

This term measures the shortcoming of the random sample in determining the correct majority class and can thus be called *sampling error*. It must be non-negative since failure to determine one or more leaf majority classes correctly can only reduce the classification rate.

Majority class error can now be decomposed as:

$$CR(T(maj)) - CR(T) = (\ CR(T(maj)) - E(\ CR(T(ran))\ )\ ) + (\ E(\ CR(T(ran))\ ) - CR(T)\ ) . \quad (2)$$

That is:

$$majority\ class\ error = sampling\ error + selection\ bias\ error .$$

Taken together, the two decompositions in Equations 1 and 2 yield an overall decomposition of inductive error into three components as

$$inductive\ error = structure\ error + sampling\ error + selection\ bias\ error. \quad (3)$$

4 Identifying Majority Class from a Random Sample and Leaf Sampling Error

The extent of sampling error is dependent on the probability that the majority class in a random sample is the correct one. By 'correct class' is meant a class (or one of several), called a majority class, which has the largest probability of occurrence at that leaf as determined from the model. Bechofer et al. (1959) and Kesten and Morse (1959) investigated the problem of correct selection with a view to determining the least favourable distribution, defined as that which minimizes, subject to constraints, the probability that the correct class will be identified.

In a k class problem ($k \geq 2$), suppose the probability distribution of the classes at a leaf node according to the class model is $(p_1,...,p_k)$. Assume throughout this discussion, following Bechofer et al. (1959), that the p_i are re-arranged so that $p_i \leq p_{i+1}$ for all i. A *majority class* is then one having probability p_k. Given a random sample of n from $(p_1,...,p_k)$ with frequency distribution $F = (f_1,...,f_k)$ across classes, the usual estimate of majority class based on F is:

$$\hat{class}_{maj} = \arg\max(F) .$$

For various n, k and $(p_1,...,p_k)$, the probability, P_{corr}, that this selection will be correct can be calculated from the multinomial distribution as:

$$P_{corr} = P(\hat{class}_{maj} = class_{maj})$$

where $class_{maj}$ is a majority class.

4.1 Properties of P_{corr}

Henceforth, p_k will be denoted p_{maj}. $P_{corr} \geq p_{maj}$ and increases with n (unless $p_{maj} = 1/k$ in which case, $P_{corr} = 1/k$ for all n). P_{corr} has the same value for $n = 1$ as for $n = 2$. For $k = 2$ and odd n, P_{corr} has the same value for n and $n + 1$.

Intuitively, for a given n, P_{corr} should be greater in situations where $p_{k-1} \square p_{maj}$ since the majority class has less competition and, conversely, should be small when all the p_i are fairly equal. Based on the work of Kesten and Morse (1959), Marshall and Olkin (1979) used majorisation theory to establish that, for fixed and unique p_{maj}, P_{corr} is Schur-concave in the *residual* probabilities $(p_1,...,p_{k-1})$. That is, P_{corr} is non-decreasing under an equalization operation on these probabilities in the sense of de-majorisation Hickey (1996).

For $k = 2$, p_{maj} determines the complete distribution $(1-p_{maj}, p_{maj})$. For $k > 2$ and fixed p_{maj}, the greatest equalization occurs when all residual probabilities are identical, that is, each is $(1-p_{maj})/(k-1)$. This will be referred to as the *equal residue* distribution and will be denoted $D_e(p_{maj}, k)$. Thus, P_{corr} is maximised amongst all distributions on k classes with given p_{maj} by $D_e(p_{maj}, k)$. At the other end of the scale, assuming $p_{maj} > 1/2$, then concentration of the residue at a single class produces the minimum P_{corr} for that n and p_k. Note that this latter situation is identical to that of a two-class problem with distribution $(1-p_{maj}, p_{maj})$. An implication of this is that, for $p_{maj} > 0.5$ and any given n, P_{corr} for the two class problem provides a lower bound for P_{corr} over all distributions on k classes, $k > 2$.

Bechofer et al. (1959) were concerned with the probability of correct selection when a (unique) majority class had at least a given margin of probability over the next largest, expressed as a multiplicative factor, a. Kesten and Morse (1959) showed that under the constraint

$$p_{maj} \geq a p_{k-1}, \quad a > 1$$

P_{corr} is minimized by the distribution

$$\left(\frac{1}{a+k-1}, \frac{1}{a+k-1}, ..., \frac{a}{a+k-1} \right).$$

The proof of this intuitive result is quite complex. An alternative proof was provided by Marshall and Olkin (1979) using the Schur-concavity property of P_{corr} for fixed p_{maj} discussed above.

For $D_e(p_{maj}, k)$, P_{corr} increases with k for fixed p_{maj} and increases with p_{maj} for fixed k. The first of these results follows from the Schur-concavity of P_{corr} in the residual probabilities for fixed p_{maj} because for $k < k'$, $D_e(p_{maj}, k)$ can be viewed as a distribution on k' classes. The second follows immediately from the Kesten and Morse theorem stated above because when p_{maj} is increased it will still satisfy the constraint $p_{maj} \geq a p_{k-1}, \quad a > 1$ for the value of a applicable before the increase.

The value of p_{maj} has considerable impact on the value of P_{corr} for given n. For example, for k = 2 and n = 10, p_{maj} = 0.6 produces P_{corr} of approximately 73% whereas for p_{maj} = 0.8, P_{corr} will be 98%. When k = 3 and n = 10, p_{maj} = 0.6 gives $D_e(p_{maj}, 3)$ = (0.2, 0.2, 0.6) and P_{corr} = 89%. The least favourable distribution here is (0, 0.4, 0.6) with P_{corr} = 73% as noted above.

When $k > 2$, it is possible that $p_{maj} < 0.5$. In this case, accumulating the residual probabilities on a single class will result in that class becoming the majority and thus the lower bound for P_{corr} offered by the corresponding two class problem does not hold.

Also, when $k > 2$, there may be tied majority classes in the leaf class probability distribution. Any of these when identified from the sample will qualify as a correct selection. Thus it is possible for p_{maj} to be very small and yet P_{corr} be large.

Bechofer et al. (1959) provide tables of P_{corr} for various values of the multiplicative factor a in the Kesten and Morse theorem and offer a large sample approximation for P_{corr}. Frank (2000) also considered the problem for $k = 2$ and graphs 1- P_{corr} against p_{maj}.

4.2 Leaf Classification Rate and Leaf Sampling Error

The sampling error of an induced tree is contributed to by the individual classifications taking place at each leaf of the tree. Inability to determine the correct majority class at a leaf impairs the classification rate locally at the leaf. The best rate that can be obtained at a leaf, that is, its local Bayes rate, is p_{maj} from its class probability distribution. The expected actual rate from a random sample, that is, the expectation of the probability of the selected class, will be called the (expected) *leaf classification rate* (*LCR*). Thus $LCR \leq p_{maj}$.

LCR can be calculated as an expectation over two events: either the correct majority class has been identified giving a conditional percentage classification rate of 100 * p_{maj} or it has not giving a conditional rate of 100* $E_{res}(P(\overline{class}_{maj}))$, where $E_{res}(P(\overline{class}_{maj}))$ is the conditional expected probability of the estimated majority class when it is incorrect, that is, over the residual probabilities. Thus, expressed as a percentage,

$$LCR = 100 * (p_{maj} * P_{corr} + E_{res}(P(\overline{class}_{maj})) * (1 - P_{corr})). \quad (4)$$

The Schur-concavity of P_{corr} for given p_{maj} does not extend to *LCR*. In Equation 4, for given p_{maj}, P_{corr} will increase as the residue probabilities are equalized, however this may be offset by the decrease in $E_{res}(P(\overline{class}_{maj}))$ as the larger residue probabilities decrease. For example, when $k = 4$, $p_{maj} = 0.4$ and $n = 3$, the distribution (0, 0.3, 0.3, 0.4) has *LCR* = 34.2%. Equalising the residue probabilities to $D_e(0.4, 4) = (0.2, 0.2, 0.2, 0.4)$ reduces *LCR* to 29.0%. On the other hand when $k = 5$, $p_{maj} = 0.8$ and $n = 3$, the distribution (0, 0, 0, 0.2, 0.8) has *LCR* = 73.8% whereas for the equal residue distribution $D_e(0.8, 5) = (0.05, 0.05, 0.05, 0.05, 0.8)$ this increases slightly to 74.0%.
For $D_e(p_{maj}, k)$, Equation 4 becomes:

$$LCR = 100 * (p_{maj} * P_{corr} + (1 - p_{maj}) * (1 - P_{corr}) / (k - 1)). \quad (5)$$

Using results stated above for $D_e(p_{maj}, k)$, it is straightforward to show that, for given n and k, *LCR* in Equation 5 increases with p_{maj}.

The shortfall $100 * p_{maj}$ - *LCR* is the expected loss in classification rate at a leaf due to the determination of majority class from a random sample and thus can be called the *leaf sampling error* (*LSE*). As noted above, the expectation of *LSE* over leaves in a tree is the sampling error as defined in Section 3.2.

In Table 1, *LCR* and *LSE* for $D_e(p_{maj}, k)$ are shown for $n = 1$, 2, 4 and 10 for several values of k and a range of values of p_{maj}. For given n and k it is seen that, although *LCR* increases with p_{maj}, as noted above, *LSE* increases and decreases again. When p_{maj} is large, majority class is

very likely to be correctly determined and hence sampling error is low. When p_{maj} is low, then the consequence of wrongful determination of majority class, although more likely, is cushioned by the complimentary probability being only slightly less than p_{maj} and so loss of classification rate is again minimal. As k and n increase, the maximum value of LSE tends to occur at smaller p_{maj} .

k	100* p_{maj}								
	10	20	30	40	50	60	70	80	90
					n = 1, 2				
2						52.0+8.0	58.0+**12.0**	68.0+**12.0**	82.0+8.0
3				34.0+ 6.0	37.5+12.5	44.0+16.0	53.5+**16.5**	66.0+14.0	81.5+8.5
4			25.3+4.7	28.0+12.0	33.3+16.7	41.3+**18.7**	52.0+18.0	65.3+14.7	81.3+8.7
5			21.3+8.7	25.0+15.0	31.3+18.7	40.0+**20.0**	51.3+18.7	65.0+15.0	81.3+8.7
10		11.1+8.9	14.4+15.6	20.0+20.0	27.8+**22.2**	37.8+**22.2**	50.0+20.0	64.4+15.5	81.1+8.9
15	6.8+3.2	8.6+11.4	12.5+17.5	18.6+21.4	26.8+**23.2**	37.1+22.9	49.6+20.4	64.3+15.7	81.1+8.9
					n = 4				
2						53.0+7.0	61.4+**8.6**	73.8+6.2	87.8+2.2
3				34.3+5.7	39.8+10.2	49.4+**10.6**	61.8+8.2	75.4+4.6	88.7+1.3
4			25.5+4.5	29.7+10.3	38.0+**12.0**	49.3+10.7	62.6+7.4	76.2+3.8	89.0+1.0
5			21.9+8.1	27.9+12.1	37.4+**12.6**	49.6+10.4	63.1+6.9	76.7+3.3	89.2+0.8
10		11.6+8.4	16.8+13.2	25.5+**14.5**	37.1+12.9	50.4+9.6	64.3+5.7	77.6+2.4	89.5+0.5
15	6.8+3.2	9.4+10.6	15.5+14.5	25.0+**15.0**	37.2+12.8	50.8+9.2	64.7+5.3	77.8+2.2	89.6+0.4
					n = 10				
2						54.7+**5.3**	66.0+4.0	78.8+1.2	89.9+0.1
3				35.0+ 5.0	43.2+ **6.8**	55.5+ 4.5	68.3+ 1.7	79.7+ 0.3	90.0+ 0.0
4			25.8+ 4.2	32.3+ **7.7**	43.8+ 6.2	56.9+ 3.1	69.0+ 1.0	79.9+ 0.1	90.0+ 0.0
5			23.1+ 6.9	32.2+ **7.8**	44.7+ 5.3	57.7+ 2.3	69.4+ 0.6	79.9+ 0.1	90.0+ 0.0
10		12.8+ 7.2	21.5+ **8.5**	34.0+ 6.0	47.0+ 3.0	59.0+ 1.0	69.8+ 0.2	80.0+ 0.0	90.0+ 0.0
15	6.9+ 3.1	11.7+ **8.3**	22.1+ 7.9	35.0+ 5.0	47.8+ 2.2	59.3+ 0.7	69.9+ 0.1	80.0+ 0.0	90.0+ 0.0

Table 1: Leaf classification rate (LCR) and leaf sampling error (LSE) for leaf sample size n = 1, 2, 4 and 10 for various k and p_{maj} under $D_e(p_{maj}, k)$. Cell format is LCR + LSE, both expressed as percentages; largest LSE for each k is shown bolded.

For k = 2, LSE reaches a maximum of approximately 12% when n = 1, 2 and p_{maj} lies between 0.7 and 0.8. For $k > 2$, LSE has the potential to be much larger than for k = 2 when n is small. For n = 1, 2, $P_{corr} = p_{maj}$ and, from Equation 5, LCR for $D_e(p_{maj}, k)$ can be expressed as:

$$LCR = 100 * (p_{maj}{}^2 + (1 - p_{maj})^2 / (k - 1))$$

which decreases with k to $100 * p_{maj}{}^2$. Thus, as k increases, LSE for $D_e(p_{maj}, k)$ tends to

$$100 * p_{maj} - 100 * p_{maj}{}^2 = 100 * p_{maj}(1 - p_{maj})$$

which has a maximum value of 25% at $p_{maj} = 0.5$. Table 1 shows that LSE can be 20% or above for $k \geq 5$.

For $n > 2$ and given p_{maj}, LCR for $D_e(p_{maj}, k)$ can increase or decrease with k. This is because LCR, in Equation 5, is a convex combination of p_{maj} and $(1 - p_{maj})/(k - 1))$ weighted by P_{corr} and $1 - P_{corr}$ respectively. As k increases, P_{corr} increases and $(1 - p_{maj})/(k - 1))$, which by definition is less than p_{maj}, decreases but its weight, $1 - P_{corr}$, is also decreasing.

Since the sampling error of the tree is the average of its leaf sampling errors, the behaviour of leaf sampling error should be reflected in the overall sampling error. As training set size increases, induced trees will tend to have more structure and so the leaf class probability distributions will be more informative with the result that individual p_{maj} in the leaf distributions will tend to increase. Thus the pattern of increase and decrease in sampling error with increases in p_{maj} noted above for LSE should be observable in the sampling error for the whole tree.

5 The LED Domain Revisited

To illustrate the error decompositions described in Section 3 and to motivate further discussion, decompositions will be calculated for decision trees induced using training data generated from the LED artificial domain (Breiman et al., 1984). An LED display for digits has seven binary indicators as illustrated in Figure 1. Each of these is corrupted, that is, inverted, independently with a given probability. If each digit has the same prior probability of being selected for display, then a complete probability model on attributes (x_1 , … , x_7, *class*) has been defined. The class model can be derived and represented, extensionally, as a set of 128 rules whose conditions express the instantiation of (x_1, …, x_7) and associate this with a probability distribution on the vector of ten classes, (*1*, … , *9*, *0*).

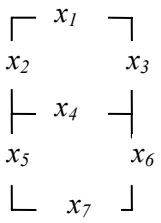


Figure 1: Mapping of attributes in the LED display.

5.1 Experimental Results

Experiments to induce ID3 trees were carried out on data generated from the LED domain with corruption probability 0.1 for which the Bayes rate is 74.0%. These were repeated on the 24

attribute domain obtained by augmenting the seven attributes with 17 mutually independent binary pure noise attributes (Breiman et al., 1984). In a final series of experiments, random attribute selection was used for induction on the 24 attribute domain.

A number of replications were performed at each of several points along the learning curve varying from 10000 at sample size 25 down to 10 at sample size 10000. For each trial, the error decomposition in Equation 1 and the sub-decomposition of majority class error in Equation 2 were obtained yielding the overall decomposition in Equation 3.

Sampling error was estimated in a tree by replacing each leaf with a freshly drawn random sample of the same size and obtaining the classification rate of the resulting tree. This produces an unbiased estimate of sampling error over replications and is more efficient than calculating the exact sampling error from the leaf class probability distribution, particularly when the sample reaching a leaf is large. The results are shown in Table 2.

For the seven attribute domain shown in Table 2(a), structure error decreases with sample size and is virtually eliminated at size 1000. For most of the learning curve, it is dominated by majority class error and the sub-decomposition shows that this is due mostly to sampling error with selection bias being either negative or approximately zero. Sampling error decreases with sample size due to the rapid increase in examples reaching the leaves. For sample sizes 25 and 50 the negative selection biases are two-tailed significant at the 5% level. The attribute selection competition here is aiding the determination of majority class: the sample reaching a leaf is better able to determine majority class than is an independent random sample.

Table 2 (b) shows that, with the addition of 17 pure noise attributes, the full trees are now much larger and that total error, structure and majority class error are considerably larger than for the seven attribute domain. There is still structure error at sample size 10000. Core trees are initially smaller but become larger as they inflate with locked in pure noise attributes. The large majority class error is due to both sampling and selection bias errors. Because of the availability of attributes for expansion, leaf sample sizes do not increase to reduce sampling error.

There is also some evidence of an increase and then decrease in sampling error due to a gradual increase in information in the leaf distributions as noted Section 4.2. In contrast to the seven attribute case, selection bias is now two-tailed significant at the 1% level along the learning curve apart from size 25, where, as for the seven attribute domain, it is significantly negative. As tree depth increases, there are fewer attributes available for selection, yet selection bias continues to increase along the learning curve suggesting that it accumulates with depth, that is, a leaf inherits a selection bias from its parent and adds to it.

Comparing Table 2 (c) with Table 2 (b) shows that random attribute selection produces much larger trees with fewer examples reaching each leaf. Core trees are also considerably inflated as is to be expected. The increase in error is accounted for by the much greater structure error. In contrast, the majority class error is generally much lower due to the reduction in selection bias error, which more than compensates for the larger sampling error. There is a modest increase in selection bias error along the learning curve becoming statistically different from zero at the 5% level from size 250 onwards. The process of tree expansion produces child node frequency configurations constrained to add up to that of the parent and which are, therefore, not genuinely independent of one another. Thus although there is no attribute selection competition, the process of repeatedly sub-dividing a single overall sample into progressively smaller constrained sub-samples does produce a small bias.

The experiments above were repeated for corruption probabilities 0.05 and 0.2. The results (not shown) exhibit similar characteristics to those reported above. Errors, particularly majority class error, worsen as corruption probability increases.

	Full Tree			Core Tree					Err Decomp		Maj Err Sub Decomp	
Samp Size	*No. Leaves*	*Av. Leaf Size*	*Av. Depth*	*No. Leaves*	*Av. Leaf Size*	*Av. Depth*	*CR (%)*	*Err (%)*	*Struct Err (%)*	*Maj Err (%)*	*Samp Err (%)*	*Sel Bias Err (%)*
(a) ID3 inductions on the seven attribute LED domain with corruption probability 0.1												
25	13	2.0	4.0	10	2.6	3.5	53.9	20.1	13.6	6.5	11.0	-4.5
50	20	2.5	4.7	13	4.1	4.0	61.6	12.4	6.4	6.0	7.3	-1.3
100	32	3.1	5.4	16	6.4	4.3	66.6	7.4	3.6	3.8	3.9	-0.1
250	55	4.6	6.0	23	11	4.9	69.6	4.4	1.3	3.1	3.0	0.1
500	72	6.9	6.4	26	20	5.1	71.6	2.4	0.4	2.0	2.1	-0.1
1000	88	11	6.6	27	38	5.1	73.0	1.0	0.1	0.9	0.9	0.0
2500	108	23	6.8	29	87	5.3	73.6	0.4	0.0	0.4	0.4	0.0
5000	118	43	6.9	29	174	5.2	73.8	0.2	0.0	0.2	0.1	0.1
10000	124	81	7.0	28	363	5.2	73.9	0.1	0.0	0.1	0.1	0.0
(b) ID3 inductions on the 24 attribute LED domain with corruption probability 0.1												
25	11	2.3	3.7	6.1	4.4	2.7	33.0	41.0	36.8	4.2	6.0	-1.8
50	19	2.7	4.6	8.4	6.2	3.3	42.4	31.6	22.1	9.5	7.0	2.5
100	34	3.0	5.6	10	10	3.6	47.9	26.1	13.0	13.2	6.9	6.3
250	76	3.3	6.8	14	19	4.2	51.7	22.3	8.8	13.5	5.7	7.8
500	145	3.5	7.9	20	26	4.8	53.2	20.8	7.4	13.4	5.2	8.2
1000	288	3.5	9.1	25	41	5.2	53.9	20.1	5.7	14.4	5.0	9.4
2500	711	3.5	10.7	32	80	5.8	54.6	19.4	4.2	15.2	5.1	10.1
5000	1447	3.5	11.9	46	112	7.1	54.8	19.3	4.1	15.2	5.0	10.2
10000	2935	3.4	13.0	60	181	7.9	54.8	19.2	3.4	15.8	5.1	10.7
(c) Inductions using random attribute selection on the 24 attribute LED domain with corruption probability 0.1												
25	30	0.9	5.8	17	1.7	4.9	17.1	56.9	52.2	4.7	4.8	-0.1
50	59	0.9	6.9	33	1.6	6.1	19.0	55.0	49.3	5.7	5.7	0.0
100	116	0.9	8.0	63	1.7	7.1	21.1	52.9	46.4	6.6	6.5	0.1
250	283	0.9	9.3	151	1.7	8.5	23.5	50.6	43.3	7.3	7.1	0.2
500	554	0.9	10.4	284	1.8	9.6	26.1	48.0	39.7	8.3	8.0	0.3
1000	1089	0.9	11.4	524	2.0	10.6	28.8	45.2	36.0	9.2	8.7	0.5
2500	2618	1.0	12.8	1192	2.2	11.9	31.8	42.2	31.7	10.5	9.5	1.0
5000	5082	1.0	13.8	2285	2.2	12.9	34.5	39.5	28.7	10.8	9.8	1.0
10000	9956	1.0	14.8	4441	2.3	13.9	36.1	37.9	26.5	11.4	10.1	1.3

Table 2: Tree statistics (*No. Leaves* = number of leaves in the tree; *Av. Leaf Size* = average number of examples in the leaves of the tree; *Av. Depth* = average depth of the tree), classification rate (*CR*), inductive error (*Err*) and error decompositions for tree inductions on examples generated from the LED domain. All results are averages over replications.

6 Experiments with the Autouniv Classification Model Generator

It is important to establish the extent to which the results from the LED domain, regarding the behaviour of the error decompositions, hold in general and how they change under different model characteristics. To investigate this, an artificial model generator, Autouniv, was built.

6.1 An Outline of the Autouniv Procedure

Autouniv produces a class model together with a joint distribution of the description attributes. At present the generator is implemented for discrete attributes only. To create a model, the number of informative attributes, pure noise attributes and classes are specified; the number of values for an attribute is specified as either a range across attributes or as the same fixed value for all attributes.

To create the joint attribute distribution, attributes are separated randomly into independent factors with the maximum number of attributes allowable in a factor also being specified. A pure noise attribute cannot be in the same factor as an informative attribute. The joint probability distribution for each factor is then generated at random. If the number of values for the attributes was specified as a range then, for each attribute, the actual number is randomized separately within this range.

To create the class model, a decision tree is generated and a class distribution is built at each leaf. The tree is then converted to a rule set. The tree is built in a random fashion as follows. At each expansion, an available attribute is selected at random from one of the informative factors. Pure noise attributes are never used for expansion. A minimum depth for the tree is set. After the tree has been built to this depth, further expansion along a path is controlled by a stopping probability, which is chosen at random between specified lower and upper limits and is generated independently at each leaf. This probability is then used in a 'coin toss' to determine whether the current node will be expanded. Finally, lower and upper limits are specified for the number of leaves of the tree. A tree will be rejected if its size is outside these limits. It will also be rejected if there is an informative factor at least one of whose attributes does not appear in the tree.

The class distribution at a leaf is created in two stages. First the majority class is selected from a specified distribution; ties are possible. Then the probability of this majority class (classes) is determined at random between given limits; these limits can be set differently for different classes. For the remaining non-majority classes, a subset of these is selected at random to receive positive probability which is assigned randomly.

The Autouniv procedure was developed to facilitate simple construction of a rich variety of realistic models. The tree building mechanism permits a degree of control of model complexity through specification of the number of informative attributes, the minimum depth of the tree, stopping probability range and the number of leaves. It also guarantees that all attributes declared as informative will be informative but also, through the factoring mechanism, that some of these may be redundant (Hickey, 1996). The procedure for constructing class distributions allows for specification of noise at different levels (and hence differing Bayes rates). It also permits heterogeneity in the noise across rules within a particular model. Some classes can be made noisier than others and class base rates can vary with one or more classes made rare if required.

In spite of the control provided by the parameters, some of the properties of the generated model remain implicit. An example is the degree of interaction of the description attributes. In some models most of the information about class will be carried by a small number of attributes whereas in others it will be distributed across a large number with no one or two attributes dominating.

Because of the generality of the Autouniv procedure, which can produce models from simple to very complex, with differing noise levels and degrees of attribute interaction, there is no reason

to suppose that it might be biased towards creating models for which induced trees exhibit a particular pattern of error decomposition, such as unusually large majority class error.

Once it has been generated, a model can be queried for a supply of training examples: an example description is obtained randomly from the attribute joint distribution, the matching rule is looked up and a class determined using the class distribution for that rule.

Finally, the parameters settings can themselves be randomized between given limits. This allows for easy generation of a heterogeneous series of models for experimentation.

6.2 Experiments with Autouniv

Ten models were generated to give variety with regard to the number of attributes and classes, default classification rate, lift, noise levels and model complexity. A summary of the main characteristics of these models is given in Table 3. All but three have pure noise attributes. The first five models have two classes; the remaining five have more than two classes. The columns headed *No. Rules* and *No. conditions in a rule* give an indication of model complexity. Most models are heterogeneous in the lengths of rule conditions.

Experiments similar to those performed on the LED domains in Section 5 were carried out on these 10 models. The results are shown in Table 4 (for the first five models) and in Table 5 (for the remaining five). The principal model characteristics from Table 3 are summarized in the first columns of Tables 4 and 5 for convenience. A classification rate (*CR*) which is less than the default classification rate (*DCR*) for the model is shown in italics in Tables 4 and 5. For several models the classification rate remains below the default well into the learning curve indicating that interaction of several attributes is required for lift.

For all models, structure error falls along the learning curve and for most is almost eliminated by size 10000. The exception is model 4. From Table 3, model 4 is quite complex in that the minimum rule condition length is 8, which is greater than for the other models. Majority class error is substantial for all models and, for most, exceeds structure error in the latter part of the learning curve, remaining high even when structure error has almost been eliminated.

Model	*No. atts*	*No. rel atts*	*No. pure noise atts*	*No. of vals of an att (Min-Max or constant)*	*No. Classes*	*No. Rules*	*No. conditions in a rule (Min-Av-Max)*	*Def Rate, DCR (%)*	*Bayes Rate, BCR (%)*	*Lift (%)*
1	5	5	0	7	2	11467	3 - 5 - 5	59.5	82.7	23.2
2	8	2	6	2 - 3	2	6	2 - 2 - 2	76.9	87.6	10.7
3	30	20	10	2	2	28	3 - 7.3 - 12	51.2	92.6	41.4
4	40	20	20	2	2	438	8 - 9.4 - 17	50.9	76.8	25.9
5	50	5	45	2 - 6	2	1030	3 - 4.8 - 5	61.6	81.7	20.1
6	8	2	6	2 - 5	10	4	2 - 2 - 2	81.0	91.7	10.7
7	12	12	0	2 - 4	4	530	1 - 10.3 - 12	42.9	98.3	55.4
8	15	15	0	3	13	1981	2 - 13.9 - 15	29.0	46.3	17.3
9	23	7	16	2	3	9	3 - 3.2 - 4	59.8	79.9	20.1
10	37	12	25	2 - 4	15	74	3 - 6.3 - 12	16.9	44.1	27.2

Table 3: Details of 10 models generated by Autouniv.

		Full Tree					Err Decomp		Maj Err Sub Decomp	
Model	*Sample Size*	*No. Leaves*	*Av. Leaf Size*	*Av. Depth*	*CR (%)*	*Err (%)*	*Struct Err (%)*	*Maj Err (%)*	*Samp Err (%)*	*Sel Bias Err (%)*
1	25	34	0.8	2.1	*52.3*	30.4	23.0	7.4	7.2	0.2
	50	67	0.8	2.5	*52.7*	30.0	22.6	7.4	7.7	-0.3
No. Atts: 5	100	139	0.7	2.9	*53.4*	29.3	21.8	7.5	7.4	0.1
No. Classes: 2	500	716	0.7	3.8	*56.5*	26.2	18.0	8.2	8.9	-0.7
No. Rules: 11467	1000	1435	0.7	4.2	*57.7*	25.0	15.5	9.5	9.8	-0.3
DCR (%): 59.5	5000	5161	1.0	4.7	66.6	16.1	5.4	10.7	10.4	0.3
BCR (%): 82.7	10000	7482	1.3	4.8	70.8	11.9	2.6	9.3	9.1	0.2
2	25	9	3.1	2.8	*74.9*	12.7	2.9	9.8	5.0	4.8
	50	16	3.3	3.6	*76.5*	11.1	1.4	9.7	4.1	5.6
No. Atts: 8	100	33	3.1	4.7	77.9	9.7	0.5	9.2	3.5	5.7
No. Classes: 2	500	129	3.9	6.4	81.1	6.5	0.0	6.5	3.1	3.4
No. Rules: 6	1000	206	4.9	6.8	82.8	4.8	0.0	4.8	2.4	2.4
DCR (%): 76.9	5000	455	11.0	7.5	86.4	1.2	0.0	1.2	0.8	0.4
BCR (%): 87.6	10000	573	17.5	7.7	87.1	0.5	0.0	0.5	0.3	0.2
3	25	7	3.7	3.2	64.4	28.2	22.3	5.9	4.3	1.6
	50	12	4.5	4.2	71.3	21.3	13.2	8.1	4.0	4.1
No. Atts: 30	100	20	5.3	5.1	78.4	14.2	5.8	8.4	1.9	6.5
No. Classes: 2	500	78	6.5	7.9	82.5	10.1	1.5	8.6	1.6	7.0
No. Rules: 28	1000	149	6.8	9.5	83.7	8.9	1.0	7.9	1.6	6.3
DCR (%): 51.2	5000	760	6.6	12.6	84.0	8.6	0.4	8.2	1.2	7.0
BCR (%): 92.6	10000	1588	6.3	14.4	84.4	8.2	0.1	8.1	1.1	7.0
4	25	7	3.5	3.3	*50.2*	26.6	25.4	1.2	1.4	-0.2
	50	14	3.5	4.4	*50.3*	26.5	25.0	1.5	1.7	-0.2
No. Atts: 40	100	28	3.6	5.5	*50.4*	26.4	24.8	1.6	1.8	-0.2
No. Classes: 2	500	144	3.5	8.1	50.9	25.9	23.5	2.4	3.0	-0.6
No. Rules: 438	1000	291	3.5	9.2	51.0	25.8	23.0	2.8	2.8	0.0
DCR (%): 50.9	5000	1427	3.5	11.6	54.1	22.7	16.8	5.9	5.0	0.9
BCR (%): 76.8	10000	2846	3.5	12.7	55.3	21.5	14.6	6.9	5.0	1.9
5	25	18	1.5	1.9	*53.9*	27.8	20.0	7.8	6.6	1.2
	50	33	1.6	2.4	*54.2*	27.5	19.7	7.8	7.1	0.7
No. Atts: 50	100	66	1.5	2.8	*54.1*	27.6	19.4	8.2	7.0	1.2
No. Classes: 2	500	309	1.6	3.9	*56.9*	24.8	15.7	9.1	7.3	1.8
No. Rules: 1030	1000	607	1.7	4.6	*60.2*	21.5	12.1	9.4	7.0	2.4
DCR (%): 61.6	5000	2680	1.9	5.5	65.6	16.1	4.6	11.5	7.4	4.1
BCR (%): 81.7	10000	5133	2.0	6.1	66.4	15.3	3.0	12.3	7.8	4.5

Table 4: Tree statistics and inductive error decomposition for ID3 tree inductions on examples generated from models 1 to 5 in Table 3.

		Full Tree					Err Decomp		Maj Err Sub Decomp	
Model	*Sample Size*	*No. Leaves*	*Av. Leaf Size*	*Av. Depth*	*CR (%)*	*Err (%)*	*Struct Err (%)*	*Maj Err (%)*	*Samp Err (%)*	*Sel Bias Err (%)*
6	25	10	3.5	2.4	*79.7*	12.0	3.8	8.2	3.9	4.3
	50	19	3.3	3.3	81.8	9.9	1.7	8.2	2.2	6.0
No. Atts: 8	100	38	2.8	4.2	83.2	8.5	0.5	8.0	2.5	5.5
No. Classes: 10	500	181	2.8	5.8	84.4	7.3	0.0	7.3	2.3	5.0
No. Rules: 4	1000	320	3.1	6.2	85.4	6.3	0.0	6.3	2.4	3.9
DCR (%): 81.0	5000	1070	4.7	7.0	88.2	3.5	0.0	3.5	1.5	2.0
BCR (%): 91.7	10000	1610	6.2	7.2	89.6	2.1	0.0	2.1	1.0	1.1
7	25	11	2.5	2.1	72.8	25.5	19.3	6.2	4.7	1.5
	50	17	3.0	2.7	76.7	21.6	15.6	6.0	4.6	1.4
No. Atts: 12	100	28	3.8	3.4	83.3	15.0	9.6	5.4	4.2	1.2
No. Classes: 4	500	75	6.8	4.5	92.5	5.8	2.7	3.1	1.4	1.7
No. Rules: 530	1000	138	7.4	5.1	93.1	5.2	1.8	3.4	1.0	2.4
DCR (%): 42.9	5000	525	9.6	6.5	95.0	3.3	0.6	2.7	0.7	2.0
BCR (%): 98.3	10000	1071	9.3	7.3	95.7	2.6	0.4	2.2	0.3	1.9
8	25	22	1.1	3.0	*20.0*	26.3	14.3	12.0	12.1	-0.1
	50	43	1.2	3.7	*25.0*	21.3	10.5	10.8	10.6	0.2
No. Atts: 15	100	82	1.2	4.4	*27.6*	18.7	7.3	11.4	10.2	1.2
No. Classes: 13	500	382	1.3	5.8	31.9	14.4	1.6	12.8	11.0	1.8
No. Rules: 1981	1000	760	1.3	6.5	32.5	13.8	1.0	12.8	11.5	1.3
DCR (%): 29.0	5000	3899	1.3	8.1	33.2	13.1	0.2	12.9	11.5	1.4
BCR (%): 46.3	10000	7997	1.3	8.8	33.7	12.6	0.1	12.5	11.2	1.3
9	25	8	3.2	3.5	61.7	18.2	2.3	15.9	5.8	10.1
	50	15	3.4	4.7	63.1	16.8	0.1	16.7	5.7	11.0
No. Atts: 23	100	31	3.3	6.1	63.9	16.0	0.0	16.0	5.6	10.4
No. Classes: 3	500	161	3.1	9.1	63.8	16.1	0.0	16.1	5.6	10.5
No. Rules: 9	1000	345	2.9	10.6	64.1	15.8	0.0	15.8	5.6	10.2
DCR (%): 59.8	5000	1817	2.8	13.6	64.7	15.2	0.0	15.2	6.4	8.8
BCR (%): 79.9	10000	3718	2.7	15.0	64.9	15.0	0.0	15.0	6.6	8.4
10	25	27	1.0	2.6	*11.2*	32.9	24.1	8.8	9.0	-0.2
	50	51	1.0	3.1	*12.6*	31.5	22.0	9.5	9.7	-0.2
No. Atts: 37	100	101	1.0	3.7	*13.8*	30.3	20.3	10.0	10.1	-0.1
No. Classes: 15	500	428	1.2	5.3	23.7	20.4	8.5	11.9	10.9	1.0
No. Rules: 74	1000	786	1.3	6.1	27.6	16.5	4.9	11.6	10.4	1.2
DCR (%): 16.9	5000	3664	1.4	7.7	32.0	12.1	0.2	11.9	10.5	1.4
BCR (%): 44.1	10000	7296	1.4	8.3	33.2	10.9	0.1	10.8	9.2	1.6

Table 5: Tree statistics and inductive error decomposition for ID3 tree inductions on examples generated from models 6 to 10 in Table 3.

Sampling errors are consistent with those expected from the discussion in Section 4.2 and from Table 1. In model 4, which has the lowest initial sampling error, the default rate is 50.9% indicating that near the beginning of the learning curve there is little penalty from obtaining an incorrect majority class. As the trees acquire structure, sampling error rises while the sample size at the leaf remains constant. The largest sampling error occurs for models 8 and 10, which have 13 and 15 classes respectively and quite low default and Bayes rates. Leaf sample sizes are smallest for these models. Sampling errors, though, fall short of the maxima in Table 1.

Selection bias error shows a more complex pattern. There are instances of very high bias and of virtually non-existent bias and the extent seems comparatively unrelated to the number of attributes and other characteristics of the model. For model 1 with five attributes, it is similar to that seen for the seven attribute LED domain. For model 2, however, with eight attributes, it is fairly large at the beginning of the learning curve when the leaf sample size is small, falling later on when it increases. In contrast, for model 4, with 40 attributes, 20 of which are pure noise, selection bias error only becomes statistically significant from size 5000.

There is some indication that the occurrence of larger selection bias is associated with simpler models such as 2, 3, 6 and 9. A possible explanation for this is that when structure error is low, the attribute selection competition is taking place amongst attributes none of which can offer much information gain. The competition is then more vulnerable to spurious leaf distributions.

6.2.1 RANDOM TREES

All experiments were repeated with random attribute selection. The random trees were much larger for all models, typically having between 2 and 4 times as many leaves as the corresponding ID3 trees. For all models, except model 1, the classification rates along the learning curve were significantly lower than for the corresponding ID3 curve.

The random trees for model 1 matched the classification rates for ID3, with no significant difference all along the learning curve. Likewise there was no significant difference in decomposition errors. This is due to there being only five attributes all of which are relevant. Moreover, from Table 4, the classification rates are less than the default until after sample size 1000 so that the lift is shared amongst the attributes rather than being concentrated in one or two. Thus, as noted by Liu and White (1994), there is little benefit from selection based on maximising information gain over that offered by random selection.

For nearly all models, the selection bias error was virtually eliminated along the learning curve. Only for models 6 and 9 was here a slight increase as was observed for the LED domain.

These results broadly confirm the findings of Liu and White (1994). Their poorer performance is due to much larger structure error caused by the interference of pure noise attributes. They tend to have very much larger reduced cores indicating a high degree of inflation. Where the ID3 tree has significant selection bias error, this will be almost eliminated in the random tree but this benefit, which may be accompanied anyway by larger sampling error due to smaller leaf samples, is usually not sufficient to compensate for the increased structure error.

7 Experiments with Real Data

Determination of the error decomposition requires knowledge of the joint probability distribution of the description attributes and the class. This is necessary to calculate, for example, the Bayes rate and the correct majority class at a leaf. Only the sub-decomposition of majority class into sampling and selection bias errors was estimated from sample data for convenience as explained in Section 5. If a large data set is available then it is possible to estimate the whole decomposition. To illustrate the procedures involved, the Forest Cover data set (UCI KDD Archive) will be used.

In the Forest Cover data, seven species of tree are classified using 54 attributes that describe their location. The data consists of 581012 examples. There are no missing values. Five of the classes are comparatively rare. These were combined into a single class called 'other'. There are 40 binary attributes that describe soil type. These were eliminated. Four binary attributes describing wilderness area were combined into a single four-valued wilderness attribute.

The remaining 10 attributes are all continuous. These were made discrete by binning each into four bins with labels 1, 2, 3 and 4 as shown in Table 6.

Attribute	**Binning Ranges for Bins 1 - 4**			
	1	2	3	4
Elevation	< 2400	2400 < 3000	3000 < 3300	≥ 3300
Aspect	< 60	60 < 180	180 < 300	≥ 300
Slope	< 11	11 < 33	33 < 55	≥ 55
Horizontal_Distance_To_Hydrology	< 233	233 < 699	699 < 1165	≥ 1165
Vertical_Distance_To_Hydrology	< -44	-44 < 214	214 <472	≥ 472
Horizontal_Distance_To_Roadways	< 1187	1187 < 3559	3559 < 5930	≥ 5930
Hillshade_9am	< 42.3	42.3 < 127.0	127.0 < 211.7	≥ 211.7
Hillshade_Noon	< 42.3	42.3 < 127.0	127.0 < 211.7	≥ 211.7
Hillshade_3pm	< 42.3	42.3 < 127.0	127.0 < 211.7	≥ 211.7
Horizontal_Distance_To_Fire_Points	< 1196	1196 < 3587	3587 < 5978	≥ 5978

Table 6: Continuous to discrete conversion of 10 Forest Cover attributes.

The final data set consists of 11 discrete attributes and three classes. This was split randomly into approximately 75% to represent the model and 25% to be used as a test set. Statistics relating to these sets are shown in Table 7.

		Class Distribution		
Data Set	Size	*Lodgepole Pine*	*Spruce/Fir*	*Other*
Whole	581012	283301	211840	85871
	(100%)	(48.76%)	(36.46%)	(14.78%)
Model	436169	212587	158968	64614
	(75.1%)	(48.74%)	(36.45%)	(14.81%)
Test	144843	70714	52872	21257
	(24.9%)	(48.82%)	(36.50%)	(14.68%)

Table 7: Statistics for the revised Forest Cover data with 11 attributes and three classes obtained from a random partitioning into model and test subsets.

To estimate the Bayes rate, the CART algorithm was applied to the model set to build a decision tree. The twoing measure was used for attribute selection. Cost complexity pruning with

the 1 standard error setting was applied. The resulting tree, which is an approximation to the true model, had 1131 leaves. The rule set derived from this tree was applied to the test set producing a classification rate of 73.78% as an estimate of the true Bayes rate. From Table 7, the default classification rate in the test set is 48.82% giving a lift of 24.96%.

Experiments similar to those in Sections 5 and 6 were carried out on the revised data. Training examples were drawn randomly from the model data set. To determine structure and majority class error, an estimate of the true majority class at a leaf in an induced tree needs to be obtained. This was provided by applying the rule condition associated with the leaf to the whole of the model data set. The majority class amongst examples matching this condition was then taken as the estimate of the true majority class. Such estimates are typically based on fairly large numbers of matching examples. For example, if a tree induced from 10000 examples has a leaf containing two examples then the expectation, from Table 7, is that there would be 2 * 436169/10000 = 87 matching examples in the model set. These estimates of majority class were also used to estimate the reduced core of each induced tree through same majority class pruning.

Sampling error was calculated by the method described in Section 5 using random samples obtained from the model data set. All classification rates needed for the decomposition were estimated from the test data set.

The results, shown in Table 8, exhibit similar patterns of decomposition to those seen earlier. A sample of about 30000 is required to virtually eliminate structure error. Majority class error dominates structure error all along the learning curve. This is mostly due to sampling error. The low selection bias error across the learning curve is consistent with that observed for the more complex artificial models in Section 6.

	Full Tree			Core Tree					Err Decomp		Maj Err Sub Decomp	
Samp Size	*No. Leaves*	*Av. Leaf Size*	*Av. Depth*	*No. Leaves*	*Av. Leaf Size*	*Av. Depth*	*CR (%)*	*Err (%)*	*Struct Err (%)*	*Maj Err (%)*	*Samp Err (%)*	*Sel Bias Err (%)*
25	25	1.1	2.9	16	1.8	2.5	52.6	21.2	9.4	11.8	11.0	0.8
50	51	1.0	3.7	29	1.9	3.2	55.3	18.5	6.8	11.7	9.9	1.8
100	102	1.0	4.5	55	1.9	4.0	56.6	17.2	5.8	11.4	9.4	2.0
500	502	1.0	6.3	241	2.1	5.6	60.0	13.8	3.6	10.2	8.2	2.0
1000	908	1.1	6.8	425	2.4	6.2	62.1	11.7	2.7	9.0	7.5	1.5
2500	1746	1.4	7.3	799	3.1	6.6	65.2	8.6	1.6	7.0	5.9	1.1
5000	2633	1.9	7.7	1195	4.2	7.0	67.8	6.0	1.1	4.9	4.2	0.7
10000	3671	2.7	7.9	1597	6.3	7.2	69.8	4.0	0.6	3.4	3.1	0.3
20000	4971	4.0	8.1	2065	9.7	7.4	71.4	2.4	0.4	2.0	1.7	0.3
30000	5727	5.2	8.2	2402	12.5	7.5	72.3	1.5	0.2	1.3	1.3	0.0

Table 8: Tree statistics and inductive error decomposition for ID3 inductions on examples generated from the revised Forest Cover data.

8 Conclusion and Future Work

The contribution of this paper has been the introduction of a method of decomposition of the classification error occurring in decision tree induction. Its application has been demonstrated on both artificial and real data. Instead of comparing tree induction algorithms in terms of classification error is it now possible to provide further insight into how this arose, specifically whether it is due to failure to grow sufficient tree structure or to successfully estimate majority class at the leaves.

It has been shown that majority class error is often quite substantial and that it can be further broken down into sampling error and selection bias error with the extent of these sources being quantified.

By factoring out the effects of selection bias, the sub-decomposition of majority class error permits a statistical analysis of sampling error not previously possible because of the biased samples reaching the leaves. For two classes, sampling error appears to be limited to a maximum of about 12%. For more than two classes it could be as much as 25%. Sampling error does decrease reasonably quickly when the size of sample reaching the leaves eventually begins to increase, particularly if the level of noise in the domain is low.

In ID3, selection bias error is due to the corruption in the sample reaching a leaf caused by the multiple comparison effect of the competition to select the best attribute with which to expand the tree. It may be insignificantly different from zero along the learning curve even when there are a large number of attributes involved in the selection competition and yet may be large when there are only a small number of attributes. The initial evidence provided here supports the hypothesis that if the underlying model is sufficiently complex, then this offers some protection against selection bias error. Although regarded here as a source of majority class error, selection bias error could, conceivably, be viewed as part of the error in forming tree structure.

The results provided here offer further insight into why ID3 typically outperforms trees grown with random attribute selection. It is due to a largely successful trade-off in forming structure efficiently at the expense of creating selection bias error.

The challenge for future work is to use the decomposition to develop better tree induction algorithms. For example, it may be possible to find a better trade-off between forming structure and incurring selection bias than that offered by ID3. The decomposition can be applied to induced trees however constructed. In particular, it can be obtained for trees that have been pruned and so should enhance investigation into issues relating to overfitting avoidance (Schaffer, 1993) and the properties of methods of pruning (Oates and Jensen, 1997; 1999).

It may also be possible to extend the approach to investigate the behaviour of bagging and boosting techniques for decision trees.

Majority class error was decomposed into sampling and selection bias errors. It is possible, instead, to decompose it in a way that reflects contributions from the reduced core and from extension beyond the core. Such an alternative sub-decomposition may be especially useful in investigating overfitting avoidance. Structure error can also be decomposed to isolate the effect of pure noise attributes on the induction. Work is being undertaken on both of these.

References

Robert E. Bechhofer, Salah Elmaghraby and Norman Morse. A single sample multiple-decision procedure for selecting the multinomial event which has the highest probability. *Annals of Mathematical Statistics*, 30:102-119, 1959.

Leo Breiman, Jerome H. Friedman, Richard A. Olshen and Charles J. Stone. *Classification and Regression Trees*. Wadsworth, Pacific Grove, California, 1984.

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

Carlos R.P. Hartmann, Pramod K. Varshney, Kishan G. Mehrotra and Carl L. Gerberich. Application of information theory to the construction of efficient decision trees. *IEEE Transactions on Information Theory*, IT-28:565-577, 1982.

Ray J. Hickey. Artificial universes: towards a systematic approach to evaluating algorithms which learn from examples. In *Proceedings of the Ninth International Conference on Machine Learning*, pages 196-205, Aberdeen, Scotland, 1992.

Ray J. Hickey. Noise modelling and evaluating learning from examples. *Artificial Intelligence*, 82(1-2):157-179, 1996.

Gareth M. James. Variance and bias for general loss functions. *Machine Learning*, 51(2):115-135, 2003.

David D. Jensen and Paul R. Cohen. Multiple comparisons in induction algorithms. *Machine Learning*, 38(3):309-338, 2000.

Harry Kesten and Norman Morse. A property of the multinomial distribution. *Annals of Mathematical Statistics*, 30:120-127, 1959.

Ron Kohavi and George H. John. Wrappers for feature subset selection. *Artificial Intelligence*, 97(1-2):273-324, 1997.

Wei Zhong Liu and Allan P. White. The importance of attribute selection measures in decision tree induction. *Machine Learning*, 15(1):25-41, 1994.

Albert W. Marshall and Ingram Olkin. *Inequalities: The Theory of Majorisation and its Applications*. Academic Press, New York, 1979.

Tom M. Mitchell. *Machine Learning*. McGraw-Hill, New York, 1997.

Tim Oates and David D. Jensen. The effects of training set size on decision tree complexity. In *Proceedings of the Fourteenth International Conference on Machine Learning*, pages. 254-261, Nashville, Tennessee, 1997.

Tim Oates and David D. Jensen. Toward a theoretical understanding of why and when decision tree pruning algorithms fail. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence*, pages 372-378, Orlando, Florida, 1999.

Foster Provost and Pedro Domingos. Tree induction for probability-based ranking. *Machine Learning*, 52(3):199-215, 2003.

J. Ross Quinlan. Induction of decision trees. *Machine Learning*, 1(1):81-106, 1986.

Cullen Schaffer. Overfitting avoidance as bias. *Machine Learning*, 10(2):153-178, 1993.

UCI KDD Archive. http://kdd.ics.uci.edu

Gary M. Weiss and Haym Hirsh. A quantitative study of small disjuncts. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence*, pages 665-670, Austin, Texas, 2000.

Journal of Machine Learning Research 8 (2007) 1769-1797 Submitted 7/06; Revised 1/07; Published 8/07

Characterizing the Function Space for Bayesian Kernel Models

Natesh S. Pillai NSP2@STAT.DUKE.EDU
Qiang Wu QIANG@STAT.DUKE.EDU
Department of Statistical Science
Duke University
Durham, NC 27708, USA

Feng Liang FENG@STAT.UIUC.EDU
Department of Statistics
University of Illinois at Urbana-Champaign
Urbana-Champaign, IL 61820, USA

Sayan Mukherjee SAYAN@STAT.DUKE.EDU
Department of Statistical Science
Institute for Genome Sciences & Policy
Duke University
Durham, NC 27708, USA

Robert L. Wolpert WOLPERT@STAT.DUKE.EDU
Department of Statistical Science
Professor of the Environment and Earth Sciences
Duke University
Durham, NC 27708, USA

Editor: Zoubin Ghahramani

Abstract

Kernel methods have been very popular in the machine learning literature in the last ten years, mainly in the context of Tikhonov regularization algorithms. In this paper we study a coherent Bayesian kernel model based on an integral operator defined as the convolution of a kernel with a signed measure. Priors on the random signed measures correspond to prior distributions on the functions mapped by the integral operator. We study several classes of signed measures and their image mapped by the integral operator. In particular, we identify a general class of measures whose image is dense in the reproducing kernel Hilbert space (RKHS) induced by the kernel. A consequence of this result is a function theoretic foundation for using non-parametric prior specifications in Bayesian modeling, such as Gaussian process and Dirichlet process prior distributions. We discuss the construction of priors on spaces of signed measures using Gaussian and Lévy processes, with the Dirichlet processes being a special case the latter. Computational issues involved with sampling from the posterior distribution are outlined for a univariate regression and a high dimensional classification problem.

Keywords: reproducing kernel Hilbert space, non-parametric Bayesian methods, Lévy processes, Dirichlet processes, integral operator, Gaussian processes

1. Introduction

Kernel methods have a long history in statistics and applied mathematics (Schoenberg, 1942; Aronszajn, 1950; Parzen, 1963; de Boor and Lynch, 1966; Micchelli and Wahba, 1981; Wahba, 1990) and have had a tremendous resurgence in the machine learning literature in the last ten years (Poggio and Girosi, 1990; Vapnik, 1998; Schölkopf and Smola, 2001; Shawe-Taylor and Cristianini, 2004). Much of this resurgence was due to the popularization of classification algorithms such as support vector machines (SVMs) (Cortes and Vapnik, 1995) that are particular instantiations of the method of regularization of Tikhonov (1963). Many machine learning algorithms and statistical estimators can be summarized by the following penalized loss functional (Evgeniou et al., 2000; Hastie et al., 2001, Section 5.8)

$$\hat{f} = \arg\min_{f \in \mathcal{H}} \left[L(f, \text{data}) + \lambda \|f\|_K^2 \right],$$

where L is a loss function, $\mathcal{H}$ is often an infinite-dimensional reproducing kernel Hilbert space (RKHS), $\|f\|_K^2$ is the norm of a function in this space, and λ is a tuning parameter chosen to balance the trade-off between fitting errors and the smoothness of the function. The data is assumed to be drawn independently from a distribution $\rho(x,y)$ with $x \in \mathcal{X} \subset \mathbb{R}^d$ and $y \in \mathcal{Y} \subset \mathbb{R}$. Due to the representer theorem (Kimeldorf and Wahba, 1971) the solution of the penalized loss functional will be a kernel

$$\hat{f}(x) = \sum_{i=1}^{n} w_i K(x, x_i),$$

where $\{x_i\}_{i=1}^n$ are the n observed input or explanatory variables. The statistical learning community as well as the approximation theory community has studied and characterized properties of the RKHS for various classes of kernels (DeVore et al., 1989; Zhou, 2003).

Probabilistic versions and interpretations of kernel estimators have been of interest going back to the work of Hájek (1961, 1962) and Kimeldorf and Wahba (1971). More recently a variety of kernel models with a Bayesian framework applied to the finite representation from the representer theorem have been proposed (Tipping, 2001; Sollich, 2002; Chakraborty et al., 2005). However, the direct adoption of the finite representation is not a fully Bayesian model since it depends on the (arbitrary) training data sample size (see remark 3 for more discussion). In addition, this prior distribution is supported on a finite-dimensional subspace of the RKHS. Our coherent fully Bayesian approach requires the specification of a prior distribution over the entire space $\mathcal{H}$.

A continuous, positive semi-definite kernel on a compact space $\mathcal{X}$ is called a *Mercer* kernel. The RKHS for such a kernel can be characterized (Mercer, 1909; König, 1986) as

$$\mathcal{H}_K = \left\{ f \;\middle|\; f(x) = \sum_{j \in \Lambda} a_j \phi_j(x) \text{ with } \sum_{j \in \Lambda} {a_j}^2 / \lambda_j < \infty \right\}, \tag{1}$$

where $\{\phi_j\} \subset \mathcal{H}$ and $\{\lambda_j\} \subset \mathbb{R}_+$ are the orthonormal eigenfunctions and the corresponding non-increasing eigenvalues of the integral operator with kernel K on $L^2\big(\mathcal{X}, \mu(\mathrm{d}u)\big)$,

$$\lambda_j \phi_j(x) = \int_{\mathcal{X}} K(x,u) \phi_j(u) \mu(\mathrm{d}u) \tag{2}$$

and where $\Lambda := \big\{ j : \lambda_j > 0 \big\}$. The eigenfunctions $\{\phi_j\}$ and the eigenvalues $\{\lambda_j\}$ depend on the measure $\mu(\mathrm{d}u)$, but the RKHS does not. This suggests specifying a prior distribution over $\mathcal{H}$ by

placing one on the parameter space

$$\mathcal{A} = \left\{ \{a_j\} \,\middle|\, \sum_{j \in \Lambda} {a_j}^2 / \lambda_j < \infty \right\}$$

as in Johnstone (1998) and Wasserman (2005, Section 7.2). There are serious computational and conceptual problems with specifying a prior distribution on this infinite-dimensional set. In particular, only in special cases are the eigenfunctions $\{\phi_j\}$ and eigenvalues $\{\lambda_j\}$ available in closed form.

Another approach, the *Bayesian kernel model*, is to study the class of functions expressible as kernel integrals

$$\mathcal{G} = \left\{ f \,\middle|\, f(x) = \int_{\mathcal{X}} K(x,u)\, \gamma(\mathrm{d}u), \;\; \gamma \in \Gamma \right\}, \tag{3}$$

for some space $\Gamma \subseteq \mathcal{B}(\mathcal{X})$ of signed Borel measures. Any prior distribution on Γ induces one on $\mathcal{G}$. The natural question that arises in this Bayesian approach is:

> For what spaces Γ of signed measures is the RKHS $\mathcal{H}_K$ identical to the linear space $\mathrm{span}(\mathcal{G})$ spanned by the Bayesian kernel model?

The space $\mathcal{G}$ is the range $\mathscr{L}_K[\Gamma]$ of the integral operator $\mathscr{L}_K : \Gamma \to \mathcal{G}$ given by

$$\mathscr{L}_K[\gamma](x) := \int_{\mathcal{X}} K(x,u)\, \gamma(\mathrm{d}u). \tag{4}$$

Informally (we will be more precise in Section 2) we can characterize Γ as the range of the inverse operator $\mathscr{L}_K^{-1} : \mathcal{H}_K \to \Gamma$. The requirements on Γ for the equivalence between $\mathscr{L}_K[\Gamma]$ and $\mathcal{H}_K$ is the primary focus of this paper and in Section 2 we formalize and prove the following proposition:

Proposition 1 *For $\Gamma = \mathcal{B}(\mathcal{X})$, the space of all signed Borel measures, $\mathcal{G} = \mathcal{H}_K$.*

The proposition asserts that the Bayesian kernel model and the penalized loss model both operate in the same function space when Γ includes all signed measures.

This result lays a theoretical foundation from a function analytic perspective for the use of two commonly used prior specifications: Dirichlet process priors (Ferguson, 1973; West, 1992; Escobar and West, 1995; MacEachern and Müller, 1998; Müller et al., 2004) and Lévy process priors (Wolpert et al., 2003; Wolpert and Ickstadt, 2004).

1.1 Overview

In Section 2, we formalize and prove the above proposition. Prior distributions are placed on the space of signed measures in Section 4 using Lévy, Dirichlet, and Gaussian processes. In Section 5 we provide two examples using slightly different process prior distributions for a univariate regression problem and a high dimensional classification problem. This illustrates the use of these process priors for posterior inference. We close in Section 6 with a brief discussion.

Remark 2 *Equation (3) is a Fredholm integral equation of the first kind (Fredholm, 1900). The problem of estimating an unknown measure γ for a specified element $f \in \mathcal{H}_K$ is ill-posed (Hadamard, 1902) in the sense that small changes in f may give rise to large changes in estimates of γ. It was precisely the study of this problem that led Tikhonov (1963) to his regularization method, in a study*

of problems in numerical analysis such as interpolation or Gauss quadrature. Bayesian methods for interpolation and Gauss quadrature were explored by Diaconis (1988). A Bayesian method using Lévy process priors to address numerically ill-posed problems was developed by Wolpert and Ickstadt (2004). We will return to this relation between robust statistical estimation and numerically stable methods in the discussion.

Remark 3 *Due to the relation between regularization and Bayes estimators the finite representation is a MAP (maximal a posterior) estimator (Wahba, 1999; Poggio and Girosi, 1990). However, functions in the RKHS having a posterior probability very close to that of the MAP estimator need not have a finite representation so building a prior only on the finite representation is problematic if one wants to estimate the full posterior on the entire RKHS. Thus the prior used to derive the MAP estimate is essentially the same as those used in Sollich (2002). This will lead to serious computational and conceptual difficulties when the full posterior must be simulated.*

2. Characterizing the Function Space of the Kernel Model

In this section we formalize the relationship between the RKHS and the function space induced by the Bayesian kernel model.

2.1 Properties of the RKHS

Let $\mathcal{X} \subset \mathbb{R}^d$ be compact and $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ a continuous, positive semi-definite (*Mercer*) kernel. Consider the space of functions

$$\mathcal{H} = \left\{ f \;\middle|\; f(x) = \sum_{j=1}^{n} a_j K(x, x_j) :\ n \in \mathbb{N},\ \{x_j\} \subset \mathcal{X},\ \{a_j\} \subset \mathbb{R} \right\}$$

with an inner product $\langle \cdot, \cdot \rangle_K$ extending

$$\big\langle K(\cdot, x_i),\ K(\cdot, x_j) \big\rangle_K := K(x_i, x_j).$$

The Hilbert space closure $\mathcal{H}_K$ of $\mathcal{H}$ in this inner-product is the RKHS associated with the kernel K (Aronszajn, 1950). The kernel is "reproducing" in the sense that each $f \in \mathcal{H}_K$ satisfies

$$f(x) = \langle f, K_x \rangle_K$$

for all $x \in \mathcal{X}$, where $K_x(\cdot) := K(\cdot, x)$.

A well-known alternate representation of the RKHS is given by an orthonormal expansion (Aronszajn 1950, extended to arbitrary measures by König 1986; see Cucker and Smale 2001). Let $\{\lambda_j\}$ and $\{\phi_j\}$ be the non-increasing eigenvalues and corresponding complete orthonormal set of eigenvectors of the operator $\mathscr{L}_K$ of Equation (4), restricted to the Hilbert space $L^2\big(\mathcal{X}, \mathrm{d}u\big)$ of measures $\gamma(\mathrm{d}u) = \gamma(u)\mathrm{d}u$ with square-integrable density functions $\gamma \in L^2\big(\mathcal{X}, \mathrm{d}u\big)$. Mercer's theorem (Mercer, 1909) asserts the uniform and absolute convergence of the series

$$K(u, v) = \sum_{j=1}^{\infty} \lambda_j \phi_j(u) \phi_j(v), \tag{5}$$

whereupon with $\Lambda := \{j : \lambda_j > 0\}$ we have

$$\mathcal{H}_K = \left\{ f = \sum_{j \in \Lambda} a_j \phi_j \;\Big|\; \sum_{j \in \Lambda} {\lambda_j}^{-1} {a_j}^2 < \infty \right\}.$$

2.2 Bayesian Kernel Models and Integral Operators

Recall the Bayesian kernel model was defined by

$$\mathcal{G} = \left\{ \mathscr{L}_\kappa[\gamma](x) := \int_{\mathcal{X}} K(x,u)\ \gamma(\mathrm{d}u), \quad \gamma \in \Gamma \right\},$$

where Γ is a space of signed Borel measures on $\mathcal{X}$. We wish to characterize the space $\mathscr{L}_K^{-1}(\mathcal{H}_K)$ of Borel measures mapped into the RKHS $\mathcal{H}_K$ of Equation (1). A precise characterization is difficult and instead we will find a subclass $\Gamma \subset \mathscr{L}_K^{-1}(\mathcal{H}_K)$ which will be large enough in practice, in the sense that $\mathscr{L}_K(\Gamma)$ is dense in $\mathcal{H}_K$.

First we study the image under $\mathscr{L}_K$ of four classes of measures: (1) those with square integrable (Lebesgue) density functions; (2) all finite measures with Lebesgue density functions; (3) the set of discrete measures; and (4) linear combinations of all of these. Then we will extend these results to the general case of Borel measures (see Appendix A for proofs).

We first examine the class $L^2(\mathcal{X}, \mathrm{d}u)$, viewed as the space of finite measures on $\mathcal{X}$ with square-integrable density functions with respect to Lebesgue measure; in a slight abuse of notation we write $\gamma(\mathrm{d}u) = \gamma(u)\mathrm{d}u$, using the same letter γ for the measure and its density function. Since $\mathcal{X}$ is compact and K bounded, $\mathscr{L}_K$ is a positive compact operator on $L^2(\mathcal{X}, \mathrm{d}u)$ with a complete ortho-normal system (CONS) $\{\phi_j\}$ of eigenfunctions with non-increasing eigenvalues $\{\lambda_j\} \subset \mathbb{R}_+$ satisfying Equation (5). Each $\gamma \in L^2(\mathcal{X}, \mathrm{d}u)$ admits a unique expansion $\gamma = \sum_j a_j \phi_j$, with $\|\gamma\|_2^2 = \sum_j a_j^2 < \infty$. The image under $\mathscr{L}_K$ of the measure $\gamma(\mathrm{d}u) := \gamma(u)\,\mathrm{d}u$ with Lebesgue density function γ may be expressed as the L^2-convergent sum

$$\mathscr{L}_K[\gamma](x) = \sum_j \lambda_j a_j \phi_j(x).$$

Proposition 4 *For every $\gamma \in L^2(\mathcal{X}, \mathrm{d}u)$, $\mathscr{L}_K[\gamma] \in \mathcal{H}_K$ and*

$$\|\mathscr{L}_K[\gamma]\|_K^2 = \langle \mathscr{L}_K[\gamma], \gamma \rangle_2.$$

Consequently, $L^2(\mathcal{X}, \mathrm{d}u) \subset \mathscr{L}_K^{-1}(\mathcal{H}_K)$.

The following corollary illustrates that the space $L^2(\mathcal{X}, \mathrm{d}u)$ is too small for our purpose—that is, that important functions $f \in \mathscr{L}_K^{-1}(\mathcal{H}_K)$ fail to lie in $L^2(\mathcal{X}, \mathrm{d}u)$.

Corollary 5 *If the set $\Lambda := \{j : \lambda_j > 0\}$ is finite, then $\mathscr{L}_K(L^2(\mathcal{X}, \mathrm{d}u)) = \mathcal{H}_K$; otherwise $\mathscr{L}_K(L^2(\mathcal{X}, \mathrm{d}u)) \subsetneq \mathcal{H}_K$. The latter occurs whenever K is strictly positive definite and the RKHS is infinite-dimensional.*

Thus only for finite dimensional RKHS's is the space of square integrable functions sufficient to span the RKHS. In almost all interesting non-parametric statistics problems, the RKHS is infinite-dimensional.

Next we examine the space of integrable functions $L^1(\mathcal{X}, \mathrm{d}u)$, a larger space than $L^2(\mathcal{X}, \mathrm{d}u)$ when $\mathcal{X}$ is compact.

Proposition 6 *For every $\gamma \in L^1(\mathcal{X}, \mathrm{d}u)$, $\mathscr{L}_K[\gamma] \in \mathcal{H}_K$. Consequently, $L^1(\mathcal{X}, \mathrm{d}u) \subset \mathscr{L}_K^{-1}(\mathcal{H}_K)$.*

Another class of functions to be considered is the space of finite discrete measures,

$$\mathscr{M}_D = \left\{ \mu = \sum_j c_j \delta_{x_j} : \ \{c_j\} \subset \mathbb{R}, \ \{x_j\} \subset \mathcal{X}, \ \sum_j |c_j| < \infty \right\},$$

where δ_x is the Dirac measure supported at $x \in \mathcal{X}$ (the sum may be finite or infinite). This class will arise naturally when we examine Lévy and Dirichlet processes in Section 4.3.

Proposition 7 *For every $\mu \in \mathscr{M}_D$, $\mathscr{L}_K[\mu] \in \mathcal{H}_K$. Consequently, $\mathscr{M}_D \subset \mathscr{L}_K^{-1}(\mathcal{H}_K)$.*

By Proposition 6 and 7 the space $\mathscr{M}$ spanned by $L^1(\mathcal{X}, \mathrm{d}u) \cup \mathscr{M}_D$ is a subset of $\mathscr{L}_K^{-1}(\mathcal{H}_K)$. The range of $\mathscr{L}_K$ on just the elements of $\mathscr{M}_D$ with *finite* support is precisely $\mathcal{H}$, linear combinations of the $\left\{K_{x_j}\right\}_{x_j \in \mathcal{X}}$; thus

Proposition 8 *$\mathscr{L}_K(\mathscr{M})$ is dense in $\mathcal{H}_K$ with respect to the RKHS norm.*

Let $\mathscr{B}_+(\mathcal{X})$ denote the cone of all finite nonnegative Borel measures on $\mathcal{X}$ and $\mathscr{B}(\mathcal{X})$ the set of signed Borel measures. Clearly every $\mu \in \mathscr{B}(\mathcal{X})$ can be written uniquely as $\mu = \mu_+ - \mu_-$ with $\mu_+, \mu_- \in \mathscr{B}_+(\mathcal{X})$. The set $\mathscr{B} \backslash \mathscr{M}$ contains those Borel measures that are singular with respect to the Lebesgue measure. In this context, the set $\mathscr{M} = \mathscr{M}_D \cup L^1(\mathcal{X}, \mathrm{d}u)$ contains the Borel measures that can be used in practice. The above results, Propositions 6 and 4, also hold if we replace the Lebesgue measure with a Borel measure. It is natural to compare $\mathscr{B}(\mathcal{X})$ with $\mathscr{L}_K^{-1}(\mathcal{H}_K)$.

Proposition 9 *$\mathscr{B}(\mathcal{X}) \subset \mathscr{L}_K^{-1}(\mathcal{H}_K)$.*

We close this section by showing that even $\mathscr{B}(\mathcal{X})$ need not exactly characterize the class $\mathscr{L}_K^{-1}(\mathcal{H}_K)$. The proof of Proposition 6 implies that

$$\|\mathscr{L}_K[\gamma]\|_K^2 = \iint_{\mathcal{X} \times \mathcal{X}} K(x,u)\, \gamma(x)\, \gamma(u)\, \mathrm{d}x\, \mathrm{d}u. \tag{6}$$

¿From the above it is apparent that $\mathscr{L}_K[\gamma] \in \mathcal{H}_K$ holds only if $\mathscr{L}_K[\gamma]$ is well defined and the quantity on the right hand side of (6) is finite. If the kernel is smooth and vanishes at certain $x, u \in \mathcal{X}$, then (6) can be finite even if $\gamma \notin L^1(\mathcal{X}, \mathrm{d}u)$. For example in the case of polynomial kernels δ_x', the functional derivatives of the Dirac measure δ_x, are mapped into $\mathcal{H}_K$.

Proposition 10 *$\mathscr{B}(\mathcal{X}) \subsetneq \mathscr{L}_K^{-1}(\mathcal{H}_K(\mathcal{X}))$.*

Proof

We construct an infinite signed measure γ satisfying $\mathscr{L}_K[\gamma] \in \mathcal{H}_K$. As in Example 1 below, let

$$K(x,u) := x \wedge u - xu$$

be the covariance kernel for the Brownian bridge on the unit interval $\mathcal{X} = [0,1]$ (as usual, "$x \wedge u$" denotes the minimum of two real numbers x, u). Consider the improper $\mathbf{Be}(0,0)$ distribution

$$\gamma(\mathrm{d}u) = \frac{\mathrm{d}u}{u(1-u)},$$

with image under the integral operator

$$f(x) := \mathscr{L}_K[\gamma](x) = -x\log(x) - (1-x)\log(1-x).$$

The function $f(x)$ satisfies $f(0) = 0 = f(1)$ and has finite RKHS norm

$$\|f\|_K^2 = -2\int_0^1 \frac{\log(x)}{1-x}\,\mathrm{d}x = \frac{\pi^2}{3},$$

so $f(x)$ is in the the RKHS (see Example 1). Thus the infinite signed measure $\gamma(\mathrm{d}s)$ is in $\mathscr{L}_K^{-1}[\mathscr{H}_K]$ but not in $\mathscr{B}(\mathcal{X})$, so $\mathscr{L}_K^{-1}[\mathscr{H}_K]$ is larger than the space of finite signed measures. ■

3. Two Concrete Examples

In this section we construct two explicit examples to help illustrate the ideas of Section 2.

Example 1 (Brownian bridge) *On the space $\mathcal{X} = [0,1]$ consider the kernel*

$$K(x,u) := (x \wedge u) - xu,$$

which is jointly continuous and the covariance function for the Brownian bridge (Rogers and Williams, 1987, §IV.40) and hence a Mercer kernel. The eigenfunctions and eigenvalues of Equation (2) for Lebesgue measure $\mu(\mathrm{d}u) = \mathrm{d}u$ are

$$\lambda_j = \frac{1}{j^2\pi^2} \qquad \phi_j(x) = \sqrt{2}\sin(j\pi x).$$

The associate integral operator of Equation (4) is

$$\begin{aligned}\mathscr{L}_K[\gamma](x) &:= \int_{\mathcal{X}} K(x,u)\,\gamma(\mathrm{d}u) \\ &= (1-x)\int_{[0,x)} u\gamma(\mathrm{d}u) + x\int_{[x,1]}(1-u)\gamma(\mathrm{d}u),\end{aligned}$$

mapping any $\gamma(\mathrm{d}u) = \gamma(u)\mathrm{d}u$ with $\gamma \in L^1(\mathcal{X},\mathrm{d}u)$ to a function $f(x) = \mathscr{L}_K[\gamma](x)$ that satisfies the boundary conditions $f(0) = 0 = f(1)$ and, for almost every $x \in \mathcal{X}$,

$$\begin{aligned}f(x) &= (1-x)\int_0^x u\gamma(u)\mathrm{d}u + x\int_x^1 (1-u)\gamma(u)\mathrm{d}u, \\ f'(x) &= \int_x^1 \gamma(u)\,\mathrm{d}u - \int_0^1 u\gamma(u)\,\mathrm{d}u, \\ f''(x) &= -\gamma(x)\end{aligned}$$

and hence, by Equation (6) and integration by parts,

$$\begin{aligned}\|f\|_K^2 &= \int_0^1 f(x)\,\gamma(x)\,\mathrm{d}x \\ &= \int_0^1 -f(x)\,f''(x)\,\mathrm{d}x \\ &= \int_0^1 f'(x)^2\,\mathrm{d}x.\end{aligned}$$

Evidently the RKHS is just

$$\begin{aligned}\mathcal{H}_K &= \left\{ f(x) = \sum_{j=1}^{\infty} a_j \sqrt{2}\sin(j\pi x) \;\Big|\; \sum_{j=1}^{\infty} \pi^2 j^2 {a_j}^2 < \infty \right\} \\ &= \left\{ f \text{ in } L^2(\mathcal{X}, \mathrm{d}u) \big|\; f(0) = 0 = f(1) \text{ and } f' \in L^2(\mathcal{X}, \mathrm{d}u) \right\},\end{aligned}$$

the subspace of the Sobolev space $H_{+1}(\mathcal{X})$ satisfying Dirichlet boundary conditions (Mazja, 1985, Section 1.1.4), and

$$\begin{aligned}\mathcal{L}_K^{-1}\big(\mathcal{H}_K\big) &= \left\{ \gamma(x) = \sum_{j=1}^{\infty} a_j \sqrt{2}\sin(j\pi x) \;\Big|\; \sum_{j=1}^{\infty} \frac{{a_j}^2}{\pi^2 j^2} < \infty \right\} \\ &= \left\{ \gamma = f'' \big| f, f' \in L^2(\mathcal{X}, \mathrm{d}u),\ f(0) = 0 = f(1) \right\},\end{aligned}$$

a subspace of $H_{-1}(\mathcal{X})$.

Example 2 (Splines on a circle) *The kernel function for first order splines on the real line is*

$$K(x,u) := |x-u| \qquad x, u \in \mathbb{R}$$

and the corresponding RKHS norm is

$$\|f\|_K^2 = \int_{-\infty}^{\infty} f'(x)^2 \,\mathrm{d}x.$$

However, since the domain is not compact the spectrum of the associated integral operator on $L^2(\mathbb{R}, \mathrm{d}u)$ is continuous rather than discrete, the approach of Section 2 does not apply.

Instead we consider the case of splines with periodic boundary conditions. On the space $\mathcal{X} = [0,1]$ we consider the kernel function

$$\begin{aligned}K(x,u) &= \sum_{j=1}^{\infty} \frac{1}{2\pi^2 j^2} \cos(2\pi j |u-x|) \\ &= \frac{1}{2}\Big(|x-u| - \frac{1}{2}\Big)^2 - \frac{1}{24} \qquad 0 < x, u < 1\end{aligned}$$

The eigenfunctions and eigenvalues of Equation (2) for Lebesgue measure $\mu(\mathrm{d}u) = \mathrm{d}u$ are

$$\begin{array}{lcllcll} \phi_{2j-1}(x) & := & \sqrt{2}\sin(2\pi j x), & & \lambda_{2j-1} & = & \frac{1}{4\pi^2 j^2}, \\ \phi_{2j}(x) & := & \sqrt{2}\cos(2\pi j x) & , & \lambda_{2j} & = & \frac{1}{4\pi^2 j^2}, \end{array} \qquad j \in \mathbb{N}.$$

The RKHS norm for this kernel is

$$\|f\|_K^2 = \int_0^1 f'(x)^2 \,\mathrm{d}x$$

and the RKHS is

$$\mathcal{H}_K = \left\{ f(x) = \sum_{j=1}^{\infty} \sqrt{2}\big[a_j \sin(2\pi j x) + b_j \cos(2\pi j x)\big] \;\Big|\; \sum_{j=1}^{\infty} 4\pi^2 j^2 \,(a_j^2 + b_j^2) < \infty \right\}$$

the subspace of the Sobolev space $H_{+1}(\mathcal{X})$ satisfying periodic boundary conditions and orthogonal to the constants (Wahba, 1990, Section 2.1) and

$$\mathscr{L}_K^{-1}\left(\mathcal{H}_K\right) = \left\{ \gamma(x) = \sum_{j=1}^{\infty} \sqrt{2}\left[a_j \sin(\pi j x) + b_j \cos(\pi j x)\right] \;\middle|\; \sum_{j=1}^{\infty} \frac{a_j^2 + b_j^2}{4 j^2 \pi^2} < \infty \right\},$$

a subspace of $H_{-1}(\mathcal{X})$.

Elements in either RKHS given in the above two examples with a finite representation

$$f(x) = \sum_{i=1}^{m} c_i K(x, x_i), \;\; m < \infty$$

are splines. For the first example these functions are linear splines that vanish at $\{0,1\}$. In the second example if the coefficients sum to zero ($\sum_{i=1}^{m} c_i = 0$), then these functions are linear splines with periodic boundary conditions. If the coefficients do not sum to zero then they are quadratic splines with periodic boundary conditions.

4. Bayesian Kernel Models

Our goal from Section 1 is to present a coherent Bayesian framework for non-parametric function estimation in a RKHS. Suppose we observe data (with noise), $\{(x_i, y_i)\} \subset \mathcal{X} \times \mathbb{R}$ from the linear regression model

$$y_i = f(x_i) + \varepsilon_i \tag{7}$$

where we assume $\{\varepsilon_i\}$ are independent $\mathbf{No}(0, \sigma^2)$ random variables with unknown variance σ^2, and $f(\cdot)$ is an unknown function we wish to estimate. For a fixed kernel we assume $f \in \mathcal{H}_K$. Recall that the integral operator $\mathscr{L}_K$ maps $\mathscr{M}(\mathcal{X})$ into $\mathcal{H}_K$ and in particular $\mathscr{L}_K(\mathscr{M}(\mathcal{X}))$ is dense in $\mathcal{H}_K$. Therefore, we assume that

$$f(x) = \int_{\mathcal{X}} K(x, u) Z(du) \tag{8}$$

where $Z(du) \in \mathscr{M}(\mathcal{X})$ is a signed measure on $\mathcal{X}$. If we put a prior on $\mathscr{M}(\mathcal{X})$, we are in essence putting a prior on the functions $f \in \mathcal{G}$.

Our measurement error model (7) gives us the following likelihood for the data $D := \{(x_i, y_i)\}_{i=1}^n$

$$L(D|Z) \propto \prod_{i=1}^{n} \exp\left[-\frac{1}{2\sigma^2} \left(y_i - f(x_i)\right)^2 \right]. \tag{9}$$

With a prior distribution on Z, $\pi(Z)$, we can obtain the posterior density function given data

$$\pi(Z|D) \propto L(D|Z)\, \pi(Z), \tag{10}$$

which implies a posterior distribution for f via the integral operator (8).

4.1 Priors on $\mathscr{M}$

A random signed measure $Z(du)$ on $\mathcal{X}$ can be viewed as a stochastic process on $\mathcal{X}$. Therefore the practice of specifying a prior on $\mathscr{M}(\mathcal{X})$ via a stochastic process is ubiquitous in non-parametric Bayesian analysis. Gaussian processes and Dirichlet processes are two commonly used stochastic processes to generate random measures.

We first apply the results of Section 2 to Gaussian process priors (Rasmussen and Williams, 2006, Section 6) and then to Lévy process priors (Wolpert et al., 2003; Tu et al., 2006). We also remark that Dirichlet processes can be constructed from Lévy process priors.

4.2 Gaussian Processes

Gaussian processes are canonical examples of stochastic processes used for generating random measures. They have been used extensively in the machine learning and statistics community with promising results in practice and theory (Kimeldorf and Wahba, 1971; Chakraborty et al., 2005; Rasmussen and Williams, 2006; Ghosal and Roy, 2006).

We consider two modeling approaches using Gaussian process priors:

i. Model I: Placing a prior directly on the space of functions $f(x)$ by sampling from paths of the Gaussian process with its covariance structure defined via a kernel K;

ii. Model II: Placing a prior on the random signed measures $Z(du)$ on $\mathcal{X}$ by using a Gaussian process prior for $Z(du)$ which implies a prior on the function space defined by the kernel model in Equation (8).

For both approaches we can characterize the function space spanned by the kernel model. The first approach is the more standard approach for non-parametric Bayesian inference using Gaussian processes while the later is an example of our Bayesian kernel model. However, as pointed out by (Wahba, 1990, Section 1.4) the random functions from the first approach will be almost surely outside the RKHS induced by the kernel. However these functions will be contained in a larger RKHS, as we show in the next section.

We first state some classical results on the sample paths of Gaussian processes. We then use these properties and the results of Section 2 to characterize the function spaces of the two models.

4.2.1 SAMPLE PATHS OF GAUSSIAN PROCESSES

Consider a Gaussian process $\{Z_u,\ u \in \mathcal{X}\}$ on a probability space $\{\Omega, \mathcal{A}, \mathbf{P}\}$ having covariance functions determined by a kernel function K. Let $\mathcal{H}_K$ be the corresponding RKHS and let the mean m be contained in the RKHS, $m \in \mathcal{H}_K$. Then the following zero-one law holds:

Theorem 11 *(Kallianpur 1970, Theorem 5.1) If $Z_\bullet \equiv \{Z_u, u \in \mathcal{X}\}$ is a Gaussian process with covariance K and mean $m \in \mathcal{H}_K$, and $\mathcal{H}_K$ is infinite dimensional, then*

$$\mathbf{P}(Z_\bullet \in \mathcal{H}_K) = 0.$$

The probability measure is assumed to be complete.

Thus the sample paths of the Gaussian process are almost surely outside $\mathcal{H}_K$. However, there exists a RKHS $\mathcal{H}_R$ that is bigger than $\mathcal{H}_K$ that contains the sample paths almost surely. To construct such an RKHS we first need to define nuclear dominance.

Definition 12 *Given two kernel functions R and K, R dominates K (written as $R \succ K$) if $\mathcal{H}_K \subseteq \mathcal{H}_R$.*

Given the above definition of dominance the following operator can be defined:

Theorem 13 *(Lukić and Beder, 2001) Let $R \succ K$. Then*

$$\|g\|_R \leq \|g\|_K, \ \forall g \in \mathcal{H}_K.$$

There exists a unique linear operator $L : \mathcal{H}_R \to \mathcal{H}_R$ whose range is contained in $\mathcal{H}_K$ such that

$$\langle f, g \rangle_R = \langle Lf, g \rangle_K, \ \forall f \in \mathcal{H}_R, \forall g \in \mathcal{H}_K.$$

In particular

$$LR_u = K_u, \ \forall u \in \mathcal{X}.$$

As an operator into $\mathcal{H}_R$, L is bounded, symmetric, and positive.

Conversely, let $L : \mathcal{H}_R \to \mathcal{H}_R$ be a positive, continuous, self-adjoint operator then

$$K(s,t) = \langle LR_s, R_t \rangle_R, \ s,t \in \mathcal{X}$$

defines a reproducing kernel on $\mathcal{X}$ such that $K \leq R$.

L is the dominance operator of $\mathcal{H}_R$ over $\mathcal{H}_K$ and this dominance is called nuclear if L is a nuclear or trace class operator (a compact operator for which a trace may be defined that is finite and independent of the choice of basis). We denote nuclear dominance as $R \gtrdot K$.

4.2.2 IMPLICATIONS FOR THE FUNCTION SPACES OF THE MODELS

Model I placed a prior directly on the space of functions using sample paths from the Gaussian process with covariance structure defined by the kernel K. Theorem 11 states that sample paths from this Gaussian process are not contained in $\mathcal{H}_K$. However, there exists another RKHS $\mathcal{H}_R$ with kernel R which does contain the sample path if R has nuclear dominance over K.

Theorem 14 *(Lukić and Beder, 2001) Let K and R be two reproducing kernels. Assume that the RKHS $\mathcal{H}_R$ is separable. A necessary and sufficient condition for the existence of a Gaussian process with covariance K and mean $m \in \mathcal{H}_R$ and with trajectories in $\mathcal{H}_R$ with probability 1 is that $R \gtrdot K$.*

The implication of this theorem is that we can find a function space $\mathcal{H}_R$ that contains functions generated by the Gaussian process defined by covariance function K.

Model II places a prior on random signed measures $Z(du)$ on $\mathcal{X}$ by using a Gaussian process prior for $Z(du)$. This implies a prior of the space of functions spanned by the kernel model in Equation (8). This space $\mathcal{G}$ is contained in $\mathcal{H}_K$ by our results in Section 2. This is due to the fact that any sample path from a continuous Gaussian process on a compact domain $\mathcal{X}$ is in L^1 and therefore the corresponding function from the integral (8) is still in $\mathcal{H}_K$.

4.3 Lévy Processes

Lévy processes offer an alternative to Gaussian processes in non-parametric Bayesian modeling. Dirichlet processes and Gaussian processes with a particular covariance structure can be formulated from the framework of Lévy processes. For the sake of simplicity in exposition, we will use the univariate setting $\mathcal{X} = [0,1]$ to illustrate the construction of random signed measures using Lévy processes. The extension to the multivariate setting is straightforward and outlined in Appendix B.

A stochastic process $Z := \{Z_u \in \mathbb{R} : u \in \mathcal{X}\}$ is called a *Lévy process* if it satisfies the following conditions:

1. $Z_0 = 0$ almost surely.
2. For any integer $m \in \mathbb{N}$ and any $0 = u_0 < u_1 < ... < u_m$, the random variables $\{Z_{u_j} - Z_{u_{j-1}}\}$, $1 \leq j \leq m$ are independent. (Independent increments property)
3. The distribution of $Z_{s+u} - Z_s$ does not depend on s (Temporal homogeneity or stationary increments property).
4. The sample paths of Z are almost surely right continuous and have left limits, that is, are "càdlàg".

Familiar examples of Lévy processes include Brownian motion, Poisson processes, and gamma processes. The following celebrated theorem characterizes Lévy processes.

Theorem 15 *(***Lévy-Khintchine***) Z is a Lévy process if and only if the characteristic function of* $Z_u : u \geq 0$ *has the following form:*

$$\mathbb{E}[e^{i\lambda Z_u}] = \exp\left\{u\left[i\lambda a - \frac{1}{2}\sigma^2\lambda^2 + \int_{\mathbb{R}\setminus 0}[e^{i\lambda w} - 1 - i\lambda w 1_{\{w:|w|<1\}}(w)]\nu(dw)\right]\right\}, \tag{11}$$

where $a \in \mathbb{R}$, $\sigma^2 \geq 0$ *and* ν *is a nonnegative measure on* $\mathbb{R}\setminus 0$ *with*

$$\int_{\mathbb{R}\setminus 0}(1 \wedge |w|^2)\nu(dw) < \infty. \tag{12}$$

Note that (11) can be written as a product of two components,

$$\exp\left\{iau\lambda - \frac{u\sigma^2}{2}\lambda^2\right\} \times \exp\left\{u\int_{\mathbb{R}\setminus 0}\left[e^{i\lambda w} - 1 - i\lambda w 1_{\{w:|w|<1\}}(w)\right]\nu(dw)\right\},$$

the characteristic functions of a Gaussian process and of a partially compensated Poisson process, respectively. This observation is the essence of the Lévy-Itô theorem (Applebaum, 2004, Theorem 2.4.16), which asserts that every Lévy process can be decomposed into the sum of two independent components: a "continuous process" (Brownian motion with drift) and a (possibly compensated) "pure jump" process. The three parameters (a, σ^2, ν) in (11) uniquely determine a Lévy process where a denotes the drift term, σ^2 denotes the variance (diffusion coefficient) of the Brownian motion, and $\nu(dw)$ denotes the intensity of the jump process. The so-called "Lévy measure" ν need not be finite, but (12) implies that $\nu[(-\varepsilon,\varepsilon)^c] < \infty$ for each $\varepsilon > 0$ and so ν is at least sigma-finite.

4.3.1 PURE JUMP LÉVY PROCESSES

Pure jump Lévy processes are used extensively in non-parametric Bayesian statistics due to their computationally amenability. In this section we first state an interpretation of these processes using Poisson random fields. We then describe Dirichlet and symmetric α-stable processes.

4.3.2 POISSON RANDOM FIELDS INTERPRETATION

Any pure jump Lévy process Z has a nice representation via a Poisson random field. Set $\Delta Z_u := Z_u - \lim_{s\uparrow u} Z_s$, the jump size at the location u. Set $\Gamma = \mathbb{R}\times\mathcal{X}$, the Cartesian product of $\mathbb{R}$ with $\mathcal{X}$. For any sets $A\subset\mathbb{R}\backslash 0$ bounded away from zero and $B\subset\mathcal{X}$ we can define the counting measure

$$N(A\times B) := \sum_{s\in B} 1_A\big(\Delta Z_s\big). \tag{13}$$

The measure N defined above turns out to be a Poisson random measure on Γ, with mean measure $\nu(dw)du$ where du is the uniform reference measure on $\mathcal{X}$ (for instance the Lebesgue measure when $\mathcal{X}=[0,1]$). For any $E\subset\Gamma$ with $\mu = \int_E \nu(dw)du < \infty$ the random variable $N(E)$ has a Poisson distribution with intensity μ.

When ν is a finite measure, the total number of jumps $J\in\mathbb{N}$ of the process follows a Poisson distribution with finite intensity $\mu(\Gamma)$. When Z has a density with respect to the Lévy random field M with Lévy measure m, Z_u has finite total variation and determines a finite measure $Z(du) = dZ_u$. In this case, any realization of $Z(du)$ can be formulated as

$$Z(du) = \sum_{j=1}^{J} w_j \delta_{u_j}, \tag{14}$$

where $(w_j,u_j)\in\Gamma$ are *i.i.d.* draws from $\nu(dw)du$ representing the jump size and the jump location, respectively. Given a realization of $Z(du) = \{u_j,w_j\}_{j=1}^J$, Equation (8) reduces to

$$\int_{\mathcal{X}} K(x,u)Z(du) = \int_{\Gamma} K(x,u)N(dwdu) = \sum_{j=1}^{J} w_j K(x,u_j),$$

where $N(dwdu)$ is a Poisson random measure as defined by (13). Then the likelihood for the data $D := \{(x_i,y_i)\}_{i=1}^n$ is given by

$$L(D|Z) \propto \prod_{i=1}^{n} \exp\Big[-\frac{1}{2\sigma^2}\Big(y_i - \sum_{j=1}^{J} w_j K(x_i,u_j)\Big)^2\Big].$$

If the measure $\nu(dw)du$ has a density function $\nu(w,u)$ with respect to some finite reference measure $m(dwdu)$, then the prior density function for Z with respect to a Lévy(m) process is

$$\pi(Z) = \Big[\prod_{j=1}^{J} \nu(w_j,u_j)\Big] e^{m(\Gamma)-\nu(\Gamma)}. \tag{15}$$

Using Bayes' theorem, we can calculate the posterior distribution for Z via (10).

When ν is an infinite measure the number of jumps in the unit interval is countably infinite almost surely. However, if the Lévy measure satisfies

$$\int_{\mathbb{R}} (1\wedge|w|)\nu(dw) < \infty, \tag{16}$$

then the sequence $\{w_j\}$ is almost surely absolutely summable (i.e, $\sum_{j=1}^{\infty}|w_j|<\infty$ a.s.) and we can still represent the process Z via the summation (14). Note that condition (16) is stronger than the integrability condition (12) in the Lévy-Khintchine theorem. This allows for the existence of Lévy processes with jumps that are not absolutely summable.

4.3.3 DIRICHLET PROCESS

The Dirichlet process is commonly used in non-parametric Bayesian analysis (Ferguson, 1973, 1974) mainly due to its analytical tractability. When passing from prior to posterior computations, it has been shown that the Dirichlet process is the only conjugate member of the whole class of normalized random measures with independent increments (James et al., 2005) so the posterior can be efficiently computed. Recently it has received much attention in the machine learning literature (Blei and Jordan, 2006; Xing et al., 2004, 2006). Though Dirichlet processes are often defined via Dirichlet distributions, they can also be defined as a normalized Gamma process as noted by Ferguson (1973). A Gamma process is a pure jump Lévy process, which has the Lévy measure

$$\nu(dw)=aw^{-1}\exp\{-bw\}dw,\quad w>0,$$

so at each location u $Z_u\sim\text{Gamma}(au,b)$. Suppose Z_u is a Gamma$(a,1)$ process defined on $\mathcal{X}=[0,1]$, then

$$\tilde{Z}_u=Z_u/Z_1$$

is the $\mathbf{DP}(a\,\mathrm{d}u)$ Dirichlet process. Since the Dirichlet process is a random measure on probability distribution functions, it can be used when the target function $f(x)$ is a probability density function. Dirichlet processes can also be used to model a general smooth function $f(x)$ in combination with other random processes. For example, Liang et al. (2007) and Liang et al. (2006) consider a variation of the integral (8)

$$f(x)=\int_{\mathcal{X}}K(x,u)\,Z(du)=\int_{\mathcal{X}}w(u)\,K(x,u)\,F(du),\tag{17}$$

where the random signed measure $Z(du)$ is modeled by a random probability distribution function $F(du)$ and random coefficients $w(u)$. A Dirichlet process prior is specified for F and a Gaussian prior distribution is specified for w.

4.3.4 SYMMETRIC α-STABLE PROCESS

Symmetric α-stable processes are another class of Lévy processes, arising from symmetric α-stable distributions. The symmetric α-stable distribution has the following characteristic function:

$$\varphi(\eta)=\exp(-\gamma|\eta|^{\alpha}),$$

γ is the dispersion parameter, and $\alpha\in(0,2]$ is the characteristic exponent. The case, when $\gamma=1$ is called the standard symmetric α-stable (SαS) distribution. It has the following Lévy measure

$$\nu(dw)=\frac{\Gamma(\alpha+1)}{\pi}\sin\left(\tfrac{\pi\alpha}{2}\right)|w|^{-1-\alpha}dw\qquad\alpha\in(0,2].$$

Two important cases of SαS distributions are the Gaussian when $\alpha=2$ and the Cauchy when $\alpha=1$. Thus SαS processes allow us to model heavy or light tail processes by varying α. One can verify that the Lévy measure is infinite for $0<\alpha\leq 2$ since $\nu(\mathbb{R})=\int_{\mathbb{R}}\nu(dw)=2\int_{(0,\infty]}\alpha w^{-1-\alpha}\,dw=\infty$.

Hence the process has an infinite number of jumps in any finite time interval. However by a limiting argument, we can ignore the jumps of negligible size (say $< \varepsilon$). Hence our space reduces to

$$\Gamma_\varepsilon = (-\varepsilon, \varepsilon)^c \times [0,1].$$

Given the jumps sizes $\{w_j\}$, jump locations $\{u_j\}$, and the number of jumps J, the prior probability density function (15) is

$$\pi(Z) = \left[\Pi_{j=1}^J |w_j|\right]^{1-\alpha} e^{2(\varepsilon^{-1}-\varepsilon^{-\alpha})}\alpha^J, \quad |w_j| \geq \varepsilon \tag{18}$$

with respect to a Cauchy random field.

Using this prior is essentially the same as using a penalty term in a regularization approach. For the SαS process, we have

$$\log \pi(Z) \propto J \log \alpha + (1-\alpha)\Big(\sum_j \log |u_j| 1_{|u_j|>\varepsilon}\Big) + \text{constant}.$$

The first term is an AIC like penalty for the number of knots J and the second term is a LASSO-type penalty in log-scale. There is also a hidden penalty which shrinks all the coefficients with magnitude less than ε to zero.

4.4 Computational and Modeling Considerations

The computational and modeling issues involved in choosing process priors, especially in high dimensional settings, are at the heart of non-parametric Bayesian modeling. In this section we discuss these issues for the models discussed in the previous section.

A main challenge with Gaussian process models is that a finite dimensional representation of the sample path is required for computation. For low dimensional problems (say $d \leq 3$), a reasonable approach is to place a grid on $\mathcal{X}$. Then we can approximate a continuous process Z by its values on the finitely many points $\{u_j\}_{j=1}^m$ on the grid. Using this approximation, our kernel model (8) can be written as

$$f(x) = \sum_{j=1}^m w_j K(x, u_j),$$

and the implied prior distribution on $(w_1, \ldots, w_m)$ is a multivariate normal with mean and covariance structure as defined by the kernel K evaluated at points $\{u_j\}$. For low-dimensional data a grid can be placed on the input space. However, this approach is not practical in higher dimensions. This issue is addressed in Gaussian process regression models by evaluating the function at the training and future test data points. This corresponds to a fixed design setting. It is important to note however, that the prior being sampled in this model is not over $\mathcal{X}$ but the restriction of $\mathcal{X}$ to the data. Both the direct model and the kernel model will face this computational consideration and thus the computational cost will not differ significantly between models.

For pure jump processes discretization is not the bottleneck. The nature of the pure jump process ensures that the kernel model will have discrete knots. The key issue in using a pure jump processes to model multivariate data is that the knots of the model should be representative of samples drawn from the marginal distribution of the data ρ_x. This is a serious computational as well as modeling challenge, it is obvious that independently sampling each dimension will typically not be a good

idea either in terms of computational time or modeling accuracy. In Section 5.2 we provide a kernel model that addresses this issue.

A theoretical and empirical comparison of the accuracy of the various process priors on a variety of function classes and data sets would be of interest, but is beyond the scope of this paper. Due to the extensive literature on Gaussian process models from theoretical as well as practical perspectives (Rasmussen and Williams, 2006; Ghosal and Roy, 2006) our simulations will focus on two pure jump process models.

5. Posterior Inference

For the case of regression our model is

$$y_i = f(x_i) + \varepsilon_i \qquad \text{for } x_i \in \mathcal{X}$$

with $\{\varepsilon_i\}$ as normal independent random variables and the unknown regression function f (which is assumed to be in $\mathcal{H}_K$) is modeled as

$$f(x) = \int_{\mathcal{X}} K(x,u) Z(du).$$

In the case of binary regression we can use a probit model

$$\mathbf{P}(y_i = 1 | x_i) = \Phi[f(x_i)],$$

where $\Phi[\cdot]$ is the cumulative distribution function of the standard normal distribution.

In Section 4, we discussed specifying a prior on $\mathcal{H}_K$ via the random measure $Z(du)$. The observed data add to our knowledge of both the "true function" $f(\cdot)$ and the distribution of $Z(du)$. This information is used to update the prior and obtain the posterior density $\pi(Z|D)$. For pure jump measures $Z(du)$ and most non-parametric models this update is computationally difficult because there is no closed-form expression for the posterior distribution. However, Markov chain Monte Carlo (MCMC) methods can be used to simulate the posterior distribution.

We will apply a Dirichlet process model to a high-dimensional binary regression problem and illustrate the use of Lévy process models on a univariate regression problem.

5.1 Lévy Process Model

Posterior inference for Lévy random measures have been less explored than Dirichlet and Gaussian processes. Wolpert et al. (2003) is a recent comprehensive reference on this topic. We use the methodology developed in this work for our model.

The random measure $Z(du)$ is given by

$$Z(du) \sim \text{Lévy}(\nu(dw)du)$$

where

$$\nu(dw) = \frac{\Gamma(\alpha+1)}{\pi} \sin\left(\tfrac{\pi\alpha}{2}\right) |w|^{-1-\alpha} \, 1_{\{w:|w|>\varepsilon\}} \, dw \qquad \alpha \in (0,2]$$

is the Lévy measure (truncated) for the SαS process. As explained in Section 4.3.4, since $\nu(dw)$ is not a finite measure on $\mathbb{R}$, we ignore jumps of size smaller than ε. Any realization of the random

measure $Z(du)$ is an element of the parameter space Θ

$$\Theta := \bigcup_{J=0}^{\infty} \Big((-\varepsilon, \varepsilon)^c \times [0,1] \Big)^J$$

with the prior probability density function given by Equation(18), with respect to a Cauchy random field.

5.1.1 TRANSITION PROBABILITY PROPOSAL

In this section, we describe an MCMC algorithm to simulate from Θ according to the posterior distribution. We construct an irreducible transition probability distribution $Q(d\theta^*|\theta)$ on the parameter space Θ such that the stationary distribution of the chain will be the posterior distribution.

Two different realizations from the parameter space Θ may not have the same number of jumps. Hence the number of jumps J is modeled a birth-death process. At any iteration step t the parameter space consists of J jump locations $\{u_j\}$ of size $\{w_j\}$, $\theta_t = \{w_j, u_j\}_{j=1}^{J}$. The (weighted) transition probability algorithm, Algorithm 1, computes the weighted transition probability to a new state θ^* given the current state θ.

Algorithm 1: Weighted transition probability algorithm $Q(\theta)$.

input : $0 < p_b, p_d < 1, \tau > 0$, current state $\theta \in \Theta$

return: proposed new state θ^* and its weighted transition probability $Q(\theta^*|\theta)\pi(\theta)$

Draw $t \sim U[0,1]$;
if $t < 1 - p_b$ **then**
  draw uniformly $j \in \{1,...,J\}$; draw $\gamma_1, \gamma_2 \sim \mathbf{No}(0, \tau^2)$;
  $w_* \leftarrow w_j + \gamma_1$; $u_* \leftarrow u_j + \gamma_2$;
  if *(*$|w_*| < \varepsilon$ **or** $t < p_d$*)* **then**
    $J \leftarrow J - 1$; delete (w_j, u_j);
    $Q(\theta^*|\theta)\pi(\theta) \leftarrow \frac{(J+1)p_b}{2\varepsilon^{-\alpha}\Big((1-p_b-p_d)\Big[\Phi(\frac{w_j+\varepsilon}{\tau}) - \Phi(\frac{w_j-\varepsilon}{\tau})\Big] + p_d\Big)}$;
  else
    $Q(\theta^*|\theta)\pi(\theta) \leftarrow \left|\frac{w_*}{w_j}\right|$; $w_j \leftarrow w_*$; $u_j \leftarrow u_*$;
else
  $J \leftarrow J + 1$; $u_J \sim U[X]$; $w_J \sim$ Birth;
  $Q(\theta^*|\theta)\pi(\theta) \leftarrow \frac{2\varepsilon^{-\alpha}\Big((1-p_d-p_b)\Big[\Phi(\frac{w_J+\varepsilon}{\tau}) - \Phi(\frac{w_J-\varepsilon}{\tau})\Big] + p_d\Big)}{p_b J}$;

In the above algorithm, $\mathbf{No}(0, \tau^2)$ denotes the normal distribution with mean 0 and variance τ^2 and $\Phi(\cdot)$ denotes the distribution function of the standard normal distribution. The variables (p_b, p_d) stand for probability of birth step and death step respectively. There is an implicit update step, where a chosen point(u_j) is 'updated' with another point(u_*) with probability $1 - p_b - p_d$. In the birth step, a new point is sampled according to the density

$$\frac{\alpha |w|^{-1-\alpha}}{2\varepsilon^{-\alpha}} \quad \varepsilon > 0.$$

5.1.2 THE MCMC ALGORITHM

The MCMC algorithm, Algorithm 2, simulates draws from the posterior distribution. This is done by Metropolis-Hastings sampling using the weighted transition probability algorithm above to generate a Markov chain whose equilibrium density is the posterior density.

Algorithm 2: MCMC algorithm

input : data D, number of iterations T, weighted transition probability algorithm $Q(\theta)$

return: parameters drawn from the posterior $\{\theta_i\}_{i=1}^{T}$

$J \sim \mathbf{Po}(2\varepsilon^{-\alpha})$; `// initialize J`
for $j \leftarrow 1$ **to** J **do**
  `// initialize` $\theta(0)$
  $u_j \sim U[\mathcal{X}]$; $w_j \sim$ Birth;

for $t \leftarrow 1$ **to** T **do**
  `//` t`-th iteration of the Markov chain`
  $\{\theta_*, Q(\theta_*|\theta_t)\pi(\theta_t)\} \leftarrow Q(\theta(t))$; `// call the weighted transition probability algorithm`
  $\log\pi(\theta_*|D) - \log\pi(\theta_t|D) = \log\frac{L(D|\theta_*)}{L(D|\theta_t)} + \log\frac{\pi(\theta_*)}{\pi(\theta_t)}$;
  $\zeta_* \leftarrow \log\pi(\theta_*|D) + \log Q(\theta_t|\theta_*) - \log\pi(\theta_t|D) - \log Q(\theta_*|\theta_t)$; `// the Metropolis-Hastings log acceptance probability`
  $e \sim \mathbf{Ex}(1)$;
  if $e + \zeta_{t+1} > 0$ **then** $\theta_{t+1} \leftarrow \theta_*$ **else** $\theta_{t+1} \leftarrow \theta_t$;

The MCMC algorithm will provide us with T realizations of the jump parameters $\{\theta_t\}_{t=1}^{T}$. We assume that the chain reaches its stationary distribution after b iterations ($b \ll T$). For each of the $T-b$ realizations, we have a corresponding function

$$\hat{f}_t(x) = \sum_{i=1}^{J_t} w_{it} K(x, u_{it}),$$

where for the t-th realization J_t is the number of jumps, w_{it} is the magnitude of the i-th jump, and u_{it} is the position of the i-th jump. Point estimates can be made by averaging $\hat{f}$ and credible intervals can be computed from the distribution of $\hat{f}$ to provide an estimate of uncertainty.

5.1.3 ILLUSTRATION ON SIMULATED DATA

Data is generated from a noisy sinusoid

$$f(x_i) = \sin(2\pi x_i) + \varepsilon_i \text{ for } x \in [0,1], \tag{19}$$

with $\varepsilon_i \overset{iid}{\sim} \mathbf{No}(0, .01)$, $\{x_i\}_{i=1}^{100}$ points equally spaced in $[0,1]$, and $\{y_i\}_{i=1}^{100}$ are computed by Equation (19). We applied the SαS model with $\alpha = 1.5$ and a Gaussian kernel $K(x,u) = \exp\{(x-u)^2\}$

to this data. We set $\varepsilon = 0.01$ and $(p_b, p_u, p_d) = (0.4, 0.2, 0.4)$, in algorithms 1 and 2. In Figure 1a-d we plot the target sinusoid, the function realized at an iteration t of the Markov chain, and the jump locations and magnitudes of the random measure. In Figure 1e,f we provide a plot of the target function, realization of the data, and the 95% point-wise credible band—the 95% credible interval at each point x_i.

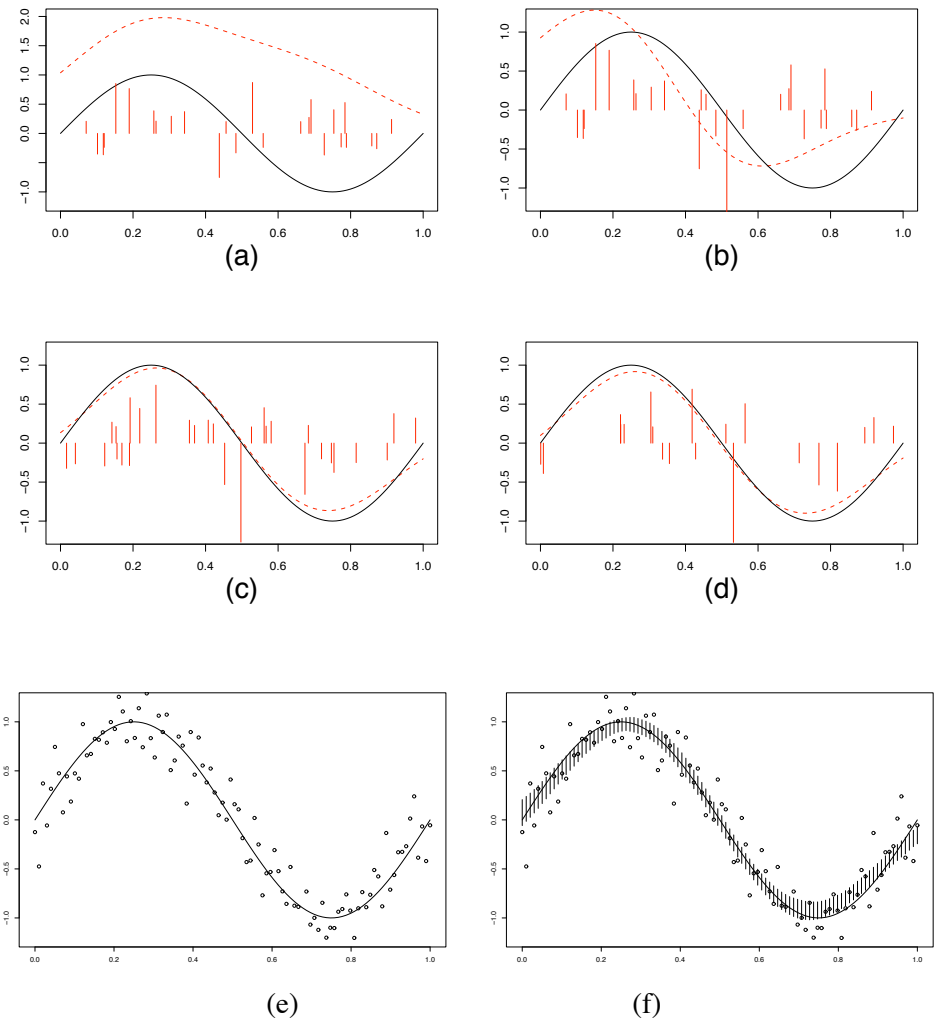


Figure 1: Plots of the target sinusoid (solid line), the function realized at an iteration t of the Markov chain (dashed line), and the jump locations and magnitudes of the measure (spikes) for (a) $t = 1$, (b) $t = 10$, (c) $t = 5 \times 10^3$, and (d) $t = 10^4$. (e) A realization of the simulated data (circles) and the underlying target sinusoid (solid line). (f) The 95% point-wise credible band for the data and the target sinusoid.

5.2 Classification of Gene Expression Data

For Dirichlet processes there is extensive literature on exact posterior inference using MCMC methods (West, 1992; Escobar and West, 1995; MacEachern and Müller, 1998; Müller et al., 2004) as well as work on approximate inference using variational methods (Blei and Jordan, 2006). Recently Dirichlet process priors have been applied to a Bayesian kernel model for high dimensional data. For example in Liang et al. (2006) and Liang et al. (2007) the Bayesian kernel model was used to classify gene expression data as well as digits, the MNIST database. We apply this model to gene expression data consisting of microarray gene expression profiles from 190 cancer samples and 90 normal samples (Ramaswamy et al., 2001; Mukherjee et al., 2003), over 16,000 genes.

The model is based upon the integral operator given in Equation (17)

$$f(x) = \int_{\mathcal{X}} K(x,u)\, Z(du) = \int_{\mathcal{X}} w(u)\, K(x,u)\, F(du),$$

where the random signed measure $Z(du)$ is modeled by a random probability distribution function $F(du)$ and a random weight function $w(u)$. We assume that the support of $Z(du)$ and $w(u)F(du)$ are equal. A key point in our model will be that if our estimate of F is discrete and puts masses w_i at support points (or "knots") u_i, then the expression for $f(\cdot)$ is simply

$$f(x) = \sum_i w(u_i)\, K(x,u_i).$$

The above model, in which basis functions are placed at random locations and a joint distribution is specified for the coefficients, has been considered previously in the literature (see Neal, R. M. 1996 and Liang et al. 2007). In Liang et al. (2007) uncertainty about F is expressed using a Dirichlet process prior, $\mathrm{Dir}(\alpha, F_0)$. The posterior after marginalization is also a Dirichlet distribution and given data $(x_1,\ldots,x_n)$ the posterior will have the following representation (Liang et al., 2007, 2006)

$$\hat{f}(x) = \frac{\alpha}{\alpha+n} \int w(u) K(x,u) F_0(du) + \frac{1}{\alpha+n} \sum_{i=1}^{n} w(x_i) K(x,x_i),$$

which can be approximated by the following discrete summation

$$\hat{f}(x) \approx \sum_{i=1}^{n} w_i K(x,x_i) \tag{20}$$

when $\frac{\alpha}{n}$ is small and $w_i = \frac{w(x_i)}{\alpha+n}$. We specify a mixture-normal prior on the coefficients w_i as in Liang et al. (2007) and use the same MCMC algorithm to simulate the posterior.

Note that although Equation (20) has the same form as the representer theorem, it is derived from a very different formulation. In fact, when there is unlabeled data available—$(x_{n+1},\ldots,x_{n+m})$ drawn from the margin ρ_X—our model has the following discrete representation

$$\hat{f}(x) = \sum_{i=1}^{n} w_i K(x,x_i) + \sum_{i=1}^{m} w_{i+n} K(x,x_{i+n}),$$

where $w_\ell = \frac{w(x_\ell)}{\alpha+m+n}$. The above form is identical to the one obtained via the manifold regularization framework (Belkin and Niyogi, 2004; Belkin et al., 2006). The two derivations are from different

perspectives. This simple incorporation of unlabeled data into the model further illustrates the advantage of placing the prior over random measures in the Bayesian kernel model.

In our experiments we first applied a standard variation filter to reduce the number of genes to $p = 2800$. We then randomly assigned 20% of the samples from the cancer and normal groups to training data and use the remaining 80% as test data. We used a linear kernel in the model and we used the classification model detailed in Liang et al. (2007).

We performed two analyses on this data:

Analysis I—The training data were used in the model and the posterior probability was simulated for each point in the test set. A linear kernel was used.

Analysis II—The training and unlabeled test data were used in the model and the posterior probability was simulated for each point in the test set. A linear kernel was used.

The classification accuracy for Analyses I and II were 73% and 85%, respectively. The accuracy of the predictive models in Analysis I is comparable to that obtained for support vector machines in Mukherjee et al. (2003). Figure 2 displays boxplots of the posterior mean of the 72 the normal and 152 cancer samples for the two analyses.

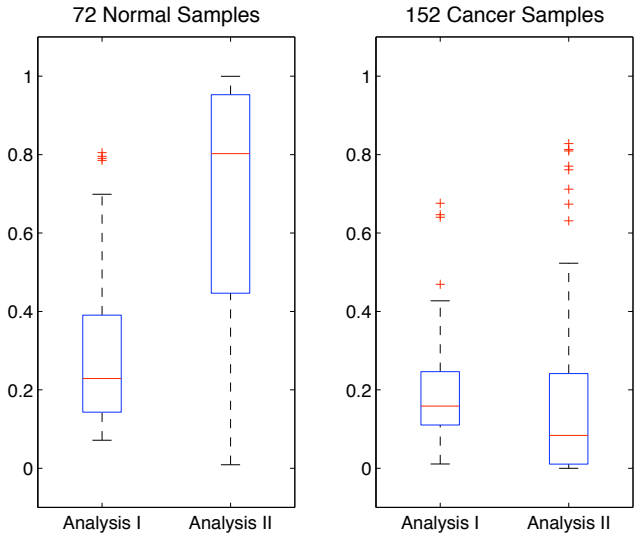


Figure 2: Boxplots of the posterior mean for normal and cancer samples with just the training data (Analysis I) and the training and unlabeled test data (Analysis II). (In the above boxplots, the box ranges from the first quartile (F.Q.) to the third quartile (T.Q.) of the data, while the line shows the median. The dots denote the outliers, which are points which lie beyond 1.5*(T.Q. - F.Q.) on either side of the box.)

6. Discussion

The modeling objective underlying this paper is to formulate a coherent Bayesian perspective for regression using a RHKS model. This requires a firm theoretical foundation characterizing the function space that the Bayesian kernel model spans and the relation of this space to the RKHS. Our results in Section 2 are interesting in their own right, in addition to providing this foundation.

We examined the function class defined by the Bayesian kernel model, the integral of a kernel with respect to a signed Borel measure

$$\mathcal{G} = \left\{ f \;\middle|\; f(x) = \int_{\mathcal{X}} K(x,u)\,\gamma(\mathrm{d}u),\;\; \gamma \in \Gamma \right\},$$

where $\Gamma \subseteq \mathscr{B}(\mathcal{X})$. We stated an equivalence under certain conditions of the function class $\mathcal{G}$ and the RKHS induced by the kernel. This implies: (a) a theoretical foundation for the use of Gaussian processes, Dirichlet processes, and other jump processes for non-parametric Bayesian kernel models, (b) an equivalence between regularization approaches and the Bayesian kernel approach, and (c) an illustration of why placing a prior on the distribution is natural approach in Bayesian non-parametric modelling.

Coherent non-parametric methods have been of great interest in the Bayesian community, however function analytic issues have not been considered. Conversely theoretical studies of RKHS have not approached the approximation and estimation problems from a Bayesian perspective (the exception to both of these are the works of Wahba 1990 and Diaconis 1988). It is our view that the interface of these perspectives is a promising area of research for statisticians, computer scientists, and mathematicians and has both theoretical and practical implications.

A better understanding of this interface may lead to a better understanding of the following research problems:

1. Posterior consistency: It is natural to expect the posterior distribution to concentrate around the true function since the posterior distribution is a probability measure on the RKHS. A natural idea is to use the equivalence between the RKHS and our Bayesian model to exploit the well understood theory of RKHS in proving posterior consistency of the Bayesian kernel model. Tools such as concentration inequalities, uniform Glivenko-Cantelli classes, and uniform central limit theorems may be helpful.

2. Priors on function spaces: In this paper we discuss general function classes without concern for more subtle smoothness properties. An obvious question is can we use the same ideas to relate priors on measures and the kernel to specific classes of functions, such as Sobolev spaces. A study of the relation between integral operators and priors could lead to interesting and useful results for putting priors over specific function classes using the kernel model.

3. Comparison of process priors for modeling: A theoretical and empirical comparison of the accuracy of the various process priors on a variety of function classes and data sets would be of great practical importance and interest, especially for high dimensional problems.

4. Numerical stability and robust estimation: The original motivation for regularization methods was to provide numerical stability in solving Fredholm integral equation of the first kind. Our interest is that of providing robust non-parametric statistical estimates. A link between stability of operators and the generalization or predictive ability of regression estimates is

known (Bousquet and Elisseeff, 2002; Poggio et al., 2004). Further developing this relation is a very interesting area of research and may be of importance for the posterior consistency of the Bayesian kernel model.

Acknowledgments

We would like to thank the reviewers for many useful suggestions and comments. FL would like to acknowledge NSF grant DMS 0406115. SM and QW would like to acknowledge support for this project from the Institute for Genome Sciences & Policy at Duke as well as the NIH grant P50 HG 03391-02 for the Center for Public Genomics. RW would like to acknowledge NSF grants DMS–0112069 and DMS–0422400. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

Appendix A. Proofs of Propositions

In this appendix we provide proofs for the propositions in Section 2.

A.1 Proof for Proposition 4

It holds that

$$\|\mathscr{L}_K[\gamma]\|_K^2\| = \|\sum_{j\in\Lambda}\lambda_j a_j\phi_j\|_K^2 = \sum_{j\in\Lambda}\frac{(\lambda_j a_j)^2}{\lambda_j} = \sum_{j\in\Lambda}\lambda_j a_j^2$$

which is upper bounded by $\lambda_1\sum_j a_j^2 < \infty$. Hence $\mathscr{L}_K[\gamma] \in \mathcal{H}_K$. By direct computation, we have

$$\langle \mathscr{L}_K[\gamma], \gamma\rangle_2 = \big\langle \sum\lambda_j a_k\phi_j, \sum a_j\phi_j\big\rangle_2 = \sum\lambda_k a_j^2 = \|\mathscr{L}_K[\gamma]\|_K^2.$$

A.2 Proof for Corollary 5

The first claim is obvious since both $\mathscr{L}_K[L^2(\mathcal{X},\mathrm{d}u)]$ and $\mathcal{H}_K$ are the same finite dimensional space spanned by $\{\phi_j\}_{j\in\Lambda}$.

The second claim follows from the existence of the sequence $(c_j)_{j\in\Lambda}$ such that

$$\sum_{j\in\Lambda}\frac{c_j^2}{\lambda_j} < \infty \qquad \text{and} \qquad \sum_{j\in\Lambda}\frac{c_j^2}{\lambda_j^2} = \infty.$$

For any such sequence, the function $f = \sum_{j\in\Lambda} c_j\phi_j$ lies in $\mathcal{H}_K$. But by Proposition 4, one cannot find a $\gamma \in L^2(\mathcal{X},\mathrm{d}u)$ such that $\mathscr{L}_K[\gamma] = f$. A simple example is $(c_j)_{j\in\Lambda} = (\lambda_j)_{j\in\Lambda}$.

If K is strictly positive definite, then all its eigenvalues are positive. So the last claim holds.

A.3 Proof for Proposition 6

Since $K(u,v)$ is continuous on the compact set $\mathcal{X}\times\mathcal{X}$, it has a finite maximum $\kappa^2 := \sup_{u,v} K(u,v) < \infty$. Since $L^2(\mathcal{X},\mathrm{d}u)$ is dense in $L^1(\mathcal{X},\mathrm{d}u)$, for every $\gamma \in L^1(\mathcal{X},\mathrm{d}u)$, there exists a Cauchy sequence

$\{\gamma_n\}_{n\geq 1} \subset L^2(\mathcal{X}, \mathrm{d}u)$ which converges to γ in $L^1(\mathcal{X}, \mathrm{d}u)$. It follows from Proposition 4 that $\mathscr{L}_K[\gamma_n] \in \mathcal{H}_K$ and

$$\|\mathscr{L}_K[\gamma_n]\|_K^2 = \int_{\mathcal{X}} \int_{\mathcal{X}} K(u,v)\,\gamma_n(u)\,\mathrm{d}u\,\gamma_n(v)\,\mathrm{d}v \leq \kappa^2 \int_{\mathcal{X}} |\gamma_n(u)|\mathrm{d}u \int_{\mathcal{X}} |\gamma_n(v)|\mathrm{d}v = \kappa^2 \|\gamma_n\|_1^2 < \infty.$$

Therefore we have $\{\mathscr{L}_K[\gamma_n]\}_{n\geq 1} \subset \mathcal{H}_K$ and

$$\lim_{n\to\infty} \sup_{m>n} \|\mathscr{L}_K[\gamma_n] - \mathscr{L}_K[\gamma_m]\|_K \leq \lim_{n\to\infty} \sup_{m>n} \kappa \|\gamma_n - \gamma_m\|_1 = 0,$$

so $\{\mathscr{L}_K[\gamma_n]\}_{n\geq 1}$ is a Cauchy sequence in $\mathcal{H}_K$. By completeness it converges to some $f \in \mathcal{H}_K$. The proof will be finished if we show $\mathscr{L}_K[\gamma] = f$.

By the reproducing property of $\mathcal{H}_K$ convergence in the RKHS norm implies point-wise convergence for $x \in \mathcal{X}$, so $L_K[\gamma_n](x) \to f(x)$ for every x.

In addition, for every $x \in \mathcal{X}$, we have

$$\lim_{n\to\infty} |\mathscr{L}_K[\gamma_n](x) - \mathscr{L}_K[\gamma](x)| \leq \int_{\mathcal{X}} |K(x,u)(\gamma_n(u) - \gamma(u))|\,\mathrm{d}u \leq \kappa^2 \|\gamma_n - \gamma\|_1 = 0,$$

which implies that $\mathscr{L}_K[\gamma_n](x)$ also converges to $\mathscr{L}_K[\gamma](x)$. Hence $\mathscr{L}_K[\gamma] = f \in \mathcal{H}_K$.

A.4 Proof for Proposition 7

Let $\gamma = \sum c_i \delta_{x_i} \in \mathscr{M}_D$. Then $\mathscr{L}_K[\gamma] = \sum c_i K_{x_i}$ and

$$\|\mathscr{L}_K[\gamma]\|_K^2 = \sum_{i,j} c_i K(x_i, x_j) c_j \leq \kappa^2 \left(\sum_i |c_i| \right)^2 < \infty.$$

Therefore, our conclusion holds.

A.5 Proof for Proposition 9

The arguments for Lebesgue measure hold if we replace the Lebesgue measure with any finite Borel measure. We denote the corresponding integral operator as $\mathscr{L}_{K,\mu}$ and function space of integrable and square integrable functions as $L^1_\mu(\mathcal{X})$ and $L^2_\mu(\mathcal{X})$ respectively. Then

$$L^2_\mu(\mathcal{X}) \subset L^1_\mu(\mathcal{X}) \subset L^{-1}_{K,\mu}(\mathcal{H}_K).$$

Since the function $1_{\mathcal{X}}(x) = 1$ lies in $L^1_\mu(\mathcal{X})$ we obtain

$$\mathscr{L}_K(\mu) = \mathscr{L}_{K,\mu}(1_{\mathcal{X}}) = \int_{\mathcal{X}} K(\cdot, u)\mathrm{d}\mu(u) \in \mathcal{H}_K.$$

This implies $\mathscr{B}_+(\mathcal{X})$ lies in $L_K^{-1}(\mathcal{H}_K)$ and so does $\mathscr{B}(\mathcal{X})$.

Appendix B. Multivariate Version of Lévy-Khintchine Formula

Here we give the statement of the multivariate version of the Lévy-Khintchine formula (Applebaum, 2004, Corollary 2.4.20).

Theorem 16 *(***Lévy-Khintchine***) Let X be a d-dimensional Lévy process with characteristic function $\phi_t(u) := \mathbb{E}(e^{i\langle u, X_t\rangle}), u \in \mathbb{R}^d$. Then there exists a unique vector $a \in \mathbb{R}^d$, a $d \times d$ semi-positive definite matrix σ, and ν a positive measure on $\mathbb{R}^d \backslash 0$ with $\int_{\mathbb{R}^d}(1 \wedge |u|^2)\nu(du) < \infty$ such that,*

$$\phi_t(u) = \exp\left\{ t\left[i\langle u, a\rangle - \frac{1}{2}\langle u, \sigma u\rangle + \int_{\mathbb{R}^d\backslash 0}[e^{i\langle u,s\rangle} - 1 - i\langle u, s\rangle 1_{\{s:|s|<1\}}(s)]\nu(ds)\right]\right\}$$

where $\langle\cdot,\cdot\rangle$ denotes the standard inner product in $\mathbb{R}^d$.

The results we have presented extend to the multivariate case without complication. The simplest multivariate extension is to assume independence of the dimensions, however for small sample sizes and many dimensions this is not practical. This issue can be addressed by carefully inducing covariance structure in the model (Liang et al., 2007, 2006).

References

David Applebaum. *Lévy Processes and Stochasitic Calculus*. Cambridge Studies in Advanced Mathematics. Cambridge Univ. Press, Cambridge, UK, 2004.

Nachman Aronszajn. Theory of reproducing kernels. *T. Am. Math. Soc.*, 686:337–404, 1950.

Mikhail Belkin and Partha Niyogi. Semi-supervised learning on Riemannian manifolds. *Machine Learning*, 56(1-3):209–239, 2004.

Mikhail Belkin, Partha Niyogi, and Vikas Sindhwani. Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *J. Mach. Learn. Res.*, 7:2399–2434, 2006.

David M. Blei and Michael I. Jordan. Variational inference for Dirichlet process mixtures. *Bayesian Anal.*, 1(1):121–143 (electronic), 2006.

Olivier Bousquet and André Elisseeff. Stability and generalization. *J. Mach. Learn. Res.*, 2:499–526, 2002.

Sounak Chakraborty, Malay Ghosh, and Bani K. Mallick. Bayesian non-linear regression for large p small n problems. *J. Am. Stat. Assoc.*, 2005. Under revision.

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

Felipe Cucker and Stephen Smale. On the mathematical foundations of learning. *Bulletin of the American Mathematical Society*, 39:1–49, 2001.

Carl de Boor and Robert E. Lynch. On splines and their minimum properties. *J. Math. Mech.*, 15: 953–969, 1966.

Ronald A. DeVore, Ralph Howard, and Charles A. Micchelli. Optimal nonlinear approximation. *Manuskripta Mathematika*, 1989.

Persi Diaconis. Bayesian numerical analysis. In Shanti S. Gupta and James O. Berger, editors, *Statistical decision theory and related topics, IV*, volume 1, pages 163–175. Springer-Verlag, New York, NY, 1988.

Michael D. Escobar and Mike West. Bayesian density estimation and inference using mixtures. *J. Am. Stat. Assoc.*, 90:577–588, 1995.

Theodoros Evgeniou, Massimiliano Pontil, and Tomaso Poggio. Regularization networks and support vector machines. *Advances in Computational Mathematics*, 13:1–50, 2000.

Thomas S. Ferguson. Prior distributions on spaces of probability measures. *Ann. Stat.*, 2:615–629, 1974.

Thomas S. Ferguson. A Bayesian analysis of some nonparametric problems. *Ann. Stat.*, 1:209–230, 1973.

Erik Ivar Fredholm. Sur une nouvelle méthode pour la résolution du problèm de Dirichlet. *Euvres complètes:publiées sous les auspices de la Kungliga svenska vetensakademien par l'Institut Mittag-Leffler*, pages 61–68, 1900.

Subhashis Ghosal and Anindya Roy. Posterior consistency of Gaussian process prior for nonparametric binary regression. *Ann. Statist.*, 34(5):2413–2429, 2006.

Jacques Hadamard. Sur les problèmes aux dérivées partielles et leur signification physique. *Princeton University Bulletin*, pages 49–52, 1902.

Jaroslav Hájek. On linear statistical problems in stochastic processes. *Czechoslovak Math. J.*, 12 (87):404–444, 1962.

Jaroslv Hájek. On a property of normal distributions of any stochastic process. *Select. Transl. Math. Statist. and Probability*, 1:245–252, 1961.

Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. Springer-Verlag, 2001.

Lancelot F. James, Antonio Lijoa, and Igor Prünster. Conjugacy as a distinctive feature of the Dirichlet process. *Scand. J. Stat.*, 33:105–120, 2005.

Iain Johnstone. Function estimation in Gaussian noise: sequence models. Draft of a monograph, 1998.

Gopinath Kallianpur. The role of reproducing kernel Hilbert spaces in the study of Gaussian processes. *Advances in Probability and Related Topics*, 2:49–83, 1970.

Hermann König. *Eigenvalue distribution of compact operators*, volume 16 of *Operator Theory: Advances and Applications*. Birkhäuser, Basel, CH, 1986.

George S. Kimeldorf and Grace Wahba. A correspondence between Bayesian estimation on stochastic processes and smoothing by splines. *Ann. Math. Statist.*, 41(2):495–502, 1971.

Feng Liang, Sayan Mukherjee, and Mike West. Understanding the use of unlabelled data in predictive modeling. *Stat. Sci.*, 2006. To appear.

Feng Liang, Ming Liao, Kai Mao, Sayan Mukherjee, and Mike West. Non-parametric Bayesian kernel models. Discussion Paper 2007-10, Duke University ISDS, Durham, NC, 2007. URL {\emwww.stat.duke.edu/research/papers/}.

Milan N. Lukić and Jay H. Beder. Stochasitic processes with sample paths in reproducing kernel Hilbert spaces. *T. Am. Math. Soc.*, 353(10):3945–3969, 2001.

Stephen MacEachern and Peter Müller. Estimating mixture of Dirichlet process models. *J. Comput. Graph. Stat.*, pages 223–238, 1998.

Vladimir G. Mazja. *Sobolev Spaces*. Springer-Verlag, New York, NY, 1985.

Peter Müller, Fernando Quintana, and Gary Rosner. A method for combining inference across related nonparametric Bayesian models. *J. Am. Stat. Assoc.*, pages 735–749, 2004.

James Mercer. Functions of positive and negative type and their connection with the theory of integral equations. *Philosophical Transactions of the Royal Society, London A*, 209:415–446, 1909.

Charles A. Micchelli and Grace Wahba. Design problems for optimal surface interpolation. In Zvi Ziegler, editor, *Approximation Theory and Applications*, pages 329–348, 1981.

Sayan Mukherjee, Pablo Tamayo, Simon Rogers, Ryan M. Rifkin, Anna Engle, Colin Campbell, Todd R. Golub, and Jill P. Mesirov. Estimating dataset size requirements for classifying DNA Microarray data. *Journal of Computational Biology*, 10:119–143, 2003.

Neal, R. M. *Bayesian Learning for Neural Networks*. Springer, New York, 1996. Lecture Notes in Statistics 118.

Emanuel Parzen. Probability density functionals and reproducing kernel Hilbert spaces. In Murray Rosenblatt, editor, *Proceedings of the Symposium on Time Series Analysis*, pages 155–169, New York, NY, 1963. John Wiley & Sons.

Tomaso Poggio and Federico Girosi. Regularization algorithms for learning that are equivalent to multilayer networks. *Science*, 247:978–982, 1990.

Tomaso Poggio, Ryan M. Rifkin, Sayan Mukherjee, and Partha Niyogi. General conditions for predictivity in learning theory. *Nature*, 428:419–422, 2004.

Sridhar Ramaswamy, Pablo Tamayo, Ryan M. Rifkin, Sayan Mukherjee, Chen-Hsiang Yeang, Michael Angelo, Christine Ladd, Michael Reich, Eva Latulippe, Jill P. Mesirov, Tomaso Poggio, William Gerald, Massimo Loda, Eric S. Lander, and Todd R. Golub. Multiclass cancer diagnosis using tumor gene expression signatures. *Proc. Nat. Aca. Sci.*, 98:149–54, 2001.

Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning*. MIT Press, Cambridge, MA, 2006.

L. Chris G. Rogers and David Williams. *Diffusions, Markov Processes, and Martingales*, volume 2. John Wiley & Sons, New York, NY, 1987. ISBN 0-471-91482-7.

Bernhard Schölkopf and Alexander J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, Cambridge, MA, 2001.

Isaac J. Schoenberg. Positive definite functions on spheres. *Duke Mathematics Journal*, 9:96–108, 1942.

John S. Shawe-Taylor and Nello Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge Univ. Press, Cambridge, UK, 2004.

Peter Sollich. Bayesian methods for support vector machines: Evidence and predictive class probabilities. *Machine Learning*, 46(1-3):21–52, 2002.

Andrei Nikolaevich Tikhonov. Solution of incorrectly formulated problems and the regularization method. *Soviet Doklady*, 4:1035–1038, 1963.

Michael E. Tipping. Sparse Bayesian learning and the relevance vector machine. *J. Mach. Learn. Res.*, 1:211–244, 2001.

Chong Tu, Merlise A. Clyde, and Robert L. Wolpert. Lévy adaptive regression kernels. Discussion Paper 2006-08, Duke University ISDS, Durham, NC, 2006. URL `http://www.stat.duke.edu/research/papers/`.

Vladimir N. Vapnik. *Statistical Learning Theory*. John Wiley & Sons, New York, NY, 1998.

Grace Wahba. *Splines Models for Observational Data*, volume 59 of *Series in Applied Mathematics*. SIAM, Philadelphia, PA, 1990.

Grace Wahba. Support vector machines, reproducing kernel Hilbert spaces, and randomized GACV. In Bernhard Schölkopf, Alexander J. Smola, Christopher J. C. Burges, and Rosanna Soentpiet, editors, *Advances in Kernel Methods: Support Vector Learning*, pages 69–88. MIT Press, Cambridge, MA, 1999.

Larry Wasserman. *All of Nonparametric Statistics*. Springer-Verlag, 2005.

Mike West. Hyperparameter estimation in Dirichlet process mixture models. Discussion Paper 1992-03, Duke University ISDS, Durham, NC, 1992. URL `http://www.stat.duke.edu/research/papers/`.

Robert L. Wolpert and Katja Ickstadt. Reflecting uncertainty in inverse problems: A Bayesian solution using Lévy processes. *Inverse Problems*, 20(6):1759–1771, 2004.

Robert L. Wolpert, Katja Ickstadt, and Martin Bøgsted Hansen. A nonparametric Bayesian approach to inverse problems (with discussion). In José Miguel Bernardo, Maria Jesus Bayarri, James O. Berger, A. Phillip Dawid, David Heckerman, Adrian F. M. Smith, and Mike West, editors, *Bayesian Statistics 7*, pages 403–418, Oxford, UK, 2003. Oxford Univ. Press. ISBN 0-19-852615-6.

Eric P. Xing, Roded Sharan, and Michael I. Jordan. Bayesian haplotype inference via the Dirichlet process. In Carla E. Brodley, editor, *Machine Learning, Proceedings of the 21st International Conference (ICML 2004), Banff, Canada*, New York, NY, 2004. ACM Press. URL http://www.aicml.cs.ualberta.ca/_banff04/icml/pages/accepted.htm.

Eric P. Xing, Kyung-Ah Sohn, Michael I. Jordan, and Yee-Whye Teh. Bayesian multi-population haplotype inference via a hierarchical Dirichlet process mixture. In William Cohen and Andrew Moore, editors, *Machine Learning, Proceedings of the 23rd International Conference (ICML 2006), Pittsburgh, PA*, New York, NY, 2006. ACM Press. URL http://www.icml2006.org/icml2006/technical/accepted.html.

Ding-Xuan Zhou. Capacity of reproducing kernel spaces in learning theory. *IEEE T. Inform. Theory*, 49:1743–1752, 2003.

Journal of Machine Learning Research 8 (2007) 1799-1833 Submitted 09/06; Revised 5/07; Published 8/07

"Ideal Parent" Structure Learning for Continuous Variable Bayesian Networks

Gal Elidan GALEL@CS.STANFORD.EDU
Department of Computer Science
Stanford University
Stanford, CA 94305, USA

Iftach Nachman INACHMAN@CGR.HARVARD.EDU
FAS Center for Systems Biology
Harvard University
Cambridge, MA 02138, USA

Nir Friedman NIR@CS.HUJI.AC.IL
School of Computer Science and Engineering
Hebrew University
Jerusalem 91904, Israel

Editor: David Maxwell Chickering

Abstract

Bayesian networks in general, and continuous variable networks in particular, have become increasingly popular in recent years, largely due to advances in methods that facilitate automatic learning from data. Yet, despite these advances, the key task of learning the structure of such models remains a computationally intensive procedure, which limits most applications to parameter learning. This problem is even more acute when learning networks in the presence of missing values or hidden variables, a scenario that is part of many real-life problems. In this work we present a general method for speeding structure search for continuous variable networks with common parametric distributions. We efficiently evaluate the approximate merit of candidate structure modifications and apply time consuming (exact) computations only to the most promising ones, thereby achieving significant improvement in the running time of the search algorithm. Our method also naturally and efficiently facilitates the addition of useful new hidden variables into the network structure, a task that is typically considered both conceptually difficult and computationally prohibitive. We demonstrate our method on synthetic and real-life data sets, both for learning structure on fully and partially observable data, and for introducing new hidden variables during structure search.

Keywords: Bayesian networks, structure learning, continuous variables, hidden variables

1. Introduction

Probabilistic graphical models have gained wide-spread popularity in recent years with the advance of techniques for learning these models directly from data. The ability to learn allows us to overcome lack of expert knowledge about domains and adapt models to a changing environment, and can also lead to scientific discoveries. Indeed, Bayesian networks in general, and continuous variable

A preliminary version of this paper appeared in the Proceedings of the Twentieth Conference on Uncertainty in Artificial, 2004 (UAI '04).

networks in particular, are now being used in a wide range of applications, including fault detection (e.g., U. Lerner and Koller, 2000), modeling of biological systems (e.g., Friedman et al., 2000) and medical diagnosis (e.g., Shwe et al., 1991).

A key task in learning these models from data is adapting the structure of the network based on observations. This NP-complete problem (Chickering, 1996a) is typically treated as a combinatorial optimization problem that is addressed by heuristic search procedures, such as greedy hill climbing. This procedure examines local modifications to single edges at each step, evaluates them using some score, and proceeds to apply the one that leads to the largest improvement in score, until a local maximum is reached. Even with this simple approach structure learning is computationally challenging for all but small networks due to the large number of possible modifications that can be evaluated, and the cost of evaluating each one. To make things worse, the problem is even harder in the (realistic) presence of missing values, as non-linear optimization is required to evaluate different structure modification candidates during the search. Learning is particularly problematic when we also want to allow for hidden variables and want to effectively add them during the learning process. Thus, in practice, most applications are still limited to parameter estimation.

Of particular interest to us is learning continuous variable networks, which are crucial for a wide range of real-life applications. One case that received scrutiny in the literature is learning *linear Gaussian* networks (Geiger and Heckerman, 1994; Lauritzen and Wermuth, 1989). In this case, we can use sufficient statistics to summarize the data, and a closed form equation to evaluate the score of candidate structure modifications. In general, however, we are also interested in non-linear interactions. These do not have sufficient statistics, and require applying parameter optimization to evaluate the score of candidate structures. These difficulties severely limit the applicability of standard heuristic structure search procedures to rich non-linear models.

In this work, we present a general method for speeding structure search for continuous variable networks. In contrast to innovative structure learning methods that modify the space explored by the search algorithm (e.g., Chickering, 1996b; Moore and Wong, 2003; Teyssier and Koller, 2005), our method leverages on the parametric structure of the conditional distributions in order to efficiently approximate the benefit of an *individual* structure candidate. As such, our method can be used to speed up many existing structure learning algorithms and heuristics.

The basic idea is straightforward and is inspired from the notion of *residues* in regression (McCullagh and Nelder, 1989). For each variable, we construct an *ideal parent profile* of a new hypothetical parent that would lead to the best possible prediction of that variable. Intuitively, a candidate parent of a variable is useful if it is similar to the ideal parent. Using basic principles, we derive a similarity measure for efficiently comparing a candidate parent to the ideal profile. We show that this measure approximates the improvement in score that would result from the addition of that parent to the network structure. This provides us with a fast method for scanning many potential parents and focuses more careful evaluation (exact scoring) on a smaller number of promising candidates.

The ideal parent profiles we construct during search also provide new leverage on the problem of introducing new hidden variables during structure learning. Basically, if the ideal parent profiles of several variables are sufficiently similar, and are not similar to one of their current parents, we can consider adding a new hidden variable that serves as a parent of all these variables. The ideal profile allows us to estimate the impact this new variable will have on the score, and suggest the values it takes in each instance. The method therefore provides a guided approach for introducing new variables during search and allows for contrasting them with alternative search steps in a computationally efficient manner.

We apply our method using linear Gaussian and non-linear Sigmoid Gaussian conditional probability distributions to several tasks: learning structure with complete data; learning structure with missing data; and learning structure while allowing for the automatic introduction of new hidden variables. We evaluate all tasks on both realistic synthetic experiments and real-life problems in the field of computational biology.

The rest of the paper is structured as follows: In Section 2 we provide a brief summary of continuous variable networks. In Section 3 we present the "Ideal Parent" concept as it applies to the simple case of linear Gaussian models. In Section 4 we discuss how our method is used within a structure learning algorithm. In Section 5 we show how our method can be leveraged in order to introduce new useful hidden variables during learning, and in Section 6 we discuss the computational modifications needed to address both the presence of missing values and hidden variables. In Section 7 we show how our entire framework can be generalized to the challenging case of more general non-linear distributions. In Section 8 we present a further extension to conditional probability distributions that use non-additive noise models. In Section 9 we present our experimental results for both synthetic and real-life data. We conclude with a discussion of related works and future directions in Section 10.

2. Continuous Variable Networks

Consider a finite set $\mathcal{X} = \{X_1, \ldots, X_n\}$ of random variables. A *Bayesian network* (BN) is an annotated directed acyclic graph G that represents a joint probability distribution over $\mathcal{X}$. The nodes of the graph correspond to the random variables and are annotated with a conditional probability density (CPD) of the random variable given its parents $\mathbf{U_i}$ in the graph G. The joint distribution is the product over families (variable and its parents)

$$P(X_1, \ldots, X_n) = \prod_{i=1}^{n} P(X_i | \mathbf{U_i}).$$

The graph G represents independence properties that are assumed to hold in the underlying distribution: Each X_i is independent of its non-descendants given its parents $\mathbf{U_i}$.

Unlike the case of discrete variables, when the variable X and some or all of its parents are real valued, there is no representation that can capture all conditional densities. A common choice is the use of *linear Gaussian* conditional densities (Geiger and Heckerman, 1994; Lauritzen and Wermuth, 1989), where each variable is a linear function of its parents with Gaussian noise. When all the variables in a network have linear Gaussian conditional densities, the joint density over $\mathcal{X}$ is a multivariate Gaussian (Lauritzen and Wermuth, 1989). In many real world domains, such as in neural or gene regulation network models, the dependencies are known to be non-linear (for example, a saturation effect is expected). In these cases, we can still use Gaussian conditional densities, but now the mean of the density is expressed as a non-linear function of the parents (for example, a sigmoid).

Given a training data set $\mathcal{D} = \{\mathbf{x}[1], \ldots, \mathbf{x}[M]\}$, where the mth instance $\mathbf{x}[m]$ assigns values to the variables in $\mathcal{X}$, the problem of learning a Bayesian network is to find a structure and parameters that maximize the likelihood of $\mathcal{D}$ given the graph, typically along with some regularization constraints. Given a data set $\mathcal{D}$ and a network structure G, we define

$$\ell(\mathcal{D} : G, \theta) = \log P(\mathcal{D} : G, \theta) = \sum_m \log P(\mathbf{x}[m] : G, \theta)$$

Algorithm 1: Greedy Hill-Climbing Structure Search for Bayesian Networks

```
Input  : 𝒟   // training set
         G_0 // initial structure
Output : A final structure G

G_best ← G_0
repeat
    G ← G_best
    foreach Operator Add,Delete,Reverse,Replace edge in G do
        if Operator does not create a directed cycle then
            G' ← ApplyOperator(G)
            if Score(G' : 𝒟) > Score(G_best : 𝒟) then
                G_best ← G'
            end
        end
    end foreach
until G_best == G
return G_best
```

to be the log-likelihood function, where θ are the model parameters. In estimating the *maximum likelihood* parameters of the network, we aim to find the parameters $\hat{\theta}$ that maximize this likelihood function. When the data is *complete* (all variables are observed in each instance), the log-likelihood can be rewritten as a sum of *local* likelihood functions,

$$\ell(\mathcal{D}:G,\theta)=\sum \ell_i(\mathcal{D}:\mathbf{U_i},\theta_i)$$

where $\ell_i(\mathcal{D}:\mathbf{U_i},\theta_i)$ is a function of the choice of $\mathbf{U_i}$ and the parameters θ_i of the corresponding CPD: it is the log-likelihood of regressing X_i on $\mathbf{U_i}$ in the data set with the particular choice of CPD. Due to this decomposition, we can find the maximum likelihood parameters of each CPD independently by maximizing the local log-likelihood function. For some CPDs, such as linear Gaussian ones, there is a closed form expression for the maximum likelihood parameters. In other cases, finding these parameters is a continuous optimization problem that is typically addressed by gradient based methods.

Learning the structure of a network is a significantly harder task. The common approach is to introduce a scoring function that balances the likelihood of the model and its complexity and then attempt to maximize this score using a heuristic search procedure that considers local changes (e.g., adding and removing edges). A commonly used score is the *Bayesian Information Criterion* (BIC) score (Schwarz, 1978)

$$BIC(\mathcal{D},G)=\max_{\theta}\ell(\mathcal{D}:G,\theta)-\frac{\log M}{2}\mathrm{Dim}[G] \tag{1}$$

where M is the number of instances in $\mathcal{D}$, and $\mathrm{Dim}[G]$ is the number of parameters in G. The BIC score is actually an approximation to the more principled full Bayesian score, that integrates over all possible parameterizations of the CPDs. While a closed form for the Bayesian score, with a suitable prior, is known for Gaussian networks (Geiger and Heckerman, 1994), numerical computation of

this score is extremely demanding for the non-linear case. Thus, we adopt the common approach and focus on the BIC approximation from here on.

A common search procedure for optimizing the score is the greedy hill-climbing procedure outlined in Algorithm 1. This procedure can be augmented with mechanisms for escaping local maxima, such as random walk perturbations upon reaching a local maxima (also known as random restarts), and using a TABU list (Glover and Laguna, 1993).

3. The "Ideal parent" Concept

Our goal is to speed up a generic structure search algorithm for a Bayesian network with continuous variables. The complexity of any such algorithm is rooted in the need to score each candidate structure change, which in turn may require non-linear parameter optimization. Thus, we want to somehow efficiently approximate the benefit of each candidate and score only the most promising of these candidates. The manner in which this helps us to discover new hidden variables will become evident in Section 5.

3.1 Basic Framework

Consider adding Z as a new parent of X whose current parents in the network are $\mathbf{U}$. Given a training data $\mathcal{D}$ of M instances, to evaluate the change in score, when using the BIC score of Eq. (1), we need to compute the change in the log-likelihood

$$\Delta_{X|\mathbf{U}}(Z) = \max_{\theta_{X|\mathbf{U},Z}} \ell_X(\mathcal{D} : \mathbf{U} \cup \{Z\}, \theta_{X|\mathbf{U},Z}) - \ell_X(\mathcal{D} : \mathbf{U}, \widehat{\theta}_{X|\mathbf{U}}) \tag{2}$$

where $\widehat{\theta}_{X|\mathbf{U}}$ are the maximum likelihood parameters of X given $\mathbf{U}$ and $\theta_{X|\mathbf{U},Z}$ are the parameters for the family where Z is an additional parent of X. The change in the BIC score is this difference combined with the change in the model complexity penalty terms. Thus, to evaluate this difference, we need to compute the maximum likelihood parameters of X given the new choice of parents. Our goal is to speed up this computation.

The basic idea of our method is straightforward. For a given variable, we want to construct a hypothetical ideal parent Y that would best predict the variable. We will then compare each existing candidate parent Z to this imaginary one using a similarity measure $C(\vec{y}, \vec{z})$ (which we describe below). Finally, we will fully score only the most promising candidates: those that are most similar to the ideal parent. Figure 1 illustrates this process. In order for this approach to be beneficial, we want the similarity score to approximate the actual change in likelihood defined in Eq. (2). Furthermore, we want to be able to compute the similarity measure in a fraction of the time it takes to fully score a candidate parent.

3.1.1 CONDITIONAL PROBABILITY DISTRIBUTION

To make our discussion concrete, we focus on networks where we represent X as a function of its parents $\mathbf{U} = \{U_1, \ldots, U_k\}$ with a conditional probability distribution (CPD) that has the following general form:

$$X = g(\alpha_1 \mathbf{u}_1, \ldots, \alpha_k \mathbf{u}_k : \theta) + \varepsilon \tag{3}$$

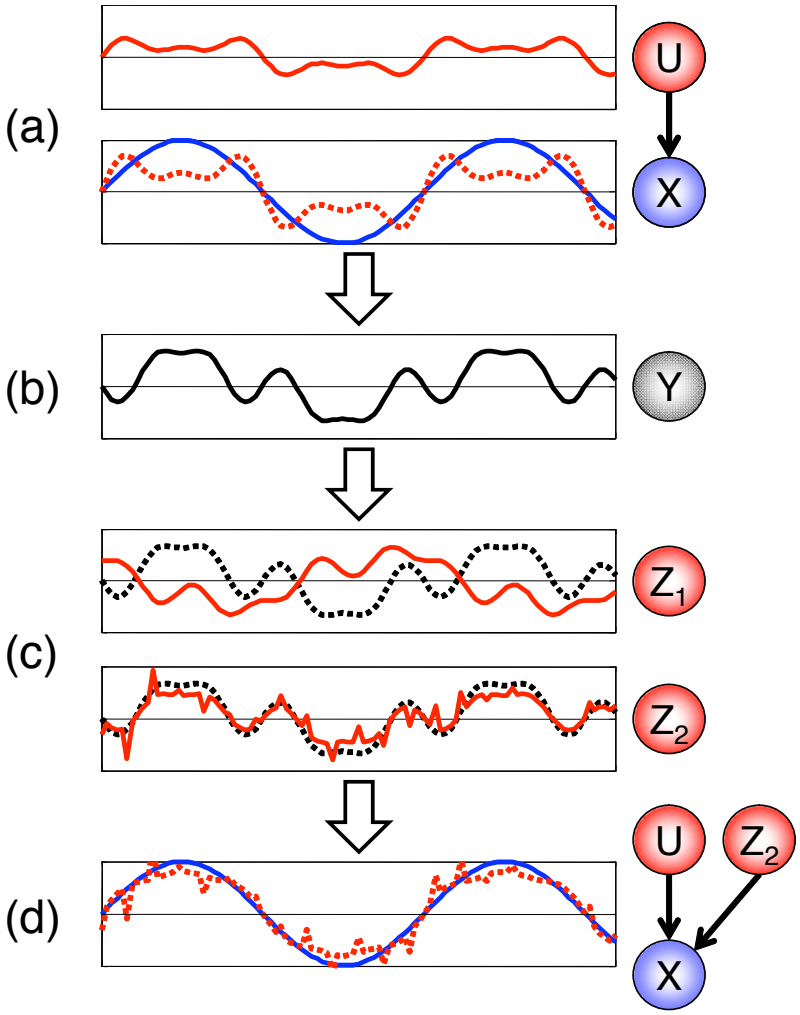


Figure 1: **The "Ideal Parent" Concept**: Illustration of the "Ideal Parent" approach for a variable with a single parent U and a linear Gaussian conditional distribution. The top panel of (a) shows the profile (assignment in all instances) of the parent. The panel below shows the profile of the child node along with the profile predicted for the child based on its parent (dotted red). (b) shows the profile of the ideal hypothetical parent that would lead to zero error in prediction of the child variable if added to the current model. In the linear Gaussian case, this profile is simply the residual of the two curves shown in (a). (c) shows the profiles of two candidate parents, compared to the profile of the ideal parent (dotted black). (d) shows the child profile along with its prediction based on the original parent *and* the new chosen parent from the candidate in (c) that was most similar to the ideal profile of (b). Note that the prediction is not perfect as the profile of the parent chosen does not, in general, match the profile of the ideal parent exactly.

where g is a *link function* that integrates the contributions of the parents with additional parameters θ, α_i that are scale parameters applied to each of the parents, and ε that is a noise random variable with zero mean. In the following discussion, we assume that ε is Gaussian with variance σ^2.

When the function g is the sum of its arguments, this CPD is the standard linear Gaussian CPD. However, we can also consider non-linear choices of g. For example,

$$g(\alpha_1 \mathbf{u}_1, \ldots, \alpha_k \mathbf{u}_k : \theta) \equiv \theta_1 \frac{1}{1 + e^{-\sum_i \alpha_i u_i}} + \theta_0 \tag{4}$$

is a sigmoid function where the response of X to its parents' values is saturated when the sum is far from zero.

3.1.2 LIKELIHOOD FUNCTION

Given the above form of CPDs, we can now write a concrete form of the log-likelihood function

$$\begin{aligned} \ell_X(\mathcal{D} : \mathbf{U}, \theta) &= -\frac{1}{2} \sum_{m=1}^{M} \left[\log(2\pi) + \log(\sigma^2) + \frac{1}{\sigma^2} (x[m] - g(\mathbf{u}[m]))^2 \right] \\ &= -\frac{1}{2} \left[M \log(2\pi) + M \log(\sigma^2) + \frac{1}{\sigma^2} \sum_m (x[m] - g(\mathbf{u}[m]))^2 \right] \end{aligned} \tag{5}$$

where, for simplicity, we absorbed each scaling factor α_j into each value of $u_j[m]$. Similarly, when the new parent Z is added with coefficient α_z, the new likelihood is

$$\ell_X(\mathcal{D} : \mathbf{U} \cup \{Z\}, \alpha_z, \theta,) = -\frac{1}{2} \left[M \log(2\pi) + M \log(\sigma_z^2) + \frac{1}{\sigma_z^2} \sum_m (x[m] - g(\mathbf{u}[m], \alpha_z z[m]))^2 \right]$$

where σ_z^2 is used to denote the variance parameter after Z is added. Consequently, the difference in likelihood of Eq. (2) takes the form of

$$\begin{aligned} \Delta_{X|\mathbf{U}}(Z) &= -\frac{M}{2} \left[\log \sigma_z^2 - \log \sigma^2 \right] \\ &\quad -\frac{1}{2} \left[\frac{1}{\sigma_z^2} \sum_m (x[m] - g(\mathbf{u}[m], \alpha_z z[m]))^2 - \frac{1}{\sigma^2} \sum_m (x[m] - g(\mathbf{u}[m]))^2 \right]. \end{aligned} \tag{6}$$

3.1.3 THE "IDEAL PARENT"

We now define the ideal parent for X

Definition 3.1: Given a data set $\mathcal{D}$, and a CPD for X given its parents $\mathbf{U}$, with a link function g and parameters θ and α, the *ideal parent* Y of X is such that for each instance m,

$$x[m] = g(\alpha_1 u_1[m], \ldots, \alpha_k u_k[m], y[m] : \theta).$$

∎

Under mild conditions, the *ideal parent profile* (i.e., value of Y in each instance) can be computed for almost any uni-modal parametric conditional distribution. The only requirement from g is that it should be invertible w.r.t. each one of the parents. Note that in this definition, we implicitly assume

that $x[m]$ lies in the image of g. If this is not the case, we can substitute $x[m]$ with $x_g[m]$, the point in g's image closest to $x[m]$. This guarantees that the prediction's mode for the current set of parents and parameters is as close as possible to X.

The resulting profile for the hypothetical ideal parent Y is the optimal set of values for the $(k+1)$th parent, in the sense that it would maximize the likelihood of the child variable X. This is true since by definition, X is equal to the mode of the function of its parents defined by g. Intuitively, if we can efficiently find a candidate parent Z that is similar to the hypothetically optimal parent, we can improve the model by adding an edge from this parent to X. We are now ready to instantiate the similarity measure $C(\vec{y},\vec{z})$. Below, we demonstrate how this is done for the case of a linear Gaussian CPD. We extend the framework for non-linear CPDs in Section 7.

3.2 Linear Gaussian

Let X be a variable in the network with a set of parents $\mathbf{U}$, and a *linear Gaussian* conditional distribution. In this case, g in Eq. (3) takes the form

$$g(\alpha_1\mathbf{u}_1,\ldots,\alpha_k\mathbf{u}_k:\theta) \equiv \sum_i \alpha_i\mathbf{u}_i + \theta_0.$$

To choose promising candidate parents for X, we start by computing the ideal parent Y for X given its current set of parents. This is done by inverting the linear link function g with respect to this additional parent Y (note that we can assume, without loss of generality, that the scale parameter of this additional parent is 1). This results in

$$y[m] = x[m] - \sum_j \alpha_j u_j[m] - \theta_0.$$

We can summarize this in vector notation, by using $\vec{x} = \langle x[1],\ldots,x[M]\rangle$, and so we get

$$\vec{y} = \vec{x} - \mathcal{U}\vec{\alpha} - \theta_0$$

where $\mathcal{U}$ is the matrix of parent values on all instances, and $\vec{\alpha}$ is the vector of scale parameters.

Having computed the *ideal parent profile*, we now want to efficiently evaluate its similarity to the profile of candidate parents. Intuitively, we want the similarity measure to reflect the likelihood gain by adding Z as a parent of X. Ideally, we want to evaluate $\Delta_{X|\mathbf{U}}(Z)$ for each candidate parent Z. However, instead of re-estimating all the parameters of the CPD after adding Z as a parent, we approximate this difference by only fitting the scaling factor associated with the new parent and freezing all other parameters of the CPD (the scaling parameters of the current parents $\mathbf{U}$ and the variance parameter σ^2).

Theorem 3.2 *Suppose that X has parents $\mathbf{U}$ with a set $\vec{\alpha}$ of scaling factors. Let Y be the ideal parent as described above, and Z be some candidate parent. Then the change in the log-likelihood of X in the data, when adding Z as a parent of X, while freezing all scaling and variance parameters except the scaling factor of Z, is*

$$\begin{aligned} C_1(\vec{y},\vec{z}) &\equiv \max_{\alpha_Z} \ell_X(\mathcal{D}:\mathbf{U}\cup\{Z\},\widehat{\theta}_{X|\mathbf{U}}\cup\{\alpha_Z\}) - \ell_X(\mathcal{D}:\mathbf{U},\widehat{\theta}_{X|\mathbf{U}}) \\ &= \frac{1}{2\sigma^2}\frac{(\vec{y}\cdot\vec{z})^2}{\vec{z}\cdot\vec{z}}. \end{aligned} \tag{7}$$

Proof: In the linear Gaussian case $y[m] = x[m] - g(\mathbf{u}[m])$ by definition and $g(\mathbf{u}[m], \alpha_z z[m]) = g(\mathbf{u}[m]) + \alpha_z z[m]$ so that Eq. (6) can be written as

$$\begin{aligned}\Delta_{X|\mathbf{U}}(Z) &= -\frac{M}{2}\left[\log\sigma_z^2 - \log\sigma^2\right] - \frac{1}{2}\left[\frac{1}{\sigma_z^2}\sum_m (y[m] - \alpha_z z[m])^2 - \frac{1}{\sigma^2}\sum_m y[m]^2\right] \\ &= -\frac{M}{2}\left[\log\sigma_z^2 - \log\sigma^2\right] - \frac{1}{2}\left[\frac{1}{\sigma_z^2}\left(\vec{y}\cdot\vec{y} - 2\alpha_z\vec{z}\cdot\vec{y} + \alpha_z^2\vec{z}\cdot\vec{z}\right) - \frac{1}{\sigma^2}\vec{y}\cdot\vec{y}\right]. \qquad (8)\end{aligned}$$

Since $\sigma_z = \sigma$ this reduces to

$$\begin{aligned}\Delta_{X|\mathbf{U}}(Z:\alpha_z) &\equiv \ell_X(\mathcal{D}:\mathbf{U}\cup\{Z\}, \widehat{\theta}_{X|\mathbf{U}}\cup\{\alpha_Z\}) - \ell_X(\mathcal{D}:\mathbf{U}, \widehat{\theta}_{X|\mathbf{U}}) \\ &= -\frac{1}{2\sigma^2}\left(-2\alpha_z\vec{z}\cdot\vec{y} + \alpha_z^2\vec{z}\cdot\vec{z}\right). \qquad (9)\end{aligned}$$

To optimize our only free parameter α_z, we use

$$\frac{\partial\Delta_{X|\mathbf{U}}(Z:\alpha_z)}{\partial\alpha_z} = -\frac{1}{2\sigma^2}\left(-2\vec{z}\cdot\vec{y} + 2\alpha_z\vec{z}\cdot\vec{z}\right) = 0 \quad \Rightarrow \quad \alpha_z = \frac{\vec{z}\cdot\vec{y}}{\vec{z}\cdot\vec{z}}.$$

Plugging this into Eq. (9), we get

$$\begin{aligned}C_1(\vec{y},\vec{z}) &\equiv \max_{\alpha_z}\Delta_{X|\mathbf{U}}(Z:\alpha_z) \\ &= -\frac{1}{2\sigma^2}\left(-2\frac{\vec{z}\cdot\vec{y}}{\vec{z}\cdot\vec{z}}\vec{z}\cdot\vec{y} + \left(\frac{\vec{z}\cdot\vec{y}}{\vec{z}\cdot\vec{z}}\right)^2\vec{z}\cdot\vec{z}\right) \\ &= \frac{1}{2\sigma^2}\frac{(\vec{z}\cdot\vec{y})^2}{\vec{z}\cdot\vec{z}}.\end{aligned}$$

The form of the similarity measure can be even further simplified

Proposition 3.3 *Let $C_1(\vec{y},\vec{z})$ be as defined above and let σ be the maximum likelihood parameter* before *Z is added as a new parent of X. Then*

$$C_1(\vec{y},\vec{z}) = \frac{M}{2}\frac{(\vec{y}\cdot\vec{z})^2}{(\vec{z}\cdot\vec{z})(\vec{y}\cdot\vec{y})} = \frac{M}{2}\cos^2\phi_{\vec{y},\vec{z}}$$

where $\phi_{\vec{y},\vec{z}}$ is the angle between the ideal parent profile vector $\vec{y}$ and the candidate parent profile vector $\vec{z}$.

Proof: To recover the maximum likelihood value of σ we differentiate the log-likelihood function as written in Eq. (5)

$$\begin{aligned}\frac{\partial\ell_X(\mathcal{D}:\mathbf{U},\theta)}{\partial\sigma^2} &= -\frac{M}{2\sigma^2} + \frac{1}{\sigma^4}\sum_m (x[m] - g(\mathbf{u}[m]))^2 = 0 \\ &\Rightarrow \quad \sigma^2 = \frac{1}{M}\sum_m (x[m] - g(\mathbf{u}[m]))^2 = \frac{1}{M}\vec{y}\cdot\vec{y}\end{aligned}$$

where the last equality follows from the definition of $\vec{y}$. The result follows immediately by plugging this into Theorem 3.2 and from the fact that $\cos^2\phi_{\vec{y},\vec{z}} \equiv \frac{(\vec{y}\cdot\vec{z})^2}{(\vec{z}\cdot\vec{z})(\vec{y}\cdot\vec{y})}$ ∎

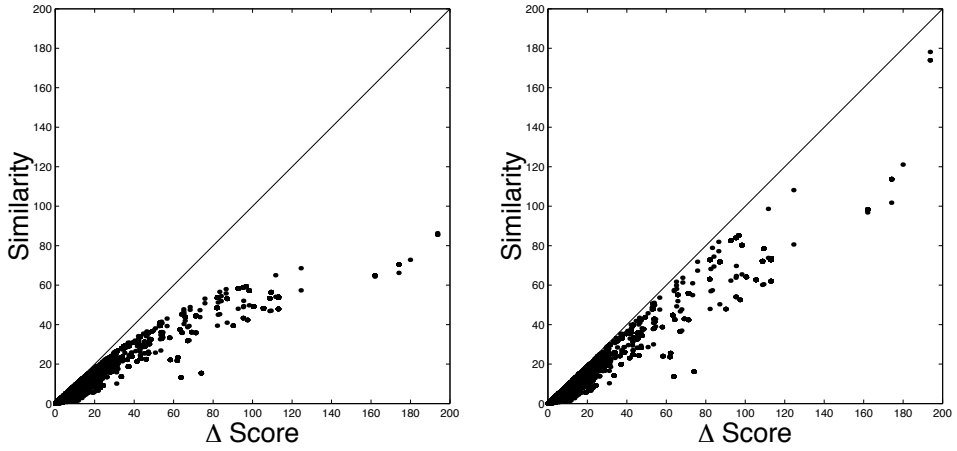


Figure 2: Demonstration of the (a) C_1 and (b) C_2 bounds for linear Gaussian CPDs. The x-axis is the true change in score as a result of an edge modification. The y-axis is the lower bound of this score. Points shown correspond to several thousand edge modifications in a run of the ideal parent method on real-life Yeast gene expressions data.

Thus, there is an intuitive geometric interpretation to the measure $C_1(\vec{y},\vec{z})$: we prefer a profile $\vec{z}$ that is similar to the ideal parent profile $\vec{y}$, regardless of its norm. It can easily be shown that $\vec{z} = c\vec{y}$ (for any constant c) maximizes this similarity measure. We retain the less intuitive form of $C_1(\vec{y},\vec{z})$ in Theorem 3.2 for compatibility with later developments.

Note that, by definition, $C_1(\vec{y},\vec{z})$ is a *lower bound* on $\Delta_{X|\mathbf{U}}(Z)$, the improvement on the log-likelihood by adding Z as a parent of X: When we add the parent we optimize all the parameters, and so we expect to attain a likelihood as high, or higher than, the one we attain by freezing some of the parameters. This is illustrated in Figure 2(a) that plots the true likelihood improvement vs. C_1 for several thousand edge modifications taken from an experiment using real life Yeast gene expression data (see Section 9).

We can get a better lower bound by optimizing additional parameters. In particular, after adding a new parent, the errors in predictions change, and so we can readjust the variance term. As it turns out, we can perform this readjustment in closed form.

Theorem 3.4 *Suppose that X has parents $\mathbf{U}$ with a set $\vec{\alpha}$ of scaling factors. Let Y be the ideal parent as described above, and Z be some candidate parent. Then the change in the log-likelihood of X in the data, when adding Z as a parent of X, while freezing all other parameters except the scaling factor of Z and the variance of X, is*

$$\begin{aligned} C_2(\vec{y},\vec{z}) &\equiv \max_{\alpha_Z,\sigma_Z} \ell_X(\mathcal{D}:\mathbf{U}\cup\{Z\},\widehat{\theta}_{X|\mathbf{U}}\cup\{\alpha_Z,\sigma_Z\}) - \ell_X(\mathcal{D}:\mathbf{U},\widehat{\theta}_{X|\mathbf{U}}) \\ &= -\frac{M}{2}\log\sin^2\phi_{\vec{y},\vec{z}} \end{aligned}$$

where $\phi_{\vec{y},\vec{z}}$ is the angle between $\vec{y}$ and $\vec{z}$.

Proof: To optimize σ_z we again consider Eq. (8) and set

$$\frac{\partial \Delta_{X|\mathbf{U}}(Z)}{\partial \sigma_z} = -\frac{M}{\sigma_z} + \frac{1}{\sigma_z^3}\left[\vec{y}\cdot\vec{y} - 2\alpha_z\vec{z}\cdot\vec{y} + \alpha_z^2\vec{z}\cdot\vec{z}\right] = 0.$$

Solving for σ_z and plugging the maximum likelihood parameter α_z from the development of $C_1(\vec{y},\vec{z})$ (which does not depend on σ_z), we get

$$\sigma_z^2 = \frac{1}{M}\left[\vec{y}\cdot\vec{y} - 2\alpha_z\vec{z}\cdot\vec{y} + \alpha_z^2\vec{z}\cdot\vec{z}\right] = \frac{1}{M}\left[\vec{y}\cdot\vec{y} - \frac{(\vec{z}\cdot\vec{y})^2}{\vec{z}\cdot\vec{z}}\right].$$

As in the case of Proposition 3.3 where $\sigma = \frac{1}{M}\vec{y}\cdot\vec{y}$, the variance term σ_z^2 "absorbs" the sum of squared errors when optimized. Thus, the second term in Eq. (8) becomes zero and we can write

$$\begin{aligned} C_2(\vec{y},\vec{z}) &= -\frac{M}{2}\left[\log(\sigma_z^2) - \log(\sigma^2)\right] \\ &= \frac{M}{2}\log\left(\frac{\vec{y}\cdot\vec{y}}{\vec{y}\cdot\vec{y} - \frac{(\vec{z}\cdot\vec{y})^2}{\vec{z}\cdot\vec{z}}}\right) = \frac{M}{2}\log\left(\frac{1}{1 - \frac{(\vec{z}\cdot\vec{y})^2}{(\vec{z}\cdot\vec{z})(\vec{y}\cdot\vec{y})}}\right) = \frac{M}{2}\log\left(\frac{1}{1-\cos^2\phi_{\vec{y},\vec{z}}}\right) \\ &= -\frac{M}{2}\log\sin^2\phi_{\vec{y},\vec{z}}. \end{aligned}$$

∎

It is important to note that both C_1 and C_2 are monotonic functions of $\frac{(\vec{y}\cdot\vec{z})^2}{\vec{z}\cdot\vec{z}}$, and so they consistently rank candidate parents of the same variable. However, when we compare changes that involve different ideal parents, such as adding a parent to X_1 compared to adding a parent to X_2, the ranking by these two measures might differ. Due to the choice of parameters we freeze in each of these measures, we have

$$C_1(\vec{y},\vec{z}) \leq C_2(\vec{y},\vec{z}) \leq \Delta_{X|\mathbf{U}}(Z)$$

and so C_2 can provide better guidance to some search algorithms. Indeed, Figure 2(b) clearly shows that C_2 is a tighter bound than C_1, particularly for promising candidates.

4. Ideal Parents in Search

The technical developments of the previous section show that we can approximate the score of candidate parents for X by comparing them to the ideal parent Y using the similarity measure. Is this approximate evaluation useful?

When performing a local heuristic search such as the one illustrated in Algorithm 1, at each iteration we have a current candidate structure and we consider some operations on that structure. These operations might include edge addition, edge replacement, edge reversal and edge deletion. We can readily use the ideal profiles and similarity measures developed to speed up two of these: edge addition and edge replacement. In a network with N nodes, there are in the order of $O(N^2)$ possible edge additions, $O(E \cdot N)$ edge replacement where E is the number of edges in the model, and only $O(E)$ edge deletions and reversals. Thus our method can be used to speed up the bulk of edge modifications considered by a typical search algorithm.

When considering adding an edge $Z \rightarrow X$, we use the ideal parent profile for X and compute its similarity to Z. We repeat this for every candidate parent for X. We then compute the full score

only for the K most similar candidates, and insert them (and the associated change in score) into a queue of potential operations. In a similar way, we can use the ideal parent profile for considering edge replacement for X. Suppose that $\mathbf{U_i} \in \mathbf{U}$ is a parent of X. We can define the ideal profile for replacing U while freezing all other parameters of the CPD of X.

Definition 4.1: Given a data set $\mathcal{D}$, and a CPD for X given its parents $\mathbf{U}$, with a link function g, parameters θ and α, the *replace ideal parent* Y of X and $\mathbf{U_i} \in \mathbf{U}$ is such that for each instance m,

$$x[m] = g(\alpha_1 u_1[m], \ldots, \alpha_{i-1} u_{i-1}, \alpha_{i+1} u_{i+1}, \ldots, \alpha_k u_k[m], y[m] : \theta).$$

∎

The rest of the developments of the previous section remain the same. For each current parent of X we compute a separate ideal profile that corresponds to the replacement of that parent with a new one. We then use the same policy as above for examining the replacement of each one of the parents. In particular, we freeze the scale parameters computed with $\mathbf{U}$ as the parent set of X, take out the parameter corresponding to $\mathbf{U_i}$, and use the C_1 or the C_2 measures to rank candidate replacements for $\mathbf{U_i}$.

For both operations, we can trade off between the accuracy of our evaluations and the speed of the search, by changing K, the number of candidate changes per family for which we compute a full score. Using $K = 1$, we only score the best candidate according to the ideal parent method ranking, thus achieving the largest speedup, However, since our ranking only approximates the true score difference, this strategy might miss good candidates. Using higher values of K brings us closer to the standard search algorithm both in terms of candidate selection quality but also in terms of computation time.

In the experiments in Section 9, we integrated the changes described above into a greedy hill climbing heuristic search procedure. This procedure also examines candidate structure changes that remove an edge and reverse an edge, which we evaluate in the standard way. The greedy hill climbing procedure applies the best available move at each iteration (among those that were chosen for full evaluation) as in Algorithm 1. Importantly, the ideal parent method is independent of the specifics of the search procedure and simply pre-selects promising candidates for the search algorithm to consider. Algorithm 2 outlines a generalization of the basic greedy structure search algorithm of Algorithm 1 to include a candidate ranking/selection algorithm such as our "Ideal Parent" method.

5. Adding New Hidden Variables

Somewhat unexpectedly, the "Ideal Parent" method also offers a natural solution to the difficult challenge of detecting new hidden variables. Specifically, the ideal parent profiles provide a straightforward way to find when and where to add hidden variables to the domain in continuous variable networks. The intuition is fairly simple: if the ideal parents of several variables are similar to each other, then we know that a similar input is predictive of all of them. Moreover, if we do not find a variable in the network that is close to these ideal parents, then we can consider adding a new hidden variable that will serve as their combined input, and, in addition, have an informed initial estimate of its profile. Figure 3 illustrates this idea.

To introduce a new hidden variable, we would like to require that it be beneficial for several children at once. The difference in log-likelihood due to adding a new parent with profile $\vec{z}$ is the

Algorithm 2: Greedy Hill-Climbing Structure Search with Candidate Ranking/Selection

```
Input  : 𝒟   // training set
         𝒢0  // initial structure
         CE  // candidate evaluation method such as our "Ideal Parent"
         K   // number of candidates to evaluate
Output : A final structure G

𝒢best ← 𝒢0
repeat
    G ← 𝒢best
    L ← ø   // initialize list of modifications to evaluate
    // for each family, choose the top 'add' and 'replace' candidates for evaluation
    foreach Xi node in G do
        Q ← ø   // initialize family specific queue
        foreach Add,Replace parent of Xi in G do
            score ← CE.Score(Operator)
            Q ← (Operator,score)
        end foreach
        foreach top K Operators in Q do
            L ← (Operator)
        end foreach
    end foreach
    // add all delete and reverse operations
    foreach Delete,Reverse edge in G do
        L ← (Operator)
    end foreach
    // process all candidate operations chosen for evaluation
    foreach Operator in L do
        if Operator does not create a directed cycle then
            𝒢' ← ApplyOperator(G)
            if Score(𝒢' : 𝒟) > Score(𝒢best : 𝒟) then
                𝒢best ← 𝒢'
            end
        end
    end foreach
until 𝒢best == G
return 𝒢best
```

sum of differences between the log-likelihoods of families it is involved in:

$$\Delta_{X_1,\ldots,X_L}(Z) = \sum_{i}^{L} \Delta_{X_i|\mathbf{U_i}}(Z)$$

where we assume, without loss of generality, that the members of the cluster (children set of the candidate hidden variable) are $X_1,\ldots,X_L$. To score the network with Z as a new hidden variable, we

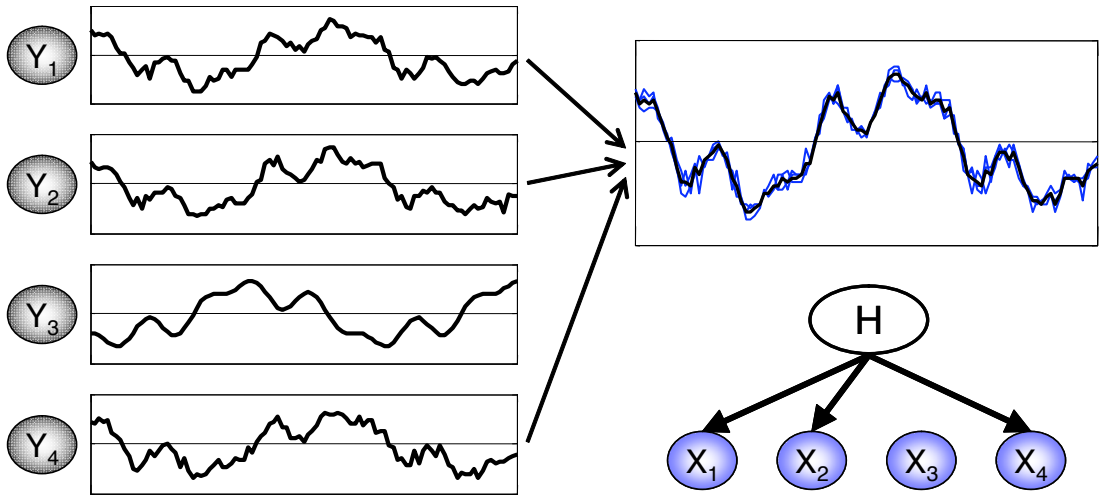


Figure 3: Illustration of how the ideal parent profiles can be used to suggest new hidden variables. Shown on the left are the ideal parent profiles $Y_1 \ldots Y_4$ of the variables $X_1 \ldots X_4$, respectively. These correspond to the residual information of these variables that is not explained by the current model. As can be seen, the first, second and fourth variables have similar ideal profiles. These profiles are averaged, resulting in a candidate hidden parent profile of these three variables (top right). Assuming that there is no variable in the network with a similar profile, our method will propose adding this hidden variable to the network as shown on the bottom right. Note that the average ideal profile of these variables provides an informed starting point for the EM algorithm.

also need to deal with the difference in the complexity penalty term, and the likelihood of Z as a root variable. These terms, however, can be readily evaluated. The difficulty is in finding the profile $\vec{z}$ that maximizes $\Delta_{X_1,\ldots,X_L}(Z)$. Using the C_1 ideal parent approximation, we can lower bound this improvement by

$$\sum_i^L C_1(\vec{y}_i,\vec{z}) \equiv \sum_i^L \frac{1}{2\sigma_i^2}\frac{(\vec{z}\cdot\vec{y}_i)^2}{\vec{z}\cdot\vec{z}} \leq \Delta_{X_1,\ldots,X_L}(Z) \tag{10}$$

and so we want to find $\vec{z}^*$ that maximizes this bound. We will then use this optimized bound as our approximate cluster score. That is we want to find

$$\vec{z}^* = \arg\max_{\vec{z}} \sum_i \frac{1}{2\sigma_i^2}\frac{(\vec{z}\cdot\vec{y}_i)^2}{\vec{z}\cdot\vec{z}} \equiv \arg\max_{\vec{z}} \frac{\vec{z}^T\mathcal{Y}\mathcal{Y}^T\vec{z}}{\vec{z}^T\vec{z}} \tag{11}$$

where $\mathcal{Y}$ is the matrix whose columns are y_i/σ_i. Note that the vector $\vec{z}^*$ must lie in the *column span* of $\mathcal{Y}$ since any component orthogonal to this span increases the denominator of the right hand term but leaves the numerator unchanged, and therefore does not obtain a maximum. We can therefore express the solution as:

$$\vec{z}^* = \sum_i v_i \frac{y_i}{\sigma_i} = \mathcal{Y}\vec{v} \tag{12}$$

where $\vec{v}$ is a vector of coefficients. Furthermore, the objective in Eq. (11) is known as the *Rayleigh quotient* of the matrix $\mathcal{Y}\mathcal{Y}^T$ and the vector $\vec{z}$. The optimum of this quotient is achieved when $\vec{z}$ equals the eigenvector of $\mathcal{Y}\mathcal{Y}^T$ corresponding to its largest eigenvalue (Wilkinson, 1965). Thus, to solve for $\vec{z^*}$ we want to solve the following eigenvector problem

$$(\mathcal{Y}\mathcal{Y}^T)\vec{z^*} = \lambda\vec{z^*}. \tag{13}$$

Note that the dimension of $\mathcal{Y}\mathcal{Y}^T$ is M (the number of instances), so that, in practice, this problem cannot be solved directly. However, by plugging Eq. (12) into Eq. (13), multiplying on the right by $\mathcal{Y}$, and defining $A \equiv \mathcal{Y}^T\mathcal{Y}$, we get a reduced generalized eigenvector problem [1]

$$AA\vec{v} = \lambda A\vec{v}.$$

Although this problem can now be solved directly, it can be further simplified by noting that A is only singular if the residue of observations of two or more variables are linearly dependent along *all* of the training instances. In practice, for continuous variables, A is indeed non-singular, and we can multiply both sides A^{-1} and end up with a simple eigenvalue problem:

$$A\vec{v} = \lambda\vec{v}$$

which is numerically simpler and easy to solve as the dimension of A is L, the number of variables in the cluster, which is typically relatively small. Once we find the L dimensional eigenvector $\vec{v}^*$ with the largest eigenvalue λ^*, we can express with it the desired parent profile $\vec{z^*}$.

We can get a better bound of $\Delta_{X_1,\ldots,X_L}(Z)$ if we use C_2 similarity rather than C_1. Unfortunately, optimizing the profile $\vec{z}$ with respect to this similarity measure is a harder problem that we do not have a closed solution for. Since the goal of the cluster identification is to provide a good starting point for the following iterations that will eventually adapt the structure, we use the closed form solution for Eq. (11). Note that once we optimize the profile z using the above derivation, we can still use the C_2 similarity score to provide a better bound on the quality of this profile as a new parent for $X_1,\ldots,X_L$.

Now that we can approximate the benefit of adding a new hidden parent to a cluster of variables, we still need to consider different clusters to find the most beneficial one. As the number of clusters is exponential, we adapt a heuristic *agglomerative clustering* approach (e.g., Duda and Hart, 1973) to explore different clusters. We start with each variable as an individual cluster and repeatedly merge the two clusters that lead to the best expected improvement (or the least decrease) in the BIC score. This procedure potentially involves $O(N^3)$ merges, where N is the number of possible variables. We save much of the computations by pre-computing the matrix $\mathcal{Y}^T\mathcal{Y}$ only once, and then using the relevant sub-matrix in each merge. In practice, the time spent in this step is insignificant in the overall search procedure.

6. Learning with Missing Values

In real-life domains, it is often the case that the data is incomplete and some of the observations are missing. Furthermore, once we add a hidden variable to the network structure, we have to copy with missing values in subsequent structure search even if the original training data was complete.

[1] In the *Generalized Eigenvector Problem*, we want to find eigenpairs $(\lambda, \vec{v})$ so that $B\vec{v} = \lambda A\vec{v}$ holds.

To deal with this problem, we use an Expectation Maximization approach (Dempster et al., 1977) and its application to network structure learning (Friedman, 1997). At each step in the search we have a current network that provides an estimate of the distribution that generated the data, and use it to compute a distribution over possible completions of the data. Instead of maximizing the BIC score, we attempt to maximize the expected BIC score

$$\boldsymbol{E}_Q[BIC(\mathcal{D},G) \mid \mathcal{D}_o] = \int Q(\mathcal{D}_h \mid \mathcal{D}_o) BIC(\mathcal{D},G) d\mathcal{D}_h$$

where $\mathcal{D}_o$ is the observed data, $\mathcal{D}_h$ is the unobserved data, and Q is the distribution represented by the current network. As the BIC score is a sum over local terms, we can use linearity of expectations to rewrite this objective as a sum of expectations, each over the scope of a single CPD. This implies that when learning with missing values, we need to use the current network to compute the posterior distribution over the values of variables in each CPD we consider. Using these posterior distributions we can estimate the expectation of each local score, and use them in standard structure search. Once the search algorithm converges, we use the new network for computing expectations and reiterate until convergence (see Friedman, 1997).

How can we combine the ideal parent method into this structural EM search? Since we do not necessarily observe X and all of its parents, the definition of the ideal parent cannot be applied directly. Instead, we define the ideal parent to be the profile that will match the expectations given Q. That is, we choose $y[m]$ so that

$$\boldsymbol{E}_Q[x[m] \mid \mathcal{D}_o] = \boldsymbol{E}_Q[g(\alpha_1 u_1[m], \ldots, \alpha_k u_k[m], y[m] : \theta) \mid \mathcal{D}_o].$$

In the case of linear CPDs, this implies that

$$\vec{y} = \boldsymbol{E}_Q[\vec{x} \mid \mathcal{D}_o] - \boldsymbol{E}_Q[\mathcal{U} \mid \mathcal{D}_o]\vec{\alpha}.$$

Once we define the ideal parent, we can use it to approximate changes in the expected BIC score (given Q). For the case of a linear Gaussian, we get terms that are similar to C_1 and C_2 of Theorem 3.2 and Theorem 3.4, respectively. The only change is that we apply the similarity measure on the expected value of $\vec{z}$ for each candidate parent Z. This is in contrast to exact evaluation of $\boldsymbol{E}_Q\left[\Delta_{X|\mathbf{U}}(Z) \mid \mathcal{D}_o\right]$, which requires the computation of the expected sufficient statistics of $\mathbf{U}, X$, and Z. To facilitate efficient computation, we adopt an approximate variational *mean-field* form (e.g., Jordan et al., 1998; Murphy and Weiss, 1999) for the posterior. This approximation is used both for the ideal parent method and the standard greedy approach used in Section 9. This results in computations that require only the first and second moments for each instance $z[m]$, and thus can be easily obtained from Q.

Finally, we note that the structural EM iterations are still guaranteed to converge to a local maximum. In fact, this does *not* depend on the fact that C_1 and C_2 are lower bounds of the true change to the score, since these measures are only used to pre-select promising candidates which are scored before actually being considered by the search algorithm. Indeed, the ideal parent method is a modular structure candidate selection algorithm and can be used as a black-box by any search algorithm.

7. Non-linear CPDs

We now address the important challenge of non-linear CPDs. In the class of CPDs we are considering, this non-linearity is mediated by the link function g, which we assume here to be invertible.

Examples of such functions include the sigmoid function shown in Eq. (4) and hyperbolic functions that are suitable for modeling gene transcription regulation (Nachman et al., 2004), among many others. When we learn with non-linear CPDs, parameter estimation is harder. To evaluate a potential parent P for X we have to perform non-linear optimization (e.g., conjugate gradient) of all of the α coefficients of all parents as well as other parameters of g. In this case, a fast approximation can boost the computational cost of the search significantly.

As in the case of linear CPDs, we compute the ideal parent profile $\vec{y}$ by inverting g. (We assume that the inversion of g can be performed in time that is proportional to the calculation of g itself as is the case for the CPDs considered above.) Suppose we are considering the addition of a parent to X in addition to its current parents $\mathbf{U}$, and that we have computed the value of the ideal parent $y[m]$ for each sample m by inversion of g. Now consider a particular candidate parent Z whose value at the mth instance is $Z[m]$. How will the difference between the ideal value and the value of Z reflect in the prediction of X for this instance?

As we have seen for the linear case in Section 3, the difference $z[m] - y[m]$ translated through g to a prediction error. In the non-linear case, the effect of the difference on predicting X depends on other factors, such as the values of the other parents. To see this, consider again the sigmoid function g of Eq. (4). If the sum of the arguments to g is close to 0, then g locally behaves like a sum of its arguments. On the other hand, if the sum is far from 0, the function is in one of the saturated regions, and big differences in the input almost do not change the prediction. This complicates our computations and does not allow the development of similarity measures as in Theorem 3.2 and Theorem 3.4 directly.

We circumvent this problem by approximating g with a linear function around the value of the ideal parent profile. We use a first-order Taylor expansion of g around the value of $\vec{y}$ and write

$$g(\vec{\mathbf{u}}, \vec{z}) \approx g(\vec{\mathbf{u}}, \vec{y}) + (\alpha_z \vec{z} - \vec{y}) \cdot \left. \frac{\partial g(\vec{\mathbf{u}}, \vec{z})}{\partial \alpha_z \vec{z}} \right|_{\alpha_z \vec{z} = \vec{y}}.$$

As a result, the "penalty" for a distance between $\vec{z}$ and $\vec{y}$ depends on the gradient of g at the particular value of $\vec{y}$, given the value of the other parents. In instances where the derivative is small, larger deviations between $y[m]$ and $z[m]$ have little impact on the likelihood of $x[m]$, and in instances where the derivative is large, the same deviations may lead to worse likelihood.

To understand the effect of this approximation in more detail we consider a simple example with a sigmoid Gaussian CPD as defined in Eq. (4), where X has no parents in the current network and Z is a candidate new parent. Figure 4(a) shows the sigmoid function (dotted) and its linear approximation at $Y = 0$ (solid) for an instance where $X = 0.5$. The computation of $Y = \log\left(\frac{1}{0.5} - 1\right) = 0$ by inversion of g is illustrated by the dashed lines. (b) is the same for a different sample where $X = 0.85$. In (c),(d) we can see the effect of the approximation for these two different samples on our evaluation of the likelihood function. For a given probability value, the likelihood function is more sensitive to changes in the value of Z around Y when $X = 0.5$ when compared to the instance $X = 0.85$. This can be seen more clearly in (e) where equi-potential contours are plotted for the sum of the approximate log-likelihood of these two instances. To recover the setup where our sensitivity to Z does *not* depend on the specific instance as in the linear case, we consider a skewed version of $Z \cdot \partial g / \partial y$ rather than Z directly. The result is shown in Figure 4(f). We can generalize the example above to develop a similarity measure for the general non-linear case:

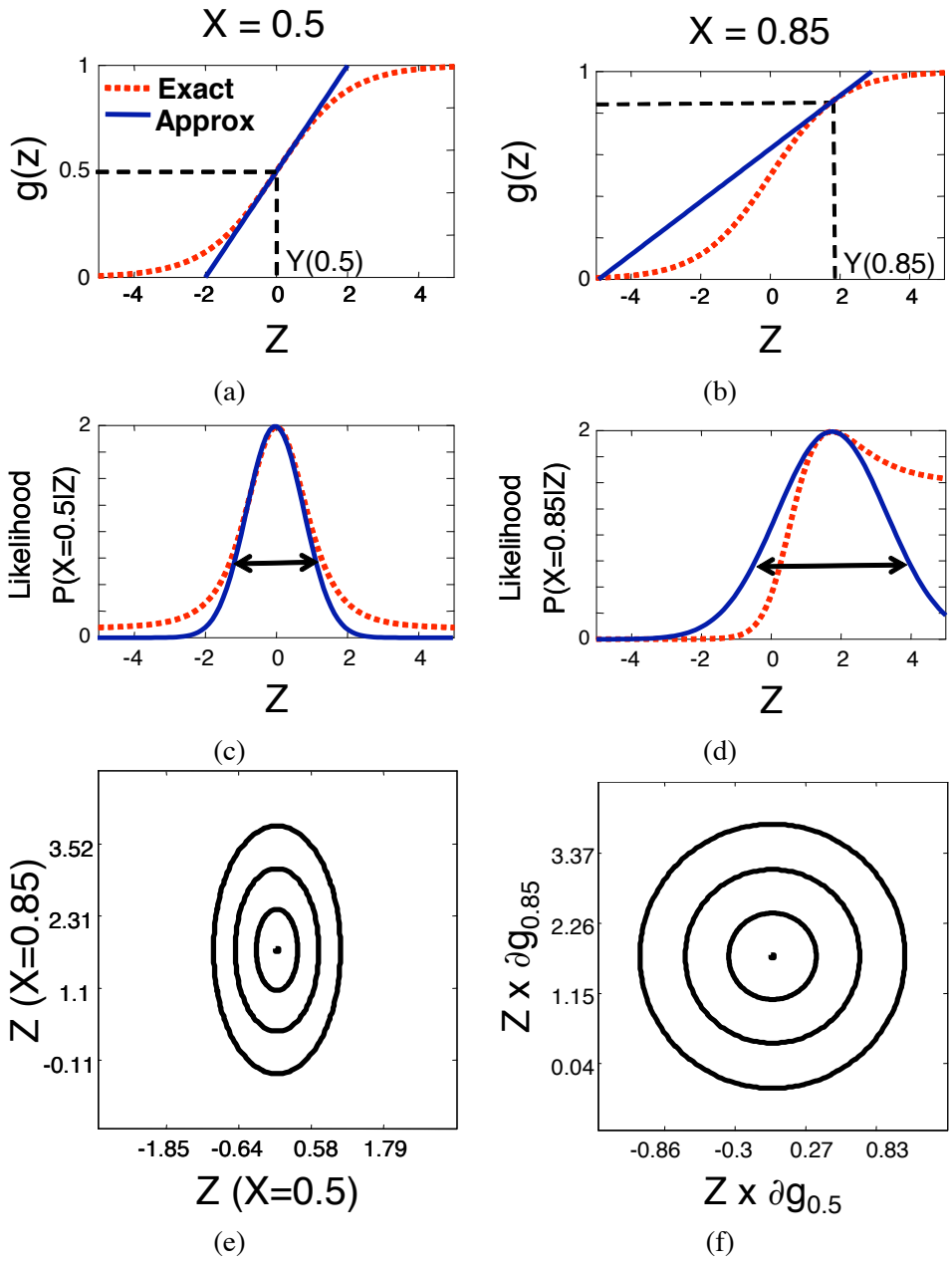


Figure 4: A simple example of the effect of the linear approximation for a sigmoid CPD where X has no parents in the current network and Z is considered as a new candidate parent. Two samples (a) and (b) show the function $g(y_1,\ldots,y_k:\theta) \equiv \theta_1 \frac{1}{1+e^{-\sum_i y_i}} + \theta_0$ for two instances where $X = 0.5$ and $X = 0.85$, respectively, along with their linear approximation at the ideal parent value Y of X. (c) and (d) show the corresponding likelihood function and its approximation. (e) shows the equi-potential contours of the sum of the log-likelihood of the two instances as a function of the value of Z in each of these instances. (f) is the same as (e) when the axes are skewed using the gradient of g with respect to the value of Y.

Theorem 7.1 *Suppose that X has parents* $\mathbf{U}$ *with a set* $\vec{\alpha}$ *of scaling factors. Let Y be the ideal parent as described above, and Z be some candidate parent. Then the change in log-likelihood of X in the data, when adding Z as a parent of X, while freezing all other parameters, is approximately*

$$C_1(\vec{y}\circ g'(\vec{y}),\vec{z}\circ g'(\vec{y}))-\frac{1}{2\sigma^2}(k_1-k_2) \tag{14}$$

where $g'(\vec{y})$ *is the vector whose mth component is* $\partial g(\vec{\alpha}\mathbf{u},y)/\partial y\,|_{\mathbf{u}[m],y[m]}$*, and* $\circ$ *denotes component-wise product. Similarly, if we also optimize the variance, then the change in log-likelihood is approximately*

$$C_2(\vec{y}\circ g'(\vec{y}),\vec{z}\circ g'(\vec{y}))-\frac{M}{2}\log\frac{k_1}{k_2}.$$

In both cases,

$$k_1=(\vec{y}\circ g'(\vec{y}))\cdot(\vec{y}\circ g'(\vec{y}))\;;\;k_2=(\vec{x}-g(\vec{\mathbf{u}}))\cdot(\vec{x}-g(\vec{\mathbf{u}}))$$

do not depend on $\vec{z}$*.*

Thus, we can use exactly the same measures as before, except that we "distort" the geometry with the weight vector $g'(y)$ that determines the importance of different instances. To approximate the likelihood difference, we also add the correction term which is a function of k_1 and k_2. This correction is not necessary when comparing two candidates for the same family, but is required for comparing candidates from different families, or when adding hidden variable. Note that unlike the linear case, and as a result of the linear approximation of g, our theorem now involves an approximation of the difference in likelihood.

Proof: Using the general form of the Taylor linear approximation for a non-linear link function g, Eq. (6) can be written as

$$\begin{aligned}
&\Delta_{X|\mathbf{U}}(Z)\\
&\approx -\frac{M}{2}\log\frac{\sigma_z^2}{\sigma^2}-\frac{1}{2}\left[\frac{1}{\sigma_z^2}\left[\vec{x}-g(\vec{\mathbf{u}},\vec{y})-(\alpha_z\vec{z}-\vec{y})\circ g'\right]^2-\frac{1}{\sigma^2}\left[\vec{x}-g(\vec{\mathbf{u}})\right]^2\right]\\
&= -\frac{M}{2}\log\frac{\sigma_z^2}{\sigma^2}-\frac{1}{2\sigma_z^2}\left[\alpha_z^2(\vec{z}\circ g')^2-2\alpha_z(\vec{z}\circ g')\cdot(\vec{y}\circ g')+(\vec{y}\circ g')^2\right]+\frac{1}{2\sigma^2}\left[\vec{x}-g(\vec{\mathbf{u}})\right]^2\\
&= -\frac{M}{2}\log\frac{\sigma_z^2}{\sigma^2}-\frac{1}{2\sigma_z^2}\left[\alpha_z^2\vec{z}_\star\cdot\vec{z}_\star-2\alpha_z\vec{z}_\star\cdot\vec{y}_\star+\vec{y}_\star\cdot\vec{y}_\star\right]+\frac{1}{2\sigma^2}\left[\vec{x}-g(\mathbf{u})\right]^2
\end{aligned} \tag{15}$$

where we use the fact that $\vec{x}-g(\vec{\mathbf{u}},\vec{y})=0$ by construction of $\vec{y}$, and we denote for clarity $\vec{y}_\star\equiv\vec{y}\circ g'$ and $\vec{z}_\star\equiv\vec{z}\circ g'$. To optimize α_z we use

$$\frac{\partial\Delta_{X|\mathbf{U}}(Z)}{\partial\alpha_z}\approx-\frac{1}{2\sigma}\left[2\alpha_z\vec{z}_\star\cdot\vec{z}_\star-2\vec{z}_\star\cdot\vec{y}_\star\right]\qquad\Rightarrow\qquad\alpha_z=\frac{\vec{z}_\star\cdot\vec{y}_\star}{\vec{z}_\star\cdot\vec{z}_\star}.$$

Plugging this into Eq. (15) we get

$$\begin{aligned}
\Delta_{X|\mathbf{U}}(Z) &\approx \frac{1}{2\sigma^2}\frac{(\vec{z}_\star\cdot\vec{y}_\star)^2}{\vec{z}_\star\cdot\vec{z}_\star}-\frac{1}{2\sigma^2}\vec{y}_\star\cdot\vec{y}_\star+\frac{1}{2\sigma^2}\left[\vec{x}-g(\vec{\mathbf{u}})\right]^2\\
&= C_1(\vec{y}_\star,\vec{z}_\star)-\frac{1}{2\sigma^2}(k_1-k_2)
\end{aligned}$$

which proves Eq. (14). When we also optimize that variance, as noted before, the variance terms absorbs the sum of squared errors, so that

$$\sigma_z = \frac{1}{M}\left[\vec{y}_\star \cdot \vec{y}_\star - \frac{(\vec{z}_\star \cdot \vec{y}_\star)^2}{\vec{z}_\star \cdot \vec{z}_\star}\right].$$

Plugging this into Eq. (15) results in

$$\begin{aligned}
\Delta_{X|\mathbf{U}}(Z) &\approx -\frac{M}{2}\log\frac{\sigma^2}{\sigma_z^2} \\
&= \frac{M}{2}\log\frac{[\vec{x}-g(\mathbf{u})]^2}{\vec{y}_\star \cdot \vec{y}_\star - \frac{(\vec{z}_\star \cdot \vec{y}_\star)^2}{\vec{z}_\star \cdot \vec{z}_\star}} = \frac{M}{2}\log\frac{[\vec{x}-g(\mathbf{u})]^2}{\vec{y}_\star \cdot \vec{y}_\star\left[1 - \frac{(\vec{z}_\star \cdot \vec{y}_\star)^2}{\vec{z}_\star \cdot \vec{z}_\star \vec{y}_\star \cdot \vec{y}_\star}\right]} \\
&= \frac{M}{2}\log\frac{1}{1-\frac{(\vec{z}_\star \cdot \vec{y}_\star)^2}{\vec{z}_\star \cdot \vec{z}_\star \vec{y}_\star \cdot \vec{y}_\star}} + \frac{M}{2}\log[\vec{x}-g(\mathbf{u})]^2 - \frac{M}{2}\log(\vec{y}_\star \cdot \vec{y}_\star) \\
&= C_2(\vec{y}_\star, \vec{z}_\star) - \frac{M}{2}\log\frac{k_1}{k_2}.
\end{aligned}$$

∎

As in the linear case, the above theorem allows us to efficiently evaluate promising candidates for the *add edge* step in the structure search. The *replace edge* step can also be approximated with minor modifications. As before, the significant gain in speed is that we only perform a few parameter optimizations (that are expected to be costly as the number of parents grows), rather than $O(N)$ such optimizations for each variable.

Adding a new hidden variable with non-linear CPDs introduces further complications. We want to use, similar to the case of a linear model, the structure score of Eq. (10) with the distorted C_1 measure. Optimizing this measure has no closed form solution in this case and we need to resort to an iterative procedure or an alternative approximation. We use an approximation where the correction terms of Eq. (14) are omitted so that a form that is similar to the linear Gaussian case is used, with the "distorted" geometry of $\vec{y}$. Having made this approximation, the rest of the details are the same as in the linear Gaussian case.

8. Other Noise Models

So far, we only considered conditional probability distributions of the form of Eq. (3) where the uncertainty is modeled using an additive Gaussian noise term. In some cases, such as when modeling biological processes related to regulation, using a multiplicative noise model may be more appropriate, as most noise sources in these domains are of multiplicative nature (Nachman et al., 2004). We can model such a noise process using CPDs of the form

$$X = g(\alpha_1\mathbf{u}_1, \ldots, \alpha_k\mathbf{u}_k : \theta)(1+\varepsilon) \tag{16}$$

where, as in Eq. (3), ε is a noise random variable with zero mean. Another popular choice for modeling multiplicative noise is the log-normal form:

$$\log(X) = \log(g(\alpha_1\mathbf{u}_1, \ldots, \alpha_k\mathbf{u}_k : \theta)) + \varepsilon$$

where the log of the random variable is distributed normally. In this section we present a formulation that generalizes the concepts introduced so far to these more general scenarios. We present explicit derivations for the multiplicative noise CPD of Eq. (16) in Section 8.3.

8.1 General Framework

To cope with CPDs that use a multiplicative noise model, we first formalize the general form of a CPD we consider. We then generalize the concept of the ideal parent to accommodate this general form of distributions and state the approximation to the likelihood we make based on this new definition. We will then show that our generalized ideal parent definition leads, as before, to a natural similarity measure that includes our previous results as a special case.

Concretely, we consider conditional density distributions of the following general form

$$P(X \mid \mathbf{U}) = q(X : g(\alpha_1 u_1[m], \ldots, \alpha_k u_k[m] : \theta), \phi)$$

where g is the link function with parameters θ as before, and q is the "noise" distribution with parameters ϕ (e.g., variance parameters). In the additive case of Eq. (3) we have $q = \mathcal{N}(X; g, \sigma^2)$. In the multiplicative case of Eq. (16) we have $q = \mathcal{N}(X; g, (g\sigma)^2)$.

We now revisit our idea of the ideal parent. Recall that our definition of the ideal parent profile $\vec{y}$ was motivated by the goal of maximizing the likelihood of the child variable profile $\vec{x}$. However, unlike the case of additive noise, in general and in the case of the multiplicative noise model, g is not necessarily the mode of q. To accommodate this, we generalize our definition of an ideal parent:

Definition 8.1: Let $\mathcal{D}$ be a data set and let $P(X \mid \mathbf{U}) = q(X : g(\mathbf{U} : \theta), \phi)$ be a CPD for X given its parents $\mathbf{U}$ with parameters θ, α and ϕ, where both q and g are twice differentiable and g is invertible with respect to each one of the parents $\mathbf{U}$. The *ideal parent* Y of X is such that for each instance m,

$$\left.\frac{\partial q(x[m]; g, \phi)}{\partial g}\right|_{g=g(\alpha_1 u_1[m],\ldots,\alpha_k u_k[m],y[m]:\theta)} = 0. \tag{17}$$

∎

That is, $\vec{y}$ is the vector that makes $g(\mathbf{u}, \vec{y})$ maximize the likelihood of the child variable at each instance. Since $\frac{\partial q}{\partial z} = \frac{\partial q}{\partial g}\frac{\partial g}{\partial z}\Big|_{z=y} = 0$, this definition also means that the ideal parent maximizes the likelihood w.r.t. the values of a new parent. The above definition is quite general and allows for a wide range of link functions and uni-modal noise models. We note that in the case where the distribution is a simple Gaussian with any choice of g, this definition coincides with Definition 3.1. As an example of a conditional form that does not fall into our framework, $g = \sin(\sum_i \alpha_i \mathbf{u}_i)$ is not only not invertible but also allows for infinitely many "ideal" parents. As we show below, this more complex definition is useful as it will allow us to efficiently evaluate candidate parents for the general CPDs we consider in this section.

The above new definition of the ideal parent motivates us to use a different approximation than the one used in the case of non-linear CPDs with additive noise. Specifically, instead of simply approximating g, we now approximate the likelihood directly around $\vec{y}$, using a second order approximation:

$$\log P(\vec{x} \mid \mathbf{u}, \alpha_z z) \approx \log P(\vec{x} \mid \mathbf{u}, \vec{y}) + (\alpha_z \vec{z} - \vec{y}) \cdot \nabla_{\alpha_z \vec{z}} \log P(\vec{x} \mid \mathbf{u}, \vec{z})\big|_{\alpha_z \vec{z} = \vec{y}} + \frac{1}{2}(\alpha_z \vec{z} - \vec{y})^T H (\alpha_z \vec{z} - \vec{y}) \tag{18}$$

where H is the Hessian matrix of $\log P(\vec{x} \mid \mathbf{u}, \vec{z})$ at the point $\alpha_z \vec{z} = \vec{y}$.

8.2 Evaluating the benefit of a Candidate Parent

With the generalized definition of an ideal parent of Eq. (17) and the approximation chosen for the likelihood function in Eq. (18) we can approximately evaluate the benefit of a candidate parent:

Theorem 8.2 *Suppose that X has parents $\mathbf{U}$ with a set $\vec{\alpha}$ of scaling factors. Let Y be the ideal parent as defined in Eq. (17), and Z be some candidate parent. Then the change in log-likelihood of X in the data, when adding Z as a parent of X, while freezing all other parameters except the scaling factor of Z, is approximately*

$$\begin{aligned} C_1(\vec{y},\vec{z}) &\approx \log P(\vec{x} \mid \mathbf{u},\vec{y}) - \max_{\alpha_Z} \frac{1}{2} K(\alpha_z\vec{z}-\vec{y},\alpha_z\vec{z}-\vec{y}) - \log P(\vec{x} \mid \mathbf{u}) \\ &= \log P(\vec{x} \mid \mathbf{u},\vec{y}) - \frac{1}{2}K(\vec{y},\vec{y}) + \frac{1}{2}\frac{(K(\vec{y},\vec{z}))^2}{K(\vec{z},\vec{z})} - \log P(\vec{x} \mid \mathbf{u}) \end{aligned} \quad (19)$$

where $K(.,.)$ is an inner product of two vectors defined as:

$$K(\vec{a},\vec{b}) = \sum_m a[m]b[m]\frac{-1}{q_m}\frac{\partial^2 q_m}{\partial {g_m}^2}\left({g'}_m\right)^2$$

and

$$\begin{aligned} g_m &= g(\mathbf{u}[m], y[m] : \theta) \\ q_m &= q(x[m] : g_m, \phi) \\ {g'}_m &= \frac{\partial g(\mathbf{u}, y : \theta)}{\partial y} \mid_{\mathbf{u}[m],y[m]} . \end{aligned}$$

Before proving this result, we first consider its elements and how they relate to our previous results of Theorem 3.2 and Theorem 7.1. The inner product K captures the deformation for the general case: The factor $({g'}_m)^2$ weighs each vector by the gradient of g, as explained in Section 7. The new factor $\frac{-1}{q_m}\frac{\partial^2 q_m}{\partial {g_m}^2}$ measures the sensitivity of q_m to changes in g_m for each instance. This factor is always positive as a maximum point of q_m is involved. Note that in the Gaussian noise models we considered in the previous sections, this term is constant: $\frac{1}{\sigma^2}$. In non-Gaussian models, this sensitivity can vary between instances.

It is easy to see that the generalized definition of C_1 coincides with our previous results. As in the linear Gaussian case, the (approximate) difference in likelihood $C_1(\vec{y},\vec{z})$ is expressed as a function of some distance between the new parent $\alpha_z\vec{z}$ and the ideal parent $\vec{y}$. This distance is then deformed by a sample dependent weight similarly to the non-linear case discussed in Section 7. In the case of a linear Gaussian CPD, we have ${g'}_m = 1$, and so $K(\vec{a},\vec{b}) = \frac{1}{\sigma^2}\vec{a}\cdot\vec{b}$. All terms which do not depend on $\vec{z}$ cancel out in this case, resulting in our original definition for C_1 in Eq. (7). For the non-linear Gaussian with additive noise, we have $K(\vec{a},\vec{b}) = \frac{1}{\sigma^2}(\vec{a}\circ g'(y))\cdot(\vec{b}\circ g'(y))$, and the form of Eq. (14) is recovered.

Importantly, we note that our new formulation is applicable to a wide range of link functions and uni-modal noise models (with the minimal restrictions detailed above). The difference between different choices simply manifest as difference in the form of the derivatives that appear in the kernel function K, and in the additional $\log P$ terms in Eq. (19). Finally, we note that we cannot derive a similarly general expression for C_2, since it requires optimizing both σ and α_z, and the solution to this problem depends on the form of the distribution q.

For completeness, we now prove the result of Theorem 8.2.

Proof: The first term in the second order approximation of Eq. (18) vanishes since, by our definition, $\frac{\partial q}{\partial z}|_{\alpha_z \vec{z}=\vec{y}} = 0$, which implies also $\frac{\partial \log(q)}{\partial z}|_{\alpha_z \vec{z}=\vec{y}} = 0$. Using the chain rule, we derive the expression for the Hessian:

$$\begin{aligned} H_{m,n} &= \left. \frac{\partial^2 \log P(\vec{x} \mid \mathbf{u}, \vec{z})}{\partial \alpha_z z[m] \partial \alpha_z z[n]} \right|_{\alpha_z \vec{z}=\vec{y}} \\ &= \delta_{mn} \frac{1}{q_m^2} \left(- \left(\frac{\partial q_m}{\partial g_m} \frac{\partial g_m}{\partial y[m]} \right)^2 + q_m \left\{ \frac{\partial^2 q_m}{\partial {g_m}^2} \left(\frac{\partial g_m}{\partial y[m]} \right)^2 + \frac{\partial q_m}{\partial g_m} \frac{\partial^2 g_m}{\partial y[m]^2} \right\} \right). \end{aligned}$$

The Hessian matrix is always diagonal, since each term in the log-likelihood involves $y[m]$ that corresponds to a single sample m. After eliminating terms involving $\frac{\partial q}{\partial g}$, the diagonal elements of the Hessian simplify to:

$$H_{m,m} = \frac{1}{q_m} \frac{\partial^2 q_m}{\partial {g_m}^2} \left({g'}_m \right)^2$$

where ${g'}_m \equiv \frac{\partial g_m}{\partial y[m]}$. With this simplification of the Hessian, the approximation of the log-likelihood can be written as

$$\log P(\vec{x} \mid \mathbf{u}, \alpha_z \vec{z}) \approx \log P(\vec{x} \mid \mathbf{u}, \vec{y}) + \frac{1}{2} \sum_m \frac{(\alpha_z z[m] - y[m])^2}{q_m} \frac{\partial^2 q_m}{\partial {g_m}^2} \left({g'}_m \right)^2. \tag{20}$$

The difference in the log-likelihood with and without a new parent z can now be immediately retrieved and equals to the second term of the right hand side of Eq. (20). Denoting this difference by $C_1(\vec{y}, \vec{z})$ and replacing α_z with its maximum likelihood estimator $\frac{K(\vec{y},\vec{z})}{K(\vec{z},\vec{z})}$, we get the desired result.
∎

8.3 Multiplicative Noise CPD

We now complete the detailed derivation of the general framework we presented in the previous section for the case of the multiplicative noise conditional density of Eq. (16). Written explicitly, the CPD has the following form:

$$q(x : g, \sigma^2) = \frac{1}{\sqrt{2\Pi}\sigma|g|} \exp\left(-\frac{1}{2\sigma^2} \left(\frac{x}{g} - 1 \right)^2 \right).$$

To avoid singularity, we will restrict the values of g to be positive. The partial derivatives of q_m are:

$$\begin{aligned} \frac{\partial q_m}{\partial g_m} &= \left[-\frac{1}{g_m} + \frac{1}{\sigma^2} \left(\frac{x}{g_m} - 1 \right) \frac{x}{g_m^2} \right] q_m \\ \frac{\partial^2 q_m}{\partial {g_m}^2} &= \left[-\frac{1}{g_m} + \frac{1}{\sigma^2} \left(\frac{x}{g_m} - 1 \right) \frac{x}{g_m^2} \right]^2 q_m + \left[\frac{1}{g_m^2} + \frac{1}{\sigma^2} \left(\frac{x}{g_m} - 1 \right) \frac{-2x}{g_m^3} - \frac{1}{\sigma^2} \frac{x^2}{g_m^4} \right] q_m. \end{aligned}$$

By the definition of $\vec{y}$ the first derivative is zero so that

$$-\frac{1}{g_m} + \frac{1}{\sigma^2} \left(\frac{x}{g_m} - 1 \right) \frac{x}{g_m^2} = 0 \tag{21}$$

which is equivalent to requiring that the following holds:

$$g(\alpha_1 u_1[m],\ldots,\alpha_k u_k[m],\vec{y}[m]:\theta) = x[m]\frac{-1+\sqrt{1+4\sigma^2}}{2\sigma^2}. \quad (22)$$

Note that the negative solution is discarded due to the constraint $g > 0$. Also note that the link function in this case is in fact, as can be expected, a scaled version of $x[m]$. We can now extract $y[m]$ as before by simply inverting g_m.

The terms of the second derivative can now also be simplified:

$$\begin{aligned}\frac{\partial^2 q}{\partial g^2} &= \left[\frac{1}{g^2}-\frac{1}{\sigma^2}\left(\frac{x}{g}-1\right)\frac{2x}{g^3}-\frac{1}{\sigma^2}\frac{x^2}{g^4}\right]q \\ &= -\frac{1}{g^2}\left[1+\frac{1}{\sigma^2}\frac{x^2}{g^2}\right]q \\ &= -\frac{1}{g^2}k_\sigma q\end{aligned}$$

where the second and third equalities result from substituting Eq. (21) and Eq. (22), respectively, and k_σ is a positive constant function of σ. We can now express K in a dot product compact form

$$K(\vec{a},\vec{b}) = k_\sigma\left(\vec{a}\circ\frac{g'(y)}{g(y)}\right)\cdot\left(\vec{b}\circ\frac{g'(y)}{g(y)}\right)$$

where $\frac{g'(y)}{g(y)}$ is the vector whose mth component is $\frac{g'_m}{g_m}$. Note that this instance specific weight is similar to the one we used for the non-linear additive Gaussian case of Theorem 7.1. In this more general setting, each instance m is additionally scaled by g_m. This has an intuitive explanation in the case of the multiplicative conditional density: the noise level is expected to go up with g and so all samples are rescaled to the same noise level.

For completeness, we write the additional $\log P$ terms in the expression of Eq. (19) for C_1 in the case of the multiplicative conditional density:

$$\begin{aligned}\log P(\vec{x}\mid\mathbf{u},\vec{y}) - \log P(\vec{x}\mid\mathbf{u}) &= -\sum\log(\sigma' g(\mathbf{u}[m],y[m])) + \sum\log(\sigma g(\mathbf{u}[m])) - \\ &\quad \frac{1}{2\sigma'^2}\sum(\frac{x[m]}{g(\mathbf{u}[m],y[m])}-1)^2 + \frac{1}{2\sigma^2}\sum(\frac{x[m]}{g(\mathbf{u}[m])}-1)^2 \\ &= -M\log\frac{-1+\sqrt{1+4\sigma'^2}}{2\sigma'} - \sum\log x[m] + \sum\log\sigma g(\mathbf{u}[m]) - \\ &\quad \frac{M}{2\sigma'^2}\left(\frac{2\sigma'^2}{-1+\sqrt{1+4\sigma'^2}}-1\right)^2 + \frac{1}{2\sigma^2}\sum(\frac{x[m]}{g(\mathbf{u}[m])}-1)^2\end{aligned}$$

where σ' denotes the new variance parameter.

9. Experiments

We now examine the impact of the ideal parent method in two settings. In the first setting, we use this method for pruning the number of potential moves that are evaluated by greedy hill climbing structure search. We use this learning procedure to learn the structure over the observed or partially

observed variables. In the second setting, we use the ideal parent method as a way of introducing new hidden variables, and also as a guide to reduce the number of evaluations when learning structure that involves hidden variables and observed ones, using a Structural EM search procedure.

9.1 Structure learning with Known Variables

In the first setting, we applied standard greedy hill climbing search (Greedy) and greedy hill climbing supplemented by the ideal parent method as discussed in Section 4 (Ideal). In using the ideal parent method, we used the C_2 similarity measure described in Section 3 to rank candidate edge additions and replacements, and then applied full scoring only to the top K ranking candidates per variable.

We first want to evaluate the impact of the approximation we make on the quality of the model learned. To do so, we start with a synthetic experiment where we know the true underlying network structure. In this setting we can evaluate the magnitude of the performance cost that is the result of the approximation we use. (We examine the speedup gain of our method on more interesting real-life examples below.) To make the synthetic experiment realistic, for the generating distribution we used a network learned from real data (see below) with 44 variables. From this network we can generate data sets of different sizes and apply our method with different values of K. Figure 5 compares the ideal parent method and the standard greedy procedure for linear Gaussian CPDs (left column) and sigmoid CPDs (right column). Using $K = 5$ is, as we expect, closer to the performance of the standard greedy method both in terms of training set [(a),(e)] and test set [(b),(f)] performance than $K = 2$. For linear Gaussian CPDs test performance is essentially the same for both methods. Using sigmoid CPDs we can see a slight advantage for the standard greedy method. When considering the percent of true edges recovered [(c),(g)], as before, the standard method shows some advantage over the ideal method with $K = 5$. However, by looking at the total number of edges learned [(d),(h)], we can see that the standard greedy method achieves this by using close to 50% more edges than the original structure for sigmoid CPDs. Thus, a relatively small advantage in performance comes at a high complexity price (and as we demonstrate below, at a significant speed cost).

We now examine the effect of the method on learning from real-life data sets. We base our data sets on a study that measures the expression of the baker's yeast genes in 173 experiments (Gasch et al., 2000). In this study, researchers measured the expression of 6152 yeast genes in its response to changes in the environmental conditions, resulting in a matrix of 173×6152 measurements. In the following, for practical reasons, we use two sets of genes. The first set consists of 639 genes that participate in general metabolic processes (Met), and the second is a subset of the first with 354 genes which are specific to amino acid metabolism (AA). We choose these sets since part of the response of the yeast to changes in its environment is in altering the activity levels of different parts of its metabolism. For some of the experiments below, we focused on subsets of genes for which there are no missing values, consisting of 89 and 44 genes, respectively. On these data sets we can consider two tasks. In the first, we treat genes as variables and experiments as instances. The learned networks indicate possible regulatory or functional connections between genes (Friedman et al., 2000). A complementary task is to treat the 173 experiments as variables (Cond). In this case the network encodes relationships between different conditions.

In Table 1 we summarize differences between the Greedy search and the Ideal search with K set to 2 and 5, for the linear Gaussian CPDs as well as sigmoid CPDs. Since the C_2 similarity is only a lower bound of the *BIC* score difference, we expect the candidate ranking of the two to be different.

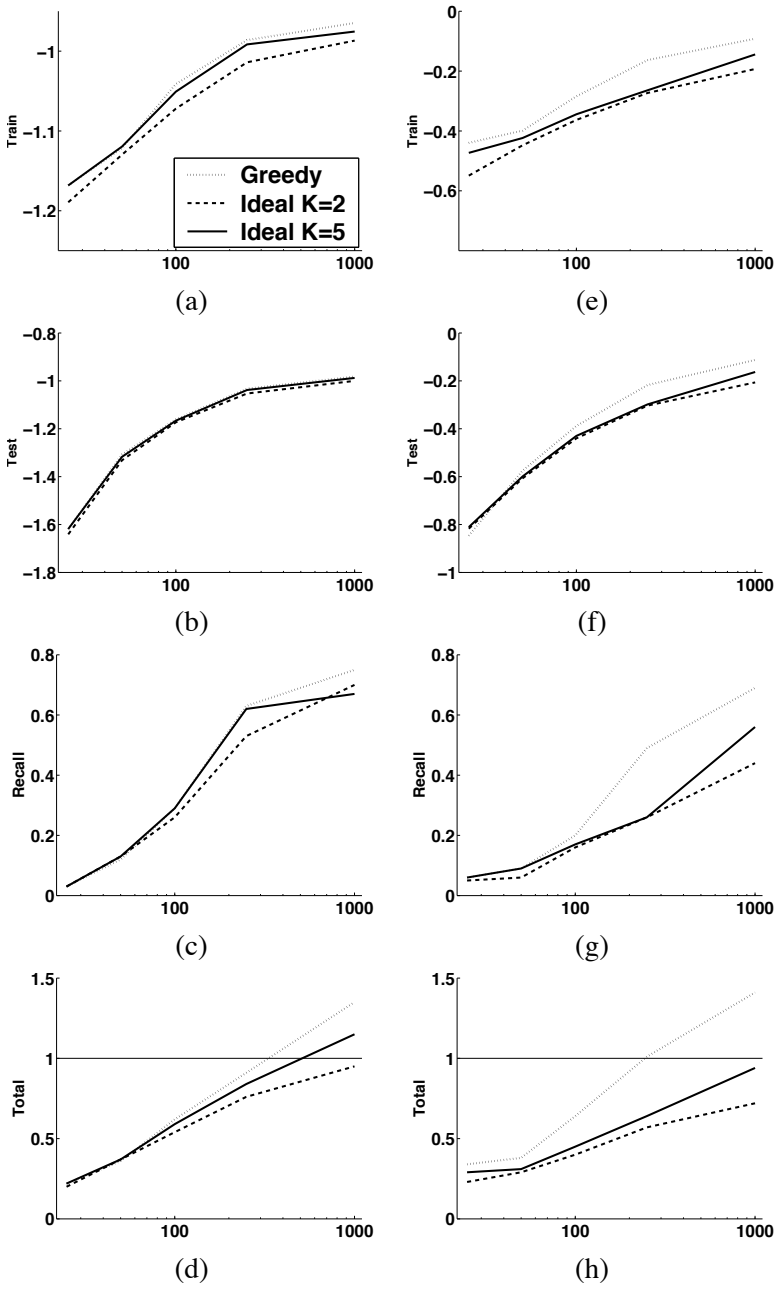


Figure 5: Evaluation of Ideal search on synthetic data generated from a real-life like network with 44 variables. We compare Ideal search with $K = 2$ (dashed) and $K = 5$ (solid), against the standard Greedy procedure (dotted). The figures show, as a function of the number of instances (x-axis), for linear Gaussian CPDs: (a) average training log-likelihood per instance per variable; (b) same for test; (c) fraction of true edges obtained in learned structure; (d) total number of edges learned as fraction of true number of edges; (e)-(h) same for sigmoid CPDs.

			Greedy		Ideal $K=2$ vs Greedy						Ideal $K=5$ vs Greedy					
Data set	vars	inst	train	test	Δtrain	Δtest	edge	move	ev	sp	Δtrain	Δtest	edge	move	ev	sp
Linear Gaussian with complete data																
AA	44	173	-0.90	-1.07	-0.024	0.006	87.1	96.5	3.6	2	-0.008	0.007	94.9	96.5	9.3	2
AA Cond	173	44	-0.59	-1.56	-0.038	0.082	92.2	92.6	1.2	2	-0.009	0.029	96.9	98.2	2.9	2
Met	89	173	-0.79	-1.00	-0.033	-0.024	88.7	91.5	1.6	3	-0.013	-0.016	94.5	96.9	4.4	2
Met Cond	173	89	-0.59	-1.06	-0.035	-0.015	91.3	98.0	1.0	2	-0.007	-0.023	98.9	98.5	2.4	2
Linear Gaussian with missing values																
AA	354	173	-0.13	-0.50	-0.101	-0.034	81.3	95.2	0.4	5	-0.048	-0.022	90.7	96.0	0.9	5
AA Cond	173	354	-0.20	-0.38	-0.066	-0.037	74.7	87.5	0.4	14	-0.033	-0.021	86.3	101.1	1.6	11
Sigmoid with complete data																
AA	44	173	0.03	-0.12	-0.132	-0.065	49.7	59.4	2.0	38	-0.103	-0.046	60.4	77.6	6.1	18
AA Cond	173	44	-0.12	-0.81	-0.218	0.122	62.3	76.7	1.0	36	-0.150	0.103	73.7	79.4	2.3	21
Met	89	173	0.12	-0.08	-0.192	-0.084	47.9	58.3	0.9	65	-0.158	-0.059	56.6	69.8	2.6	29
Met Cond	173	89	0.22	-0.17	-0.207	-0.030	60.5	69.5	0.8	53	-0.156	-0.042	69.8	77.7	2.2	29

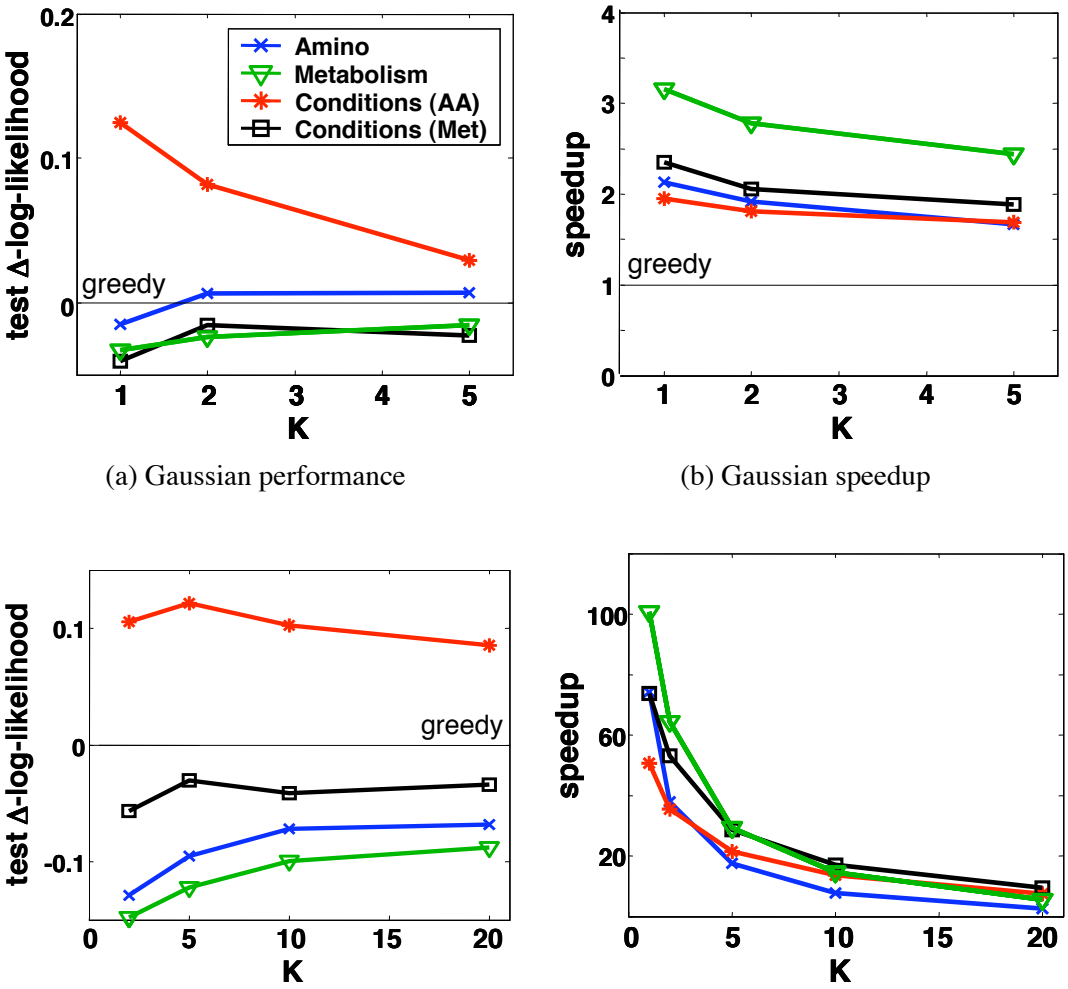
Table 1: Performance comparison of the Ideal parent search with $K=2$, $K=5$ and Greedy on real data sets. *vars* - number of variables in the data set; *inst* - the number of instances in the data set; *train* - average training set log-likelihood per instance per variable; *test* - same for test set; Δ*train* - average difference in training set log-likelihood per instance per variable; Δ*test* - same for test set; *edges* - percent of edges learned by Ideal with respect to those learned by Greedy; *moves* - percent of structure modifications taken during the search; *ev* - percent of moves evaluated; *sp* - speedup of Ideal over greedy method. All numbers are averages over 5 fold cross validation sets.

As most of the difference comes from freezing some of the parameters, a possible outcome is that the Ideal search is less prone to over-fitting. Indeed, as we see, though the training set log-likelihood in most cases is lower for Ideal search, the test set performance is only marginally different than that of the standard greedy method, and often surpasses it.

Of particular interest is the tradeoff between accuracy and speed when using the ideal parent method. In Figure 6 we examine this tradeoff in four of the data sets described above using linear Gaussian and sigmoid CPDs. For both types of CPDs, the performance of the ideal parent method approaches that of Greedy as K is increased. As we can expect, in both types of CPDs the ideal parent method is faster even for $K=5$. However, the effect on total run time is much more pronounced when learning networks with non-linear CPDs. In this case, most of the computation is spent in optimizing the parameters for scoring candidates. Indeed, careful examination of the number of structural moves taken and the number of moves evaluated in Table 1, shows that the dramatic speedup is mostly a result of the reduction in the number of candidates evaluated. Importantly, this speedup in non-linear networks makes previously "intractable" real-life learning problems (like gene regulation network inference) more accessible.

9.2 Learning Hidden Variables

In the second experimental setting, we examine the ability of our algorithm to learn structures that involve hidden variables and introduce new ones during the search. In this setting, we focus on *two layered networks* where the first layer consists of hidden variables, all of which are assumed to be roots, and the second layer consists of observed variables. Each of the observed variables is a leaf and can depend on one or more hidden variables. Learning such networks involves introducing



(a) Gaussian performance

(b) Gaussian speedup

(c) sigmoid performance

(d) sigmoid speedup

Figure 6: Evaluation of Ideal search on real-life data using 5-fold cross validation. (a) average difference in log-likelihood per instance on test data when learning with linear Gaussian CPDs relative to the Greedy baseline (y-axis) vs. the number of ideal candidates for each family K (x-axis). (b) Relative speedup over Greedy (y-axis) against K (x-axis). (c),(d) same for sigmoid CPDs.

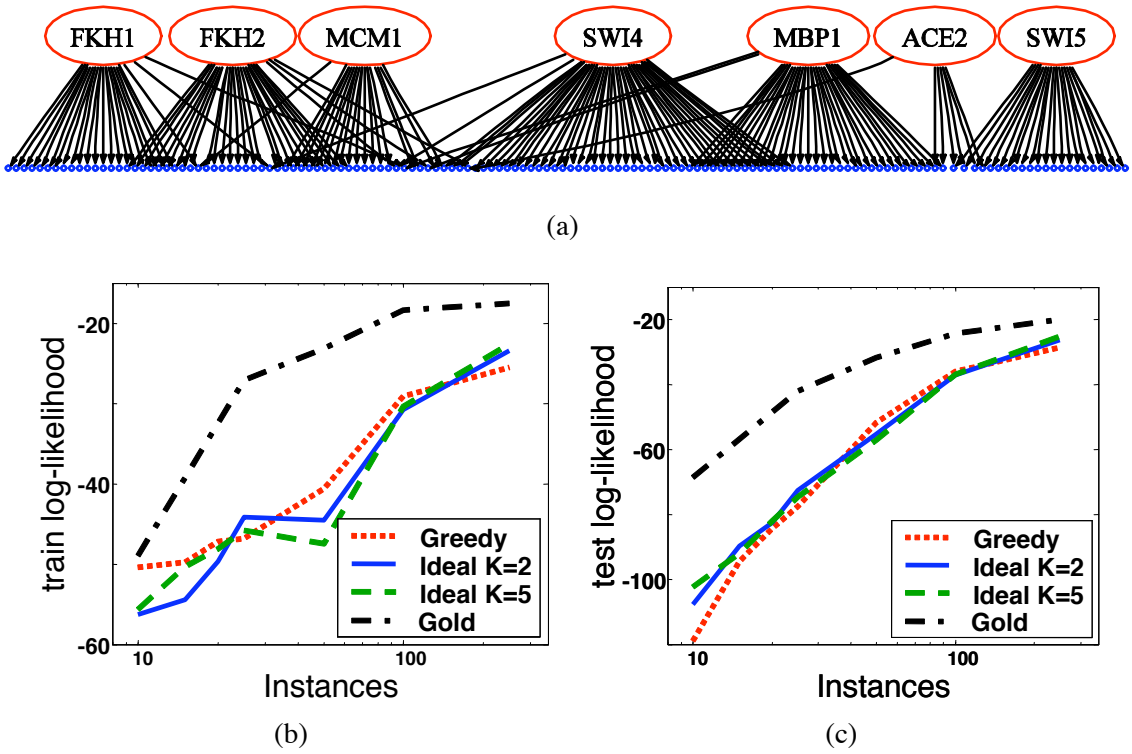


Figure 7: Evaluation of performance in two-layer network experiments. (a) Gold structure with 141 which was curated by a biological expert and used to generate synthetic data; (b) average log-likelihood per instance on *training* data (y-axis) for Greedy , Ideal search with $K = 2$ and Ideal search with $K = 5$, when learning with linear Gaussian CPDs against the number of training samples (x-axis); (c) Same for *test* set.

different hidden variables, and determining for each observed variable which hidden variables it depends on.

As in the case of standard structure learning, we first want to evaluate the impact of our approximation on learning. To test this, we used a network topology that is curated (Nachman et al., 2004) from biological literature for the regulation of cell-cycle genes in yeast. This network involves 7 hidden variables and 141 observed variables. We learned the parameters for the network from a cell cycle gene expression data set (Spellman et al., 1998). From the learned network we then sampled data sets of varying sizes, and tried to recreate the regulation structure using either greedy search or ideal parent search. In both search procedures we introduce hidden variables in a gradual manner. We start with a network where a single hidden variable is connected as the only parent to all observed variables. After parameter optimization, we introduce another hidden variable - either as a parent of all observed variables (in greedy search), or to members of the highest scoring cluster (in ideal parent search, as explained in Section 5). We then let the structure search modify edges (subject to the two-layer constraints) until no beneficial moves are found, at which point we introduce

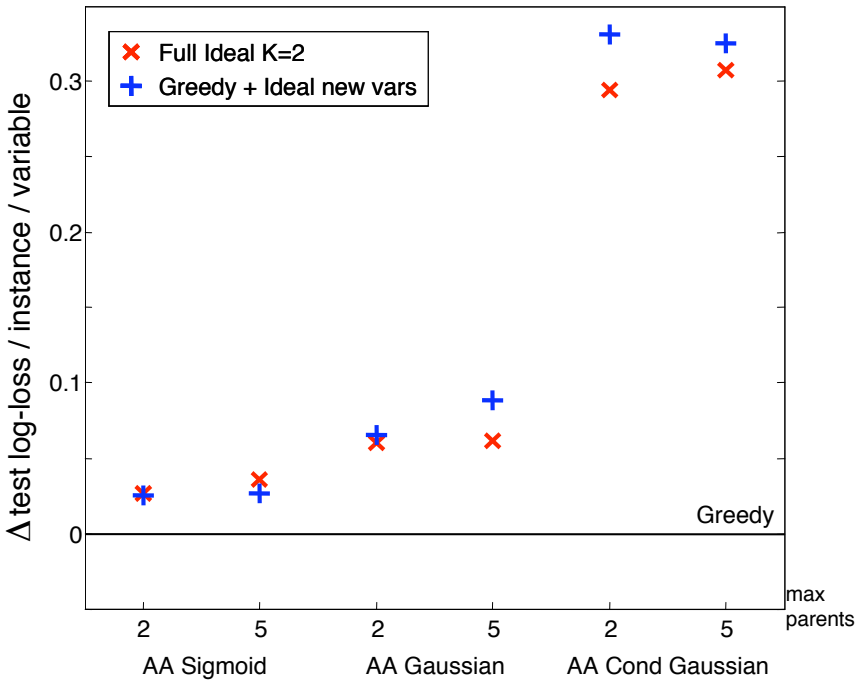


Figure 8: Structure learning of bipartite networks where the parents are new hidden variables and the children are the observed variables. The different data sets of the baker's Yeast include: AA with 44 variables for both Gaussian and sigmoid Gaussian CPDs; AA Cond with 173 variables and Gaussian CPDs. For each data set a structure with up to 2 or 5 parents was considered. Shown is the test log-likelihood per instance per variable relative to the baseline of the standard greedy structure learning algorithm.

another hidden variable, and so on. The search terminates when it is no longer beneficial to add a new variable.

Figure 7 shows the performance of the ideal parent search and the standard greedy procedure as a function of the number of instances, for linear Gaussian CPDs. As can be seen, although there are some differences in training set likelihood, the performance on test data is essentially the same. Thus, as in the case of the yeast experiments considered above, there was no degradation of performance due to the approximation made by our method.

We then considered the application of the algorithms to real-life data sets. Figure 8 shows the test set results for several of the data sets of the baker's yeast (Gasch et al., 2000) described above, for both Gaussian and sigmoid Gaussian CPDs. The full ideal parent method (red 'x') with $K = 2$ and the ideal method for adding new hidden variables is consistently better than the baseline greedy procedure. To demonstrate that the improvement is in large part due to the guided method for adding hidden variables we also ran the baseline greedy procedure for structure changes augmented with the ideal method for adding new hidden variables (blue '+'). As can be seen, the performance of this method is typically slightly better than the full ideal method, since it does not approximate the structural adaptation stage. In this setup, the only difference from the greedy baseline is the way

that new hidden variables are introduced. Thus, these results support our hypothesis that the ideal method is able to introduce effective new hidden variables, that are preferable to a hidden variables that are naively introduced into the network structure.

The superiority of the sigmoid Gaussian over the Gaussian model for the AA data set (in the order of 1 bit per instance per variable) motivates us to pursue learning of models with non-linear CPDs. We could not compare the different methods for the larger data sets as the greedy method was several orders of magnitudes slower than our ideal parent method and did not complete runs given several days of CPU time (in the linear Gaussian case the ideal parent method was roughly 5 times faster than the standard greedy approach). We believe that the ability of the ideal method to avoid over-fitting will only increase its strength in these more challenging cases.

We also considered the application of our algorithm to the real-life cell-cycle gene expression data described in the previous section with linear Gaussian CPDs. Although this data set contains only 17 samples, it is of high interest from a biological perspective to try and infer from it as much as possible on the structure of regulation. We performed leave-one-out cross validation and compared the ideal parent method with $K = 2$ and $K = 5$ to the standard greedy method. To help avoid over-fitting, we limited the number of hidden parents for each observed variable to 2. In terms of training log-likelihood per instance per variable, the greedy method is better than the ideal method by 0.4 and 0.42 bits per instance, for $K = 5$ and $K = 2$, respectively. However, its test log-likelihood performance is significantly worse as a result of high over-fitting of two particular instances, and is worse by 0.72 bits per instance than the ideal method with $K = 5$ and by 0.88 bits per instance than the ideal method with $K = 2$. As we have demonstrated in the synthetic example above, the ability of the ideal method to avoid over-fitting via a guided search, does not come at the price of diminished performance when data is more plentiful. When the observed variables were allowed to have up to 5 parents, all methods demonstrated over-fitting, which for Greedy was far more severe.

10. Discussion and Future Work

In this work we set out to learn the structure of Bayesian networks with continuous variables. Our contribution is twofold: First, we showed how to speed up structure search, particularly for non-linear conditional probability distributions. This speedup is essential as it makes structure learning feasible in many interesting real life problems. Second, we presented a principled way of introducing new hidden variables into the network structure. We used the concept of an ideal parent for both of these tasks and demonstrated its benefits on both synthetic and real-life biological domains. In particular, we showed that our method is able to effectively learn networks with hidden variables that improve generalization performance. In addition, it allowed us to cope with domains where the greedy method proved too time consuming.

Several works in recent years have tried to address the complexities involved in structure learning using different approaches. To name a few examples, Chickering (1996b) suggests searching the smaller space of Bayesian network equivalence classes. Moore and Wong (2003) suggest innovative global search operators that completely sever and reinsert a variable into the network structure. They take advantage of the fact that the set of children can be computed efficiently and use a branch and bound technique for computing the parent set. Koivisto and Sood (2004) were the first to show how the problem of exact structure learning can be made less than super-exponential by conditioning on the ordering of variables and the use of dynamic programming. Singh and Moore (2005) propose a different dynamic programming approach for learning the exact structure of Bayesian networks by

considering an alternative recursive formulation. They compare the complexity of their approach to that of Koivisto and Sood (2004) under different settings. Silander and Myllym (2006) build on the same order based idea and propose a somewhat simpler algorithm that recursively builds the network structure from the "sinks" of the optimal structure toward the roots. Teyssier and Koller (2005) perform an intelligent order-based search that is not guaranteed to find the optimal structure but significantly reduces the running time of the search procedure, and finds high scoring structures in practice.

In contrast to these approaches that focus on the search strategy, our "Ideal Parent" approach leverages on the parametric structure of the conditional distributions. This allows us to get a fast approximation of the contribution of a search operator. In here, we applied this approach in conjunction with a greedy search algorithm. However, it can also be supplemented to many other search procedures as a way of dramatically reducing the number of candidate moves that are carefully evaluated.

Two works are of particular interest and relevance to ours. Della Pietra et al. (1997) suggest an efficient method for incrementally inducing features of Markov random fields. To efficiently approximate the merit of candidate feature, they evaluate the improvement in likelihood when the only parameter that can change is the one associated with the new feature. Thus, all other parameters of the model are held fixed during the evaluation. For binary features, they find a closed-form solution for the improvement. For more general features, they use non-linear optimization to perform the evaluation. The idea of freezing some parameters in order to facilitate approximate but efficient computations is also the basis for our development of the approximate score. The context of continuous Bayesian networks, as well as the details of the likelihood functions involved in computations, however, are quite different.

Another connection is to the "Sparse Candidate" procedure of Friedman et al. (1999), which limits the number of candidate parents considered by the search procedure. While sharing the motivation of our work, their pre-pruning of candidates does not take advantage of the form of the conditional distribution nor does it try to approximate the benefit of a candidate directly. Instead, they used statistical signals as a surrogate for the benefit of a candidate parent. Thus, these methods are in fact orthogonal and it would be intriguing to see if the "Ideal Parent" method can help the "Sparse Candidate" method during the pruning stage.

The parametric form of CPDs we examined here are specific instances of *generalized linear models* (GLMs) (McCullagh and Nelder, 1989). This class of CPDs uses a function g that is applied to the sum of its arguments, called the *link function* in the GLM literature. However, we can also consider more complex functions, as long as they are well defined for any desired number of parents. For example, in Nachman et al. (2004) models based on chemical reaction models are considered, where the function g does not have a GLM form. An example of a two variable function of this type is:

$$g(y_1, y_2 : \theta) = \theta \frac{y_1 y_2}{(1+y_1)(1+y_2)}.$$

We also note that GLM literature deals extensively with different forms of noise. While we mainly focus here on the case of additive Gaussian noise, and briefly addressed other noise models, the ideas we propose here can be extended to many of these noise distributions.

Few works touched on the issue of when and how to add a hidden variable in the network structure (e.g., Elidan et al., 2001; Elidan and Friedman, 2003; Martin and VanLehn, 1995; Zhang, 2004). Only some of these methods are potentially applicable to continuous variable networks, and

none have been adapted to this context. To our knowledge, this is the first method to address this issue in a general context of continuous variable networks.

Many challenges remain. First, instead of scoring the top K candidate parents of each variable, we could evaluate only the K most promising candidates over *all* possible structure modifications. In doing so we could make use of the superiority of the $C2$ measure over the $C1$ measure, and further improve the speed of our method, possibly by another order of magnitude. Second, the "Ideal Parent" method can be combined as a plug-in for candidate selection with other innovative search procedures. Third, we want to adapt our method for additional and more complex conditional probability distributions (e.g., Nachman et al., 2004), and extend it to multi-modal distributions. Fourth, we want to improve the approximation for adding new hidden variables in the non-linear case. Finally, it might be possible to leverage on the connection to Generalized Linear Models for handling more elaborate noise models.

Acknowledgments

We thank Shai Shwartz and the anonymous reviewers for comments on earlier versions of this manuscript. This work was supported, in part, by grants from the Israeli Ministry of Science and US-Israel Bi-national Foundation. I. Nachman and G. Elidan were also supported by the Horowitz fellowship. N. Friedman was also supported in part by the Bauer Center for Genomics Research, Harvard University.

References

D. M. Chickering. Learning Bayesian networks is NP-complete. In D. Fisher and H. J. Lenz, editors, *Learning from Data: Artificial Intelligence and Statistics V*, pages 121–130. Springer-Verlag, New York, 1996a.

D. M. Chickering. Learning equivalence classes of Bayesian network structures. In E. Horvitz and F. Jensen, editors, *Proc. Twelfth Conference on Uncertainty in Artificial Intelligence (UAI '96)*, pages 150–157, San Francisco, 1996b. Morgan Kaufmann.

S. Della Pietra, V. Della Pietra, and J. Lafferty. Inducing features of random fields. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 19(4):380–393, 1997.

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*, B 39:1–39, 1977.

R. O. Duda and P. E. Hart. *Pattern Classification and Scene Analysis*. John Wiley & Sons, New York, 1973.

G. Elidan and N. Friedman. The information bottleneck EM algorithm. In C. Meek and U. Kjærulff, editors, *Proc. Nineteenth Conference on Uncertainty in Artificial Intelligence (UAI '03)*, pages 200–208, San Francisco, 2003. Morgan Kaufmann.

G. Elidan, N. Lotner, N. Friedman, and D. Koller. Discovering hidden variables: A structure-based approach. In T. K. Leen, T. G. Dietterich, and V. Tresp, editors, *Advances in Neural Information Processing Systems 13*, pages 479–485, Cambridge, Mass., 2001. MIT Press.

N. Friedman. Learning belief networks in the presence of missing values and hidden variables. In D. Fisher, editor, *Proc. Fourteenth International Conference on Machine Learning*, pages 125–133. Morgan Kaufmann, San Francisco, 1997.

N. Friedman, M. Linial, I. Nachman, and D. Pe'er. Using Bayesian networks to analyze expression data. *Computational Biology*, 7:601–620, 2000.

N. Friedman, I. Nachman, and D. Pe'er. Learning Bayesian network structure from massive data sets: The 'sparse candidate" algorithm. In K. Laskey and H. Prade, editors, *Proc. Fifteenth Conference on Uncertainty in Artificial Intelligence (UAI '99)*, page 206–215, San Francisco, 1999.

A. P. Gasch, P. T. Spellman, C. M. Kao, O. Carmel-Harel, M. B. Eisen, G. Storz, D. Botstein, and P. O. Brown. Genomic expression program in the response of yeast cells to environmental changes. *Molecular Biology of the Cell*, 11:4241–4257, 2000.

D. Geiger and D. Heckerman. Learning Gaussian networks. In R. López de Mantarás and D. Poole, editors, *Proc. Tenth Conference on Uncertainty in Artificial Intelligence (UAI '94)*, pages 235–243, San Francisco, 1994. Morgan Kaufmann.

F. Glover and M. Laguna. Tabu search. In C. Reeves, editor, *Modern Heuristic Techniques for Combinatorial Problems*, Oxford, England, 1993. Blackwell Scientific Publishing.

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*. Kluwer, Dordrecht, Netherlands, 1998.

M. Koivisto and K. Sood. Exact Bayesian structure discovery in Bayesian networks. *Journal of Machine Learning Research*, 5:549–573, 2004.

S. L. Lauritzen and N. Wermuth. Graphical models for associations between variables, some of which are qualitative and some quantitative. *Annals of Statistics*, 17:31–57, 1989.

J. Martin and K. VanLehn. Discrete factor analysis: Learning hidden variables in Bayesian networks. Technical report, Department of Computer Science, University of Pittsburgh, 1995.

P. McCullagh and J.A. Nelder. *Generalized Linear Models*. Chapman & Hall, London, 1989.

A. Moore and W. 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 (ICML '03)*, pages 552–559, Menlo Park, California, 2003.

K. Murphy and Y. Weiss. Loopy belief propagation for approximate inference: An empirical study. In K. Laskey and H. Prade, editors, *Proc. Fifteenth Conference on Uncertainty in Artificial Intelligence (UAI '99)*, page 467–475, San Francisco, 1999. Morgan Kaufmann.

I. Nachman, A. Regev, and N. Friedman. Inferring quantitative models of regulatory networks from expression data. *Bioinformatics*, 20(Suppl 1):S1248–1256, 2004.

G. Schwarz. Estimating the dimension of a model. *Annals of Statistics*, 6:461–464, 1978.

M.A. Shwe, B. Middleton, D.E. Heckerman, M. Henrion, E.J. Horvitz, H.P. Lehmann, and G.F. Cooper. Probabilistic diagnosis using a reformulation of the INTERNIST-1/QMR knowledge base. I. The probabilistic model and inference algorithms. *Methods of Information in Medicine*, 30:241–55, 1991.

T. Silander and P. Myllym. A simple approach for finding the globally optimal Bayesian network structure. In Dechter and Richardson, editors, *Proc. Twenty Second Conference on Uncertainty in Artificial Intelligence (UAI '06)*, San Francisco, 2006. Morgan Kaufmann.

A. Singh and A. Moore. Finding optimal Bayesian networks by dynamic programming. Technical report, Carnegie Mellon University, 2005.

P. T. Spellman, G. Sherlock, M. Q. Zhang, V. R. Iyer, K. Anders, M. B. Eisen, P. O. Brown, D. Botstein, and B. Futcher. Comprehensive identification of cell cycle-regulated genes of the yeast saccharomyces cerevisiae by microarray hybridization. *Molecular Biology of the Cell*, 9(12): 3273–97, 1998.

M. Teyssier and D. Koller. Ordering-based search: A simple and effective algorithm for learning Bayesian networks. In F. Bacchus and T. Jaakkola, editors, *Proc. Twenty First Conference on Uncertainty in Artificial Intelligence (UAI '05)*, pages 584–590, San Francisco, 2005. Morgan Kaufmann.

R. Parr U. Lerner and D. Koller. Bayesian fault detection and diagnosis in dynamic systems. In *Proc. of the Seventeenth National Conference on Artificial Intelligence (AAAI)*, pages 531–537, 2000.

J. Wilkinson. *The Algebric Eigenvalue Problem*. Claderon Press, Oxford, 1965.

N.L. Zhang. Hierarchical latent class models for cluster analysis. *Journal of Machine Learning Research*, 5:697–723, 2004.

Journal of Machine Learning Research 8 (2007) 1835-1865 Submitted 10/06; Revised 4/07; Published 8/07

Behavioral Shaping for Geometric Concepts

Manu Chhabra MCHHABRA@CS.ROCHESTER.EDU
Department of Computer Science
University of Rochester
Rochester, NY 14627, USA

Robert A. Jacobs ROBBIE@BCS.ROCHESTER.EDU
Department of Brain and Cognitive Sciences
University of Rochester
Rochester, NY 14627, USA

Daniel Štefankovič STEFANKO@CS.ROCHESTER.EDU
Department of Computer Science
University of Rochester
Rochester, NY 14627, USA

Editor: Manfred K. Warmuth

Abstract

In a search problem, an agent uses the membership oracle of a target concept to find a positive example of the concept. In a *shaped search problem* the agent is aided by a sequence of increasingly restrictive concepts leading to the target concept (analogous to behavioral shaping). The concepts are given by membership oracles, and the agent has to find a positive example of the target concept while minimizing the total number of oracle queries. We show that for the concept class of intervals on the real line an agent using a bounded number of queries *per oracle* exists. In contrast, for the concept class of unions of two intervals on the real line no agent with a bounded number of queries per oracle exists. We then investigate the (amortized) number of queries per oracle needed for the shaped search problem over other concept classes. We explore the following methods to obtain efficient agents. For axis-parallel rectangles we use a bootstrapping technique to obtain gradually better approximations of the target concept. We show that given rectangles $R \subseteq A \subseteq \mathbb{R}^d$ one can obtain a rectangle $A' \supseteq R$ with $\mathrm{vol}(A')/\mathrm{vol}(R) \leq 2$, using only $O(d \cdot \mathrm{vol}(A)/\mathrm{vol}(R))$ random samples from A. For ellipsoids of bounded eccentricity in $\mathbb{R}^d$ we analyze a deterministic ray-shooting process which uses a sequence of rays to get close to the centroid. Finally, we use algorithms for generating random points in convex bodies (Dyer et al., 1991; Kannan et al., 1997) to give a randomized agent for the concept class of convex bodies. In the final section, we explore connections between our bootstrapping method and active learning. Specifically, we use the bootstrapping technique for axis-parallel rectangles to active learn axis-parallel rectangles under the uniform distribution in $O(d \ln(1/\varepsilon))$ labeled samples.

Keywords: computational learning theory, behavioral shaping, active learning

1. Introduction

Computational models of learning are often inspired by human cognitive phenomena. For example, the PAC model of Valiant (1984) is a model of our ability to learn concepts from positive and negative examples generated at random by the world. Human learning, however, demonstrates a richer variety of learning phenomena such as that of behavioral shaping. Behavioral shaping is a

training procedure commonly used to teach complex behaviors. Using this procedure, a complex task is taught to a learner in an incremental manner. The learner is initially rewarded for performing an easy task that coarsely resembles the target task that the teacher wants the learner to perform. Over time, the learner is rewarded for performing more difficult tasks that monotonically provide better approximations to the target task. At the end of the training sequence, the learner is rewarded only for performing the target task. Shaping was first proposed by B. F. Skinner in the 1930s (Skinner, 1938). In addition to training humans and other organisms, behavioral shaping has also been used in artificial intelligence applications such as those that arise in robotics or reinforcement learning (Dorigo and Colombetti, 1994; Mataric, 1994; Randløv and Alstrøm, 1998; Konidaris and Barto, 2006; Ng et al., 1999).

The goal of this paper is to mathematically formalize the notion of shaping, and to show that shaping makes certain tasks easier to learn. We specifically study shaping in the context of search problems (for a learning theoretic analysis of a similar search problem, see, e.g., Fine and Mansour, 2006). In a search problem, the task is to find one positive example of a concept given a membership oracle of the concept using as few queries to the oracle as possible. If a shaping sequence, which is a sequence of nested concepts, is available, it might be possible to solve the search problem with a smaller number of oracle queries. When concepts are standard geometrical concepts like rectangles, balls, ellipsoids, and convex bodies in high dimensions, we show efficient algorithms to solve the search problem using a shaping sequence.

"Reward shaping" of Ng et al. (1999) and quasi-convex optimization are related to our work. Ng et al. (1999) gave conditions under which "reward shaping" works in a reinforcement learning setting. In their framework, shaping is a transformation of the reward function and the goal is to formalize conditions under which this transformation preserves value of the underlying policies. Our framework is different from Ng et al. (1999) in at least two ways. First, we have a sequence of reward functions as compared to their one step reward transform. Second, our reward functions are binary whereas they allow general, real valued rewards.

A weaker version of the shaped search problem in which the concepts are convex and all the oracles are available to the agent simultaneously can be viewed as an instance of a quasi-convex optimization problem. Also, we cannot apply the algorithms from this area since they usually rely on an oracle (so called separation oracle) stronger than the membership oracle that we use. What makes our setting different is that the oracles are available in a fixed order, and we only have membership oracles. Our framework is motivated by behavioral shaping (Skinner, 1938), as well as practical robotics (Dorigo and Colombetti, 1994).

Our work is similar in spirit to the idea of using a helpful teacher (Goldman and Kearns, 1991; Goldman et al., 1993; Goldman and Mathias, 1993; Hegedűs, 1994). For example, Goldman and Kearns (1991) considered a model where a helpful teacher chose examples to allow a learner to uniquely identify any concept in a given concept class. Our model differs from Goldman and Kearns (1991) in the following aspects. First, the teacher in Goldman and Kearns (1991) is "active" (it is directly providing the learner with "good" examples), whereas in our model the burden of choosing good queries is on the learner. Second, the learner in Goldman and Kearns (1991) is faced with a more general task of identifying the concept, whereas our learner is solving a search problem.

The rest of the paper is organized as follows. In Section 2 we define the shaped search problem and summarize the results of the paper. In Section 3 we illustrate the shaped search problem with algorithms for the concept classes of intervals and axis-parallel rectangles. In Section 4 we use a bootstrapping algorithm to give an improved algorithm for axis-parallel rectangles. Section 5

explains and analyzes the center-point algorithm which is used in Section 6 to solve the shaped search problem for bounded eccentricity ellipsoids. Section 7 uses the techniques on sampling random points from a convex body to solve the problem for general convex bodies. In Section 8 we define the problem of one-sided active learning, and show that the bootstrapping algorithm given in Section 4 can be used to active-learn the concept class of axis-parallel rectangles with $O(d\ln(1/\varepsilon))$ labeled samples. Finally, Section 9 wraps up the paper with a discussion and possible future directions.

2. A Formal Model of Behavioral Shaping

A *search problem* is defined by a pair of sets (S,T) such that $T \subseteq S$ (S is the starting set and T is the target set). A search agent "knows" S and has to find a point $y \in T$. The set T will be given by a membership oracle. The goal of the agent is to minimize the number of queries to the oracle.

Of course without any conditions on (S,T) the agent could be searching for a needle in a haystack and require an unbounded number of queries. To make the problem reasonable we will assume that S and T come from some concept class $\mathcal{C}$ (e. g., S,T could be intervals in $\mathbb{R}$), and that the volume $\mathrm{vol}(T)$ is not too small compared to the volume $\mathrm{vol}(S)$ (i. e., T is not a needle in a haystack S).

Before formally defining a search problem we need the following standard notions from learning theory (see, e.g., Anthony and Biggs, 1992; Kearns and Vazirani, 1994). Let $(X,\mathcal{B})$ be a measurable space and let vol be a measure on $(X,\mathcal{B})$. Let $\mathcal{C} \subseteq \mathcal{B}$. The set $\mathcal{C}$ is called a *concept class* and its members are called *concepts*. We will assume that $\mathcal{C}$ comes with a *representation scheme* (see Kearns and Vazirani, 1994, Chapter 2). Examples of concept classes that we study include intervals in $\mathbb{R}$, axis-parallel rectangles in $\mathbb{R}^d$, ellipsoids in $\mathbb{R}^d$, and convex bodies in $\mathbb{R}^d$. We will restrict our attention to the Lebesgue measure on $\mathbb{R}^d$.

Definition 2.1 Let $\mathcal{C}$ be a concept class. A *search problem* is defined by a pair of concepts (S,T) such that $T \subseteq S$ and $S,T \in \mathcal{C}$. The agent has a representation of S and has to find a point in T using a membership oracle for T.

Note that for any concept class there is a natural "oblivious" randomized algorithm for the search problem: query independent uniform random points from S until you find a point in T. The expected number of queries of the algorithm is $\mathrm{vol}(S)/\mathrm{vol}(T)$. For sufficiently complicated concept classes (e. g., finite unions of intervals) the use of randomness might be inevitable—a deterministic algorithm with bounded number of queries need not exist (the question of deterministic search is related to the concept of hitting sets, see, e.g., Linial et al., 1993).

For concept classes we will consider one can find $\Omega(n)$ disjoint concepts, each of volume $\Omega(1/n)$. The following observation implies that the trivial algorithm is the best possible (up to a constant factor).

Observation 2.1 *Suppose that there exist disjoint concepts $T_1,\ldots,T_n \subseteq S$. Let i be uniformly random from $[n]$. The expected (over the choice of i) number of queries made by any (randomized) agent on (S,T_i) is $\Omega(n)$.*

In a *shaped search problem* the agent's search task will be aided by a *shaping sequence*, which is a sequence of nested sets between the S and T. The sets in the shaping sequence will be gradually

shrinking concepts from the underlying concept class $\mathcal{C}$. The rate of shrinking will be controlled by a parameter, denoted γ.

Definition 2.2 Let $\gamma \in (0,1)$. Let $\mathcal{C}$ be a concept class and let (S,T) be a search problem over $\mathcal{C}$. A sequence of concepts $S_0 \supseteq S_1 \supseteq \cdots \supseteq S_k$ is called a γ-*shaping sequence* if $S_0 = S$, $S_k = T$, and $\mathrm{vol}(S_{t+1}) \geq \gamma \mathrm{vol}(S_t)$ for all $t \in \{0,\ldots,k-1\}$.

A search agent in a shaped search problem only has access to the membership oracles of $S_1,\ldots,S_k$. However, if the agent makes a query to S_t, it can no longer make a query to S_j with $j < t$. In other words, the oracles S_t are presented to the agent in k iterations, with the agent making (zero or more) queries to the oracle S_t at iteration t. The agent successfully solves the shaped search problem if at the end of the process it outputs $x \in T$. We assume that the agent knows the value of γ and does not know the value of k. However, before the last iteration the agent is informed that it is accessing the last oracle.

We will evaluate the performance of the agent by the amortized number of membership queries per oracle (i.e., the total number of queries divided by k). We will also consider randomized agents, which are zero-error probability (Las Vegas) algorithms (i.e., they are always successful). For a randomized agent the performance is measured by the expected number of membership queries per oracle, where the expectation is taken over the coin flips of the agent. This is formalized in the following definition.

Definition 2.3 Let $\mathcal{C}$ be a concept class. Let $\mathcal{A}$ be a search agent. We say that the agent $\mathcal{A}$ solves a γ-shaped search problem using *q queries per oracle*, if for every $S,T \in \mathcal{C}$, every k, and every γ-shaping sequence $S_0,\ldots,S_k \in \mathcal{C}$ the total number of queries made by the agent is bounded by kq. If the agent is randomized we require the expected total number of queries to be bounded by kq.

Note that for $\gamma > \gamma'$ any γ-shaping sequence is a γ'-shaping sequence. Thus as $\gamma \to 1$ the shaped search problem becomes easier. We will study how γ affects the complexity of the shaped search problem.

2.1 Our Results

In order to introduce the shaped search problem, we start with a positive and a negative result for two simple concept classes (the proofs are in Section 3). First, we show that $O(1/\gamma)$ queries per oracle suffice to solve the γ-shaped search problem for the concept class of closed intervals in $\mathbb{R}$.

Proposition 2.4 *Let $\mathcal{C}$ be the concept class of closed intervals in $\mathbb{R}$. There exists a deterministic agent which for any γ uses $O(1/\gamma)$ queries per oracle to solve γ-shaped search problem.*

Next, we contrast the Proposition 2.4 by showing that for the class of "unions of two closed intervals in $\mathbb{R}$" there exists no agent that solves the γ-shaped search problem using bounded number of queries per oracle.

Proposition 2.5 *Let $\mathcal{C}$ be the concept class of unions of two closed intervals in $\mathbb{R}$. Let $\gamma \in (0,1)$. For every (randomized) agent $\mathcal{A}$ and every number q there exists a search problem (S,T), k, and a γ-shaping sequence $S_1,\ldots,S_k$ such that $\mathcal{A}$ makes more than q queries per oracle (in expectation).*

We understand the concept class of intervals completely as Proposition 2.4 can be strengthened as follows.

Proposition 2.6 *Let $\mathcal{C}$ be the concept class of closed intervals in $\mathbb{R}$. Let $f(\gamma) = 1/\gamma$ for $\gamma \leq 1/2$, and $f(\gamma) = \ln(1/\gamma)$ for $\gamma > 1/2$. There exists a deterministic agent which for any $\gamma \in (0,1)$ uses $O(f(\gamma))$ queries per oracle to solve γ-shaped search problem. On the other hand, for any $\gamma \in (0,1)$, any (randomized) agent makes at least $\Omega(f(\gamma))$ queries per oracle.*

The shaped search problem for axis-parallel rectangles in $\mathbb{R}^d$ turns out to be more complicated. Here we do not understand the dependence of the complexity of the γ-shaped search problem on γ. We present three algorithms; each algorithm works better than the other two for a certain range of γ.

We say that a randomized agent is *oblivious* if for every oracle S_t the queries to S_t which lie in S_t are independent and uniformly random in S_t.

Theorem 2.7 *Let $\mathcal{C}$ be the concept class of axis-parallel rectangles in $\mathbb{R}^d$.*

1. *For any γ there exists a randomized agent using $O(\frac{1}{\gamma} + (d + \ln\frac{1}{\gamma})\ln d)$ queries per oracle.*
2. *For any γ there exists an (oblivious) randomized agent using $O(d/\gamma)$ queries per oracle.*
3. *For any constant $\gamma > 1/2$ there exists a deterministic agent using $O(\ln d)$ queries per oracle.*

The following table compares the number of queries used by the algorithms for various values of γ.

	Alg. 1.	Alg. 2.	Alg. 3.
$\gamma = 3/4$	$O(d \ln d)$	$O(d)$	$O(\ln d)$
$\gamma = 1/4$	$O(d \ln d)$	$O(d)$	N/A
$\gamma = 1/\ln d$	$O(d \ln d)$	$O(d \ln d)$	N/A
$\gamma = 1/d$	$O(d \ln d)$	$O(d^2)$	N/A

Note that the deterministic algorithm for part 3. uses less queries than the randomized algorithm for part 2., but it only works in a very restricted range of γ. It relies on the following fact: the centroid of an axis-parallel rectangle of volume 1 is contained in every axis-parallel sub-rectangle of volume $\geq 1/2$. It would be interesting to know whether the logarithmic dependence on d could be extended for constants $\gamma \leq 1/2$, or, perhaps, a lower bound could be shown.

Question 1 *Are $\Omega(d)$ queries per oracle needed for $\gamma < 1/2$?*

The simple agent for the part 1) of Theorem 2.7 is described in Section 3.

In Section 4 we introduce the concept of "bootstrap-learning algorithm". A bootstrap-learning algorithm, given an approximation A_1 of an unknown concept $C \in \mathcal{C}$ and a membership oracle for C, outputs a better approximation A_2 of C. We show an efficient bootstrap-learning algorithm for the concept class of axis-parallel rectangles and use it to prove part 2) of Theorem 2.7.

Part 3) of Theorem 2.7 is proved in Section 5. We show how an approximate center of an axis-parallel rectangle can be maintained using only $O(\ln d)$ (amortized) queries per oracle. If γ is not too small, the center of S_t will remain inside S_{t+1} and can be "recalibrated".

The results of Section 5 suggest that maintaining an approximate centroid of the S_t is a useful approach for solving the shaped search problem. For a centrally symmetric convex body K the

following process can be used to get close to a centroid of K: pick a line ℓ through the current point, move the current point to the center of $\ell \cap K$ and repeat. If ℓ is uniformly random the process converges to the centroid of K. It would be interesting to know what parameters influence the convergence rate of this process.

Question 2 *How many iterations of the random ray-shooting are needed to get ε-close to the centroid of a (isotropic), centrally symmetric convex body K?*

We will consider the following deterministic version of the ray-shooting approach: shoot the rays in the axis directions $e_1, \ldots, e_d$, in a round-robin fashion.

Question 3 *How many iterations of the deterministic round-robin ray-shooting are needed to get ε-close to the centroid of a (isotropic), centrally symmetric convex body K?*

In Section 6 we study a variant of the deterministic ray-shooting which moves to a weighted average of the current point and the center of $K \cap \ell$. We analyze the process for the class of ellipsoids of bounded eccentricity. As a consequence we obtain:

Theorem 2.8 *Let $\mathcal{C}$ be the concept class of ellipsoids with eccentricity bounded by L. Let $\gamma > 1/2$. The γ-shaped search problem can be solved by a deterministic agent using $O(L^2 \cdot d^{3/2} \ln d)$ queries per ray-shooting oracle (a ray-shooting oracle returns the points of intersection of K with a line through a point $x \in K$)*

The requirement $\gamma > 1/2$ in Theorem 2.8 can be relaxed. Similarly, as in the axis-parallel rectangle case, we need a condition on the volume of a sub-ellipsoid of an ellipsoid E which guarantees that the sub-ellipsoid contains the centroid of E. We do not determine this condition (which is a function of L and d).

To prove Theorem 2.8 we need the following interesting technical result.

Lemma 2.9 *Let $v_1, \ldots, v_d \in \mathbb{R}^d$ be orthogonal vectors. Let $\alpha \in (0,2)$ and $L \geq 1$. Let D be an $d \times d$ diagonal matrix with diagonal entries from the interval $[1/L, 1]$. Let*

$$M(\alpha) = \prod_{i=1}^{n} \left(I - \alpha \cdot \frac{D v_i v_i^{\mathrm{T}} D}{v_i^{\mathrm{T}} D^2 v_i} \right).$$

Then

$$\|M(1/\sqrt{d})\|_2^2 \leq 1 - \frac{1}{5L^2\sqrt{d}}.$$

Using random walks and approximating ellipsoids (Bertsimas and Vempala, 2004; Kannan et al., 1997; Grötschel et al., 1988; Lovász, 1986) we can show that convexity makes the shaped search problem solvable. We obtain the following result (a sketch of the proof is in Section 7):

Theorem 2.10 *Let $\mathcal{C}$ be the concept class of compact convex bodies in $\mathbb{R}^d$. For any $\gamma \in (0,1)$ there exists a randomized agent for the γ-shaped search problem using $O(\mathrm{poly}(d, 1/\gamma))$ queries per oracle.*

3. Basic Results for Intervals and Axis-parallel Rectangles

Now we show that for intervals $O(1/\gamma)$ queries per oracle are sufficient to solve the γ-shaped search problem. For each S_t we will compute an interval $[a_t, b_t]$ containing S_t such that the length of $[a_t, b_t]$ is at most three times longer than the length of S_t. By querying S_{t+1} on a uniform set of $O(1/\gamma)$ points in $[a_t, b_t]$ we will obtain $[a_{t+1}, b_{t+1}]$.

Proof of Proposition 2.4:
The agent will compute an interval approximating S_t for $t = 0, \ldots, k$. More precisely it will compute a_t and b_t such that $S_t \subseteq [a_t, b_t]$ and $\mathrm{vol}(S_t) \geq (b_t - a_t)/3$. Initially we have $S = S_0 =: [a_0, b_0]$ and $\mathrm{vol}(S_0) = (b_0 - a_0) \geq (b_0 - a_0)/3$.

Suppose that $S_t \subseteq [a_t, b_t]$ and $\mathrm{vol}(S_t) \geq (b_t - a_t)/3$. Using an affine transformation we can, w.l.o.g., assume $a_t = 0$ and $b_t = 1$. Thus $\mathrm{vol}(S_t) \geq 1/3$ and $\mathrm{vol}(S_{t+1}) \geq \gamma/3$.

Let $Q_0 = \{0, 1\}, Q_1 = \{0, 1/2, 1\}, \ldots, Q_i = \{\, j/2^i \mid j = 0, \ldots, 2^i\}$. The agent will query S_{t+1} on all points in Q_i, $i = 0, 1, \ldots$, until it finds the smallest j such that $|Q_j \cap S_{t+1}| \geq 2$. Choose a_{t+1} and b_{t+1} such that

$$Q_j \cap S_{t+1} = \{a_{t+1} + 2^{-j}, \ldots, b_{t+1} - 2^{-j}\}.$$

For this a_{t+1}, b_{t+1} we have $S_{t+1} \subseteq [a_{t+1}, b_{t+1}]$ and $\mathrm{vol}(S_{t+1}) \geq (b_{t+1} - a_{t+1})/3$.

Note that if $2^{-i} \leq \gamma/6$ then $|A_i \cap S_{t+1}| \geq 2$. By the minimality of j we have $2^j \leq 12/\gamma$ and hence the total number of queries per oracle is $O(1/\gamma)$. ■

Proof of Proposition 2.6:
First we show the upper bound of $O(\ln(1/\gamma))$ for $\gamma > 1/2$. Let $\ell = \lceil -\frac{\ln 2}{\ln \gamma} \rceil$. Note that $\gamma^\ell \geq 1/4$. Now we use the agent from Proposition 2.4 on oracles $S_0, S_\ell, S_{2\ell}, \ldots$, and we do not query the rest of the oracles at all. The number of queries per oracle is $O(1/\ell) = O(\ln(1/\gamma))$.

Next we show the lower bound of $\Omega(1/\gamma)$ for $\gamma < 1/2$. We take a shaping sequence of length $k = 1$. Note that there exist $\lfloor 1/\gamma \rfloor$ disjoint intervals of length γ in $[0, 1]$ and hence, by Observation 2.1, the agent needs to make $\Omega(\lfloor 1/\gamma \rfloor)$ queries (per oracle).

Finally, the lower bound of $\Omega(\ln(1/\gamma))$ will follow by an information-theoretic argument. Assume that the agent is deterministic. Fix an integer k. There exist $\Omega(1/\gamma^k)$ disjoint intervals of length γ^k. For each of these intervals there exists a shaping sequence of length k ending with that interval. We will randomly pick one of these shaping sequences and present it to the agent. The agent, using Q queries, identifies which interval (out of the $\Omega(1/\gamma^k)$ intervals) we chose. This implies $E[Q] = \Omega(k \ln(1/\gamma))$, and hence the number of queries per oracle is $\Omega(\ln(1/\gamma))$. The lower bound for a randomized agent follows by Yao's min-max lemma (see, e.g., Motwani and Raghavan, 1995, Chapter 2). ■

For unions of two intervals the agent can be forced to make many queries per oracle. If one of the intervals is large and one is small then the small interval can abruptly shrink. We will use this to "hide" the target T. Then we will shrink the large interval until it disappears. Now the agent is forced to find the hidden target T, which requires a large number of queries.

Proof of Proposition 2.5:
Let $\gamma \in (0, 1)$. Let n be the smallest positive integer such that $(1 + \gamma^n)\gamma \leq 1$. Let ℓ be a positive integer such that $\gamma^\ell < 1/2$. Let T be a random interval of length $\gamma^{n+\ell}$ in $[0, \gamma^n]$. Let $S = [-1, 1]$. The

γ-shaping sequence will be the following:

$$S_t = \begin{cases} [-1,0] \cup [0,\gamma^t] & \text{for } t = 0,\ldots,n, \\ [-\gamma^{t-n-1},0] \cup T & \text{for } t = n+1,\ldots,3n+\ell+1, \\ T & \text{for } t = 3n+\ell+2. \end{cases}$$

Note that S_t is always a union of two intervals. In the first $n+1$ iterations, the right hand-side interval is shrinking. When the right-hand side interval is sufficiently small we can replace it by the "hidden" interval T. After that we shrink the left-hand side until we make it disappear.

For the sake of the lower bound argument, we will help the agent by telling it the general shape of the shaping sequence, but we will keep the location of T secret. Now, w.l.o.g, we can assume that the agent only queries points in $[0,\gamma^n]$ on the oracle for T (because for all the other queries the agent can figure the answer himself). By Observation 2.1 the agent needs $\Omega(1/\gamma^\ell)$ queries to find a point in T. Hence the number of queries per oracle is

$$\Omega\left(\frac{1/\gamma^\ell}{3n+\ell+2}\right).$$

Letting $\ell \to \infty$ we obtain that the number of queries per oracle is unbounded. ■

Now we describe an agent for axis-parallel rectangles. Let A_t be a tight rectangle containing S_t (i.e., a rectangle such that $\mathrm{vol}(A_t) \leq C \cdot \mathrm{vol}(S_t)$ for some constant C). We will sample random points in A_t until we get a point y inside S_{t+1}. Then we will shoot rays from y in the axis-parallel directions to find the approximate boundary of S_{t+1}. From this we will obtain a tight rectangle A_{t+1} containing S_{t+1}.

Proof of the part 1) of Theorem 2.7:
We will compute axis-parallel rectangles $A_0,\ldots,A_k$ such that $S_t \subseteq A_t$ and $\mathrm{vol}(A_t) \leq \mathrm{e}\,\mathrm{vol}(S_t)$ (for $t = 0,\ldots,k$). Clearly we can take $A_0 = S_0$.

Suppose that we have A_t such that $S_t \subseteq A_t$, and $\mathrm{vol}(A_t) \leq \mathrm{e}\,\mathrm{vol}(S_t)$. Using an affine transformation we can, w.l.o.g, assume $A_t = [0,1]^d$. Since the S_t form a γ-shaping sequence we have $\mathrm{vol}(S_{t+1}) \geq \gamma/\mathrm{e}$. We will sample random points from A_t, until we get a point x inside S_{t+1}. In expectation we will need to query at most e/γ points to find x.

Now that we have x we will try to approximate S_{t+1} in each dimension separately. We will find the smallest $j \geq 0$ such that $x + 2^{-j}e_1 \in S_{t+1}$, or $x - 2^{-j}e_1 \in S_{t+1}$. Only $O(-\ln w_1(S_{t+1}))$ queries are needed for this step (where $w_1(S_{t+1})$ is the width of S_{t+1} in the 1-st dimension).

Then using binary search on $[0,2^{1-j}]$ we will find y such that $x + ye_1 \in S_{t+1}$ and $x + (y + 2^{-j-1}/d)e_1 \notin S_{t+1}$. This step uses only $O(\ln d)$ queries. Similarly we will find $z \in [0,2^{1-j}]$ such that $x - ze_1 \in S_{t+1}$ and $x - (z + 2^{-j-1}/d)e_1 \notin S_{t+1}$. Note that

$$I_1 := [x - (z + 2^{-j-1}/d), x + (y + 2^{-j-1}/d)],$$

contains the projection of S_{t+1} into the 1-st dimension, and the the length of I_1 is at most $(1 + 1/d)w_1(S)$.

Analogously we compute the I_i for $i = 1,\ldots,d$. The total number of queries is

$$O(d \ln d) + O\left(-\sum_{i=1}^{d} \ln w_1(S_{t+1})\right) = O\left(d \ln d + \ln \frac{1}{\gamma}\right).$$

Let $A_{t+1} = I_1 \times \cdots \times I_d$. We have $S_{t+1} \subseteq A_{t+1}$, and $\mathrm{vol}(A_{t+1}) \leq (1 + 1/d)^d \leq \mathrm{e}$. ■

4. (α,β)-bootstrap Learning Algorithms

In this section we prove the part 2) of Theorem 2.7. We cast the proof in a general setting that we call "bootstrap-learning algorithms".

Definition 4.1 Let $\mathcal{C}, \mathcal{C}_A$ be concept classes. Let $\alpha > \beta \geq 1$. An (α,β)-*bootstrap learning algorithm for* $\mathcal{C}$ *using* $\mathcal{C}_A$ takes as an input a representation of a concept $A_1 \in \mathcal{C}_A$ and an oracle for a concept $R \in \mathcal{C}$. The concepts A_1 and R are guaranteed to satisfy $R \subseteq A_1$ and $\mathrm{vol}(A_1) \leq \alpha \cdot \mathrm{vol}(R)$. The algorithm outputs a representation of a concept $A_2 \in \mathcal{C}_A$ such that $R \subseteq A_2$ and $\mathrm{vol}(A_2) \leq \beta \cdot \mathrm{vol}(R)$. The efficiency of the algorithm is measured by the worst-case (expected) number T of oracle queries to R (i.e., we take the worst A_1 and R from $\mathcal{C}$).

If an efficient (α,β)-bootstrap learning algorithm exists for a concept class $\mathcal{C}$ then it can be used for the shaped search problem as follows.

Lemma 4.2 *Let $\mathcal{C}$, $\mathcal{C}_A$ be concept classes. Let $\alpha > \beta \geq 1$. Assume that there exists an (α,β)-bootstrap learning algorithm for $\mathcal{C}$ using $\mathcal{C}_A$ using T queries. Suppose that for every $C \in \mathcal{C}$ there exists $A \in \mathcal{C}_A$ such that $C \subseteq A$ and $\mathrm{vol}(A) \leq \beta \cdot \mathrm{vol}(C)$. Then there exists an agent which for any $\gamma \geq \beta/\alpha$ solves the γ-shaped search problem (over $\mathcal{C}$) using T queries per oracle.*

Proof :
We will compute $A_0, \ldots, A_k \in \mathcal{C}_A$ such that $S_t \subseteq A_t$ and $\mathrm{vol}(A_t) \leq \beta \cdot \mathrm{vol}(S_t)$. By the assumption of the lemma the first A_0 exists. (The starting concept S in a shaped search problem is known in advance and hence the agent can pre-compute A_0.)

Suppose that we have A_t. Then $S_{t+1} \subseteq A_t$, and $\mathrm{vol}(A_t) \leq (\beta/\gamma)\mathrm{vol}(S_{t+1}) \leq \alpha\mathrm{vol}(S_{t+1})$. Hence using the (α,β) boot-strap algorithm we can find A_{t+1}, using only T queries. ■

If one uses Lemma 4.2 to obtain an agent for the shaped search problem for $\mathcal{C}$, one should choose $\mathcal{C}_A$ which allows for an efficient (α,β)-bootstrap algorithm. Later in this section we will show that for axis-parallel rectangles one can take $\mathcal{C}_A = \mathcal{C}$, and obtain an efficient (α,β)-bootstrap algorithm. Are there concept classes for which it is advantageous to choose $\mathcal{C}_A \neq \mathcal{C}$? More generally one can ask:

Question 4 *For given concept class $\mathcal{C}$, and $\alpha > \beta \geq 1$, which concept classes $\mathcal{C}_A$ allow for an efficient (α,β)-bootstrap learning algorithm?*

We will study the following algorithm for the (α,β)-bootstrap learning problem.

input : a representation of $A_1 \in \mathcal{C}_A$ and an oracle for $R \in \mathcal{C}$
assume : $R \subseteq A_1$ and $\text{vol}(A_1) \leq \alpha \cdot \text{vol}(R)$.
output : a representation of $A_2 \in \mathcal{C}_A$, such that
$R \subseteq A_2$ and $\text{vol}(A_2) \leq \beta \cdot \text{vol}(R)$.

$S^+ \leftarrow \emptyset, S^- \leftarrow \emptyset$
repeat
  pick a random point $p \in A_1$
  if $p \in R$ **then** $S^+ \leftarrow S^+ \cup \{p\}$ **else** $S^- \leftarrow S^- \cup \{p\}$ **fi**
  $\text{Possible}_R \leftarrow \{C \in \mathcal{C} \mid S^+ \subseteq C \subseteq A_1 \setminus S^-\}$
  $v \leftarrow$ the minimal volume of a concept in Possible_R
  $A_2 \leftarrow$ a concept of minimal volume in $\mathcal{C}_A$ containing all $C \in \text{Possible}_R$
until $\text{vol}(A_2) \leq \beta \cdot v$
output a representation of A_2

Algorithm 1: Inner-Outer algorithm for (α, β)-bootstrap learning

Note that the set Possible_R contains R and hence $R \subseteq A_2$, and $v \leq \text{vol}(R)$. Thus when the algorithm terminates we have $\text{vol}(A_2) \leq \beta \cdot v \leq \beta \cdot \text{vol}(R)$. Thus we have the following observation.

Proposition 4.3 *The Inner-Outer algorithm is an* (α, β)*-bootstrap learning algorithm.*

Now we analyze the Inner-Outer algorithm for the concept class of axis-parallel rectangles with $\mathcal{C}_A = \mathcal{C}$. We will need the following technical results (the proofs are in the appendix).

Lemma 4.4 *Let $X_1, \ldots, X_n$ be i.i.d. uniformly random in the interval $[0,1]$. Then*

$$E\left[-\ln\left(\max_i X_i - \min_i X_i\right)\right] = \frac{2n-1}{n(n-1)} \leq \frac{2}{n-1} \leq \frac{4}{n}.$$

Lemma 4.5 *Let K be from the binomial distribution $B(n,p)$. Let $X_1, \ldots, X_K$ be i.i.d. uniformly random in the interval $[0,1]$. Then*

$$E[\min\{1, X_1, \ldots, X_K\}] = \frac{1-(1-p)^{n+1}}{(n+1)p} \leq \frac{1}{np}.$$

Lemma 4.6 *Let $\mathcal{C}$ be the set of axis-parallel rectangles in $\mathbb{R}^d$. Let $\mathcal{C}_A = \mathcal{C}$. The expected number of oracle calls made by the Inner-Outer algorithm is bounded by $8 + 320d\alpha/\ln\beta$.*

As an immediate corollary we obtain:

Proof of the part 2) of Theorem 2.7:
Immediate from Lemma 4.6 and Lemma 4.2. ■

Proof of Lemma 4.6:
W.l.o.g. we can assume $A_1 = [0,1]^d$. Let $R = [a_1, b_1] \times \cdots \times [a_d, b_d]$, where $0 \leq a_i \leq b_i \leq 1$, for $i = 1, \ldots, d$.

For the purpose of the analysis of the Inner-Outer algorithm we will split the algorithm into two phases of length n_1 and n_2, respectively. We will then show that with probability $1/4$ the algorithm stops after these two phases. From this it will follow that the expected number of samples used by the algorithm is at most $4(n_1 + n_2)$.

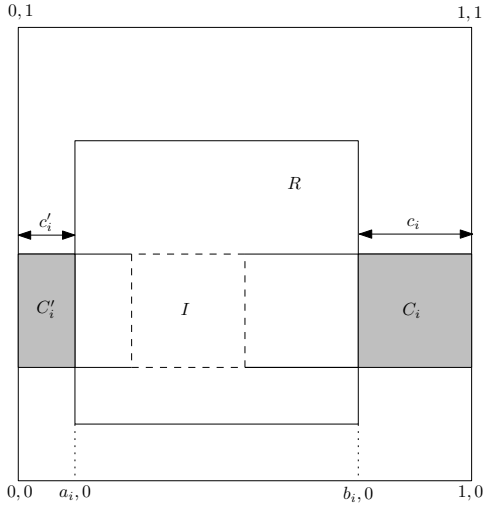


Figure 1: Schematic drawing for the proof of Lemma 4.6.

In the first phase n_1 i.i.d. random points from A_1 are sampled. The expected number of points that fall inside R is at least n_1/α. By Chernoff bound with probability at least $3/4$ we get at least $n_1/(2\alpha)$ points inside R. With probability at most $1/4$ the algorithm "fails".

From now on we assume that the algorithm did not "fail", i.e., at least $n_1/(2\alpha)$ points are inside R. Let I be the smallest rectangle containing these points. We will choose $n_1 > 4\alpha$ and hence, by Lemma 4.4, the expected logarithm of the width of I in the i-th dimension satisfies

$$E\left[-\ln \frac{w_i(I)}{b_i - a_i}\right] \leq \frac{8\alpha}{n_1}. \tag{1}$$

Summing the (1) for $i \in [d]$ we obtain (using the linearity of expectation)

$$E\left[\ln \frac{\mathrm{vol}(R)}{\mathrm{vol}(I)}\right] \leq \frac{8d\alpha}{n_1}. \tag{2}$$

Markov inequality applied to (2) yields that with probability at least $3/4$ we have

$$\ln \frac{\mathrm{vol}(R)}{\mathrm{vol}(I)} \leq \frac{32d\alpha}{n_1}. \tag{3}$$

If (3) is not satisfied we say that the algorithm "failed".

From now on we assume that (3) is true, i.e., the algorithm did not fail. Thus

$$\mathrm{vol}(I) \geq \mathrm{vol}(R) \cdot \exp\left(-\frac{32d\alpha}{n_1}\right). \tag{4}$$

Let I_i be obtained from I by stretching the i-th dimension to $[0,1]$ (for $i \in [d]$). Note that R cuts I_i into three parts. Call the parts of I_i that are outside of R, C_i and C'_i (see Figure 1). Let c_i be the width of C_i in the i-th dimension, and c'_i be the width of C'_i in the i-th dimension.

Now we sample n_2 i.i.d. random points from A_1. A point falls inside C_i with probability $\mathrm{vol}(C_i) = c_i \mathrm{vol}(I)/(b_i - a_i)$. The expected distance of the point in C_i closest to R, by Lemma 4.5, is bounded by

$$c_i \cdot \frac{1}{n_2 \cdot c_i \mathrm{vol}(I)/(b_i - a_i)} = \frac{b_i - a_i}{n_2 \mathrm{vol}(I)}. \tag{5}$$

Note that A_2 determined by the closest points in the C_i and C_i' contains $\mathrm{Possible}_R$. By (5) the expected width of A_2 in the i-th dimension satisfies $E[w_i(A_2)] \leq (b_i - a_i)(1 + 2/(n_2 \mathrm{vol}(I)))$. By Jensen's inequality

$$E\left[\ln \frac{w_i(A_2)}{b_i - a_i}\right] \leq \ln\left(1 + \frac{2}{n_2 \mathrm{vol}(I)}\right) \leq \frac{2}{n_2 \mathrm{vol}(I)}.$$

By the linearity of expectation

$$E\left[\ln \frac{\mathrm{vol}(A_2)}{\mathrm{vol}(R)}\right] \leq \frac{2d}{n_2 \mathrm{vol}(I)}.$$

By Markov inequality with probability at least $3/4$ we have

$$\ln \frac{\mathrm{vol}(A_2)}{\mathrm{vol}(R)} \leq \frac{8d}{n_2 \mathrm{vol}(I)} \leq \frac{8d\alpha}{n_2}, \tag{6}$$

and hence

$$\mathrm{vol}(R) \geq \mathrm{vol}(A_2) \cdot \exp\left(-\frac{8d\alpha}{n_2}\right). \tag{7}$$

Again, if (6) is false we say that the algorithm "failed". Note that the algorithm "failed" (in any of the three possible ways) with probability at most $3/4$. Thus with probability $1/4$ we have that (4) and (7) are true and hence

$$\mathrm{vol}(A_2) \leq \mathrm{vol}(I) \cdot \exp\left(\frac{8d\alpha}{n_2} + \frac{32d\alpha}{n_1}\right).$$

For $n_1 = \lceil 64 d\alpha / \ln \beta \rceil$ and $n_2 = \lceil 16 d\alpha / \ln \beta \rceil$ the right hand side is bounded by β and hence the algorithm will terminate.

Note that additional points in S^+ and S^- do not "hurt" the algorithm (removal of a point cannot increase v, nor can it decrease $\mathrm{vol}(A_2)$). Thus if the algorithm does not terminate, the next run of length $n_1 + n_2$ terminates with probability $\geq 1/4$, etc. Hence in expectation at most $4(n_1 + n_2)$ oracle queries suffice. ■

5. Center-point Algorithm

In this section we show how an approximate center-point of a rectangle can be maintained using only $O(\ln d)$ queries per oracle. As a corollary we obtain a proof of the part 3) of Theorem 2.7.

Given a vector $v \in \mathbb{R}^d$, let $P_i v$ be the projection of v to the i-th dimension (i.e., the vector obtained by zeroing out all the entries of v except the i-th coordinate). Let ∂S denote the boundary of the S.

Definition 5.1 Let S be an axis-parallel rectangle in $\mathbb{R}^d$. Let $\varepsilon \in (0,1)$. Let x be a point in S and let $v \in \mathbb{R}^d$, $v \geq 0$. Let $\alpha_i, \beta_i \geq 0$ be determined by $x + \alpha_i(P_i v) \in \partial S$ and $x - \beta_i(P_i v) \in \partial S$, for $i \in [d]$. We say that v is an ε-*approximate distance vector* for (S,x) if $1 - \varepsilon \leq \alpha_i \leq 1$ and $1 - \varepsilon \leq \beta_i \leq 1$ for $i \in [d]$. We say that x is an ε-*approximate center-point* of S if there exists an ε-approximate distance vector v for (S,x).

Note that if v is an ε-approximate distance vector for (S,x) then we have

$$\prod_{i=1}^{d}(2v_i) \geq \prod_{i=1}^{d}((\alpha_i+\beta_i)v_i) = \mathrm{vol}(S). \tag{8}$$

If $S' \subseteq S$ and the volume of S' is not much smaller than the volume of S then an approximate center-point x of S should be contained in S'. The next lemma formalizes this intuition.

Lemma 5.2 *Let $S' \subseteq S$ be axis-parallel rectangles. Assume that $\mathrm{vol}(S)/\mathrm{vol}(S') \leq 2-2\varepsilon$. Let $x \in S$ be an ε-approximate center-point of S. Then $x \in S'$. Moreover, for $\alpha_i', \beta_i' \geq 0$ determined by $x+\alpha_i'(P_i v) \in \partial S'$ and $x-\beta_i'(P_i v) \in \partial S'$ we have $\alpha_i'+\beta_i' \geq 1$, and $\alpha_i' \geq \varepsilon/2$, and $\beta_i' \geq \varepsilon/2$.*

Now we give an algorithm which "recalibrates" an ε-approximate center-point. Note that the first two steps of the algorithm rely on the fact that $x \in S'$ (which is guaranteed by Lemma 5.2).

input : $x \in S$, and an ε-approximate distance vector v for (S,x).
assume : $S' \subseteq S$ and $\mathrm{vol}(S) \leq (2-2\varepsilon)\mathrm{vol}(S')$
output : $x' \in S'$, and an ε-approximate distance vector v' for (S',x').

1 find $\delta^+ > 0$ such that $x+(\delta^+ + \varepsilon/8)v \notin S'$ and $x+\delta^+ v \in S'$
2 find $\delta^- > 0$ such that $x-(\delta^- + \varepsilon/8)v \notin S'$ and $x-\delta^- v \in S'$
3 let $\delta = \min\{\delta^+, \delta^-\}$, let $s=+1$ if $\delta = \delta+$ and $s=-1$ otherwise
4 **if** $\delta < 1-\varepsilon$ **then**
5 find $j \in [d]$ such that $x+s(\delta+\varepsilon/8)(P_j v) \notin S'$ and $x+s\delta(P_j v) \in S'$
6 find $\alpha > 0$ such that $x+(\alpha+\varepsilon/8)(P_j v) \notin S'$, and $x+\alpha(P_j v) \in S'$
7 find $\beta > 0$ such that $x-(\beta+\varepsilon/8)(P_j v) \notin S'$, and $x-\beta(P_j v) \in S'$
8 update $x_j \leftarrow x_j + v_j(\alpha-\beta)/2$
9 update $v_j \leftarrow (1+\varepsilon/4)((\alpha+\beta)/2)v_j$
10 go to step 1
11 return x, v

Algorithm 2: Center-point algorithm

Proof of Lemma 5.2:
Let v be an ε-approximate distance vector for (S,x). Suppose $x \notin S'$. Then there exists a coordinate $i \in [d]$ such that S' lies on one side of the hyperplane $\{z \in \mathbb{R}^d \,|\, z_i = x_i\}$. Thus the width of S' in the i-th dimension satisfies $w_i(S') \leq v_i$. On the other hand $w_i(S) \geq w_i(S') + (1-\varepsilon)v_i$. Hence $w_i(S)/w_i(S') \geq 2-\varepsilon$ which implies $\mathrm{vol}(S)/\mathrm{vol}(S') \geq 2-\varepsilon$, a contradiction. We proved $x \in S'$.

For any $i \in [d]$, using $\alpha_i, \beta_i \geq 1-\varepsilon$, we obtain

$$2-2\varepsilon \geq \frac{\mathrm{vol}(S)}{\mathrm{vol}(S')} \geq \frac{\alpha_i+\beta_i}{\alpha_i'+\beta_i'} \geq \frac{2-2\varepsilon}{\alpha_i'+\beta_i'},$$

and hence $\alpha_i'+\beta_i' \geq 1$.

Similarly, for any $i \in [d]$, using $\beta_i' \leq \beta_i \leq 1$, $\alpha_i' \leq \alpha_i$, we obtain

$$2-2\varepsilon \geq \frac{\mathrm{vol}(S)}{\mathrm{vol}(S')} \geq \frac{\alpha_i+\beta_i}{\alpha_i'+\beta_i'} \geq \frac{\alpha_i+\beta_i}{\alpha_i'+\beta_i} \geq \frac{\alpha_i+1}{\alpha_i'+1} \geq \frac{2-\varepsilon}{\alpha_i'+1}.$$

This implies $\alpha_i' \geq \varepsilon/2$. The proof of $\beta_i' \geq \varepsilon/2$ is the same. ■

By the assumptions on the input the α_i, and β_i for x, v, S are bounded by 1 from above. Hence the α_i and β_i for x, v, S' are bounded by 1 from above. Later we will show that during the execution of the algorithm the α_i and β_i for x, v, S' always remain bounded by 1 from above.

When $|\delta| \geq 1-\varepsilon$ on step 4 then $x+(1-\varepsilon)v \in S'$ and $x-(1-\varepsilon)v \in S'$, and hence the α_i and β_i are bounded by $1-\varepsilon$ from below. Thus the final v is ε-approximate distance vector for (S',x).

It remains to show that during the algorithm the α_i and β_i for x, v, S' always remain bounded by 1 from above. Let $x'_j = x_j + v_j(\alpha-\beta)/2$ and $v'_j = (1+\varepsilon/4)((\alpha+\beta)/2)v_j$, i.e., x'_j and v'_j are the new values assigned on lines 8 and 9. We have

$$x'_j + v'_j = x_j + v_j\left(\alpha + \frac{\varepsilon(\alpha+\beta)}{8}\right) \geq x_j + v_j(\alpha+\varepsilon/8), \tag{9}$$

and

$$x'_j + (1-\varepsilon)v'_j = x_j + v_j\left(\frac{\alpha-\beta}{2} + (1+\varepsilon/4)(1-\varepsilon)\frac{\alpha+\beta}{2}\right) \leq x_j + \alpha v_j. \tag{10}$$

From (9) and (10) and our choice of α on line 6 it follows that on line 10 the value of α_j for the new x and v satisfies $1-\varepsilon \leq \alpha_j \leq 1$. Similar argument establishes $1-\varepsilon \leq \beta_j \leq 1$. Note that

$$\frac{v'_j}{v_j} = (1+\varepsilon/4)\frac{\alpha+\beta}{2} \leq (1+\varepsilon/4)(1-\varepsilon/2) \leq 1-\varepsilon/4. \tag{11}$$

We will use (11) to bound the amortized number of steps we spend in our application of the center-point algorithm.

Proof of the part 3) of Theorem 2.7:
W.l.o.g., assume $S = S_0 = [0,1]^d$. Let $x^{(0)} = v^{(0)} = (1/2,\ldots,1/2)$. Note that $v^{(0)}$ is an ε-approximate distance vector for $(S_0, x^{(0)})$. We will use the center-point algorithm to compute $x^{(t)}, v^{(t)}$ such that $v^{(t)}$ is an ε-approximate distance vector for $(S_t, x^{(t)})$.

Before we start analyzing the algorithm let us emphasize that we defined "queries per oracle" to be an "amortized" quantity (as opposed to a "worst-case" quantity). Thus it is fine if the algorithm makes $\Theta(d)$ queries going from S_t to S_{t+1}, as long as the average number of queries per oracle is $O(\ln d)$.

Now we analyze the number of queries per oracle. Let

$$\Phi_t = \frac{\prod_{i=1}^d (2v_i^{(t)})}{\mathrm{vol}(S_t)}.$$

Note that $\Phi_0 = 1$, and, by (8), $\Phi_t \geq 1$ for every $t = 0,\ldots,k$.

From (11) it follows that every time the step on line 10 is executed, the value of Φ_t decreases by a factor of $(1-\varepsilon/4)$. The denominators can contribute a factor at most $(1/\gamma)^k$ to $\Phi_k \geq 1$. Thus the step 10 is executed at most $\ln\gamma^k / \ln(1-\varepsilon/4)$ times. Therefore the steps 1-9 are executed at most $k(1+\ln\gamma/\ln(1-\varepsilon/4))$ times. The steps $1,2,6,7$ use a binary search on $[0,1]$ with precision $\varepsilon/8$ and hence use $O(\ln 1/\varepsilon)$ queries.

Now we will argue that step 5 can be implemented using $O(\ln d)$ queries using binary search. Let $v = v_1 + v_2$, where the last $\lfloor d/2 \rfloor$ coordinates of v_1 are zero, and the first $\lceil d/2 \rceil$ coordinates of v_2 are zero. We know that $x + s\delta v_1 \in S'$ and $x + s\delta v_2 \in S'$ (we used the fact that S' is an axis-parallel rectangle). We also know that $x + s(\delta+\varepsilon/8)v_1 \notin S'$ or $x + s(\delta+\varepsilon/8)v_2 \notin S'$. If $x + s(\delta+\varepsilon/8)v_1 \notin S'$ then we proceed with binary search on v_1, otherwise we proceed with binary search on v_2.

Thus the total number of queries is

$$O\left(\left(\ln\frac{1}{\varepsilon} + \ln d\right) k \left(1 + \frac{\ln\gamma}{\ln(1-\varepsilon/4)}\right)\right) = O(k \ln d),$$

since $\gamma < 1/2$ is a constant, and we can take $\varepsilon = (1/2-\gamma)/4$. ■

6. Bounded-Eccentricity Ellipsoids

For a bounded-eccentricity ellipsoid K the following process converges to the centroid of K: pick a line ℓ through the current point, move the current point to the center of $\ell \cap K$ and repeat. We analyze the process in the case when the lines ℓ are chosen in axis-parallel directions in a round-robin fashion.

We say that an ellipsoid E has eccentricity bounded by L if the ratio of its axis is bounded by L. Let A be a positive definite matrix with eigenvalues from $[1/L^2, 1]$. Let E be the ellipsoid given by $x^{\mathrm{T}}Ax = 1$ (note that E has eccentricity at most L). If the current point is x and the chosen line is $\ell = \{x + \beta y \,|\, \beta \in \mathbb{R}\}$ then the midpoint of $E \cap \ell$ is

$$x' = \left(I - \frac{yy^{\mathrm{T}}A}{y^{\mathrm{T}}Ay}\right)x.$$

The process described above moves from x to x'. A more cautious process would move somewhere between x and x'. The point $y = (1-\alpha)x + \alpha x'$ is given by

$$y = \left(I - \alpha\frac{yy^{\mathrm{T}}A}{y^{\mathrm{T}}Ay}\right)x.$$

Thus one d-step round of the cautious process takes point x to the point $S(\alpha)x$, where

$$S(\alpha) = \prod_{i=1}^{d}\left(I - \alpha\frac{e_i e_i^{\mathrm{T}}A}{A_{ii}}\right) = A^{-1/2}\left(\prod_{i=1}^{d}\left(I - \alpha\frac{A^{1/2}e_i e_i^{\mathrm{T}}A^{1/2}}{e_i^{\mathrm{T}}Ae_i}\right)\right)A^{1/2}.$$

We will consider the following quantity as a measure of "distance" from the centroid: $\|A^{1/2}x\|_2^2$. After the move we have

$$\|A^{1/2}S(\alpha)x\|_2^2 = \left\|\left(\prod_{i=1}^{d}\left(I - \alpha\frac{A^{1/2}e_i e_i^{\mathrm{T}}A^{1/2}}{e_i^{\mathrm{T}}Ae_i}\right)\right)(A^{1/2}x)\right\|_2^2. \tag{12}$$

Let $A^{1/2} = V^{\mathrm{T}}DV$, where V is orthogonal. Note that the entries of D are between $1/L$ and 1. Let $v_i = Ve_i$. Now we can apply Lemma 2.9 on (12), and obtain that for $\alpha = 1/\sqrt{d}$ we have

$$\frac{\|A^{1/2}S(\alpha)x\|_2^2}{\|A^{1/2}x\|_2^2} \le 1 - \frac{1}{5L^2\sqrt{d}}. \tag{13}$$

Now we use (13) to prove Theorem 2.8.

Proof of Theorem 2.8:

The agent will compute a sequence of points $x_0, \ldots, x_k$ such that $x_t \in S_t$.

Suppose that we have $x_t \in S_t$. The ray-shooting process is invariant under translations and uniform stretching and hence, w.l.o.g., we can assume that the centroid of S_t is at 0, and $S_t = \{y \,|\, y^{\mathrm{T}}Ay \le 1\}$, where the eigenvalues of A are from $[1/L^2, 1]$. Let $\alpha = 1/\sqrt{d}$. From (13) it follows that if we apply the ray shooting process $5L^2\sqrt{d}\cdot c$ times we obtain a point z such that $\|A^{1/2}z\|_2^2 \le \mathrm{e}^{-c}$. We choose c so that $\mathrm{e}^{-c/2} \le (\gamma - 1/2)/2$.

Now we apply affine transformation such that S_t becomes a unit ball and z becomes a point at distance at most $e^{-c/2}$ from the center of S_t. Since

$$\text{vol}(S_{t+1}) \geq \gamma \text{vol}(S_t) \geq \left(\frac{1}{2} + 2e^{-c/2}\right) \text{vol}(S_t),$$

it follows that z is inside S_{t+1}, and we can take $x_{t+1} = z$. ■

Remark 1 Somewhat surprisingly the cautious process with $\alpha = 1/\sqrt{d}$ can get closer to the centroid than the original process (i.e., the one with $\alpha = 1$), see Observation A.3.

7. General Convex Bodies

In this section we show that the shaped search problem can be solved for general convex bodies. The algorithm is a simple combination of known sophisticated algorithms (e.g., ball-walk algorithm, and shallow-cut ellipsoid algorithm).

We start with an informal description of the algorithm. The agent will keep two pieces of information:

1. a collection of independent nearly-uniform random points in S_t, and
2. a weak Löwner-John pair (E, E') of ellipsoids, $E \subseteq S_t \subseteq E'$.

The random points in S_t will be so abundant that with high probability many of them will fall inside S_{t+1}. In the unlikely event that only few (or none) of the points fall inside S_{t+1} we will use E' to generate further random points in S_{t+1} (this will be very costly but unlikely).

Then we will use the random points in S_{t+1} to find an affine transformation which brings S_{t+1} into a near-isotropic position. As a consequence we will obtain a centering of S_{t+1} and we can use the shallow-cut ellipsoid algorithm (with just a membership oracle) to find a weak Löwner-John pair of ellipsoids for S_{t+1}. Finally, we will use the ball-walk algorithm (see, e.g., Kannan et al., 1997) to generate independent nearly-uniform random points inside S_{t+1}.

We will need the following definitions and results. As usual, $B(c,r)$ denotes the ball with center c and radius r.

Algorithms which deal with convex bodies given by membership oracles often require the body to be sandwiched between balls, in the following precise sense.

Definition 7.1 A (r_1, r_2)-*centered convex set* is a convex set $K \subseteq \mathbb{R}^d$ together with a point $c \in K$ such that $B(c, r_1) \subseteq K \subseteq B(c, r_2)$.

Not every convex body can be efficiently centered (e.g., if it is thin in some direction). However when we allow affine transformations of balls (i.e., ellipsoids), every convex body can be efficiently sandwiched. We will use the following relaxed notion of sandwiching.

Definition 7.2 A pair of ellipsoids (E, E') is called a *weak Löwner-John pair* for a convex body K, if $E \subseteq K \subseteq E'$, the centers of E and E' coincide, and E is obtained from E' by shrinking by a factor of $1/((d+1)\sqrt{d})$.

The following property is useful for understanding when an efficient centering is possible.

Definition 7.3 A convex set K is in *near-isotropic position* if the eigenvalues of the covariance matrix of the uniform distribution over K are from $[1/2, 3/2]$.

Our algorithm will need random samples from the uniform distribution over a convex body K. Unfortunately, uniform distribution can be difficult to achieve. The total variation distance will be used to measure the distance from uniformity.

Definition 7.4 The total *variation distance* between distributions π and μ is

$$d_{\mathrm{TV}}(\pi,\mu) = \sup_{A\subseteq\Omega}(\pi(A)-\mu(A)).$$

We will say that a distribution μ is δ-*nearly-uniform* in K if the total variation between μ and the uniform distribution on K is bounded by δ.

Some subroutines used in our algorithm require a centered convex body on their input. To be able to use these subroutines we need to find an affine transformation which makes a centering possible. Sufficiently many random points immediately will give such a transformation. The theorem below is a restatement of Corollary 11 in Bertsimas and Vempala (2004), which is based on Bourgain (1999), Rudelson (1999) and Kannan et al. (1997).

Theorem 7.5 *Using $s = O((d\ln d)\ln^2(1/\delta))$ independent samples from a δ-nearly-uniform distribution in K, one can find an affine transformation A such that $A(K)$ is in nearly-isotropic position, with probability at least $1-s\delta$.*

Once we have the convex body in a nearly isotropic position we immediately obtain a centering. The following result is Corollary 5.2 (with $\vartheta = 1/4$) in Kannan et al. (1997).

Theorem 7.6 *Assume that K is in nearly isotropic position. Then*

$$B(0,1/2) \subseteq K \subseteq B(0,2(d+1)).$$

Once the convex body K is centered we can use shallow-cut ellipsoid algorithm to sandwich K between ellipsoids. The following is Theorem 2.4.1 in Lovász (1986) (combined with Theorem 2.2.14 in Lovász, 1986).

Theorem 7.7 *Let K be a (r_1,r_2)-centered convex body given by a membership oracle. A weak Löwner-John pair for K can be found in time polynomial in d and r_2/r_1.*

Finally, we will need to be able to generate random points from convex bodies. We will use the ball-walk algorithm (see Kannan et al., 1997, Theorem 2.2).

Theorem 7.8 *Let K be a (r_1,r_2)-centered convex body. A random point from a distribution ε-close to uniform can be found in time polynomial is d, r_2/r_1, and $\ln(1/\varepsilon)$.*

Now we describe a Las Vegas algorithm for one step of the shaped search problem. The input of the algorithm is a set W of d^8/γ independent δ-nearly-uniform random set of points in S_t, and a weak Löwner-John pair (E,E') for S_t. The algorithm has access to a membership oracle of S_{t+1}, where $S_{t+1} \subseteq S_t$ and $\mathrm{vol}(S_{t+1}) \geq \gamma\mathrm{vol}(S_t)$. The output is a set of d^7/γ independent δ-nearly-uniform

random set of points in S_{t+1}, and a weak Löwner-John pair (F,F') for S_{t+1}. The algorithm runs in expected polynomial time.

We will use following objects in the algorithm. Let $Z \subseteq \mathbb{R}^d$ be the $2d$ points in which $B(0,1/2)$ intersects the axis of $\mathbb{R}^d$, i.e.,

$$Z = \{(1/2,0,\ldots,0),(-1/2,0,\ldots,0),\ldots,(0,\ldots,0,-1/2),(0,\ldots,0,1/2)\}.$$

Let r_1 be the radius of the largest ball contained in the convex hull of Z (i.e., $r_1 = 1/\sqrt{4d}$).

Finally, let $\delta = \exp(-\Theta(d^2))$.

1 $W' \leftarrow W \cap S_{t+1}$
2 Find an affine transformation A of Theorem 7.5, using points from W'. If W' does not contain enough points, let A be the identity.
3 Let r_2 be the radius of the smallest ball containing $A(E')$.
4 **if** Z is not inside $A(S_{t+1})$ or $r_2 > 4(d+1)\sqrt{d}$ **then**
5 generate independent uniformly random points from E' until we obtain d^6 random samples from S_{t+1}, let W' be the set of these new points. Go to step 2)
6 Use Theorem 7.7 to find a weak Löwner-John pair (F,F') for S_{t+1}, using centering $B(0,r_1) \subseteq A(S_{t+1}) \subseteq B(0,r_2)$.
7 Use Theorem 7.8 to find d^8/γ independent δ-nearly-uniform random points in S_{t+1}.

Algorithm 3: One step in the shaped-search algorithm for general convex bodies.

Theorem 7.9 *The algorithm 3 is correct, and its expected running-time is bounded by a polynomial in d.*

Proof :
Once we are on line 6 of the algorithm, we have

$$B(0,r_1) \subseteq Z \subseteq A(S_{t+1}) \subseteq A(E') \subseteq B(0,r_2),$$

and $r_2/r_1 \leq 8(d+1)d$. Thus $A(S_{t+1})$ is centered and the shallow-cut ellipsoid algorithm finds a weak Löwner-John pair (F,F') for S_{t+1}. Similarly the ball-walk algorithm gives δ-nearly-uniform samples from S_{t+1}. It remains to analyze lines 1-5 of the algorithm.

We enter line 5 only if $A(S_{t+1})$ is not nearly-isotropic. This can happen for two reasons: the number of points in W' is smaller than d^6, or the algorithm of Theorem 7.5 fails. Both these events have probability bounded by $\exp(-\Omega(d^2))$. The cost per sample on line 5 is $\mathrm{vol}(E')/\mathrm{vol}(S_{t+1}) = \exp(O(d\ln d))$. Hence the total contribution of line 5 to the total number of queries is $O(1)$. ■

8. Active Learning

One of the earliest works in which the framework allows the learner to choose examples is by Eisenberg and Rivest (1990). In this work, the learner does not have access to unlabeled samples, but is allowed to query the membership oracle with any instance of its choice. (see also Angluin, 1988; Bshouty and Eiron, 2003; Jackson, 1997, for a similar setting). They showed a negative result stating that there are certain concept classes which are "dense in themselves", meaning that a small number of queries (even if chosen by the learner) are not enough to determine the target concept

well. This result gave rise to a further line of work, the query by committee algorithm of Freund et al. (1997), in which the learner has access to an oracle of unlabeled samples also. Further, the learner is allowed to selectively query labels of some of these samples generated from the unlabeled oracle. Under this setting, it was shown that certain "dense in themselves" classes, for example homogeneous perceptrons under the uniform distribution, are efficiently learnable using a small number of labeled queries. Much modern active learning work uses this framework of having an unlabeled oracle and a membership oracle that can answer queries from examples generated from the unlabeled oracle. For example, Dasgupta et al. (2005) showed again (using a simpler method) that homogeneous perceptrons are learnable using only $O^*(d\ln(1/\varepsilon))$ labeled queries. Several other results are presented in Castro et al. (2006) and Dasgupta (2006). An exception to this framework is the recent work on active sampling by Fine and Mansour (2006). In their task, the learner has access to the oracle of a multi-valued function and has to find at least one instance of each example. Their work is related to our work in at least two ways. First, like us, they do not want to learn the concept, rather have just one positive example of the concept. Second, they allow the learner to choose its own examples.

The concept class of rectangles has been popular in the machine learning literature as rectangles are geometrically simple objects and also yield excellent results experimentally (Dietterich et al., 1997). Several theoretical results also exist for rectangles. For example, Auer et al. (1998) give an algorithm to PAC learn rectangles in $O(d/\varepsilon)$ queries, which matches the lower bound up to a multiplicative factor. Goldberg et al. (1994) give algorithms to learn the more complicated class of union of rectangles.

In this section, we show that rectangles are active learnable by using a variant of the bootstrap algorithm (Algorithm 1) in $O(d\ln(1/\varepsilon))$ labeled queries. We adopt the standard active learning framework of having an oracle that generates random samples and another oracle that can label these samples on request. Note that our current bootstrap algorithm does not use this flexibility and gets labeled samples uniformly at random from inside the outer body A_1 (see Algorithm 1). However, it clearly gives a (weak) upper bound to active learning the concept class of rectangles in $O(d/\varepsilon)$ labeled samples under the uniform distribution. In this section, we give a variant of the bootstrapping algorithm and show how it can be repeatedly used to active learn rectangles using both labeled and unlabeled oracles with only $O(d\ln(1/\varepsilon))$ labeled queries. Our algorithm for active learning rectangles is a one-sided active learning algorithm, that is, it outputs a hypothesis which is a superset of the target concept. We now define one-sided active learning.

Definition 8.1 A concept class $\mathcal{C}$ is *one-sided active learnable* under the uniform distribution over the instance space X if there is an algorithm, that for any concept $c \in \mathcal{C}$ and $0 < \varepsilon < 1$, gets $O(1/\varepsilon)$ samples from the uniform distribution on X and uses the membership oracle of c to label $O(\ln(1/\varepsilon))$ of these samples, and outputs a concept h such that $c \subseteq h$, and $P(h(x) \neq c(x)) < \varepsilon$.

Observation 8.1 *The concept class of axis-parallel rectangles inside the d-dimensional cube $[0,1]^d$ is not one-sided active learnable.*

Consider a rectangle with volume ε. Then we need, in expectation, $O(1/\varepsilon)$ labeled samples just to find one point inside the rectangle. Thus we are making exponentially more queries to the membership oracle than desired. Note that this is going to be a problem in learning *any* concept class which has concepts which have a small measure. For example, Dasgupta (2005) pointed out that learning non-homogeneous perceptrons when X is a unit sphere requires $\Omega(1/\varepsilon)$ labeled

samples. They overcame this problem by restricting the class to only homogeneous perceptrons (passing through the center of the sphere).

In the same spirit, we assume that our concept class has rectangles which are larger than some constant value and show that active learning is possible in this case.

Definition 8.2 The concept class $\mathcal{C}$ of *big rectangles* is a set of axis-parallel rectangles R such that $R \subset [0,1]^d$ and $\text{vol}(R) > 1/2$.

8.1 The Concept Class of Big Rectangles is One-sided Active Learnable

Throughout this section $\mathcal{C}$ denotes the concept class of axis-parallel rectangles. Note that the bootstrapping algorithm (Algorithm 1) for $\mathcal{C}$ takes as input an outer rectangle A_1 and two parameters α and β such that $\text{vol}(A_1) < \alpha \cdot \text{vol}(R)$ and outputs a rectangle A_2 such that $\text{vol}(A_2) < \beta \cdot \text{vol}(R)$. The algorithm samples $O(d\alpha/\ln\beta)$ points from the membership oracle of R. Notice that the algorithm actually constructs the minimal volume rectangle (call it B_2) containing all positive samples and guarantees that $\text{vol}(A_2) < \beta \cdot \text{vol}(B_2)$. We make use of this inner approximation in the active learning algorithm.

input : A representation $A_1 \in \mathcal{C}$ and $B_1 \in \mathcal{C}$. An oracle for $R \in \mathcal{C}$. A sampler S which samples uniformly at random points from A_1. A number $\beta > 1$.
assume : $B_1 \subseteq R \subseteq A_1$.
output : a representation of $B_2 \in \mathcal{C}$ and $A_2 \in \mathcal{C}$, such that $B_2 \subseteq R \subseteq A_2$ and $\text{vol}(A_2) \leq \beta \cdot \text{vol}(B_2)$.

1 $S^+ \leftarrow \emptyset, S^- \leftarrow \emptyset$
2 **repeat**
3 pick a random point $p \in A_1$ using S;
4 **if** $p \notin A_1 - B_1$ **then goto** step 3
5 **if** $p \in R$ **then** $S^+ \leftarrow S^+ \cup \{p\}$ **else** $S^- \leftarrow S^- \cup \{p\}$ **fi**
6 $\text{Possible}_R \leftarrow \{C \in \mathcal{C} \mid S^+ \subseteq C \subseteq A_1 \setminus S^-\}$
7 $B_2 \leftarrow$ the smallest axis-parallel rectangle containing S^+
8 $A_2 \leftarrow$ the axis-parallel rectangle of minimal volume containing all $C \in \text{Possible}_R$
9 **until** $\text{vol}(A_2) \leq \beta \cdot \text{vol}(B_2)$
10 output a representation of A_2 and B_2

Algorithm 4: Modified Inner-Outer algorithm used for active learning

Lemma 8.3 *Let $\alpha > \beta > 1$. Let E be the expected number of membership-oracle calls of the modified Inner-outer algorithm on input $B_1 \subseteq R \subseteq A_1$ and β.*

1. *If* $\text{vol}(A_1) \leq \alpha \cdot \text{vol}(R)$ *then* $E = O(d\alpha/\ln\beta)$.
2. *If* $\text{vol}(A_1) \leq \alpha \cdot \text{vol}(B_1)$ *then* $E = O(d(\alpha-1)/\ln\beta)$.

Proof :
By direct application of Lemma 4.6, the expected number of random points picked in step 3 of the algorithms is bounded by $8 + d\alpha/\ln\beta$. This proves part 1.

For part 2., note the modification made in step 4 of algorithm. We only query points which lie in the region between A_1 and B_1, thus ignoring at least $1/\alpha$ fraction of the region A_1 (as $\text{vol}(A_1) \leq \alpha \cdot \text{vol}(B_1)$). Hence the expected number of queries to the membership oracle is $(1 - 1/\alpha)(8 +$

$d\alpha/\ln\beta)$, which is $O(1+d(\alpha-1)/\ln\beta) = O(d(\alpha-1)/\ln\beta)$ (in the last containment we used $d(\alpha-1)/\ln\beta \geq d(\alpha-1)/\ln\alpha \geq d = \Omega(1)$). ■

The algorithm above requires a sampler S that samples uniformly random points from A_1. This sampler can be easily obtained from the unlabeled sampler of the instance space X using rejection sampling. This increases the number of samples needed by a constant factor, as $\text{vol}(A_1) \geq \text{vol}(R) > 1/2$.

We now repeatedly apply this bootstrapping procedure to do active learning.

input : An oracle O to generate unlabeled samples from the instance space $X = [0,1]^d$. The membership oracle of an axis-pallalel rectangle R. A parameter $\varepsilon > 0$.
assume : $\text{vol(R)} \geq 1/2$.
output : a representation of an axis-parallel rectangle A, such that $R \subseteq A$ and $\text{vol}(A) < (1+\varepsilon)\text{vol}(R)$

1 $(A,B) \leftarrow$ output of Algorithm 4 with $A_1 = X, B_1 = \emptyset$, and $\beta = 1+\varepsilon 2^{\lceil \log_2 1/\varepsilon \rceil}$
2 **for** $i \leftarrow \lceil \log_2 1/\varepsilon \rceil$ **to** 1 **do**
3 $\quad (A,B) \leftarrow$ output of Algorithm 4 with $A_1 = A, B_1 = B, \beta = 1+\varepsilon 2^{i-1}$
4 output a representation of A

Algorithm 5: Algorithm to do one-sided active learning

Theorem 8.4 *The expected number membership-queries made by Algorithm 5 is bounded by* $O(d\ln(1/\varepsilon))$.

Proof :
By Lemma 8.3, part 1., the number of membership queries at step 1. is bounded by $O(d)$.

By Lemma 8.3, part 2., at each iteration the number membership queries by Algorithm 4 on step 3. is bounded by

$$O(d(1+2^i\varepsilon-1)/\ln(1+2^{i-1}\varepsilon)) = O(d),$$

where in the last step we used the fact that $\ln(1+x) \geq x - x^2/2$ for $x \geq 0$. The total number of iterations is $\lceil \log_2(1/\varepsilon) \rceil$. Hence the total number of calls to the membership oracle is $O(d\ln(1/\varepsilon))$. ■

At the end of learning, $\text{vol}(A) \leq (1+\varepsilon)\cdot\text{vol}(R)$. Hence, $\text{vol}(A) - \text{vol}(R) < \varepsilon \cdot \text{vol}(R) < \varepsilon$. Further, $R \subset A$. Hence, big rectangles are one-sided active learnable.

Note that a trivial algorithm to learn big rectangles would be to learn each face at a time. This can be done by doing binary search starting from $(1/2,\ldots 1/2)$. As there are d faces, this will take $O(d\ln\frac{d}{\varepsilon})$ labeled samples (since precision ε/d is required along each dimension).

9. Discussion and Future Work

In this paper, we introduced a new framework of learning using oracles of increasingly restrictive concepts. Our framework has been inspired from the biological phenomenon of behavioral shaping in which a target behavior is taught to a subject by teaching it successively better approximations of the behavior, eventually converging to the target behavior. Analogous to behavioral shaping, in a shaped search problem, the learner is given access to a sequence of membership oracles of increasingly restrictive concepts and the learner is required to output one sample from the target concept.

We gave efficient algorithms to solve the shaped search problem for the concept class of intervals, axis-parallel rectangles, bounded eccentricity ellipsoids, and general convex bodies. While we have matching lower and upper bounds for the concept class of intervals, for other concept classes we do not understand the complexity of the shaped search problem (i.e., our lower and upper bounds do not match). The concept class of axis-parallel rectangles is a natural question to consider next.

Question 5 *Let $\mathcal{C}$ be the concept class of axis-parallel rectangles in $\mathbb{R}^d$. What is the complexity of the shaped search problem?*

The bootstrapping technique of Section 4 was useful for both the shaped search problem and active learning for axis-parallel rectangles. Whether efficient bootstrapping algorithms exist for other concept classes is an interesting problem.

Question 6 *For which concept classes is bootstrapping possible?*

Another technique that was useful in our setting was a deterministic ray shooting algorithm. By keeping track of the centroid we were able to solve the shaped search problem for bounded eccentricity ellipsoids. One can imagine that such an approach might work for any centrally symmetric convex body.

Question 7 *Let K be a centrally symmetric convex body given by a membership oracle. Can the centroid of K be found by an efficient deterministic algorithm?*

Our solution of the shaped search problem for general convex bodies samples random points from convex bodies and hence heavily relies on randomization. Is the use of randomness inevitable?

Question 8 *Let $\mathcal{C}$ be the concept class of (centrally symmetric) convex bodies in $\mathbb{R}^d$. Does there exist a deterministic agent for the γ-shaped search problem, using $O(\mathrm{poly}(d, 1/\gamma))$ queries per oracle?*

Another interesting direction of future work is to apply this model to active learning. Active learning, in general, does not provide any advantage for concepts which have a "small volume". For example, it was observed in Dasgupta (2005) that when the instance space is a unit ball, the concept class of non-homogeneous perceptrons is not active learnable (in the sense that any active learning scheme requires a large number of labeled samples). This is because a perceptron, in general, can pass through the sphere in such a way that it leaves a very small "cap" of the ball on one side, and just sampling one example from this cap might require a large number of labeled samples. Our shaping model can be seen as a way of directing search to such events of low probability. One way to remedy this problem is to consider a sequence of perceptrons which divide the ball into increasingly asymmetric parts and ultimately lead to the final perceptron. Whether non-homogeneous perceptrons are learnable under this framework is an interesting direction.

As our model has been inspired from behavioral shaping, we restrict the oracles to be presented in a temporal fashion. One could consider a framework in which all the oracles are present simultaneously and the agent can query any oracle at any time. This simultaneous oracle model can be viewed as a special case of the "reward shaping" of Ng et al. (1999). Under what conditions are these models equivalent?

Acknowledgments

The authors thank to the anonymous referees for many helpful corrections, and suggestions.

The authors are listed in alphabetical order.

Appendix A.

In this section we prove Lemmas 4.4 and 4.5 (in Subsection A.1), and Lemma 2.9 (in Subsection A.2). Finally we comment on the optimality of Lemma 2.9 (in Subsection A.3).

A.1 Technical Results about Maxima and Minima of Random Variables

Proof of Lemma 4.4:
Let Y be the minimum and Z be the maximum of the X_i. For $0 \leq y \leq z \leq 1$, the density function of (Y,Z) is

$$-\frac{\partial}{\partial y}\frac{\partial}{\partial z}(z-y)^n,$$

and hence

$$E[-\ln(Z-Y)] = \int_0^1 \int_y^1 \ln(z-y)\frac{\partial}{\partial y}\frac{\partial}{\partial z}(z-y)^n \,\mathrm{d}z\,\mathrm{d}y = \frac{2n-1}{n(n-1)}.$$

■

Proof of Lemma 4.5:
Conditioning on the value of K we obtain

$$E[\min\{1,X_1,\ldots,X_K\} \,|\, K=k] = \int_0^1 (1-x)\frac{\partial}{\partial x}x^k \,\mathrm{d}x = \frac{1}{k+1},$$

and hence

$$E[\min\{1,X_1,\ldots,X_K\}] = \sum_{k=0}^{n}\binom{n}{k}p^k(1-p)^{n-k}\frac{1}{k+1} = \frac{1-(1-p)^{n+1}}{(n+1)p}.$$

■

A.2 Bounding the 2-norm of the Ray-shooting Matrix

In this section we prove Lemma 2.9. Our goal is to understand the square of the 2-norm of

$$M(\alpha) = \prod_{i=1}^{n}\left(I - \alpha\cdot\frac{Dv_iv_i^{\mathrm{T}}D}{v_i^{\mathrm{T}}D^2v_i}\right)$$

as a function of α (the 2-norm of $M(\alpha)$ measures how much closer to the centroid does a point get in the ray-shooting algorithm of Theorem 2.8).

We can understand the value of $\|M(\alpha)\|_2^2$ for $\alpha = 0$ and $\alpha = 1$ but this does not give us much information about $\|M(\alpha)\|_2^2$ for other values of α. In order to obtain this information, we are going show $\|M(\alpha)\|_2^2 \leq 1 - L^2\alpha(2-\alpha)/||G(\alpha)||_2^2$, where the entries of $G(\alpha)$ are linear functions of α (Equations 16 and 17). It will turn out that for $G(\alpha)$ we can understand $\|G(\alpha)\|_2^2$ for $\alpha = 0$ and

$\alpha = 1$. Now, since dependence of $G(\alpha)$ on α is much simpler than the dependence of $M(\alpha)$ on α, we will be able to obtain an upper bound on $\|G(\alpha)\|_2^2$ for values of $\alpha \in [0,1]$, which, in turn, will imply an upper bound on $\|M(\alpha)\|_2^2$.

Note that scaling the v_i does not change M and hence we will, w.l.o.g., assume $\|v_i\|_2 = 1$. Let $\gamma_{ij} = v_i^{\mathrm{T}} D^2 v_j$. Let $G(\alpha)$ be the upper triangular $d \times d$ matrix defined by

$$G_{ij} = \begin{cases} \sqrt{\gamma_{jj}} & \text{for } i = j, \\ (\gamma_{ij}/\sqrt{\gamma_{jj}}) \cdot \alpha & \text{for } i < j, \\ 0 & \text{otherwise.} \end{cases} \tag{14}$$

For $\alpha \in (0,2)$ we let

$$\Gamma(\alpha) = G(\alpha)/\sqrt{\alpha(2-\alpha)}. \tag{15}$$

Let V be the $d \times d$ matrix with columns $v_1, \ldots, v_d$. To prove Lemma 2.9 we will need the following two auxiliary results.

Lemma A.1 *Let $\Gamma = \Gamma(1/\sqrt{d})$ be the matrix defined by* (15) *(with $\alpha = 1/\sqrt{d}$). Then*

$$\|\Gamma\|_2^2 \leq 5\sqrt{d}.$$

Lemma A.2 *For M, D, V, Γ defined above $\|Mx\|_2^2 = \|x\|_2^2 - \|\Gamma^{-1} V^{\mathrm{T}} D x\|_2^2$. Moreover*

$$\|M\|_2^2 = 1 - \frac{1}{\lambda_{\max}(\Gamma^{\mathrm{T}} V^{\mathrm{T}} D^{-2} V \Gamma)}, \tag{16}$$

(where $\lambda_{\max}(A)$ is the largest eigenvalue of A).

We postpone the proof of Lemma's A.1 and A.2 after the proof of Lemma 2.9.

Proof of Lemma 2.9:
Since $D^{-2} \preceq L^2 \cdot I$ we have

$$\lambda_{\max}(\Gamma^{\mathrm{T}} V^{\mathrm{T}} D^{-2} V \Gamma) \leq L^2 \cdot \lambda_{\max}(\Gamma^{\mathrm{T}} V^{\mathrm{T}} I V \Gamma) = L^2 \cdot \lambda_{\max}(\Gamma^{\mathrm{T}} \Gamma) = L^2 \cdot \|\Gamma\|_2^2. \tag{17}$$

Now applying Lemma A.1 (with $\alpha = 1/\sqrt{d}$) we get

$$\lambda_{\max}(\Gamma^{\mathrm{T}} V^{\mathrm{T}} D^{-2} V \Gamma) \leq 5L^2\sqrt{d}.$$

Now, using Lemma A.2 we obtain the result. ■

Proof of Lemma A.2:
Let Γ_k be the $k \times k$ top-left submatrix of Γ. Let V_k be the $d \times k$ matrix consisting of the first k columns of V. Let

$$M_k = \prod_{i=1}^{k} \left(I - \alpha \cdot \frac{D v_i v_i^{\mathrm{T}} D}{v_i^{\mathrm{T}} D^2 v_i} \right).$$

By induction on k we will show that for any x

$$\|M_k x\|_2^2 = \|x\|_2^2 - \|\Gamma_k^{-1} V_k^{\mathrm{T}} D x\|_2^2. \tag{18}$$

For $k = 1$ we have

$$M_1 = I - \alpha \frac{Dv_1 v_1^{\mathrm{T}} D}{v_1^{\mathrm{T}} D^2 v_1}, \quad M_1 x = x - \alpha \frac{(v_1^{\mathrm{T}} Dx)}{v_1^{\mathrm{T}} D^2 v_1} Dv_1, \quad \|M_1 x\|_2^2 = \|x\|_2^2 - \alpha(2-\alpha) \frac{\left(v_1^{\mathrm{T}} Dx\right)^2}{v_1^{\mathrm{T}} D^2 v_1}.$$

Moreover $V_1^{\mathrm{T}} = v_1^{\mathrm{T}}$, $\Gamma_1^{-2} = \alpha(2-\alpha)/(v_1^{\mathrm{T}} D^2 v_1)$, and hence

$$\|\Gamma_1^{-1} V_1^{\mathrm{T}} Dx\|_2^2 = \frac{\alpha(2-\alpha)}{v_1^{\mathrm{T}} D^2 v_1} (v_1^{\mathrm{T}} Dx)^2.$$

We showed that (18) holds for $k = 1$. Now we assume $k > 1$.

If $v_k^{\mathrm{T}} Dx = 0$ then

$$\left(I - \alpha \cdot \frac{Dv_k v_k^{\mathrm{T}} D}{v_k^{\mathrm{T}} D^2 v_k}\right) x = x,$$

and hence $M_{k-1} x = M_k x$. Moreover the last entry of $V_k^{\mathrm{T}} Dx$ is zero and hence $\|\Gamma_{k-1}^{-1} V_{k-1}^{\mathrm{T}} Dx\|_2 = \|\Gamma_k^{-1} V_k^{\mathrm{T}} Dx\|_2$. Thus (18) holds by the induction hypothesis.

Now we assume $v_k^{\mathrm{T}} Dx \neq 0$. W.l.o.g., we can assume $v_k^{\mathrm{T}} Dx = 1$. Let

$$x' = \left(I - \alpha \cdot \frac{Dv_k v_k^{\mathrm{T}} D}{v_k^{\mathrm{T}} D^2 v_k}\right) x = x - \alpha \cdot \frac{Dv_k}{\gamma_{kk}}. \tag{19}$$

We have $\|x'\|_2^2 = \|x\|_2^2 - \alpha(2-\alpha)/\gamma_{kk}$.

Let $b = V_k^{\mathrm{T}} Dx$ and $b' = V_{k-1}^{\mathrm{T}} Dx'$. If we show

$$\|\Gamma_k^{-1} b\|_2^2 = \frac{\alpha(2-\alpha)}{\gamma_{kk}} + \|\Gamma_{k-1}^{-1} b'\|_2^2 \tag{20}$$

then

$$\|M_k x\|_2^2 = \|M_{k-1} x'\|_2^2 = \|x'\|_2^2 - \|\Gamma_{k-1}^{-1} b'\|_2^2 = \|x\|_2^2 - \frac{\alpha(2-\alpha)}{\gamma_{kk}} - \|\Gamma_{k-1}^{-1} b'\|_2^2 = \|x\|_2^2 - \|\Gamma_k^{-1} b\|,$$

and we are done. Thus it remains to show (20).

From (19) we have for $i = 1, \ldots, k-1$,

$$b_i' = b_i - \alpha \frac{\gamma_{ik}}{\gamma_{kk}}.$$

Let Z be the $(k-1) \times k$ matrix taking b to b', i.e.,

$$Z = \begin{cases} 1 & \text{for } i = j, \\ -\alpha(\gamma_{ik}/\gamma_{kk}) & \text{for } j = k, \\ 0 & \text{otherwise.} \end{cases}$$

Let $y = \Gamma_k^{-1} b$ and $y' = \Gamma_{k-1}^{-1} b'$. Note that $y' = \Gamma_{k-1}^{-1} Z \Gamma_k y$.

If the last coordinate of y is zero then the last coordinate of $\Gamma_k y$ is zero as well and hence Z acts as identity on $\Gamma_k y$. Thus

$$\left(\Gamma_{k-1}^{-1} Z \Gamma_k\right)(y_1, \ldots, y_{k-1}, 0)^{\mathrm{T}} = \Gamma_{k-1}^{-1} I \Gamma_k (y_1, \ldots, y_{k-1}, 0)^{\mathrm{T}} = \Gamma_{k-1}^{-1} I \Gamma_{k-1} (y_1, \ldots, y_{k-1})^{\mathrm{T}} = (y_1, \ldots, y_{k-1})^{\mathrm{T}}.$$

Thus the left $(k-1)\times(k-1)$ submatrix of $\Gamma_{k-1}^{-1}Z\Gamma_k$ is the identity matrix.

For any $i=1,\ldots,k-1$ we have

$$e_i^{\mathrm{T}}Z\Gamma_k e_k = \left(e_i^{\mathrm{T}} - \alpha\frac{\gamma_{ik}}{\gamma_{kk}}e_k^{\mathrm{T}}\right)\Gamma_k e_k = \frac{\gamma_{ik}}{\sqrt{\gamma_{kk}}}\sqrt{\alpha/(2-\alpha)} - \alpha\frac{\gamma_{ik}}{\gamma_{kk}}\sqrt{\gamma_{kk}}/\sqrt{\alpha(2-\alpha)} = 0.$$

Thus the last column of $\Gamma_{k-1}^{-1}Z\Gamma_k$ is zero. Hence

$$\|y'\|_2^2 = \|y\|_2^2 - y_k^2. \tag{21}$$

Since $y=\Gamma_k^{-1}b$, $b_k=1$, and Γ_k is upper triangular matrix we have that

$$y_k = (\Gamma_k^{-1})_{kk} = \frac{1}{(\Gamma_k)_{kk}} = \sqrt{\frac{\alpha(2-\alpha)}{\gamma_{kk}}}.$$

Plugging y_k into (21) we obtain (20). We completed the induction step and hence (18) is true for all $k=1,\ldots,n$. We proved the first part of the lemma.

To show the second part of the lemma we observe

$$\|M\|_2^2 = \max_{\|x\|_2=1}\|Mx\|_2^2 = 1 - \min_{\|x\|_2=1}\|\Gamma^{-1}V^{\mathrm{T}}Dx\|_2^2 = 1-\lambda_{\min}\left(D^{-1}V\Gamma^{-T}\Gamma^{-1}V^{\mathrm{T}}D\right).$$

Let $A=\Gamma^{-1}V^{\mathrm{T}}D=(D^{-1}V\Gamma)^{-1}$. We have

$$\lambda_{\min}(A^{\mathrm{T}}A) = \lambda_{\min}(AA^{\mathrm{T}}) = \frac{1}{\lambda_{\max}(A^{-T}A^{-1})} = \frac{1}{\lambda_{\max}(\Gamma^{\mathrm{T}}V^{\mathrm{T}}D^{-2}V\Gamma)}.$$

■

Lemma A.3 *Let A be an $d\times d$ symmetric matrix such that $0\preceq A\preceq I$ (i.e., A is positive semi-definite and all its eigenvalues are ≤ 1). Then*

$$\sum_{i=1}^{d}\sum_{j=1}^{d}\frac{A_{ij}^2}{A_{jj}} \leq d. \tag{22}$$

Proof :
The eigenvalues of A are between 0 and 1 and hence we have $A^2\preceq A$. Thus

$$\sum_{i=1}^{d}A_{ij}^2 = (A^2)_{jj} \leq A_{jj}. \tag{23}$$

Dividing both sides of (23) by A_{jj} and summing over $j\in[d]$ we obtain the result. ■

Consider $G(1)$ defined by (14) with $\alpha=1$. We can bound $\|G(1)\|_F^2$ (the square of the Frobenius norm) by (22) where we take $A=V^{\mathrm{T}}DV=(\gamma_{ij})_{i,j=1}^d$. Using $\|A\|_2\leq\|A\|_F$ we obtain the following bound.

Corollary A.4 *Let $G(1)$ be defined by* (14) *with $\alpha=1$. Then*

$$\|G(1)\|_2 \leq \sqrt{d}. \tag{24}$$

For $\alpha = 0$ the matrix G is diagonal with all the diagonal entries ≤ 1. Hence we have:

Observation A.1 *Let $G(0)$ be defined by* (14) *with $\alpha = 0$. Then*

$$\|G(0)\|_2 \leq 1. \tag{25}$$

Let $F(\alpha) = G(\alpha)^{\mathrm{T}} G(\alpha)$. Then

$$\frac{\mathrm{d}}{\mathrm{d}\alpha} F(0) = \begin{cases} \gamma_{ij}\sqrt{\gamma_{ii}}/\sqrt{\gamma_{jj}} & \text{if } i < j, \\ \gamma_{ij}\sqrt{\gamma_{jj}}/\sqrt{\gamma_{ii}} & \text{if } i > j, \\ 0 & \text{if } i = j. \end{cases}$$

Note that $\gamma_{ii} \leq 1$ for $i \in [n]$. Hence, using Lemma A.3 we obtain the following bound.

Observation A.2

$$\left\| \frac{\mathrm{d}}{\mathrm{d}\alpha} F(0) \right\|_2^2 \leq \left\| \frac{\mathrm{d}}{\mathrm{d}\alpha} F(0) \right\|_F^2 \leq 2n. \tag{26}$$

Now we can finally prove Lemma A.1.

Proof of Lemma A.1:
Let x be such that $\|x\|_2 = 1$. Let $f(\alpha) = x^{\mathrm{T}} G(\alpha)^{\mathrm{T}} G(\alpha) x$. Note that f is a quadratic function of α. Let

$$f(\alpha) = a\alpha^2 + b\alpha + c.$$

From (24), (25), (26) we get that

$$a + b + c = f(1) \leq d,$$

$$c = f(0) \leq 1, \quad \text{and}$$

$$|b| = |f'(0)| \leq \sqrt{2d}.$$

Hence

$$f(1/\sqrt{d}) \leq (d + \sqrt{2d}) \cdot \left(\frac{1}{\sqrt{d}}\right)^2 + \sqrt{2d}\frac{1}{\sqrt{d}} + 1 \leq 5.$$

Let $\alpha = 1/\sqrt{d}$, and $G = G(\alpha)$. Since x was arbitrary we get

$$\|G\|_2^2 = \max_{\|x\|_2 = 1} x^{\mathrm{T}} G^{\mathrm{T}} G x \leq 5.$$

Finally,

$$\|\Gamma\|_2^2 = \frac{\|G\|_2^2}{\alpha(2-\alpha)} \leq 5\sqrt{d}.$$

■

A.3 Optimality of Lemma 2.9

Now we show that Lemma 2.9 cannot be improved (up to a constant factor).

Let D be a diagonal matrix with $D_{11} = 1$ and $D_{ii} = \varepsilon$ for $i = 2, \ldots, d$. Let

$$v_1 \propto (\sqrt{1/2}, \sqrt{\varepsilon}, \ldots, \sqrt{\varepsilon}, \sqrt{1/2}),$$

and let $v_1, \ldots, v_d$ be orthogonal. Let V have columns $v_1, \ldots, v_d$. Then $V^{\mathrm{T}} D^2 V = (1-\varepsilon^2) v_1 v_1^{\mathrm{T}} + \varepsilon^2 I$. From the definition (15) we immediately obtain

$$\Gamma = \Gamma(\alpha) = \frac{1}{\sqrt{\alpha(2-\alpha)}} \left(B + O(\varepsilon^{1/2}) \right),$$

where

$$B_{ij} = \begin{cases} \sqrt{1/2} & \text{if } i = j = 1 \text{ or } i = j = d, \\ \alpha\sqrt{1/2} & \text{if } i = 1 \text{ and } j \geq 2, \\ 0 & \text{otherwise.} \end{cases}$$

A short calculation yields

$$\Gamma^{\mathrm{T}} V^{\mathrm{T}} D^{-2} V \Gamma = \frac{1}{4\alpha(2-\alpha)} \cdot \frac{1}{\varepsilon^2} \cdot \left(w w^{\mathrm{T}} + O(\varepsilon) \right),$$

where $w = (1, -\alpha, \ldots, -\alpha, \alpha - 1)$, and hence

$$\lambda_{\max}(\Gamma^{\mathrm{T}} V^{\mathrm{T}} D^{-2} V) = \frac{1}{4\varepsilon^2} \left(\frac{(d-1)\alpha^2 - 2\alpha + 2}{\alpha(2-\alpha)} + O(\varepsilon) \right)$$

The minimum of $\frac{(d-1)\alpha^2 - 2\alpha + 2}{\alpha(2-\alpha)}$ occurs at $\alpha = (-1 + \sqrt{2d-3})/(d-2)$. For this value of α we have

$$\frac{(d-1)\alpha^2 - 2\alpha + 2}{\alpha(2-\alpha)} \sim \sqrt{d/8}$$

as $d \to \infty$. Thus we have the following.

Observation A.3 *For any* $\alpha \in (0, 2)$,

$$\lambda_{\max}(\Gamma^{\mathrm{T}} V^{\mathrm{T}} D^{-2} V) \gtrapprox \frac{\sqrt{d/8}}{4\varepsilon^2}.$$

For $\alpha = 1$

$$\lambda_{\max}(\Gamma^{\mathrm{T}} V^{\mathrm{T}} D^{-2} V) \approx \frac{d}{4\varepsilon^2}.$$

References

Dana Angluin. Queries and concept learning. *Machine Learning*, 2(4):319–342, 1988.

Martin Anthony and Norman Biggs. *Computational Learning Theory: An Introduction*. Cambridge University Press, New York, NY, USA, 1992.

Peter Auer, Philip M. Long, and Aravind Srinivasan. Approximating hyper-rectangles: Learning and pseudorandom sets. *Journal of Computer and System Sciences*, 57(3):376–388, 1998.

Dimitris Bertsimas and Santosh Vempala. Solving convex programs by random walks. *Journal of the ACM*, 51(4):540–556, July 2004. doi: 10.1145/1008731.1008733.

Jean Bourgain. *Random Points in Isotropic Convex Sets*. *Convex Geometric Analysis*, pages 53–58. Cambridge University Press, Cambridge, 1999.

Nader H. Bshouty and Nadav Eiron. Learning monotone DNF from a teacher that almost does not answer membership queries. *Journal of Machine Learning Research*, 3:49–57, 2003.

Rui Castro, Rebecca Willett, and Robert Nowak. Faster rates in regression via active learning. In Y. Weiss, B. Schölkopf, and J. Platt, editors, *Advances in Neural Information Processing Systems 18*, pages 179–186. MIT Press, Cambridge, MA, 2006.

Sanjoy Dasgupta. Analysis of a greedy active learning strategy. In Lawrence K. Saul, Yair Weiss, and Léon Bottou, editors, *Advances in Neural Information Processing Systems 17*, pages 337–344. MIT Press, Cambridge, MA, 2005.

Sanjoy Dasgupta. Coarse sample complexity bounds for active learning. In Y. Weiss, B. Schölkopf, and J. Platt, editors, *Advances in Neural Information Processing Systems 18*, pages 235–242. MIT Press, Cambridge, MA, 2006.

Sanjoy Dasgupta, Adam Tauman Kalai, and Claire Monteleoni. Analysis of perceptron-based active learning. In *Proceedings of the eighteenth Annual Conference on Learning Theory*, pages 249–263, 2005.

Thomas G. Dietterich, Richard H. Lathrop, and Tomás Lozano-Pérez. Solving the multiple instance problem with axis-parallel rectangles. *Artificial Intelligence*, 89(1-2):31–71, 1997. ISSN 0004-3702.

Marco Dorigo and Marco Colombetti. Robot shaping: Developing autonomous agents through learning. *Artificial Intelligence*, 71(2):321–370, 1994.

Martin Dyer, Alan Frieze, and Ravi Kannan. A random polynomial-time algorithm for approximating the volume of convex bodies. *Journal of the ACM*, 38(1):1–17, 1991.

Bonnie Eisenberg and Ronald L. Rivest. On the sample complexity of PAC-learning using random and chosen examples. In *Proceedings of the Third Annual Workshop on Computational Learning Theory*, pages 154–162, San Francisco, CA, USA, 1990. Morgan Kaufmann Publishers Inc.

Shai Fine and Yishay Mansour. Active sampling for multiple output identification. In *Proceedings of the Nineteenth Annual Conference on Learning Theory, COLT 2006, Pittsburgh, PA, USA, June 2006*, volume 4005 of *Lecture Notes in Artificial Intelligence*, pages 620–634. Springer, Berlin, 2006.

Yoav Freund, H. Sebastian Seung, Eli Shamir, and Naftali Tishby. Selective sampling using the query by committee algorithm. *Machine Learning*, 28(2-3):133–168, 1997.

Paul W. Goldberg, Sally A. Goldman, and H. David Mathias. Learning unions of boxes with membership and equivalence queries. In *COLT '94: Proceedings of the Seventh Annual Conference on Computational Learning Theory*, pages 198–207, New York, NY, USA, 1994. ACM Press. ISBN 0-89791-655-7.

Sally A. Goldman and Michael J. Kearns. On the complexity of teaching. In *COLT '91: Proceedings of the Fourth Annual Workshop on Computational Learning Theory*, pages 303–314, San Francisco, CA, USA, 1991. Morgan Kaufmann Publishers Inc. ISBN 1-55860-213-5.

Sally A. Goldman and H. David Mathias. Teaching a smart learner. In *COLT '93: Proceedings of the Sixth Annual Conference on Computational Learning Theory*, pages 67–76, New York, NY, USA, 1993. ACM Press. ISBN 0-89791-611-5.

Sally A. Goldman, Ronald L. Rivest, and Robert E. Schapire. Learning binary relations and total orders. *SIAM J. Comput.*, 22(5):1006–1034, 1993. ISSN 0097-5397.

Martin Grötschel, László Lovász, and Alexander Schrijver. *Geometric Algorithms and Combinatorial Optimization*. Springer-Verlag, Berlin, 1988.

Tibor Hegedűs. Combinatorial results on the complexity of teaching and learning. In *MFCS '94: Proceedings of the 19th International Symposium on Mathematical Foundations of Computer Science 1994*, pages 393–402, London, UK, 1994. Springer-Verlag. ISBN 3-540-58338-6.

Jeffrey C. Jackson. An efficient membership-query algorithm for learning DNF with respect to the uniform distribution. *Journal of Computer and System Sciences*, 55(3):414–440, 1997. doi: http://dx.doi.org/10.1006/jcss.1997.1533.

Ravi Kannan, László Lovász, and Miklós Simonovits. Random walks and an $O^*(n^5)$ volume algorithm for convex bodies. *Random Structures and Algorithms*, 11(1):1–50, 1997.

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

George Konidaris and Andrew Barto. Autonomous shaping: Knowledge transfer in reinforcement learning. In *Proceedings of the Twenty-third International Conference on Machine Learning*, pages 489–496, New York, NY, USA, 2006. ACM Press.

Nati Linial, Michael Luby, Michael Saks, and David Zuckerman. Efficient construction of a small hitting set for combinatorial rectangles in high dimension. In *Proceedings of the Twenty-fifth Annual ACM Symposium on Theory of Computing*, pages 258–267, New York, NY, USA, 1993. ACM Press.

László Lovász. *An Algorithmic Theory of Numbers, Graphs and Convexity*, volume 50 of *CBMS-NSF Regional Conference Series in Applied Mathematics*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 1986.

Maja J. Mataric. Reward functions for accelerated learning. In *Proceedings of the Eleventh International Conference on Machine Learning*, pages 181–189, 1994.

Rajeev Motwani and Prabhakar Raghavan. *Randomized Algorithms*. Cambridge University Press, Cambridge, 1995. ISBN 0-521-47465-5.

Andrew Y. Ng, Daishi Harada, and Stuart J. Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In *Proceedings of the Sixteenth International Conference on Machine Learning*, pages 278–287, San Francisco, CA, USA, 1999. Morgan Kaufmann Publishers Inc.

Jette Randløv and Preben Alstrøm. Learning to drive a bicycle using reinforcement learning and shaping. In *Proceedings of the Fifteenth International Conference on Machine Learning*, pages 463–471, San Francisco, CA, USA, 1998. Morgan Kaufmann Publishers Inc.

Mark Rudelson. Random vectors in isotropic position. *Journal of Functional Analysis*, 164(1): 60–72, 1999.

Burrhus F. Skinner. *The Behavior of Organisms*. Appleton-Century-Crofts, New York, NY, USA, 1938.

Leslie G. Valiant. A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142, 1984.

Journal of Machine Learning Research 8 (2007) 1867-1891 Submitted 2/06; Revised 1/07; Published 8/07

Large Margin Semi-supervised Learning

Junhui Wang WANGJH@STAT.UMN.EDU
Xiaotong Shen XSHEN@STAT.UMN.EDU
School of Statistics
University of Minnesota
Minneapolis, MN 55455, USA

Editor: Tommi Jaakkola

Abstract

In classification, semi-supervised learning occurs when a large amount of unlabeled data is available with only a small number of labeled data. In such a situation, how to enhance predictability of classification through unlabeled data is the focus. In this article, we introduce a novel large margin semi-supervised learning methodology, using grouping information from unlabeled data, together with the concept of margins, in a form of regularization controlling the interplay between labeled and unlabeled data. Based on this methodology, we develop two specific machines involving support vector machines and ψ-learning, denoted as SSVM and SPSI, through difference convex programming. In addition, we estimate the generalization error using both labeled and unlabeled data, for tuning regularizers. Finally, our theoretical and numerical analyses indicate that the proposed methodology achieves the desired objective of delivering high performance in generalization, particularly against some strong performers.

Keywords: generalization, grouping, sequential quadratic programming, support vectors

1. Introduction

In many classification problems, a large amount of unlabeled data is available, while it is costly to obtain labeled data. In text categorization, particularly web-page classification, a machine is trained with a small number of manually labeled texts (web-pages), as well as a huge amount of unlabeled texts (web-pages), because manually labeling is impractical; compare with Joachims (1999). In spam detection, a small group of identified e-mails, spam or non-spam, is used, in conjunction with a large number of unidentified e-mails, to train a filter to flag incoming spam e-mails, compare with Amini and Gallinari (2003). In face recognition, a classifier is trained to recognize faces with scarce identified and enormous unidentified faces, compare with Balcan et al. (2005). In a situation as such, one research problem is how to enhance accuracy of prediction in classification by using both unlabeled and labeled data. The problem of this sort is referred to as semi-supervised learning, which differs from a conventional "missing data" problem in that the size of unlabeled data greatly exceeds that of labeled data, and missing occurs only in response. The central issue that this article addresses is how to use information from unlabeled data to enhance predictability of classification.

In semi-supervised learning, a sample $\{Z_i = (X_i, Y_i)\}_{i=1}^{n_l}$ is observed with labeling $Y_i \in \{-1, 1\}$, in addition to an independent unlabeled sample $\{X_j\}_{j=n_l+1}^{n}$ with $n = n_l + n_u$, where $X_k = (X_{k1}, \cdots, X_{kp})$; $k = 1, \cdots, n$ is an p-dimensional input. Here the labeled sample is independently and identically distributed according to an unknown joint distribution $P(x, y)$, and the unlabeled sample is

independently and identically distributed from distribution $P(x)$ that may not be the marginal distribution of $P(x,y)$.

A number of semi-supervised learning methods have been proposed through some assumptions relating $P(x)$ to the conditional distribution $P(Y=1|X=x)$. These methods include, among others, co-training (Blum and Mitchell, 1998), the EM method (Nigam, McCallum, Thrun and Mitchell, 1998), the bootstrap method (Collins and Singer, 1999), information-based regularization (Szummer and Jaakkola, 2002), Bayesian network (Cozman, Cohen and Cirelo, 2003), Gaussian random fields (Zhu, Ghahramani and Lafferty, 2003), manifold regularization (Belkin, Niyogi and Sindhwani, 2004), and discriminative-generative models (Ando and Zhang, 2004). Transductive SVM (TSVM; Vapnik, 1998) uses the concept of margins.

Despite progress, many open problems remain. Essentially all existing methods make various assumptions about the relationship between $P(Y=1|X=x)$ and $P(x)$ in a way for an improvement to occur when unlabeled data is used. Note that an improvement of classification may not be expected when simply imputing labels of X through an estimated $P(Y=1|X=x)$ from labeled data, compare with Zhang and Oles (2000). In other words, the potential gain in classification stems from an assumption, which is usually not verifiable or satisfiable in practice. As a consequence, any departure from such an assumption is likely to degrade the "alleged" improvement, and may yield worse performance than classification with labeled data alone.

The primary objective of this article is to develop a large margin semi-supervised learning methodology to deliver high performance of classification by using unlabeled data. The methodology is designed to adapt to a variety of situations by identifying as opposed to specifying a relationship between labeled and unlabeled data from data. It yields an improvement when unlabeled data can reconstruct the optimal classification boundary, and yields a no worse performance than its supervised counterpart otherwise. This is in contrast to the existing methods.

Through three key ingredients, our objective is achieved, including (1) comparing all possible grouping boundaries from unlabeled data for classification, (2) using labeled data to determine label assignment for classification as well as a modification of the grouping boundary, and (3) interplay between (1) and (2) through tuning to connect grouping to classification for seeking the best classification boundary. These ingredients are integrated in a form of regularization involving three regularizers, each controlling classification with labeled data, grouping with unlabeled data, and interplay between them. Moreover, we introduce a tuning method using unlabeled data for tuning the regularizers.

Through the proposed methodology and difference convex programming, we develop two specific machines based on support vector machines (SVM; Cortes and Vapnik, 1995) and ψ-learning (Shen, Tseng, Zhang and Wong, 2003), denoted as SSVM and SPSI. Numerical analysis indicates that SSVM and SPSI achieve the desired objective, particularly against TSVM and a graphical method in simulated and benchmark examples. Moreover, a novel learning theory is developed to quantify SPSI's generalization error as a function of complexity of the class of candidate decision functions, the sample sizes (n_l, n_u), and the regularizers. To our knowledge, this is the first attempt to relate a classifier's generalization error to (n_l, n_u) and regularizers in semisupervised learning. This theory not only explains SPSI's performance, but also supports our aforementioned discussion concerning the interplay between grouping and classification, as evident from Section 5 that SPSI can recover the optimal classification performance at a speed in n_u because of grouping from unlabeled data.

This article is organized in eight sections. Section 2 introduces the proposed semi-supervised learning methodology. Section 3 treats non-convex minimization through difference convex programming. Section 4 proposes a tuning methodology that uses both labeled and unlabeled data to enhance of accuracy of estimation of the generalization error. Section 5 presents some numerical examples, followed by a novel statistical learning theory in Section 6. Section 7 contains a discussion, and the appendix is devoted to technical proofs.

2. Methodology

In this section, we present our proposed margin-based semi-supervised learning method as well its connection to other existing popular methodologies.

2.1 Proposed Methodology

We begin with our discussion in linear margin classification with labeled data $(X_i,Y_i)_{i=1}^{n_l}$ alone. Given a class of linear decision functions of the form $f(x)=\tilde{w}_f^T x+w_{f,0}\equiv(1,x^T)w_f$, a cost function $C\sum_{i=1}^{n_l}L(y_i f(x_i))+J(f)$ is minimized with respect to $f\in\mathcal{F}$, a class of candidate decision functions, to obtain the minimizer $\hat{f}$ yielding a classifier $\mathrm{Sign}(\hat{f})$, where $J(f)=\|\tilde{w}_f\|^2/2$ is the reciprocal of the L_2 geometric margin, and $L(\cdot)$ is a margin loss defined by functional margins $z_i=y_i f(x_i)$; $i=1,\cdots,n_l$.

Different learning methodologies are defined by different margin losses. Margin losses include, among others, the hinge loss $L(z)=(1-z)_+$ for SVM with its variants $L(z)=(1-z)_+^q$ for $q>1$; compare with Lin (2002); the ρ-hinge loss $L(z)=(\rho-z)_+$ for nu-SVM (Schölkopf, Smola, Williamson and Bartlett, 2000) with $\rho>0$ to be optimized; the ψ-loss $L(z)=\psi(z)$, with $\psi(z)=1-\mathrm{Sign}(z)$ if $z\geq 1$ or $z<0$, and $2(1-z)$ otherwise, compare with Shen et al. (2003), the logistic loss $L(z)=\log(1+e^{-z})$, compare with Zhu and Hastie (2005); the sigmoid loss $L(z)=1-\tanh(cz)$; compare with Mason, Baxter, Bartlett and Frean (2000). A margin loss $L(z)$ is said to be a large margin if $L(z)$ is nonincreasing in z, which penalizes small margin values.

In order to extract useful information about classification from unlabeled data, we construct a loss $U(\cdot)$ for a grouping decision function $g(x)=(1,x^T)w_g\equiv\tilde{w}_g^T x+w_{g,0}$, with $\mathrm{Sign}(g(x))$ indicating grouping. Towards this end, we let $U(z)=\min_{\{y=\pm1\}}L(yz)$ by minimizing y in $L(\cdot)$ to remove its dependency of y. As shown in Lemma 1, $U(z)=L(|z|)$, which is symmetric in z and indicates that it can only determine the grouping boundary that occurs near in an area with low value of $U(z)$ but provide no information regarding labeling.

While U can be used to extract the grouping boundary, it needs to yield the Bayes decision function $f^*=\arg\min_{f\in\mathcal{F}}EL(Yf(X))$ in order for it to be useful for classification, where E is the expectation with respect to (X,Y). More specifically, it needs $f^*=\arg\min_{g\in\mathcal{F}}EU(g(X))$. However, it does not hold generally since $\arg\min_{g\in\mathcal{F}}EU(g(X))$ can be any $g\in\mathcal{F}$ satisfying $|g(x)|\geq 1$. Generally speaking, U gives no information about labeling Y. To overcome this difficulty, we regularize U and introduce our regularized loss for semi-supervised learning to induce a relationship between classification f and grouping g:

$$S(f,g;C)=C_1L(yf(x))+C_2U(g(x))+\frac{C_3}{2}\|w_f-w_g\|^2+\frac{1}{2}\|\tilde{w}_g\|^2, \tag{1}$$

where $C=(C_1,C_2,C_3)$ are non-negative regularizers, and $\|w_f-w_g\|^2=\|\tilde{w}_f-\tilde{w}_g\|^2+(w_{f,0}-w_{g,0})^2$ is the usual L_2-Euclidean norm in R^{p+1}. Whereas $L(yf(x))$ regularizes the contribution from labeled

data, $U(g(x))$ controls the information extracted from unlabeled data, and $\|w_f - w_g\|^2$ penalizes the disagreement between f and g, specifying a loose relationship between f and g. The interrelation between f and g is illustrated in Figure 3. Note that in (1) the geometric margin $\frac{2}{\|\tilde{w}_f\|^2}$ does not enter as it is regularized implicitly through $\frac{2}{\|w_f - w_g\|^2}$ and $\frac{2}{\|\tilde{w}_g\|^2}$.

In nonlinear learning, a kernel $K(\cdot,\cdot)$ that maps from $S \times S$ to $\mathcal{R}^1$ is usually introduced for flexible representations: $f(x) = (1, K(x,x_1), \cdots, K(x,x_n))w_f$ and $g(x) = (1, K(x,x_1), \cdots, K(x,x_n))w_g$ with $w_f = (\tilde{w}_f, w_{f,0})$ and $w_g = \tilde{w}_g + w_{g,0}$. Then nonlinear surfaces separate instances of two classes, implicitly defined by $K(\cdot,\cdot)$, where the reproducing kernel Hilbert spaces (RKHS) plays an important role; compare with Wahba (1990) and Gu (2000). The forgoing treatment for the linear case is applicable when the Euclidean inner product $\langle x_i, x_j \rangle$ is replaced by $K(x_i, x_j)$. In this sense, the linear case may be regarded as a special case of nonlinear learning.

Lemma 1 says that the regularized loss (1) allows U to yield precise information about the Bayes decision function f^* when after tuning. Specifically, U targets at the Bayes decision function in classification when C_1 and C_3 are large, and grouping can differ from classification at other C values.

Lemma 1 *For any large margin loss $L(z)$, $U(z) = \min_{y\in\{-1,1\}} L(yz) = L(|z|)$, where $y = \text{Sign}(z) = \arg\min_{y\in\{-1,1\}} L(yz)$ for any given z. Additionally,*

$$(f_C^*, g_C^*) = \arg \inf_{f,g\in\mathcal{F}} ES(f,g;C) \to (f^*, f^*) \text{ as } C_1, C_3 \to \infty.$$

In the case that (f_C^*, g_C^*) is not unique, we choose it as any minimizer of $ES(f,g;C)$.

Through (1), we propose our cost function for semi-supervised learning:

$$s(f,g) = C_1 \sum_{i=1}^{n_l} L(y_i f(x_i)) + C_2 \sum_{j=n_l+1}^{n} U(g(x_j)) + \frac{C_3}{2}\|f-g\|^2 + \frac{1}{2}\|g\|_-^2, \tag{2}$$

where in the linear case, $\|g\|_- = \|\tilde{w}_g\|$ and $\|f-g\| = \|w_f - w_g\|$; in the nonlinear case $\|g\|_-^2 = \tilde{w}_g^T K \tilde{w}_g$, $\|f-g\|^2 = (\tilde{w}_f - \tilde{w}_g)^T K(\tilde{w}_f - \tilde{w}_g) + (\tilde{w}_{f,0} - \tilde{w}_{g,0})^2$ is the RKHS norm, with an $n \times n$ matrix $\mathbf{K}$ whose ijth element is $K(x_i, x_j)$. Minimization of (2) with respect to (f,g) yields an estimated decision function $\hat{f}$ thus classifier $\text{Sign}(\hat{f})$. The constrained version of (2), after introducing slack variables $\{\xi_k \geq 0; k = 1, \cdots, n\}$, becomes

$$C_1 \sum_{i=1}^{n_l} \xi_i + C_2 \sum_{j=n_l+1}^{n} \xi_j + \frac{C_3}{2}\|f-g\|^2 + \frac{1}{2}\|g\|_-^2, \tag{3}$$

subject to $\xi_i - L(y_i f(x_i)) \geq 0$; $i = 1, \cdots, n_l$; $\xi_j - U(g(x_j)) \geq 0$; $j = n_l+1, \cdots, n$. Minimization of (2) with respect to (f,g), equivalently, minimization of (3) with respect to $(f, g, \xi_k; k = 1, \cdots, n)$ subject to the constraints gives our estimated decision function $(\hat{f}, \hat{g})$, where $\hat{f}$ is for classification.

Two specific machines SSVM and SPSI will be further developed in what follows. In (2), SSVM uses $L(z) = (1-z)_+$ and $U(z) = (1-|z|)_+$, and SPSI uses $L(z) = \psi(z)$ and $U(z) = 2(1-|z|)_+$.

2.2 Connection Between SSVM and TSVM

To better understand the proposed methodology, we now explore the connection between SSVM and TSVM. In specific, TSVM uses a cost function in the form of

$$C_1 \sum_{i=1}^{n_l} (1 - y_i f(x_i))_+ + C_2 \sum_{j=n_l+1}^{n} (1 - y_j f(x_j))_+ + \frac{1}{2}\|f\|_-^2,$$

where minimization with respect to $(y_j : j = n_l+1, \cdots, n; f)$ yields the estimated decision function $\hat{f}$. It can be thought of as the limiting case of SSVM as $C_3 \to \infty$ forcing $f = g$ in (2).

SSVM in (3) stems from grouping and interplay between grouping and classification, whereas TSVM focuses on classification. Placing TSVM in the framework of SSVM, we see that SSVM relaxes TSVM in that it allows grouping (g) and classification (f) to differ, whereas $f \equiv g$ for TSVM. Such a relaxation yields that $|e(\hat{f}, f^*)| = |GE(\hat{f}) - GE(f^*)|$ is bounded by $|e(\hat{f}, \hat{g})| + |e(\hat{g}, g_C^*)| + |e(g_C^*, f^*)|$, with $|e(\hat{f}, \hat{g})|$ controlled by C_3, the estimation error $|e(\hat{g}, g_C^*)|$ controlled by $C_2 n_u^{-1}$ and the approximation error $|e(g_C^*, f^*)|$ controlled by C_1 and C_3. As a result, all these error terms can be reduced simultaneously with a suitable choice of (C_1, C_2, C_3), thus delivering better generalization. This aspect will be demonstrated by our theory in Section 6 and numerical analysis in Section 5. In contrast, TSVM is unable to do so, and needs to increase the size of one error in order to reduce the other error, and vice versa, compare with Wang, Shen and Pan (2007). This aspect will be also confirmed by our numerical results.

The forgoing discussion concerning SSVM is applicable to (2) with a different large margin loss L as well.

3. Non-convex Minimization Through Difference Convex Programming

Optimization in (2) involves non-convex minimization, because of non-convex $U(z)$ and/or possibly $L(z)$ in z. On the basis of recent advances in global optimization, particularly difference convex (DC) programming, we develop our minimization technique. Key to DC programming is decomposition of our cost function into a difference of two convex functions, based on which iterative upper approximations can be constructed to yield a sequence of solutions converging to a stationary point, possibly an ε-global minimizer. This technique is called DC algorithms (DCA; An and Tao, 1997), permitting a treatment of large-scale non-convex minimization.

To use DCA for SVM and ψ-learning in (2), we construct DC decompositions of the cost functions of SPSI and SSVM s^Ψ and s^{SVM} in (2):

$$s^\Psi = s_1^\Psi - s_2^\Psi; \quad s^{SVM} = s_1^{SVM} - s_2^{SVM},$$

where $L(z) = \psi(z)$ and $U(z) = 2(1-|z|)_+$ for SPSI,

$$\begin{array}{rcl} s_1^\Psi & = & C_1 \sum_{i=1}^{n_l} \psi_1(y_i f(x_i)) + C_2 \sum_{j=n_l+1}^{n} 2U_1(g(x_j)) + \frac{C_3}{2}\|f-g\|^2 + \frac{1}{2}\|g\|_-^2, \\ s_2^\Psi & = & C_1 \sum_{i=1}^{n_l} \psi_2(y_i f(x_i)) + C_2 \sum_{j=n_l+1}^{n} 2U_2(g(x_j)); \end{array}$$

and $L(z) = (1-z)_+$ and $U(z) = (1-|z|)_+$ for SSVM,

$$\begin{array}{rcl} s_1^{SVM} & = & C_1 \sum_{i=1}^{n_l} (1-y_i f(x_i))_+ + C_2 \sum_{j=n_l+1}^{n} U_1(g(x_j)) + \frac{C_3}{2}\|f-g\|^2 + \frac{1}{2}\|g\|_-^2, \\ s_2^{SVM} & = & C_2 \sum_{j=n_l+1}^{n} U_2(g(x_j)). \end{array}$$

These DC decompositions are obtained through DC decompositions of $(1-|z|)_+ = U_1(z) - U_2(z)$ and $\psi(z) = \psi_1(z) - \psi_2(z)$, where $U_1 = (|z|-1)_+$, $U_2 = |z|-1$, $\psi_1 = 2(1-z)_+$, and $\psi_2 = 2(-z)_+$. The decompositions are displayed in Figure 1.

With these decompositions, we treat the nonconvex minimization in (2) by solving a sequence of quadratic programming (QP) problems. Algorithm 1 solves (2) for SPSI and SSVM.

Algorithm 1: (Sequential QP)

Step 1. (Initialization) Set initial values $f^{(0)} = g^{(0)}$ as the solution of SVM with labeled data alone,

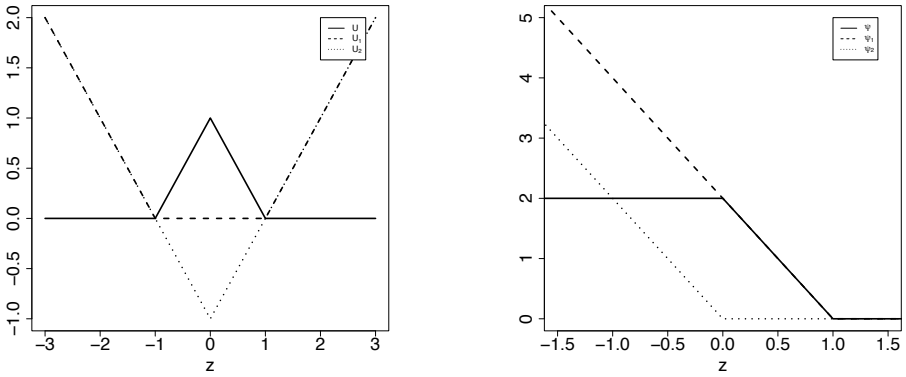


Figure 1: The left panel is a plot of U, U_1 and U_2, for the DC decomposition of $U = U_1 - U_2$. Solid, dotted and dashed lines represent U, U_1 and U_2, respectively. The right panel is a plot of ψ, ψ_1 and ψ_2, for the DC decomposition of $\psi = \psi_1 - \psi_2$. Solid, dotted and dashed lines represent ψ, ψ_1 and ψ_2, respectively.

and an precision tolerance level $\varepsilon > 0$.

Step 2. (Iteration) At iteration $k+1$, compute $(f^{(k+1)}, g^{(k+1)})$ by solving the corresponding dual problems given in (4).

Step 3. (Stopping rule) Terminate when $|s(f^{(k+1)}, g^{(k+1)}) - s(f^{(k)}, g^{(k)})| \le \varepsilon$.

Then the estimate $(\hat{f}, \hat{g})$ is the best solution among $(f^{(l)}, g^{(l)})_{l=1}^{k+1}$.

At iteration $k+1$, after omitting constants that are independent of (4), the primal problems are required to solve

$$\begin{aligned} &\min_{w_f, w_g} s_1^{\psi}(f,g) - \langle (f,g), \nabla s_2^{\psi}(f^{(k)}, g^{(k)}) \rangle, \\ &\min_{w_f, w_g} s_1^{SVM}(f,g) - \langle (f,g), \nabla s_2^{SVM}(f^{(k)}, g^{(k)}) \rangle. \end{aligned} \tag{4}$$

Here $\nabla s_2^{SVM} = (\nabla_{1f}^{SVM}, \nabla_{2f}^{SVM}, \nabla_{1g}^{SVM}, \nabla_{2g}^{SVM})$ is the gradient vector of s_2^{SVM} with respect to (f,g), with $\nabla_{1g}^{SVM} = C_2 \sum_{j=n_l+1}^{n} \nabla U_2(g(x_j)) x_j$, $\nabla_{2g}^{SVM} = C_2 \sum_{j=n_l+1}^{n} \nabla U_2(g(x_j))$, $\nabla_{1f}^{SVM} = \mathbf{0}_p$, and $\nabla_{2f}^{SVM} = 0$, where $\nabla U_2(z) = 1$ if $z > 0$, and $\nabla U_2(z) = -1$ otherwise. Similarly, $\nabla s_2^{\psi} = (\nabla_{1f}^{\psi}, \nabla_{2f}^{\psi}, \nabla_{1g}^{\psi}, \nabla_{2g}^{\psi})$ is the gradient vector of s_2^{ψ} with respect to (w_f, w_g), with $\nabla_{1f}^{\psi} = C_1 \sum_{i=1}^{n_l} \nabla \psi_2(y_i f(x_i)) y_i x_i$, $\nabla_{2f}^{\psi} = C_1 \sum_{i=1}^{n_l} \nabla \psi_2(y_i f(x_i)) y_i$, $\nabla_{1g}^{\psi} = 2\nabla_{1g}^{SVM}$, and $\nabla_{2g}^{\psi} = 2\nabla_{2g}^{SVM}$, where $\nabla \psi_2(z) = 0$ if $z > 0$ and $\nabla \psi_2(z) = -2$ otherwise. By Karush-Kuhn-Tucker(KKT)'s condition, the primal problems in (4) are equivalent to their dual forms, which are generally easier to work with and given in the Appendix C.

By Theorem 3 of Liu, Shen and Wong (2005), $\lim_{k\to\infty} \|f^{(k+1)} - f^{(\infty)}\| = 0$ for some $f^{(\infty)}$, and convergence of Algorithm 1 is superlinear in that $\lim_{k\to\infty} \|f^{(k+1)} - f^{(\infty)}\| / \|f^{(k)} - f^{(\infty)}\| = 0$ and $\lim_{k\to\infty} \|g^{(k+1)} - g^{(\infty)}\| / \|g^{(k)} - g^{(\infty)}\| = 0$, if there does not exist an instance $\tilde{x}$ such that $f^{(\infty)}(\tilde{x}) = g^{(\infty)}(\tilde{x}) = 0$ with $f^{(\infty)}(x) = (1, K(x,x_1), \cdots, K(x,x_n)) w_f^{(\infty)}$ and $g^{(\infty)}(x) = (1, K(x,x_1), \cdots, K(x,x_n)) w_g^{(\infty)}$. Therefore, the number of iterations required for Algorithm 1 is $o(\log(1/\varepsilon))$ to achieve the precision $\varepsilon > 0$.

4. Tuning Involving Unlabeled Data

This section proposes a novel tuning method based on the concept of generalized degrees of freedom (GDF) and the technique of data perturbation (Shen and Huang, 2006; Wang and Shen, 2006), through both labeled and unlabeled data. This permits tuning of three regularizers $C = (C_1, C_2, C_3)$ in (2) to achieve the optimal performance.

The generalization error (GE) of a classification function f is defined as $GE(f) = P(Yf(X) < 0) = EI(Y \neq \text{Sign}(f(X)))$, where $I(\cdot)$ is the indicator function. The $GE(f)$ usually depends on the unknown truth, and needs to be estimated. Minimization of the estimated $GE(f)$ with respect to the range of the regularizers gives the optimal regularization parameters.

For tuning, write $\hat{f}$ as $\hat{f}_C$, and write $(X^l, Y^l) = (X_i, Y_i)_{i=1}^{n_l}$ and $X^u = \{X_j\}_{j=n_l+1}^{n}$. By Theorem 1 of Wang and Shen (2006), the optimal estimated $GE(\hat{f}_C)$, after ignoring the terms independent of $\hat{f}_C$, has the form of

$$EGE(\hat{f}_C) + \frac{1}{2n_l}\sum_{i=1}^{n_l} \text{Cov}(Y_i, \text{Sign}(\hat{f}_C(X_i))|X^l) + \frac{1}{4}D_1(X^l, \hat{f}_C). \tag{5}$$

Here, $EGE(\hat{f}_C) = \frac{1}{2n_l}\sum_{i=1}^{n_l}(1 - Y_i\,\text{Sign}(\hat{f}_C(X_i)))$ is the training error, and $D_1(X^l, \hat{f}_C) = E\big(E(\triangle(X)) - \frac{1}{n_l}\sum_{i=1}^{n_l}\triangle(X_i)|X^l\big)$ with $\triangle(X) = (E(Y|X) - \text{Sign}(\hat{f}_C(X)))^2$, where $E(\cdot|X)$ and $E(\cdot|X^l)$ are conditional expectations with respect to Y and Y^l respectively. As illustrated in Wang and Shen (2006), the estimated (5) based on GDF is optimal in the sense that it performs no worse than the method of cross-validation and other tuning methods; see Efron (2004).

In (5), $\text{Cov}(Y_i, \text{Sign}(\hat{f}_C(X_i))|X^l); i = 1\cdots, n_l$ and $D_1(X^l, \hat{f}_C)$ need to be estimated. It appears that $\text{Cov}(Y_i, \text{Sign}(\hat{f}_C(X_i))|X^l)$ is estimated only through labeled data, for which we apply the data perturbation technique of Wang and Shen (2006). On the other hand, $D_1(X^l, \hat{f}_C)$ is estimated directly through (X^l, Y^l) and X^u jointly.

Our method proceeds as follows. First generate pseudo data Y_i^* by perturbing Y_i:

$$Y_i^* = \begin{cases} Y_i & \text{with probability } 1-\tau, \\ \tilde{Y}_i & \text{with probability } \tau, \end{cases} \tag{6}$$

where $0 < \tau < 1$ is the size of perturbation, and $(\tilde{Y}_i + 1)/2$ is sampled from a Bernoulli distribution with $\hat{p}(x_i)$, an rough probability estimate of $p(x_i) = P(Y = 1|X = x_i)$, which may be obtained through the same classification method that defines $\hat{f}_C$ or through logistic regression when it doesn't yield an estimated $p(x)$, such as SVM and ψ-learning. The estimated covariance is proposed to be

$$\widehat{\text{Cov}}(Y_i, \text{Sign}(\hat{f}_C(X_i))|X^l) = \frac{1}{k(Y_i, \hat{p}(X_i))}\text{Cov}^*(Y_i^*, \text{Sign}(\hat{f}_C^*(X_i))|X^l);\ i = 1, \cdots, n_l, \tag{7}$$

where $k(Y_i, \hat{p}(X_i)) = \tau + \tau(1-\tau)\frac{((Y_i+1)/2 - \hat{p}(X_i))^2}{\hat{p}(X_i)(1-\hat{p}(X_i))}$, and f_C^* is an estimated decision function through the same classification method trained through $(X_i, Y_i^*)_{i=1}^{n_l}$.

To estimate D_1, we express it as a difference between the true model error $E(E(Y|X) - Sign(\hat{f}_C(X)))^2$ and its empirical version $n_l^{-1}\sum_{i=1}^{n_l}(E(Y_i|X_i) - \text{Sign}(\hat{f}_C(X_i)))^2$, where the former can

be estimated through (X^l, Y^l) and X^u. The estimated D_1 becomes

$$\widehat{D}_1(X^l, \hat{f}_C) = E^* \left(\frac{1}{n_u} \sum_{j=n_l+1}^{n} ((2\hat{p}(X_j) - 1) - \text{Sign}(\hat{f}_C^*(X_j)))^2 - \frac{1}{n_l} \sum_{i=1}^{n_l} ((2\hat{p}(X_i) - 1) - \text{Sign}(\hat{f}_C^*(X_i)))^2 \middle| X^l \right), \tag{8}$$

Generally, $\widehat{\text{Cov}}$ in (7) and $\widehat{D}_1$ in (8) can be always computed using a Monte Carlo (MC) approximation of Cov^*, E^*, when it is difficult to obtain their analytic forms. Specifically, when Y^l is perturbed D times, a MC approximation of $\widehat{\text{Cov}}$ and $\widehat{D}_1$ can be derived:

$$\widehat{\text{Cov}}(Y_i, \text{Sign}(\hat{f}_C(X_i)) | X^l) \approx \frac{1}{D-1} \sum_{d=1}^{D} \frac{1}{k(Y_i, \hat{p}(X_i))} \text{Sign}(\hat{f}_C^{*d}(X_i))(Y_i^{*d} - \overline{Y}_i^*), \tag{9}$$

$$\widehat{D}_1(X^l, \hat{f}_C) \approx \frac{1}{D-1} \sum_{d=1}^{D} \left(\frac{1}{n_u} \sum_{j=n_l+1}^{n} ((2\hat{p}(X_j) - 1) - \text{Sign}(\hat{f}_C^{*d}(X_j)))^2 - \frac{1}{n_l} \sum_{i=1}^{n_l} ((2\hat{p}(X_i) - 1) - \text{Sign}(\hat{f}_C^{*d}(X_i)))^2 \right),$$

where $Y_i^{*d}; d = 1, \cdots, D$ are perturbed samples according to (6), $\overline{Y}_i^* = \frac{1}{D} \sum_d Y_i^{*d}$, and $\hat{f}_C^{*d}$ is trained through $(X_i, Y_i^{*d})_{i=1}^{n_l}$. Our proposed estimate $\widehat{GE}$ becomes

$$\widehat{GE}(\hat{f}_C) = EGE(\hat{f}_C) + \frac{1}{2n_l} \sum_{i=1}^{n_l} \widehat{\text{Cov}}(Y_i, \text{Sign}(\hat{f}_C(X_i)) | X^l) + \frac{1}{4} \widehat{D}_1(X^l, \hat{f}_C), \tag{10}$$

By the law of large numbers, $\widehat{GE}$ converges to (5) as $D \to \infty$. In practice, we recommend D to be at least n_l to ensure the precision of MC approximation and τ to be 0.5. In contrast to the estimated GE with labeled data alone, the $\widehat{GE}(\hat{f}_C)$ in (10) requires no perturbation of X when X^u is available. This permits more robust and computationally efficient estimation.

Minimization of (10) with respect to C yields the minimizer $\hat{C}$, which is optimal in terms of GE as suggested by Theorem 2, under similar technical assumptions as in Wang and Shen (2006).

(C.1): (Loss and risk) $\lim_{n_l \to \infty} \sup_C |GE(\hat{f}_C)/E(GE(\hat{f}_C)) - 1| = 0$ in probability.

(C.2): (Consistency of initial estimates) For almost all x, $\hat{p}_i(x) \to p_i(x)$, as $n_l \to \infty$; $i = 1, \cdots, n_l$.

(C.3): (Positivity) Assume that $\inf_C E(GE(\hat{f}_C)) > 0$.

Theorem 2 *Under Conditions C.1-C.3,* $\lim_{n_l, n_u \to \infty} \left(\lim_{\tau \to 0^+} GE(\hat{f}_{\hat{C}}) / \inf_C GE(\hat{f}_C) \right) = 1.$

Theorem 2 says the ideal optimal performance $\inf_C GE(\hat{f}_C)$ can be realized by $GE(\hat{f}_{\hat{C}})$ when $\tau \to 0^+$ and $n_l, n_u \to \infty$ against any other tuning method.

5. Numerical Examples

This section examines effectiveness of SSVM and SPSI and compare them against SVM with labeled data alone, TSVM and a graphical method of Zhu, Ghahramani and Lafferty (2003), in both simulated and benchmark examples. A test error, averaged over 100 independent replications, is used to measure their performances.

For simulation comparison, we define the amount of improvement of a method over SVM with labeled data alone as the percent of improvement in terms of the Bayesian regret,

$$\frac{(T(SVM)-T(Bayes))-(T(\cdot)-T(Bayes))}{T(SVM)-T(Bayes)}, \tag{11}$$

where $T(\cdot)$ and $T(Bayes)$ are the test error of any method and the Bayes error. This metric seems to be sensible, which is against the baseline—the Bayes error $T(Bayes)$, which is approximated by the test error over a test sample of large size, say 10^5.

For benchmark comparison, we define the amount of improvement over SVM as

$$\frac{T(SVM)-T(\cdot)}{T(SVM)}, \tag{12}$$

which underestimates the amount of improvement in absence of the Bayes rule.

Numerical analyses are performed in R2.1.1. For TSVM, SVM^{light} (Joachims, 1999) is used. For the graphical method, a MATLAB code provided in Zhu, Ghahramani and Lafferty (2003) is employed. In the linear case, $K(s,t)=\langle s,t\rangle$; in the Gaussian kernel case, $K(s,t)=\exp\left(-\frac{\|s-t\|^2}{\sigma^2}\right)$, where σ^2 is set to be p, a default value in the "svm" routine of R, to reduce computational cost for tuning σ^2.

5.1 Simulations and Benchmarks

Two simulated and three benchmark examples are examined. In each example, we perform a grid search to minimize the test error of each classifier with respect to tuning parameters, in order to eliminate the dependency of the classifier on these parameters. Specifically, one regularizer for SVM and one tuning parameter σ in the Gaussian weight matrix for the graphical method, two regularization regularizers for TSVM, and three regularizers for SSVM and SPSI are optimized over $[10^{-2},10^3]$. For SSVM and SPSI, C is searched through a set of unbalanced grid points, based on our small study of the relative importance among (C_1,C_2,C_3). As suggested by Figure 2, C_3 appears to be most crucial to $\widehat{GE}(\hat{f}_C)$, whereas C_2 is less important than (C_1,C_3), and C_1 is only useful when its value is not too large. This leads to our unbalanced search over C, that is, $C_1\in\{10^{-2},10^{-1},1,10,10^2\}$, $C_2\in\{10^{-2},1,10^2\}$, and $C_3\in\{10^{m/4};m=-8,-7,\cdots,12\}$. This strategy seems reasonable as suggested by our simulation. Clearly, a more refined search is expected to yield better performance for SSVM and SPSI.

Example 1: A random sample $\{(X_{i1},X_{i2},Y_i);i=1,\cdots,1000\}$ is generated as follows. First, 1000 independent instances (Y_i,X_{i1},X_{i2}) are sampled according to $(Y_i+1)/2\sim Bernoulli(0.5)$, $X_{i1}\sim Normal(Y_i,1)$, and $X_{i2}\sim Normal(0,1)$. Second, 200 instances are randomly selected for training, and the remaining 800 instances are retained for testing. Next, 190 unlabeled instances (X_{i1},X_{i2}) are obtained by removing labels from a randomly chosen subset of the training sample, whereas the remaining 10 instances are treated as labeled data. The Bayes error is 0.162.

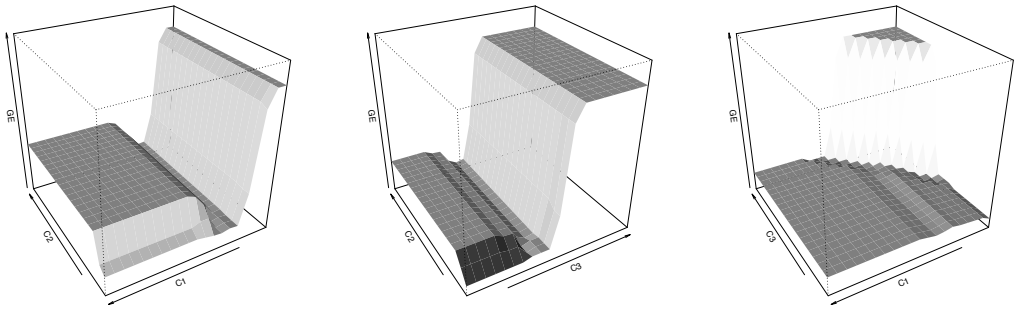


Figure 2: Plot of $\widehat{GE}(\hat{f}_C)$ as a function of (C_1, C_2, C_3) for one random selected sample of the WBC example. The top left, the top right and the bottom left are plots of $\widehat{GE}(\hat{f}_C)$ versus (C_1, C_2), (C_2, C_3) and (C_3, C_1), respectively. Here (C_1, C_2, C_3) take values in set $\{10^{-2+m/4}; m = 0, 1, \cdots, 20\}$.

Example 2: A random sample $\{(X_{i1}, X_{i2}, Y_i); i = 1, \cdots, 1000\}$ is generated. First, a random sample (X_{i1}, X_{i2}) of size 1000 is generated: $X_{i1} \sim Normal(3\cos(k_i\pi/2 + \pi/8), 1)$, $X_{i2} \sim Normal(3\sin(k_i\pi/2 + \pi/8), 4)$, with k_i sampled uniformly from $\{1, \cdots, 4\}$. Second, their labels Y_i; $i = 1, \cdots, 1000$ are assigned: $Y_i = 1$ if $k_i \in \{1, 4\}$, and -1 if $k_i \in \{2, 3\}$. As in Example 1, we obtain 200 (10 labeled and 190 unlabeled) instances for training as well as 800 instances for testing. The Bayes error is 0.089.

Benchmarks: Three benchmark examples are examined, including Wisconsin Breast Cancer (WBC), Mushroom and Spam email, each available in the UCI Machine Learning Repository (Blake and Merz, 1998). The WBC example concerns discrimination of a benign breast tissue from a malignant tissue through 9 clinic diagnostic characteristics; the Mushroom example separates an edible mushroom from a poisonous one through 22 biological records; the Spam email example discriminates texts to identify spam emails through 57 frequency attributes such as frequencies of particular words and characters. All these benchmarks are suited for linear and Gaussian kernel semi-supervised learning (Blake and Merz, 1998).

Instances in the WBC and Mushroom examples are randomly divided into halves with 10 labeled and 190 unlabeled instances for training, and the remaining instances for testing. Instances in the Spam email example are randomly divided into halves with 20 labeled and 580 unlabeled instances for training, and the remaining instances for testing.

In each example, the smallest averaged test errors of SVM with labeled data alone, TSVM, the graphical method and our proposed methods are reported in Tables 1 and 2.

As indicated in Tables 1-2, SPSI and SSVM outperform both SVM and TSVM in all cases, and the graphical method in all examples except the Mushroom example. The amount of improvement, however, varies over examples and types of classifiers. Specifically, we make the following observations.

Data $n \times dim$	Method	SVM$_l$	TSVM Improv.	Graph Improv.	SSVM Improv.	SPSI Improv.	SVM$_c$
Example 1	Linear	.344(.0104)	.249(.0134)		.188(.0084)	.184(.0084)	.164(.0084)
1000×2			52.2%	.232(.0108)	85.7%	87.9%	
	Gaussian	.385(.0099)	.267(.0132)	61.5%	.201(.0072)	.200(.0069)	.196(.0015)
			52.9%		82.5%	83.0%	
Example 2	Linear	.333(.0129)	.222(.0128)		.129(.0031)	.128(.0031)	.115(.0032)
1000×2			45.5%	.213(.0114)	83.6%	84.0%	
	Gaussian	.347(.0119)	.258(.0157)	49.2%	.175(.0092)	.175(.0098)	.151(.0021)
			34.5%		66.7%	66.7%	

Table 1: Averaged test errors as well as the estimated standard errors (in parenthesis) of SVM with labeled data alone, TSVM, the graphical method, SSVM and SPSI, over 100 pairs of training and testing samples, in the simulated examples. Here Graph, SVM$_l$ and SVM$_c$ denote performances of the graphical method, SVM with labeled data alone, and SVM with complete data without missing. The amount of improvement is defined in (11), where the Bayes error serves as a baseline for comparison.

Data $n \times dim$	Method	SVM$_l$	TSVM Improv.	Graph Improv.	SSVM Improv.	SPSI	SVM$_c$
WBC	Linear	.053(.0071)	.077(.0113)		.032(.0025)	.029(.0022)	.027(.0020)
682×9			-45.3%	.080(.0235)	39.6%	45.3%	
	Gaussian	.047(.0038)	.037(.0015)	-70.2%	.030(.0005)	.030(.0005)	.030(.0004)
			21.3%		36.2%	36.2%	
Mushroom	Linear	.232(.0135)	.204(.0113)		.186(.0095)	.184(.0095)	.041(.0018)
8124×22			12.1%	.126(.0090)	19.8%	20.7%	
	Gaussian	.217(.0135)	.217(.0117)	41.9%	.173(.0126)	.164(.0123)	.021(.0014)
			0.0%		20.3%	24.4%	
Email	Linear	.216(.0097)	.227(.0120)		.191(.0114)	.189(.0107)	.095(.0022)
4601×57			-5.09%	.232(.0101)	11.6%	12.5%	
	Gaussian	.226(.0108)	.275(.0158)	-7.41%	.189(.0120)	.189(.112)	.099(.0018)
			-21.7%		16.4%	16.4%	

Table 2: Averaged test errors as well as the estimated standard errors (in parenthesis) of SVM with labeled data alone, TSVM, the graphical method, SSVM and SPSI, over 100 pairs of training and testing samples, in the benchmark examples. The amount of improvement is defined in (12), where the performance of SVM with labeled data alone serves as a baseline for comparison in absence of the Bayes error.

- In the simulated examples, the improvements of SPSI and SSVM are from 66.9% to 87.9% over SVM, while the improvements of TSVM and the graphical method are from 34.5% to 52.9% and 49.2% to 61.5%, over SVM.
- In the benchmark examples, the improvements of SPSI, SSVM, TSVM, and the graphical method, over SVM, range from 19.8% to 45.3%, from -45.3% to 21.3%, and from -70.2% to 41.9%.
- It appears that the ψ-loss performs slightly better than the SVM hinge loss in almost all examples.

- SPSI and SSVM nearly reconstruct all relevant information about labeling in the two simulated examples and the WBC example, when they are compared with SVM with full label data. This suggests that room for further improvement in these cases is small.

To understand how SPSI and SSVM perform, we examine one randomly chosen realization in Example 1 for SPSI. As displayed in Figure 3, SVM fails to provide an accurate estimate of the true decision boundaries, because of the small size of labeled data. In contrast, the grouping boundaries estimated by unlabeled covariates, almost recover the true decision boundaries for classification. This, together with the information obtained from the labeled data regarding the sign of labeling, results in much better estimated classification boundaries.

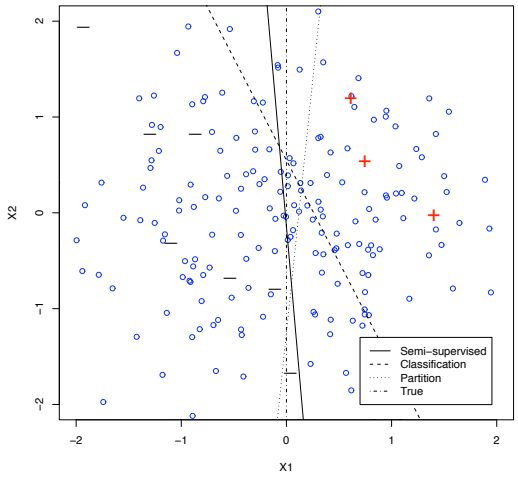


Figure 3: Illustration of SPSI in one randomly selected replication of Example 1. The solid, dashed, dotted and dotted-dashed (vertical) lines represent our ψ-learning-based decision function, the SVM decision function with labeled data alone, the partition decision function defined by unlabeled data, and the true decision boundary for classification. Here $C_1 = 0.1, C_2 = 0.01$ and $C_3 = 0.5$.

5.2 Performance After Tuning

This section compares the performances of the six methods in Section 5.1 when tuning is done using our proposed method in Section 4 and the training sample only. Particularly, SVM is tuned using the method of Wang and Shen (2006) with labeled data alone, and SPSI, SSVM , TSVM and the graphical method are tuned by minimizing the $\widehat{GE}(\hat{f}_C)$ in (10) involving both labeled and unlabeled data over a set of grid points in the same fashion as in Section 5.1. Performances of all the methods are evaluated by a test error on an independent test sample. The averaged test errors of these methods are summarized in Table 3.

As expected, SPSI and SSVM outperform both SVM with labeled data alone and TSVM in all cases, and the graphical method in all examples except Mushroom, with improvements ranging from 2.15% to 77.5% over SVM.

Data	Method	SVM$_l$	TSVM Improv.	Graph Improve.	SSVM Improv.	SPSI Improv.	SVM$_c$
Example 1	Linear	.350(.0107)	.281(.0153) 36.7%	.244(.0112)	.234(.0106) 61.7%	.233(.0106) 62.2%	.167(.0085)
	Gaussian	.395(.0101)	.331(.0211) 27.5%	56.4%	.280(.0176) 49.4%	.273(.0177) 52.4%	.258(.0102)
Example 2	Linear	.338(.0146)	.252(.0144) 34.5%	.227(.0129)	.148(.0104) 76.3%	.145(.0111) 77.5%	.118(.0084)
	Gaussian	.375(.0153)	.303(.0196) 25.2%	44.6%	.248(.0167) 44.4%	.233(.175) 49.7%	.201(.0123)
WBC	Linear	.060(.0081)	.094(.0131) -56.7%	.087(.0247)	.045(.0044) 25.0%	.042(.0035) 30.0%	.037(.0027)
	Gaussian	.051(.0039)	.044(.0047) 13.7%	-70.6%	.039(.0016) 21.6%	.039(.0018) 21.6%	.038(.0005)
Mushroom	Linear	.241(.0141)	.211(.0120) 12.4%	.137(.0101)	.209(.0108) 13.3%	.209(.0111) 13.3%	.053(.0037)
	Gaussian	.230(.0148)	.232(.0140) -0.87%	40.4%	.219(.0156) 4.78%	.210(.0131) 8.69%	.036(.0045)
Email	Linear	.236(.0109)	.241(.0128) -2.12%	.240(.0117)	.228(.0130) 3.39%	.224(.0125) 5.08%	.099(.0024)
	Gaussian	.233(.0107)	.296(.0136) -27.0%	-1.69%	.227(.0130) 2.58%	.228(.0131) 2.15%	.123(.0056)

Table 3: Averaged test errors as well as the estimated standard errors (in parenthesis) of SVM with labeled data alone, TSVM, the graphical method, SSVM and SPSI after tuning, over 100 pairs of training and testing samples, for the simulated and benchmark examples.

In conclusion, our proposed methodology achieves the desired objective of delivering high performance and is highly competitive against the top performers in the literature, where the loss $U(\cdot)$ plays a critical role in estimating decision boundaries for classification. It is also interesting to note that TSVM obtained from SVM^{light} performs even worse than SVM with labeled data alone in the WBC example for linear learning, and the Spam email example for both linear and Gaussian kernel learning. One possible explanation is that SVM^{light} may not have some difficulty in reaching good minimizers for TSVM. Moreover, the graphical method compares favorably against SVM and TSVM, but its performance does not seem to be robust in different examples. This may be due to the required Gaussian assumption.

6. Statistical Learning Theory

This section derives a finite-sample probability upper bound measuring the performance of SPSI in terms of complexity of the class of candidate decision functions $\mathcal{F}$, sample sizes (n_l, n_u) and tuning parameter C. Specifically, the generalization performance of the SPSI decision function $\hat{f}_C$ is measured by the Bayesian regret $e(f, f^*) = GE(f) - GE(f^*) \geq 0$ that is the difference between the actual performance of f and the ideal performance defined by the Bayes rule f^*. This yields SPSI's performance $\inf_C |e(\hat{f}_C, f^*)|$ after tuning.

6.1 Assumptions and Theorems

Our statistical learning theory involves risk minimization and the empirical process theory. The reader may consult Shen and Wang (2006) for a discussion about a learning theory of this kind.

First we introduce some notations. Let $(f_C^*, g_C^*) = \arg\inf_{f,g\in\mathcal{F}} ES(f,g;C)$ is a minimizer for surrogate risk $ES(f,g;C)$, as defined in Lemma 1. Let $e_f = e(f,f^*)$ be the Bayesian regret for f and $e_g = e(g,g_C^*)$ be the corresponding version for g relative to g_C^*. Denote by $V_f(X) = L(Yf(X)) - L(Yf^*(X))$ and $V_g(X) = \tilde{U}(g(X)) - \tilde{U}(g_C^*(X))$ be the differences between f and f^*, and g and g_C^* with respect to surrogate loss L and regularized surrogate loss $\tilde{U}(g) = U(g) + \frac{C_3}{2n_uC_2}\|g - f_C^*\|^2$.

To quantify complexity of $\mathcal{F}$, we define the L_2-metric entropy with bracketing. Given any $\varepsilon > 0$, denote $\{(f_m^l, f_m^u)\}_{m=1}^M$ as an ε-bracketing function set of $\mathcal{F}$ if for any $f \in \mathcal{F}$, there exists an m such that $f_m^l \le f \le f_m^u$ and $\|f_m^l - f_m^u\|_2 \le \varepsilon; m = 1,\cdots,M$, where $\|\cdot\|_2$ is the usual L_2 norm. Then the L_2-metric entropy with bracketing $H(\varepsilon,\mathcal{F})$ is defined as the logarithm of the cardinality of smallest ε-bracketing function set of $\mathcal{F}$.

Three technical assumptions are formulated based upon local smoothness of L, complexity of $\mathcal{F}$ as measured by the metric entropy, and a norm relationship.

Assumption A. (Local smoothness: Mean and variance relationship) For some some constants $0 < \alpha_h < \infty, 0 \le \beta_h < 2, a_j > 0; j = 1,2,$

$$\sup_{\{h\in\mathcal{F}:\, E(V_h(X))\le\delta\}} |e_h| \le a_1\delta^{\alpha_h}, \tag{13}$$

$$\sup_{\{h\in\mathcal{F}:\, E(V_h(X))\le\delta\}} \mathrm{Var}(V_h(X)) \le a_2\delta^{\beta_h}, \tag{14}$$

for any small $\delta > 0$ and $h = f, g$.

Assumption A describes the local behavior of mean (e_h)-and-variance $(\mathrm{Var}(V_h(X)))$ relationship. In (13), Taylor's expansion usually leads to $\alpha_h = 1$ when f and g can be parameterized. In (14), the worst case is $\beta_h = 0$ because $\max(|L(yf)|, |U(g)|) \le 2$. In practice, values for α_h and β_h depend on the distribution of (X,Y).

Let $J_0 = \max(J(g_C^*), 1)$ with $J(g) = \frac{1}{2}\|g\|_-^2$ the regularizer. Let $\mathcal{F}_l(k) = \{L(yf) - L(yf^*) : f \in \mathcal{F}, J(f) \le k\}$ and $\mathcal{F}_u(k) = \{U(g) - U(g_C^*) : g \in \mathcal{F}, J(g) \le kJ_0\}$ be the regularized decision function spaces for f's and g's.

Assumption B. (Complexity) For some constants $a_i > 0; i = 3,\cdots,5$ and ε_{n_v} with $v = l$ or u,

$$\sup_{k\ge 2} \phi_v(\varepsilon_{n_v}, k) \le a_5 n_v^{1/2}, \tag{15}$$

where $\phi_u(\varepsilon,k) = \int_{a_4T_u}^{a_3^{1/2}T_u^{\beta_g/2}} H^{1/2}(w,\mathcal{F}_u(k))dw/T_u$ with $T_u = T_u(\varepsilon,C,k) = \min(1, \varepsilon^{2/\beta_g}/2 + (n_uC_2)^{-1}(k/2-1)J_0)$, and $\phi_l(\varepsilon,k) = \int_{a_4T_l}^{a_3^{1/2}T_l^{\beta_f/2}} H^{1/2}(w,\mathcal{F}_l(k))dw/T_l$ with $T_l = T_l(\varepsilon,C,k) = \min(1, \varepsilon^{2/\beta_f}/2 + (n_lC_1)^{-1}(k/2-1)\max(J(f^*),1))$.

Although Assumption B is always satisfied by some ε_{n_v}, the smallest possible ε_{n_v} from (15) yields the best possible error rate, for given $\mathcal{F}_v$ and sample size n_v. This is to say that the rate is indeed governed by the complexity of $\mathcal{F}_v(k)$. An equation of this type, originated from the empirical process theory, has been widely used in quantifying the error rates in function estimation, see, for example, Shen and Wong (1994).

Assumption C. (Norm relationship) For some constant $a_6 > 0$, $\|f\|_1 \le a_6\|f\|$ for any $f \in \mathcal{F}$, where $\|\cdot\|_1$ is the usual L_1-norm.

Assumption C specifies a norm relationship between norm $\|\cdot\|$ defined by a RKHS and $\|\cdot\|_1$. This is usually met when $\mathcal{F}$ is a RKHS, defined, for instance, by Gaussian and Sigmoid kernels, compare with Adams (1975).

Theorem 3 *(Finite-sample probability bound for SPSI) In addition to Assumptions A-C, assume that $n_l \le n_u$. For the SPSI classifier* $\mathrm{Sign}(\hat{f}_C)$*, there exist constants $a_j > 0$; $j = 1,6,7,10,11$, and $J_l > 0$, $J_u > 0$ and $B \ge 1$ defined as in Lemma 5, such that*

$$\begin{aligned} P\big(\inf_C |e(\hat{f}_C, f^*)| \ge a_1 s_n\big) \le\ & 3.5\exp(-a_7 n_u((n_u C_2^*)^{-1} J_0)^{\max(1,2-\beta_g)}) + \\ & 6.5\exp(-a_{10} n_l((n_l C_1^*)^{-1}\min(J_l, J(f^*)))^{\max(1,2-\beta_f)}) + \\ & 6.5\exp(-a_{11} n_u((n_u C_2^*)^{-1} J_u)^{\max(1,2-\beta_g)}), \end{aligned}$$

where $s_n = \min\big(\delta_{n_l}^{2\alpha_f}, \max(\delta_{n_u}^{2\alpha_g}, \inf_{C\in\mathcal{C}}|e(g_C^*, f^*)|)\big)$, $\delta_{n_v} = \min(\varepsilon_{n_v}, 1)$ *with* $v = l, u$, $C^* = (C_1^*, C_2^*, C_3^*) = \arg\inf_{C\in\mathcal{C}}|e(g_C^*, f^*)|)$, *and* $\mathcal{C} = \{C : n_l C_1 \ge 2\delta_{n_l}^{-2}\max(J_l, J(f^*), 1), n_u C_2 \ge 2\delta_{n_u}^{-2}\max(J_0, 2C_3(2B + J(f_C^*) + J(g_C^*))), C_3 \ge a_6^2 B\delta_{n_u}^{-4}\}$.

Corollary 4 *Under the assumptions of Theorem 3, as $n_u \ge n_l \to \infty$,*

$$\inf_C |e(\hat{f}_C, f^*)| = O_p(s_n),\ \ s_n = \min\big(\delta_{n_l}^{2\alpha_f}, \max(\delta_{n_u}^{2\alpha_g}, \inf_{C\in\mathcal{C}}|e(g_C^*, f^*)|)\big).$$

Theorem 3 provides a probability bound for the upper tail of $|e(\hat{f}_C, f^*)|$ for any finite (n_l, n_u). Furthermore, Corollary 4 says that the Bayesian regret $\inf_{C\in\mathcal{C}}|e(g_C^*, f^*)|$ for the SPSI classifier $\mathrm{Sign}(\hat{f}_C)$ after tuning is of order of no larger than s_n, when $n_u \ge n_l \to \infty$. Asymptotically, SPSI performs no worse than its supervised counterpart in that $\inf_C |e(\hat{f}_C, f^*)| = O_p(\delta_{n_l}^{2\alpha_f})$. Moreover, SPSI can outperform its supervised counterpart in the sense that $\inf_C |e(\hat{f}_C, f^*)| = O_p(\min(\delta_{n_u}^{2\alpha_g}, \delta_{n_l}^{2\alpha_f})) = O_p(\delta_{n_u}^{2\alpha_g})$, when $\{g_C^* : C \in \mathcal{C}\}$ provides a good approximation to the Bayes rule f^*.

Remark: Theorem 3 and Corollary 4 continue to hold when the "global" entropy in (15) is replaced by a "local" entropy, compare with Van De Geer (1993). Let $\mathcal{F}_{l,\xi}(k) = \{L(yf) - L(yf^*) : f \in \mathcal{F}, J(f) \le k, |e(f, f^*)| \le \xi\}$ and $\mathcal{F}_{u,\xi}(k) = \{U(g) - U(g_C^*) : g \in \mathcal{F}, J(g) \le k, |e(g, g_C^*)| \le \xi\}$ be the "local" entropy of $\mathcal{F}_l(k)$ and $\mathcal{F}_u(k)$. The proof requires only a slight modification. The local entropy avoids a loss of $\log n_u$ factor in the linear case, although it may not be useful in the nonlinear case.

6.2 Theoretical Examples

We now apply the learning theory to one linear and one kernel learning examples to obtain the generalization error rates for SPSI, as measured by the Bayesian regret. We will demonstrate that the error in the linear case can be arbitrarily fast while that in the nonlinear case is fast. In either case, SPSI's performance is better than that of its supervised counterpart.

Linear learning: Consider linear classification where $X = (X_{(1)}, X_{(2)})$ is sampled independently according to the same probability density $q(z) = \frac{1}{2}(\theta + 1)|z|^\theta$ for $z \in [-1, 1]$ with $\theta \ge 1$. Given X, assign label Y to 1 if $X_{(1)} > 0$ and -1 otherwise; then Y is chosen randomly to flip with constant

probability τ for $0 < \tau < \frac{1}{2}$. Here the true decision function $f_t(x) = x_{(1)}$ yielding the vertical line as the classification boundary.

In this case, the degree of smoothness of this problem is characterized by exponent $\theta > 0$ in the density $q(z)$, which describes the level of difficulty of linear classification but may not be so in the nonlinear case.

For classification, we minimize (2) over $\mathcal{F}$, consisting of linear decision functions of form $f(x) = (1,x)^T w$ for $w \in \mathcal{R}^3$ and $x = (x_{(1)}, x_{(2)}) \in \mathcal{R}^2$. To apply Corollary 4, we verify Assumptions A-C with detailed verification given in Appendix B. In fact, Assumption A follows from the smoothness of $E(V_h(X))$ and $\mathrm{Var}(V_h(X))$ with respect to h, where a local Taylor expansion yields the degree of smoothness exponents α and β. Assumption B is automatically met, and the entropy Equation (15) is solved for the smallest possible ε_{n_v} satisfying it. Assumption C is always true for RKHS. It then follows from Corollary 4 that $\inf_C |e(\hat{f}_C, f^*)| = O_p(n_u^{-(\theta+1)/2} (\log n_u)^{(\theta+1)/2})$ as $n_u \geq n_l \to \infty$. This says that the optimal ideal performance of the Bayes rule is recovered by SPSI at speed of $n_u^{-(\theta+1)/2} (\log n_u)^{(\theta+1)/2}$ as $n_u \geq n_l \to \infty$. This rate is arbitrarily fast as $\theta \to \infty$.

Kernel learning: Consider, in the preceding case, kernel learning with a different candidate decision function class defined by the Gaussian kernel. To specify $\mathcal{F}$, we may embed a finite-dimensional Gaussian kernel representation into an infinite-dimensional space $\mathcal{F} = \{x \in \mathcal{R}^2 : f(x) = w_f^T \phi(x) = \sum_{k=0}^{\infty} w_{f,k} \phi_k(x) : w_f = (w_{f,0}, \cdots)^T \in \mathcal{R}^\infty\}$ by the representation theorem of RKHS, compare with Wahba (1990). Here $\langle \phi(x), \phi(z) \rangle = K(x,z) = \exp(-\frac{\|x-z\|^2}{2\sigma^2})$.

To apply Corollary 4, we verify Assumptions A-C as before, with detailed verification given in Appendix B. The function space $\mathcal{F}$ generated by the Gaussian kernel is rich enough to well approximate the ideal performer $\mathrm{Sign}(E(Y|X))$ (Steinwart, 2001), and yields the exponents α and β in Assumption A with smoothness and Soblev's inequality (Adams, 1975). Similarly, it follows from Corollary 4 that $\inf_C |e(\hat{f}_C, f^*)| = O_p(\min(n_l^{-1} (\log n_l J_l)^3, n_u^{-1/2} (\log n_u J_u)^{3/2}))$ as $n_u \geq n_l \to \infty$. Therefore, the optimal ideal performance of the Bayes rule is recovered by SPSI at fast speed of $\min(n_l^{-1} (\log n_l J_l)^3, n_u^{-1/2} (\log n_u J_u)^{3/2})$ as $n_u \geq n_l \to \infty$.

7. Discussion

This article proposed a novel large margin semi-supervised learning methodology that is applicable to a class of large margin classifiers. In contrast to most semi-supervised learning methods assuming various dependencies between the marginal and conditional distributions, the proposed methodology integrates labeled and unlabeled data through regularization to identify such dependencies for enhancing classification. The theoretical and numerical results show that our methodology outperforms SVM and TSVM in situations when unlabeled data provides useful information, and performs no worse when unlabeled data does not so. For tuning, further investigation of regularization paths of our proposed methodology is useful as in Hastie, Rosset, Tibshirani and Zhu (2004), to reduce computational cost.

Acknowledgments

This research is supported by NSF grants IIS-0328802 and DMS-0604394. We thank Wei Pan for many constructive comments. We also thank three referees and the editor for helpful comments and suggestions.

Appendix A. Technical Proofs

Proof of Theorem 2: The proof is similar to that of Theorem 2 of Wang and Shen (2006), and thus is omitted.

Proof of Theorem 3: The proof uses a large deviation empirical technique for risk minimization. Such a technique has been previously developed in function estimation as in Shen and Wong (1994). The proof proceeds in three steps. In **Step 1**, the tail probability of $\{e_{\widetilde{U}}(\hat{g}_C, g_C^*) \geq \delta_{n_u}^2\}$ is bounded through a large deviation probability inequality of Shen and Wong (1994). In **Step 2**, a tail probability bound of $\{|e(\hat{f}_C, f^*)| \geq \delta_{n_u}^2\}$ is induced from **Step 1** using a conversion formula between $e_{\widetilde{U}}(\hat{g}_C, g_C^*)$ and $|e(\hat{f}_C, f^*)|$. In **Step 3**, a probability upper bound for $\{|e(\hat{f}_C, f^*)| \geq \delta_{n_l}^2\}$ is obtained using the same treatment as above. The desired bound is obtained based on the bounds in **Step 2** and **Step 3**.

Step 1: It follows from Lemma 5 that $\max(\|\hat{f}_C\|^2, \|\hat{g}_C\|^2) \leq B$ for a constant $B \geq 1$, where $(\hat{f}_C, \hat{g}_C)$ is the minimizer of (2). Furthermore, $\hat{g}_C$ defined in (2) can be written as $\hat{g}_C = \arg\min_{g \in \mathcal{F}} \{C_2 \sum_{j=n_l+1}^{n} \widetilde{U}(g(x_j)) + J(g) + \frac{C_3}{2}(\|\hat{f}_C - g\|^2 - \|f_C^* - g\|^2)\}$.

By the definition of $\hat{g}_C$, $P(e_{\widetilde{U}}(\hat{g}_C, g_C^*) \geq \delta_{n_u}^2)$ is upper bounded by

$$
\begin{aligned}
& P(J(\hat{g}_C) \geq B) + P^*\Big(\sup_{g \in N} n_u^{-1} \sum_{j=n_l+1}^{n} (\widetilde{U}(g_C^*(x_j)) - \widetilde{U}(g(x_j))) + \lambda(J(g_C^*) - J(g)) \\
& \qquad + \frac{\lambda C_3}{2}(\|\hat{f}_C - g_C^*\|^2 - \|f_C^* - g_C^*\|^2 - \|\hat{f}_C - g\|^2 + \|f_C^* - g\|^2) \geq 0\Big) \\
\leq \quad & P(J(\hat{g}_C) \geq B) + P^*\Big(\sup_{g \in N} n_u^{-1} \sum_{j=n_l+1}^{n} (\widetilde{U}(g_C^*(x_j)) - \widetilde{U}(g(x_j))) + \lambda(J(g_C^*) - J(g)) \\
& \qquad + \lambda C_3(2B + J(f_C^*) + J(g_C^*)) \geq 0\Big) \equiv P(J(\hat{g}_C) \geq B) + I,
\end{aligned}
$$

where $\lambda = (n_u C_2)^{-1}$, $N = \{g \in \mathcal{F}, J(g) \leq B, e_{\widetilde{U}}(g, g_C^*) \geq \delta_{n_u}^2\}$, and P^* denotes the outer probability. By Lemma , there exists constants $a_{10}, a_{11} > 0$ such that $P(J(\hat{g}_C) \geq B) \leq 6.5\exp(-a_{10} n_l (n_l C_1)^{-1} J_l)$ $+6.5\exp(-a_{11} n_u (n_u C_2)^{-1} J_u)$, where J_l and J_u are defined in Lemma 5.

To bound I, we introduce some notations. Define the scaled empirical process as $E_u(\widetilde{U}(g_C^*) - \widetilde{U}(g)) = n_u^{-1} \sum_{j=n_l+1}^{n} \big(\widetilde{U}(g_C^*(x_j)) - \widetilde{U}(g(x_j)) + \lambda(J(g_C^*) - J(g))\big) - E(\widetilde{U}(g_C^*(X_j)) - \widetilde{U}(g(X_j)) + \lambda(J(g_C^*) - J(g))) = E_u(U(g_C^*) - U(g))$. Thus

$$
\begin{aligned}
I = P^*\Bigg(& \sup_{g \in N} E_u(U(g_C^*) - U(g)) \geq \\
& \inf_{g \in N} E(\widetilde{U}(g(X)) - \widetilde{U}(g_C^*(X))) + \lambda(J(g_C^*) - J(g)) - \lambda C_3(2B + J(f_C^*) + J(g_C^*))\Bigg).
\end{aligned}
$$

Let $A_{s,t} = \{g \in \mathcal{F} : 2^{s-1}\delta_{n_u}^2 \leq e_{\widetilde{U}}(g, g_C^*) < 2^s \delta_{n_u}^2, 2^{t-1} J_0 \leq J(g) < 2^t J_0\}$, and let $A_{s,0} = \{g \in \mathcal{F} : 2^{s-1}\delta_{n_u}^2 \leq e_{\widetilde{U}}(g, g_C^*) < 2^s \delta_{n_u}^2, J(g) < J_0\}$; $s, t = 1, 2, \cdots$. Without loss of generality, we assume that $\varepsilon_{n_u} < 1$. Then it suffices to bound the corresponding probability over $A_{s,t}$; $s, t = 1, 2, \cdots$. Toward this end, we control the first and second moment of $\widetilde{U}(g_C^*(X)) - \widetilde{U}(g(X))$ over $f \in A_{s,t}$.

For the first moment, by assumption $\delta_{n_u}^2 \geq 2\lambda \max(J_0, 2C_3(2B + J(f_C^*) + J(g_C^*)))$,

$$
\inf_{A_{s,t}} E(\widetilde{U}(g(X)) - \widetilde{U}(g_C^*(X))) + \lambda(J(g_C^*) - J(g)) \geq 2^{s-1}\delta_{n_u}^2 + \lambda(2^{t-1} - 1)J_0; s, t = 1, 2, \cdots,
$$

$$\inf_{A_{s,0}} E(\widetilde{U}(g(X)) - \widetilde{U}(g_C^*(X))) + \lambda(J(g_C^*) - J(g)) \geq (2^{s-1} - 1/2)\delta_{n_u}^2 \geq 2^{s-2}\delta_{n_u}^2; s = 1, 2, \cdots.$$

Therefore, $\inf_{A_{s,t}} E(\widetilde{U}(g(X)) - \widetilde{U}(g_C^*(X))) + \lambda(J(g_C^*) - J(g)) - \lambda C_3(2B + J(f_C^*) + J(g_C^*)) \geq M(s,t) = 2^{s-2}\delta_{n_u}^2 + \lambda(2^{t-1} - 1)J_0$, and $\inf_{A_{s,0}} E(\widetilde{U}(g(X)) - \widetilde{U}(g_C^*(X))) + \lambda(J(g_C^*) - J(g)) - \lambda C_3(2B + J(f_C^*) + J(g_C^*)) \geq M(s,0) = 2^{s-3}\delta_{n_u}^2$, for all $s,t = 1,2,\cdots$.

For the second moment, by Assumptions A,

$$\begin{aligned} & \sup_{A_{s,t}} \mathrm{Var}(\widetilde{U}(g(X)) - \widetilde{U}(g_C^*(X))) \leq \sup_{A_{s,t}} a_2 (e_{\widetilde{U}}(g, g_C^*))^{\beta_g} \leq a_2 (2^s \delta_{n_u}^2 + (2^t - 1)\lambda J_0)^{\beta_g} \\ \leq \; & a_2 2^{3\beta_g} (2^{s-2}\delta_{n_u}^2 + (2^{t-1} - 1)\lambda J_0)^{\beta_g} \leq a_3 M(s,t)^{\beta_g} = v^2(s,t), \end{aligned}$$

for and $s,t = 1,2,\cdots$ and some constant $a_3 > 0$.

Now $I \leq I_1 + I_2$ with $I_1 = \sum_{s,t=1}^{\infty} P^*(\sup_{A_{s,t}} E_u(U(g_C^*) - U(g)) \geq M(s,t))$; $I_2 = \sum_{s=1}^{\infty} P^*(\sup_{A_{s,0}} E_u(U(g_C^*) - U(g)) \geq M(s,0))$. Next we bound I_1 and I_2 separately using Theorem 3 of Shen and Wong (1994). We now verify conditions (4.5)-(4.7) there. To compute the metric entropy of $\{U(g) - U(g_C^*) : g \in A_{s,t}\}$ in (4.7) there, we note that $\int_{aM(s,t)}^{v(s,t)} H^{1/2}(w, \mathcal{F}_u(2^t))dw/M(s,t)$ is nonincreasing in s and $M(s,t)$ and hence that

$$\begin{aligned} \int_{aM(s,t)}^{v(s,t)} H^{1/2}(w, \mathcal{F}_u(2^t))dw/M(s,t) &\leq \int_{aM(1,t)}^{a_3^{1/2} M(1,t)^{\beta_g/2}} H^{1/2}(w, \mathcal{F}_u(2^t))dw/M(1,t) \\ &\leq \phi(\varepsilon_{n_u}, 2^t), \end{aligned}$$

with $a = 2a_4\varepsilon$. Assumption B implies (4.7) there with $\varepsilon = 1/2$ and some $a_i > 0$; $i = 3,4$. Furthermore, $M(s,t)/v^2(s,t) \leq 1/8$ and $T = 1$ imply (4.6), and (4.7) implies (4.5). By Theorem 3 of Shen and Wong (1994), for some constant $0 < \zeta < 1$,

$$\begin{aligned} I_1 &\leq \sum_{s,t=1}^{\infty} 3\exp\left(-\frac{(1-\zeta)n_u M^2(s,t)}{2(4v^2(s,t) + M(s,t)/3)}\right) \leq \sum_{s,t=1}^{\infty} 3\exp(-a_7 n_u (M(s,t))^{\max(1,2-\beta_g)}) \\ &\leq \sum_{s,t=1}^{\infty} 3\exp(-a_7 n_u (2^{s-1}\delta_{n_u}^2 + \lambda(2^{t-1} - 1)J_0)^{\max(1,2-\beta_g)}) \\ &\leq 3\exp(-a_7 n_u (\lambda J_0)^{\max(1,2-\beta_g)})/(1 - \exp(-a_7 n_u (\lambda J_0)^{\max(1,2-\beta_g)}))^2. \end{aligned}$$

Similarly, $I_2 \leq 3\exp(-a_7 n_u (\lambda J_0)^{\max(1,2-\beta_g)})/(1 - \exp(-a_7 n_u (\lambda J_0)^{\max(1,2-\beta_g)}))^2$. Thus $I \leq I_1 + I_2 \leq 6\exp(-a_7 n_u ((n_u C_2)^{-1} J_0)^{\max(1,2-\beta_g)})/(1 - \exp(-a_7 n_u ((n_u C_2)^{-1} J_0)^{\max(1,2-\beta_g)}))^2$, and $I^{1/2} \leq (2.5 + I^{1/2}) \exp(-a_7 n_u ((n_u C_2)^{-1} J_0)^{\max(1,2-\beta_g)})$. Thus $P(e_{\widetilde{U}}(\hat{g}_C, g_C^*) \geq \delta_{n_u}^2) \leq 3.5\exp(-a_7 n_u ((n_u C_2)^{-1} J_0)^{\max(1,2-\beta_g)}) + 6.5\exp(-a_{10} n_l ((n_l C_1)^{-1} J_l)^{\max(1,2-\beta_f)}) + 6.5\exp(-a_{11} n_u ((n_u C_2)^{-1} J_u)^{\max(1,2-\beta_f)})$.

Step 2: By Lemma 5 and Assumption C, $|e_{\widetilde{U}}(\hat{f}_C, \hat{g}_C)| \leq E|\hat{f}_C(X) - \hat{g}_C(X)| \leq a_6 \|\hat{f}_C - \hat{g}_C\| \leq a_6\sqrt{B/C_3} \leq \delta_{n_u}^2$ when $C_3 \geq a_6^2 B \delta_{n_u}^{-4}$. By Assumption A and the triangle inequality, $|e(\hat{f}_C, g_C^*)| \leq a_1 (e_{\widetilde{U}}(\hat{f}_C, g_C^*))^{\alpha_g} \leq a_1 (e_{\widetilde{U}}(\hat{g}_C, g_C^*) + |e_{\widetilde{U}}(\hat{f}_C, \hat{g}_C)|)^{\alpha_g} \leq a_1 (e_{\widetilde{U}}(\hat{g}_C, g_C^*) + \delta_{n_u}^2$, implying that $P\big(|e(\hat{f}_C, g_C^*)| \geq a_1 (2\delta_{n_u}^2)^{\alpha_g}\big) \leq P(e_{\widetilde{U}}(\hat{g}_C, g_C^*) \geq \delta_{n_u}^2), \forall C \in \mathcal{C}$. Then $P\big(\inf_C |e(\hat{f}_C, f^*)| \geq a_1 (2\delta_{n_u}^2)^{\alpha_g} + \inf_{C \in \mathcal{C}} |e(g_C^*, f^*)|\big) \leq P(e_{\widetilde{U}}(\hat{g}_{C^*}, g_{C^*}^*) \geq \delta_{n_u}^2) \leq 3.5\exp(- a_7 n_u ((n_u C_2^*)^{-1} J_0)^{\max(1,2-\beta_g)}) +$

$6.5\exp(-a_{10}n_l((n_lC_1^*)^{-1}J_l)^{\max(1,2-\beta_f)}) \;+\; 6.5\exp(-a_{11}n_u\;((n_uC_2^*)^{-1}J_u)^{\max(1,2-\beta_g)})$, where $C^* = \arg\inf_{C\in\mathcal{C}}|e(g_C^*,f^*)|$.

Step 3: Note that $\hat{f}_C = \text{argmax}_{f\in\mathcal{F}}\{C_1\sum_{i=1}^{n_l}L(y_if(x_i)) + \frac{1}{2}\|f\|_-^2\}$ when $C_2 = 0$ and $C_3 = \infty$. An application of the same treatment yields that $P(\inf_C e(\hat{f}_C,f^*) \geq a_1\delta_{n_l}^2) \leq P(\inf_C e_L(\hat{f}_C,f^*) \geq a_1\delta_{n_l}^2) \leq 3.5\exp(-a_{10}n_l((n_lC_1^*)^{-1}J(f^*))^{\max(1,2-\beta_f)})$ when $n_lC_1^* \geq 2\delta_{n_l}^{-2}\max(J(f^*),1)$. The desired result follows.

Lemma 5 *Under the assumptions of Theorem 3, for $(\hat{f}_C,\hat{g}_C)$ as the minimizer of (2), there exists constants $B>0$, depending only on C_1, such that*

$$\max(E(C_3\|\hat{f}_C-\hat{g}_C\|^2+\|\hat{g}_C\|^2), E\|\hat{f}_C\|^2, 2C_1) \quad\leq\quad B.$$

Proof: It suffices to show $E(C_3\|\hat{f}_C-\hat{g}_C\|^2+\|\hat{g}_C\|^2)\leq B$. Let $\widetilde{W}(f,g) = \frac{1}{C_1}s(f,g) = \sum_{i=1}^{n_l}\widetilde{W}_l(y_if(x_i)) + \frac{C_2}{C_1}\sum_{j=n_l+1}^{n}\widetilde{W}_u(g(x_j))$, where $\widetilde{W}_l(f(x_i)) = L(y_if(x_i)) + \frac{C_3}{4n_lC_1}\|f-g\|^2$, and $\widetilde{W}_u(g(x_j)) = U(g(x_j)) + \frac{1}{2n_uC_2}\|g\|^2 + \frac{C_3}{4n_uC_2}\|f-g\|^2$. For convenience, write $J_l(f,g) = \frac{C_3}{4}\|f-g\|^2$, $J_u(f,g) = \frac{C_3}{4}\|f-g\|^2 + \frac{1}{2}\|g\|_-^2$, $\lambda_l = (C_1n_l)^{-1}$, and $\lambda_u = (C_2n_u)^{-1}$. We then define a new empirical process $E_{l,u}(\widetilde{W}(f,g) - \widetilde{W}(f_C^*,g_C^*)) = E_l(\widetilde{W}_l(f) - \widetilde{W}_l(f_C^*)) + \frac{C_2n_u}{C_1n_l}E_u(\widetilde{W}_u(g) - \widetilde{W}_u(g_C^*))$ as

$$\frac{1}{n_l}\sum_{i=1}^{n_l}\Big(\widetilde{W}_l(f(x_i)) - \widetilde{W}_l(f_C^*(x_i)) - E(\widetilde{W}_l(f(X_i)) - \widetilde{W}_l(f_C^*(X_i)))\Big) + \frac{C_2n_u}{C_1n_l}\frac{1}{n_u}\sum_{i=n_l+1}^{n}\Big(\widetilde{W}_u(g(x_j)) - \widetilde{W}_u(g_C^*(x_i)) - E(\widetilde{W}_u(g(X_j)) - \widetilde{W}_u(g_C^*(X_i)))\Big).$$

An application of the same argument as in the proof of Theorem 3 yields that for constants $a_8,a_9>0$, $P(e_W(\hat{f}_C,\hat{g}_C;f_C^*,g_C^*)\geq\tilde{\delta}_w^2)$ is upper bounded by

$$3.5\exp(-a_8n_l((n_lC_1)^{-1}J_l)^{\max(1,2-\beta_f)}) + 3.5\exp(-a_9n_u((n_uC_2)^{-1}J_u)^{\max(1,2-\beta_g)}),$$

provided that $2J_l\leq n_lC_1\tilde{\delta}_{n_l}^2$ and $2J_u\leq n_uC_2\tilde{\delta}_{n_u}^2$, where $e_W(f,g;f_C^*,g_C^*) = e_L(f,f_C^*) + \frac{C_2}{C_1}e_U(g,g_C^*)$, $\tilde{\delta}_w^2 = \tilde{\delta}_{n_l}^2 + \frac{C_2n_u}{C_1n_l}\tilde{\delta}_{n_u}^2$, $J_l = \max(J_l(f_C^*,g_C^*),1)$ and $J_u = \max(J_u(f_C^*,g_C^*),1)$.

Without loss of generality, assume $\min(J_l(f_C^*,g_C^*),J_u(f_C^*,g_C^*))\geq 1$. Let $J(f,g) = J_l(f,g) + J_u(f,g)$ and $A_t = \{f,g\in\mathcal{F} : e_W(f,g;f_C^*,g_C^*)\leq\tilde{\delta}_w^2, 2^{t-1}J(f_C^*,g_C^*)\leq J(f,g) < 2^tJ(f_C^*,g_C^*)\}; t=1,\cdots$. Then, $P\left(J(\hat{f}_C,\hat{g}_C)\geq J(f_C^*,g_C^*)\right)$ is upper bounded by

$$\begin{aligned}
& P(e_W(\hat{f}_C,\hat{g}_C;f_C^*,g_C^*)\geq\tilde{\delta}_w^2) + \\
& \sum_{t=1}^{\infty}P^*\left(\sup_{A_t}E_{l,u}(\widetilde{W}(f_C^*,g_C^*) - \widetilde{W}(f,g)) \geq E(\widetilde{W}(f,g) - \widetilde{W}(f_C^*,g_C^*))\right) \\
\leq\; & P(e_W(\hat{f}_C,\hat{g}_C;f_C^*,g_C^*)\geq\tilde{\delta}_w^2) + \\
& \sum_{t=1}^{\infty}P^*\left(\sup_{A_t}E_{l,u}(\widetilde{W}(f_C^*,g_C^*) - \widetilde{W}(f,g)) \geq (2^{t-1}-1)\lambda_lJ(f_C^*,g_C^*) + \tilde{\delta}_w^2\right) \\
\leq\; & P(e_W(\hat{f}_C,\hat{g}_C;f_C^*,g_C^*)\geq\tilde{\delta}_w^2) + \\
& \sum_{t=1}^{\infty}P^*\left(\sup_{A_t}E_l(\widetilde{W}_l(f_C^*) - \widetilde{W}_l(f)) \geq (2^{t-1}-1)\lambda_lJ_l + \tilde{\delta}_{n_l}^2\right) + \\
& \sum_{t=1}^{\infty}P^*\left(\sup_{A_t}E_u(\widetilde{W}_u(g_C^*) - \widetilde{W}_u(g)) \geq (2^{t-1}-1)\lambda_uJ_u + \tilde{\delta}_{n_u}^2\right).
\end{aligned}$$

An application of the same argument in the proof of Theorem 3 yields that for some constants $0 < a_{10} \le a_8$ and $0 < a_{11} \le a_9$ that $P\left(J(\hat{f}_C, \hat{g}_C) \ge J(f_C^*, g_C^*)\right)$ is upper bounded by

$$\begin{aligned} & P(e_W(f,g;f_C^*,g_C^*) \ge \tilde{\delta}_w^2) + \sum_{t=1}^{\infty} \left(3\exp(-a_{11} n_l ((n_l C_1)^{-1} J_l(f_C^*, g_C^*) 2^{t-1})^{\max(1,2-\beta_f)}) + \right. \\ & \left. 3\exp(-a_{12} n_u ((n_u C_2)^{-1} J_u(f_C^*, g_C^*) 2^{t-1})^{\max(1,2-\beta_g)})\right) \\ \le \;& 6.5\exp(-a_{10} n_l ((n_l C_1)^{-1} J_l)^{\max(1,2-\beta_f)}) + 6.5\exp(-a_{11} n_u ((n_u C_2)^{-1} J_u)^{\max(1,2-\beta_g)}). \end{aligned}$$

Note that $J(\hat{f}_C, \hat{g}_C) \le s(\hat{f}, \hat{g}) \le s(1,1) \le 2C_1 n_l$. There exists a constant $B_1 > 0$ such that

$$E(C_3 \|\hat{f}_C - \hat{g}_C\|^2 + \|\hat{g}_C\|_-^2) \le J(f_C^*, g_C^*) + B_1 \le 2C_1 + B_1, \tag{16}$$

since $J(f_C^*, g_C^*) \le ES(f_C^*, g_C^*) \le ES(1,1) \le 2C_1$. It follows from the KKT condition and (16) that $E|w_{\hat{g}_C,0}|$ is bounded by a constant B_2, depending only on C_1. The desired result follows with a choice of $B = 2C_1 + B_1 + B_2^2$.

Lemma 6 *(Metric entropy in Example 6.2.1) Under the assumptions there, for $v = l$ or u,*

$$H(\varepsilon, \mathcal{F}_{v,\xi}(k)) \le O(\log(\xi^{1/(\theta+1)}/\varepsilon)).$$

Proof: We first show the inequality for $\mathcal{F}_{u,\xi}(k)$. Suppose lines $g(x) = 0$ and $g_C^*(x) = 0$ intersect lines $x_{(2)} = \pm 1$ with two points $(u_g, 1), (v_g, -1)$ and $(u_{g_C^*}, 1), (v_{g_C^*}, -1)$, respectively. Note that $e(g, g_C^*) \le \xi$ implies $P(\Delta(g, g_C^*)) \le \frac{\xi}{1-2\tau}$ with $\Delta(g, g_C^*) = \{\text{Sign}(g(x)) \ne \text{Sign}(g_C^*(x))\}$. Direct calculation yields that $P(\Delta(g, g_C^*)) \ge \frac{1}{2}\max(|u_g - u_{g_C^*}|, |v_g - v_{g_C^*}|)^{\theta+1}$, $\max(|u_g - u_{g_C^*}|, |v_g - v_{g_C^*}|) \le a' \xi^{1/(\theta+1)}$ for a constant $a' > 0$. We then cover all possible $(u_g, 1)$ and $(v_g, -1)$ with intervals of length ε^*. The covering number for these possible points is no more than $(2a'\xi^{1/(\theta+1)}/\varepsilon^*)^2$. After these points are covered, we then connect the endpoints of the covering intervals to form bracket planes $l(x) = 0$ and $u(x) = 0$ such that $l \le g \le u$, and $\|u - l\|_2 \le \|u - l\|_\infty \le \varepsilon^*$. Let $U^l(g) = 2 - 2\max(|l_{\pm 1}|, |u_{\pm 1}|)$ and $U^u(g) = 2 - 2I(l(x)u(x) > 0)\min(|l_{\pm 1}|, |u_{\pm 1}|)$, then $U^l(g) \le U(g) \le U^u(g)$ and $\|U^u(g) - U^l(g)\|_\infty \le 2\|u - l\|_\infty \le 2\varepsilon^*$. With $\varepsilon = 2\varepsilon^*$, $\{(U^l(g), U^u(g))\}$ forms an ε-bracketing set of $U(g)$. Therefore, the ε-covering number for $\mathcal{F}_{u,\xi}(k)$ is at most $(4a'\xi^{1/(\theta+1)}/\varepsilon)^2$, implying $H(\varepsilon, \mathcal{F}_{u,\xi}(k))$ is upper bounded by $O(\log(\xi^{\frac{1}{\theta+1}}/\varepsilon))$. Furthermore, it is similar to show the inequality for $\mathcal{F}_{l,\xi}(k)$ since $\left(2\min(1, 1 - \max(yl(x), yu(x))_+), 2\min(1, 1 - \min(yl(x), yu(x))_+)\right)$ forms a bracket for $L(yf(x))$ when $l \le f \le u$.

Lemma 7 *(Metric entropy in Example 6.2.2) Under the assumptions there, for $v = l$ or u,*

$$H(\varepsilon, \mathcal{F}_v(k)) \le O((\log(k/\varepsilon))^3).$$

Proof: We first show the inequality for $\mathcal{F}_u(k)$. Suppose there exist ε-brackets $(g_m^l, g_m^u)_{m=1}^M$ for some M such that for any $g \in \mathcal{F}(k) = \{g \in \mathcal{F} : J(g) \le k\}$, $g_m^l \le g \le g_m^u$ and $\|g_m^u - g_m^l\|_\infty \le \varepsilon$ for some $1 \le m \le M$. Let $U^l(g) = 2 - 2\max(|g_{m,\pm 1}^l|, |g_{m,\pm 1}^u|)$ and $U^u(g) = 2 - 2I(g_m^l g_m^u > 0)\min(|g_{m,\pm 1}^l|, |g_{m,\pm 1}^u|)$, then $U^l(g) \le U(g) \le U^u(g)$ and $\|U^u(g) - U^l(g)\|_\infty \le 2\|g_m^u - g_m^l\|_\infty \le 2\varepsilon$. Therefore, $(U^l(g) - U(g_C^*), U^u(g) - U(g_C^*))$ forms a bracket of length 2ε for $U(g) - U(g_C^*)$. The desired inequality then follows from the Example 4 in Zhou (2002) that $H_\infty(\varepsilon, \mathcal{F}(k)) \le O(\log(k/\varepsilon)^3)$ under the $L_\infty-$metric: $\|g\|_\infty = \sup_{x \in \mathcal{R}^2} |g(x)|$. Furthermore, it is similar to show the inequality for $\mathcal{F}_l(k)$ as in Lemma 6.

Lemma 8 *For any functions f, g and any constant $\rho > 0$,*

$$E|\operatorname{Sign}(f(X)) - \operatorname{Sign}(g(X))|I(|f(X)| \geq \rho) \leq 2\rho^{-1}E|f(X) - g(X)|.$$

Proof: The left hand side is $2P(|f(X)| \geq \rho, \operatorname{Sign}(f(X)) \neq \operatorname{Sign}(g(X))) \leq 2P(|f(x) - g(x)| \geq \rho) \leq 2\rho^{-1}E|f(X) - g(X)|$ by Chebyshev's inequality.

Appendix B. Verification of Assumptions A-C in the Theoretical Examples

Linear learning: Since $(X_{(1)}, Y)$ is independent of $X_{(2)}$, $ES(f,g;C) = E(E(S(f,g;C)|X_{(2)})) \geq ES(\tilde{f}_C^*, \tilde{g}_C^*;C)$ for any $f, g \in \mathcal{F}$, and $(\tilde{f}_C^*, \tilde{g}_C^*) = \arg\min_{\tilde{f},\tilde{g}\in\mathcal{F}_1} ES(\tilde{f},\tilde{g};C)$ with $\mathcal{F}_1 = \{x_{(1)} \in \mathcal{R} : \tilde{f}(x) = (1, x_{(1)})^T w : w \in \mathcal{R}^2\} \subset \mathcal{F}$. It then suffices to verify Assumptions A-C over $\mathcal{F}_1$ rather than $\mathcal{F}$. By Lemma 1, the approximation error $\inf_{C\in\mathcal{C}} e(\tilde{g}_C^*, f^*) = 0$. For (13), note that f^* minimizes $EL(Yf(X))$ and $\tilde{g}_C^*$ minimizes $E\widetilde{U}(\tilde{g})$ given $\tilde{f}_C^*$. Direct computation, together with Taylor's expansion yields that $E(V_{\tilde{h}}(X)) = (e_0, e_1)\Gamma_{\tilde{h}}(e_0, e_1)^T$ for any function $\tilde{h} = (1, x_{(1)})^T w_{\tilde{h}} \in \mathcal{F}_1$ with $w_{\tilde{h}} = w_{\tilde{h}^*} + (e_0, e_1)^T$, where $\tilde{h}^* = f^*$ or $\tilde{g}_C^*$ and $\Gamma_{\tilde{h}}$ is a positive definite matrix. Thus $E(V_{\tilde{h}}(X)) \geq \lambda_1(e_0^2 + e_1^2)$ for constant $\lambda_1 > 0$. Moreover, straightforward calculation yields that $|e_{\tilde{h}}| \leq \frac{1}{2}(1 - 2\tau)\min(|w_{\tilde{h}^*,1}|, |w_{\tilde{h}^*,1} + e_1|)^{-(\theta+1)}|e_0|^{\theta+1} \leq \lambda_2(e_0^2 + e_1^2)^{(\theta+1)/2}$ for some constant $\lambda_2 > 0$, where $w_{\tilde{h}^*} = (w_{\tilde{h}^*,0}, w_{\tilde{h}^*,1})$. A combination of these two inequalities leads to (13) with $\alpha_{\tilde{h}} = (\theta+1)/2$. For (14), note that $\operatorname{Var}(V_{\tilde{h}}(X)) \leq \|\tilde{h} - \tilde{h}^*\|_2^2 = e_0^2 + e_1^2 EX_{(1)}^2 \leq \max(1, EX_{(1)}^2)(e_0^2 + e_1^2)$. This implies (14) with $\beta_{\tilde{h}} = 1$. For Assumption B, by Lemma 6, $H(\varepsilon, \mathcal{F}_{v,\varepsilon}(k)) \leq O(\log(\varepsilon^{1/(\theta+1)})/\varepsilon)$ for any given k, thus $\phi_v(\varepsilon, k) = a_3(\log(T_v^{-\theta/2(\theta+1)}))^{1/2}/T_v^{1/2}$ with $T_v = T_v(\varepsilon, C, k)$. Hence $\sup_{k\geq 2}\phi_v(\varepsilon,k) \leq O\big((\log(\varepsilon^{-\theta/(\theta+1)}))^{1/2}/\varepsilon\big)$ in (15). Solving (15), we obtain $\varepsilon_{n_l} = (\frac{\log n_l}{n_l})^{1/2}$ when $C_1 \sim J(f^*)\delta_{n_l}^{-2} n_l^{-1} \sim \log n_l$ and $\varepsilon_{n_u} = (\frac{\log n_u}{n_u})^{1/2}$ when $C_2 \sim J_0\delta_{n_u}^{-2} n_u^{-1} \sim \log n_u$. Assumption C is fulfilled because $E(X^2) < \infty$. In conclusion, we obtain, by Corollary 4, that $\inf_C |e(\hat{f}_C, f^*)| = O_p((n_u^{-1}\log n_u)^{(\theta+1)/2})$. Surprisingly, this rate is arbitrarily fast as $\theta \to \infty$.

Kernel learning: Similarly, we restrict our attention to $\mathcal{F}_1 = \{x \in \mathcal{R} : f(x) = w_f^T\tilde{\phi}(x) = \sum_{k=0}^{\infty} w_{f,k}\tilde{\phi}_k(x) : w_f \in \mathcal{R}^\infty\}$, where $\langle\tilde{\phi}(x), \tilde{\phi}(z)\rangle = \exp(-\frac{(x-z)^2}{2\sigma^2})$.

For (13), note that $\mathcal{F}$ is rich for sufficiently large n_l in that for any continuous function f, there exists a $\tilde{f} \in \mathcal{F}$ such that $\|f - \tilde{f}\|_\infty \leq \varepsilon_{n_l}^2$, compare with Steinwart (2001). Then $f^* = \arg\min_{f\in\mathcal{F}} EL(Yf)$ implies $\|f^* - \operatorname{Sign}(E(Y|X))\|_\infty \leq \varepsilon_{n_l}^2$ and $|EL(Yf^*) - GE(f^*)| \leq 2\varepsilon_{n_l}^2$. Consequently, $|e(f, f^*)| \leq E(V_f(X)) + 2\varepsilon_{n_l}^2$ and $\alpha_f = 1$. On the other hand, $E(V_g(X)) \geq -E|g - f_C^*| - E|g_C^* - f_C^*| + \frac{C_3}{2n_uC_2}\|g - f_C^*\|^2 - \frac{C_3}{2}\|g_C^* - f_C^*\|^2$. Using the fact that (f_C^*, g_C^*) is the minimizer of $ES(f,g;C)$, we have $\frac{C_3}{2}\|g_C^* - f_C^*\|^2 \leq ES(f_C^*, g_C^*) \leq ES(1,1) \leq 2C_1$. By Sobolev's inequality (Adams, 1975), $E|g_C^* - f_C^*| \leq \lambda_3\|g_C^* - f_C^*\| \leq \lambda_3(4C_1/C_3)^{1/2}$ and $E|g - f_C^*| \leq \lambda_3\|g - f_C^*\|$, for some constant $\lambda_3 > 0$. Plugging these into the previous inequality, we have $e_{\bar{U}}(g, g_C^*) \geq \frac{C_3}{2n_uC_2}\|g - f_C^*\|^2 - \lambda_3\|g - f_C^*\| - \frac{2C_1}{n_uC_2} - \lambda_3(4C_1/C_3)^{1/2}$. By choosing suitable C, we obtain $\frac{1}{2}\|g - f_C^*\|^2 - e_{\bar{U}}(g, g_C^*)^{1/2}\|g - f_C^*\| - e_{\bar{U}}(g, g_C^*) \leq 0$. Solving this inequality yields $\|g - f_C^*\| \leq (1 + \sqrt{5})e_{\bar{U}}(g, g_C^*)^{1/2}$. Furthermore, by Lemma 8 and Sobolev's inequality, for sufficient small $\lambda_4 > 0$, $e(g, g_C^*) \leq E2\lambda_4^{-1}|f_C^*(X) - g(X)| + 2P(|f_C^*(X)| \leq \lambda_4) + e(f_C^*, g_C^*) \leq 2\lambda_4^{-1}(1 + \sqrt{5})E(V_g(X))^{1/2} + 2P(|f_C^*(X)| \leq \lambda_4) + e(f_C^*, g_C^*)$. However, by Lemma 1, $e(f_C^*, g_C^*) \to 0$, and $P(|f_C^*(X)| \leq \lambda_4) \leq P(|f^*(X)| - |f^*(X) - f_C^*(X)| \leq \lambda_4) = P(|f^*(X)| \leq |f^*(X) - f_C^*(X)| + \lambda_4) \to 0$, as $C_1, C_2, C_3 \to \infty$, because of linearity of f^*. This yields (13) with $\alpha_g = 1/2$. For (14), $\operatorname{Var}(L(Yf(X)) - L(Yf^*(X))) \leq 2E(L(Yf(X)) - L(Yf^*(X)))^2 =$

$(w_f - w_{f^*})^T \Gamma_2 (w_f - w_{f^*})$ where Γ_2 is a positive definite matrix, and similar to Example 6.2.1, $E(V_f(X)) = (w_f - w_{f^*})\Gamma_{\tilde{h}}(w_f - w_{f^*})^T$ since f^* minimizes $E(L(Yf(X)))$. Therefore, there exists a constant $\lambda_5 > 0$ such that $\mathrm{Var}(L(Yf(X)) - L(Yf^*(X))) \le \lambda_5 E(V_f(X))$. Also, $\mathrm{Var}(U(g(X)) - U(g_C^*(X))) \le \|g - g_C^*\|_2^2 \le 2(\|g - f_C^*\|^2 + \|g_C^* - f_C^*\|^2) \le 2((1+\sqrt{3})^2 e_{\tilde{U}}(g, g_C^*) + \frac{4C_1}{C_3}) \le (8 + 2(1+\sqrt{3})^2) E(V_g(X)))$, implying (14) with $\beta_f = \beta_g = 1$. For Assumption B, by Lemma 7, $H(\varepsilon, \mathcal{F}_v(k)) \le O((\log(kJ_v/\varepsilon))^3)$ for any given k. Similarly, we have $\varepsilon_{n_l} = (n_l^{-1}(\log n_l J_l)^3)^{1/2}$ when $C_1 \sim J(f^*)\delta_{n_l}^{-2} n_l^{-1} \sim (\log n_l J_l)^{-3}$ and $\varepsilon_{n_u} = (n_u^{-1}(\log n_u J_u)^3)^{1/2}$ when $C_2 \sim J_0 \delta_{n_u}^{-2} n_u^{-1} \sim (\log n_u J_u)^{-3}$. Assumption C is fulfilled with the Gaussian kernel.

Appendix C. The Dual Form of (5)

Let $\nabla^{\psi(k)} = (\nabla_1^{\psi(k)}, \nabla_2^{\psi(k)})^T$, $\nabla_1^{\psi(k)} = C_1(\nabla\psi_2(y_1 f^{(k)}(x_1))y_1, \cdots, \nabla\psi_2(y_{n_l} f^{(k)}(x_{n_l}))y_{n_l})$ and $\nabla_2^{\psi(k)} = 2C_2(\nabla U_2(g^{(k)}(x_{n_l+1})), \cdots, \nabla U_2(g^{(k)}(x_n)))$. Further, let $\alpha = (\alpha_1, \cdots, \alpha_{n_l})^T$, $\beta = (\beta_{n_l+1}, \cdots, \beta_n)^T$, $\gamma = (\gamma_{n_l+1}, \cdots, \gamma_n)^T$, $\mathbf{y}_\alpha = (y_1\alpha_1, \cdots, y_{n_l}\alpha_{n_l})^T$, and

Theorem 9 *(ψ-learning) The dual problem of (4) with respect to (α, β, γ) is*

$$\max_{\alpha,\beta,\gamma} \left\{ -\begin{pmatrix} \mathbf{y}_\alpha \\ \beta - \gamma \end{pmatrix}^T \begin{pmatrix} (1+\frac{1}{C_3})\mathbf{K}_{ll} + \frac{1}{C_3}\mathbf{I}_l & \mathbf{K}_{lu} \\ \mathbf{K}_{ul} & \mathbf{K}_{uu} \end{pmatrix} \begin{pmatrix} \mathbf{y}_\alpha \\ \beta - \gamma \end{pmatrix} + (\alpha - (\beta+\gamma))^T \mathbf{1}_n - (\mathbf{y}_\alpha - (\beta - \gamma))^T \left(\mathbf{K}\nabla^{\psi(k)} + \begin{pmatrix} \nabla_1^{\psi(k)} \\ \mathbf{0}_{n_u} \end{pmatrix} \right) \right\}, \tag{17}$$

subject to $\left(2\begin{pmatrix} \mathbf{y}_\alpha \\ \gamma - \beta \end{pmatrix} + \nabla^{\psi(k)}\right)^T \mathbf{1}_n = 0$, $\mathbf{0}_n \le \alpha \le C_1\mathbf{1}_n$, $\mathbf{0}_n \le \beta$, $\mathbf{0}_n \le \gamma$, *and* $\mathbf{0}_n \le \beta + \gamma \le C_2\mathbf{1}_n$.

Proof of Theorem 9: For simplicity, we only prove the linear case as the nonlinear case is essentially the same. The kth primal problem in (4), after introducing slack variable ξ, is equivalent to $\min_{(w_f, w_g, \xi_i, \xi_j)} C_1 \sum_{i=1}^{n_l} \xi_i + C_2 \sum_{j=n_l+1}^{n} \xi_j + \frac{C_3}{2}\|w_f - w_g\|^2 + \frac{1}{2}\|\tilde{w}_g\|^2 - \langle w, \nabla s_2^{\psi}(f^{(k)}, g^{(k)})\rangle$ subject to constraints $2(1 - y_i(\langle w_f, x_i\rangle)) \le \xi_i$, $x_i \ge 0$; $i = 1, \cdots, n_l$, and $2(|\langle w_g, x_j\rangle| - 1) \le \xi_j$, $\xi_j \ge 0$; $j = n_l + 1, \cdots, n$.

To solve this minimization problem, the Lagrangian multipliers are employed to yield

$$\begin{aligned} & L(w_f, w_g, \xi_i, \xi_j) \\ = \; & C_1 \sum_{i=1}^{n_l} \xi_i + C_2 \sum_{j=n_l+1}^{n} \xi_j + \frac{C_3}{2}\|w_f - w_g\|^2 + \frac{1}{2}\|\tilde{w}_g\|^2 - \langle w, \nabla s_2^{\psi}(w_f^{(k)}, w_g^{(k)})\rangle + \\ & 2\sum_{i=1}^{n_l} \alpha_i (1 - y_i(\langle w_f, x_i\rangle) - \frac{\xi_i}{2}) + 2\sum_{j=n_l+1}^{n} \beta_j(\langle w_g, x_j\rangle - 1 - \frac{\xi_j}{2}) - \\ & 2\sum_{j=n_l+1}^{n} \gamma_j(\langle w_g, x_j\rangle + 1 + \frac{\xi_j}{2}) - \sum_{i=1}^{n_l} \gamma_i \xi_i - \sum_{j=n_l+1}^{n} \eta_j \xi_j, \end{aligned} \tag{18}$$

where $\alpha_i \ge 0$; $i = 1, \cdots, n_l$, $\beta_j \ge 0$, $\gamma_j \ge 0$, $j = n_l + 1, \cdots, n$. Differentiate L with respect to (w_f, w_g, ξ_i, ξ_j) and let the partial derivatives be zero, we obtain that $\frac{\partial L}{\partial \tilde{w}_f} = C_3(\tilde{w}_f - \tilde{w}_g) - 2\sum_{i=1}^{n_l} \alpha_i y_i x_i - \nabla_{1f}^{\psi(k)} = 0$, $\frac{\partial L}{\partial \tilde{w}_g} = \tilde{w}_g - C_3(\tilde{w}_f - \tilde{w}_g) - 2\sum_{j=n_l+1}^{n}(\gamma_j - \beta_j)x_j - \nabla_{1g}^{\psi(k)} = 0$, $\frac{\partial L}{\partial w_{f,0}} = C_3(w_{f,0} - w_{g,0}) -$

$2\sum_{i=1}^{n_l}\alpha_i y_i = 0$, $\frac{\partial L}{\partial w_{g,0}} = -C_3(w_{f,0}-w_{g,0}) - 2\sum_{j=n_l+1}^{n}(\gamma_j-\beta_j) - \nabla_{2g}^{\psi(k)} = 0$, $\frac{\partial L}{\partial \xi_i} = C_1-\alpha_i-\gamma_i = 0$, and $\frac{\partial L}{\partial \xi_j} = C_2-\beta_j-\gamma_j-\eta_j = 0$. Solving these equations yields that $\tilde{w}_f^* = 2(1+C_3^{-1})\sum_{i=1}^{n_l}\alpha_i y_i x_i + \sum_{j=n_l+1}^{n}(\gamma_j-\beta_j)x_j + (1+C_3^{-1})\nabla_{1f}^{\psi(k)} + \nabla_{1g}^{\psi(k)}$, $\tilde{w}_g^* = 2\sum_{i=1}^{n_l}\alpha_i y_i x_i + \sum_{j=n_l+1}^{n}(\gamma_j-\beta_j)x_j + \nabla_{1f}^{\psi(k)} + \nabla_{1g}^{\psi(k)}$, $2\sum_{i=1}^{n_l}\alpha_i y_i + 2\sum_{j=n_l+1}^{n}(\gamma_j-\beta_j) + \nabla_{2f}^{\psi(k)} + \nabla_{2g}^{\psi(k)} = 0$, $\alpha_i+\gamma_i = C_1$; $i = 1,\cdots,n_l$, and $\beta_j+\gamma_j+\eta_j = 0$; $j = n_l+1,\cdots,n$. Substituting $\tilde{w}_f^*$, $\tilde{w}_g^*$ and these identities into (18), we obtain (17) after ignoring all constant terms. To derive the corresponding constraints, note that $C_1-\alpha_i-\gamma_i = 0$, $\gamma_i \geq 0$ and $\alpha_i \geq 0$ implies $0 \leq \alpha_i \leq C_1$, $\eta_j \geq 0$ and $C_2-\beta_j-\gamma_j-\eta_j = 0$ implies $\beta_j+\gamma_j \leq C_2$. Furthermore, KKT's condition requires that $\alpha_i(1-y_i(\langle w_f, x_i\rangle)-\xi_i) = 0$, $\beta_j(\langle w_g, x_j\rangle - 1 - \xi_j)$, $\gamma_j(\langle w_g, x_j\rangle + 1 + \xi_j) = 0, \gamma_i\xi_i = 0$, and $\eta_j\xi_j = 0$. That is, $\xi_i \neq 0$ implies $\gamma_i = 0$ and $\alpha_i = C_1$, and $\xi_j \neq 0$ implies $\eta_j = 0$ and $\beta_j+\gamma_j = C_2$. Therefore, if $0 < \alpha_i < C_1$, then $\xi_i = 0$ and $1-y_i(\langle w_f, x_i\rangle) = 0$, if $0 < \beta_j+\gamma_j < C_2$, then $\xi_j = 0$ and $\langle w_g, x_j\rangle + 1 = 0$ or $\langle w_g, x_j\rangle - 1 = 0$.

Write the solution of (17) as $(\alpha^{(k+1)}, \beta^{(k+1)}, \gamma^{(k+1)})$, which yields the solution of (4): $\tilde{w}_f^{(k+1)} = 2\mathbf{X}^T\begin{pmatrix}(1+\frac{1}{C_3})\mathbf{y}\alpha \\ \beta-\gamma\end{pmatrix} + \nabla^{\psi(k)}\begin{pmatrix}(1+\frac{1}{C_3})\mathbf{1}_{n_l} \\ \mathbf{1}_{n_u}\end{pmatrix}$, and $\tilde{w}_g^{(k+1)} = 2\mathbf{X}^T\begin{pmatrix}\mathbf{y}\alpha \\ \beta-\gamma\end{pmatrix} + \nabla^{\psi(k)}\mathbf{1}_n$, and $(w_{f,0}^{(k+1)}, w_{g,0}^{(k+1)})$ satisfies KKT's condition in that $y_{i_0}(K(\tilde{w}_f^{(k+1)}, x_{i_0}) + w_{f,0}^{(k+1)}) = 1$ for any i_0 with $0 < \alpha_{i_0} < C_1$, and for any j_0 with $0 < \beta_{j_0}+\gamma_{j_0} < C_2$, $K(\tilde{w}_g^{(k+1)}, x_{j_0}) + w_{g,0}^{(k+1)} = 1$ if $\beta_{j_0} > 0$ or $K(\tilde{w}_g^{(k+1)}, x_{j_0}) + w_{g,0}^{(k+1)} = -1$ if $\gamma_{j_0} > 0$. Here $K(\tilde{w}_f^{(k+1)}, x_{i_0}) = (1+\frac{1}{C_3})\sum_{i=1}^{n_l}(2\alpha_i^{(k+1)}y_i + C_1\nabla\psi_2(f^{(k)}(x_i)))K(x_i, x_{i_0}) + 2\sum_{j=n_l+1}^{n}(\gamma_j^{(k+1)} - \beta_j^{(k+1)})K(x_j, x_{i_0}) + 2C_2\sum_{j=n_l+1}^{n}\nabla U_2(g^{(k)}(x_j))K(x_j, x_{i_0})$, and $K(\tilde{w}_g^{(k+1)}, x_{j_0}) = \sum_{i=1}^{n_l}2\alpha_i^{(k+1)}y_i K(x_i, x_{j_0}) + \sum_{i=1}^{n_l}C_1\nabla\psi_2(f^{(k)}(x_i))K(x_i, x_{j_0}) + 2\sum_{j=n_l+1}^{n}(\gamma_j^{(k+1)} - \beta_j^{(k+1)} + C_2\nabla U_2(g^{(k)}(x_j)))K(x_j, x_{j_0})$. When KKT's condition is not applicable to determine $(w_{f,0}^{(k+1)}, w_{g,0}^{(k+1)})$, that is, there does not exist an i such that $0 < \alpha_i < C_1$ or an j such that $0 < \beta_j+\gamma_j < C_2$, we may compute $(w_{f,0}^{(k+1)}, w_{g,0}^{(k+1)})$ through quadratic programming by substituting $(\tilde{w}_f^{(k)}, \tilde{w}_g^{(k)})$ into (4).

Theorem 10 *(SVM) The dual problem of (4) for SVM with respect to* (α, β, γ) *is the same as (17) with* $(\alpha, \beta, \gamma, \mathbf{y}\alpha)$ *replaced by* $\frac{1}{2}(\alpha, \beta, \gamma, \mathbf{y}\alpha)$*, and* $\nabla^{\psi(k)}$ *replaced by* $\nabla^{S(k)} = (0,\cdots,0, C_2\nabla U_2(g^{(k)}(x_{n_l+1})),\cdots, C_2\nabla U_2(g^{(k)}(x_n)))^T$*. Here KKT's condition remains the same.*

Proof of Theorem 10: The proof is similar to that of Theorem 9, and thus is omitted.

References

R. A. Adams. *Sobolev Spaces*. Academic Press, New York, 1975.

M. Amini, and P. Gallinari. Semi-supervised learning with an explicit label-error model for misclassified data. In *IJCAI* 2003.

L. An and P. Tao. Solving a class of linearly constrained indefinite quadratic problems by D.C. algorithms. *J. of Global Optimization*, 11:253-285, 1997.

R. Ando and T. Zhang. A framework for learning predictive structures from multiple tasks and unlabeled data. *Technical Report RC23462*, IBM T.J. Watson Research Center, 2004.

M. Balcan, A. Blum, P. Choi, J. Lafferty, B. Pantano, M. Rwebangira and X. Zhu. Person identification in webcam images: an application of semi-supervised learning. In *ICML* 2005.

P. L. Bartlett, M. I. Jordan and J. D. McAuliffe. Convexity, classification, and risk bounds. *J. Amer. Statist. Assoc.*, 19:138-156, 2006.

M. Belkin, P. Niyogi and V. Sindhwani. Manifold Regularization : A Geometric Framework for Learning From Examples. Technical Report, Univ. of Chicago, Department of Computer Science, TR-2004-06, 2004.

C. L. Blake and C. J. Merz. UCI repository of machine learning databases [http://www.ics.ci.edu/∼mlearn/MLRepository.html]. University of California, Irvine, Department of Information and Computer Science, 1998.

A. Blum and T. Mitchell. Combining labeled and unlabeled data with co-training. In *Proceedings of the Eleventh Annual Conference on Computational Learning Theory*, 1998.

M. Collins and Y. Singer. Unsupervised models for named entity classification. In *Empirical Methods in Natural Language Processing and Very Large Corpora*, pages 100-110, 1999.

C. Cortes and V. Vapnik. Support vector networks. *Machine Learning*, 20:273-297, 1995.

F. G. Cozman, I. Cohen and M. C. Cirelo. Semi-supervised learning of mixture models and Bayesian networks. In *ICML* 2003.

B. Efron. The estimation of prediction error: Covariance penalties and cross-validation. *J. Amer. Statist. Assoc.*, 99:619-632, 2004.

C. Gu. Multidimension smoothing with splines. In M.G. Schimek, editor, *Smoothing and Regression: Approaches, Computation and Application*, 2000.

T. Hastie, S. Rosset, R. Tibshirani and J. Zhu. The entire regularization path for the support vector machine. *J. of Machine Learning Research*, 5: 1391-1415, 2004.

T. Joachims. Transductive inference for text classification using support vector machines. In *ICML* 1999.

Y. Lin. Support vector machines and the Bayes rule in classification. *Data Mining and Knowledge Discovery*, 6:259-275, 2002.

Y. Lin and L. D. Brown. Statistical properties of the method of regularization with periodic Gaussian reproducing kernel . *Ann. Statist.*, 32:1723-1743, 2004.

S. Liu, X. Shen and W. Wong. Computational development of ψ-learning. In *SIAM 2005 International Data Mining Conference*, pages 1-12, 2005.

Y. Liu and X. Shen. Multicategory ψ-learning. *J. Amer. Statist. Assoc.*, 101:500-509, 2006.

P. Mason, L. Baxter, J. Bartlett and M. Frean. Boosting algorithms as gradient descent. In *Advances in Neural Information Processing Systems 12*, pages 512-518. The MIT Press, 2000.

K. Nigam, A. McCallum, S. Thrun and T. Mitchell . Text classification from labeled and unlabeled documents using EM. In *AAAI* 1998.

B. Schölkopf, A. Smola, R. Williamson and P. Bartlett. New support vector algorithms. *Neural Computation*, 12:1207-1245, 2000.

X. Shen and W. Wong. Convergence rate of sieve estimates. *Ann. Statist.*, 22:580-615, 1994.

X. Shen. On the method of penalization. *Statist. Sinica*, 8:337-357, 1998.

X. Shen and H. C. Huang. Optimal model assessment, selection and combination. *J. Amer. Statist. Assoc.*, 101:554-568, 2006.

X. Shen, G. C. Tseng, X. Zhang and W. Wong. On psi-learning. *J. Amer. Statist. Assoc.*, 98:724-734, 2003.

X. Shen and L. Wang. Discussion of 2004 IMS Medallion Lecture: "Local Rademacher complexities and oracle inequalities in risk minimization". *Ann. Statist.*, in press.

I. Steinwart. On the influence of the kernel on the consistency of support vector machines. *J. Machine Learning Research*, 2:67-93, 2001.

M. Szummer and T. Jaakkola. Information regularization with partially labeled data. In *NIPS* 2003.

S. Van De Geer. Hellinger-consistency of certain nonparametric maximum likelihood estimators. *Ann. Statist.*, 21:14-44, 1993.

V. Vapnik. *Statistical Learning Theory*. Wiley, New York, 1998.

G. Wahba. Spline models for observational data. *Series in Applied Mathematics*, Vol. 59, SIAM, Philadelphia, 1990.

J. Wang and X. Shen. Estimation of generalization error: random and fixed inputs. *Statist. Sinica*, 16:569-588, 2006.

J. Wang, X. Shen and W. Pan. On transductive support vector machines. In *Proc. of the Snowbird Machine Learning Conference*, in press.

T. Zhang and F. Oles. A probability analysis on the value of unlabeled data for classification problems. In *ICML* 2000.

D. Zhou. The covering number in learning theory. *J. of Complexity*, 18:739-767, 2002.

J. Zhu and T. Hastie. Kernel logistic regression and the import vector machine. *J. Comp. Graph. Statist.*, 14:185-205, 2005.

X. Zhu, Z. Ghahramani and J. Lafferty. Semi-supervised learning using gaussian fields and harmonic functions. In *ICML* 2003.

X. Zhu and J. Lafferty. Harmonic mixtures: combining mixture models and graph-based methods for inductive and scalable semi-supervised learning. In *ICML* 2005.

Journal of Machine Learning Research 8 (2007) 1893-1918 Submitted 11/06; Revised 4/07; Published 8/07

Fast Iterative Kernel Principal Component Analysis

Simon Günter SIMON.GUENTER@NICTA.COM.AU
Nicol N. Schraudolph NIC.SCHRAUDOLPH@NICTA.COM.AU
S.V. N. Vishwanathan SVN.VISHWANATHAN@NICTA.COM.AU
Research School of Information Sciences and Engineering
Australian National University –and–
Statistical Machine Learning Program
National ICT Australia, Locked Bag 8001
Canberra ACT 2601, Australia

Editor: Aapo Hyvarinen

Abstract

We develop gain adaptation methods that improve convergence of the kernel Hebbian algorithm (KHA) for iterative kernel PCA (Kim et al., 2005). KHA has a scalar gain parameter which is either held constant or decreased according to a predetermined annealing schedule, leading to slow convergence. We accelerate it by incorporating the reciprocal of the current estimated eigenvalues as part of a gain vector. An additional normalization term then allows us to eliminate a tuning parameter in the annealing schedule. Finally we derive and apply stochastic meta-descent (SMD) gain vector adaptation (Schraudolph, 1999, 2002) in reproducing kernel Hilbert space to further speed up convergence. Experimental results on kernel PCA and spectral clustering of USPS digits, motion capture and image denoising, and image super-resolution tasks confirm that our methods converge substantially faster than conventional KHA. To demonstrate scalability, we perform kernel PCA on the entire MNIST data set.

Keywords: step size adaptation, gain vector adaptation, stochastic meta-descent, kernel Hebbian algorithm, online learning

1. Introduction

Principal components analysis (PCA) is a standard linear technique for dimensionality reduction. Given a matrix $\boldsymbol{X} \in \mathbb{R}^{n \times l}$ of l centered, n-dimensional observations, PCA performs an eigendecomposition of the covariance matrix $\boldsymbol{Q} := \boldsymbol{X}\boldsymbol{X}^\top$. The $r \times n$ matrix $\boldsymbol{W}$ whose rows are the eigenvectors of $\boldsymbol{Q}$ associated with the $r \leq n$ largest eigenvalues minimizes the least-squares reconstruction error

$$\|\boldsymbol{X} - \boldsymbol{W}^\top \boldsymbol{W}\boldsymbol{X}\|_F, \tag{1}$$

where $\|\cdot\|_F$ is the Frobenius norm.

As it takes $O(n^2 l)$ time to compute $\boldsymbol{Q}$ and $O(n^3)$ time to eigendecompose it, PCA can be prohibitively expensive for large amounts of high-dimensional data. Iterative methods exist that do not compute $\boldsymbol{Q}$ explicitly, and thereby reduce the computational cost to $O(rn)$ per iteration. They assume that each individual observation $\boldsymbol{x}$ is drawn from a statistical distribution[1], and the aim is to maximize the variance of $\boldsymbol{y} := \boldsymbol{W}\boldsymbol{x}$, subject to some orthonormality constraints on the weight

1. It is customary to assume that the distribution is centered, that is, $\mathbb{E}[\boldsymbol{x}] = \boldsymbol{0}$.

matrix $\boldsymbol{W}$. In particular, we obtain the so-called hierarchical PCA network if we assume that the i^{th} row of $\boldsymbol{W}$ must have unit norm and must be orthogonal to the j^{th} row, where $j = 1, \ldots, i-1$ (Karhunen, 1994). By using Lagrange multipliers to incorporate the constraints into the objective, we can rewrite the merit function $J(\boldsymbol{W})$ succinctly as (Karhunen and Joutsensalo, 1994):

$$J(\boldsymbol{W}) = \mathbb{E}[\boldsymbol{x}^\top \boldsymbol{W}^\top \boldsymbol{W} \boldsymbol{x}] + \tfrac{1}{2}\mathrm{tr}[\boldsymbol{\Lambda}(\boldsymbol{W}\boldsymbol{W}^\top - \boldsymbol{I})], \tag{2}$$

where the Lagrange multiplier matrix $\boldsymbol{\Lambda}$ is constrained to be lower triangular. Taking gradients with respect to $\boldsymbol{W}$ and setting to zero yields

$$\partial_{\boldsymbol{W}} J(\boldsymbol{W}) = \mathbb{E}[\boldsymbol{W}\boldsymbol{x}]\boldsymbol{x}^\top + \boldsymbol{\Lambda}\boldsymbol{W} = \boldsymbol{0}. \tag{3}$$

As a consequence of the KKT conditions (Boyd and Vandenberghe, 2004), at optimality

$$\boldsymbol{\Lambda}(\boldsymbol{W}\boldsymbol{W}^\top - \boldsymbol{I}) = \boldsymbol{0}. \tag{4}$$

Right multiplying (3) by $\boldsymbol{W}^\top$, using (4), and noting that $\boldsymbol{\Lambda}$ must be lower triangular yields

$$\boldsymbol{\Lambda} = -\mathrm{lt}(\mathbb{E}[\boldsymbol{W}\boldsymbol{x}]\,\boldsymbol{x}^\top \boldsymbol{W}^\top) = -\mathrm{lt}(\mathbb{E}[\boldsymbol{y}]\boldsymbol{y}^\top), \tag{5}$$

where $\mathrm{lt}(\cdot)$ makes its argument lower triangular by zeroing all elements above the diagonal. Plugging (5) into (3) and stochastically approximating the expectation $\mathbb{E}[\boldsymbol{y}]$ with its instantaneous estimate $\boldsymbol{y}_t := \boldsymbol{W}_t \boldsymbol{x}_t$, where $\boldsymbol{x}_t \in \mathbb{R}^n$ is the observation at time t, yields

$$\partial_{\boldsymbol{W}_t} J(\boldsymbol{W}) = \boldsymbol{y}_t \boldsymbol{x}_t^\top - \mathrm{lt}(\boldsymbol{y}_t \boldsymbol{y}_t^\top)\boldsymbol{W}_t. \tag{6}$$

Gradient ascent in (6) gives the *generalized Hebbian algorithm* (GHA) of Sanger (1989):

$$\boldsymbol{W}_{t+1} = \boldsymbol{W}_t + \eta_t[\boldsymbol{y}_t \boldsymbol{x}_t^\top - \mathrm{lt}(\boldsymbol{y}_t \boldsymbol{y}_t^\top)\boldsymbol{W}_t]. \tag{7}$$

For an appropriate scalar gain, η_t, (7) will tend to converge to the principal component solution as $t \to \infty$; though its global convergence is not proven (Kim et al., 2005).

A closely related algorithm by Oja and Karhunen (1985, Section 5) omits the lt operator:

$$\boldsymbol{W}_{t+1} = \boldsymbol{W}_t + \eta_t[\boldsymbol{y}_t \boldsymbol{x}_t^\top - \boldsymbol{y}_t \boldsymbol{y}_t^\top \boldsymbol{W}_t]. \tag{8}$$

This update is also motivated by maximizing the variance of $\boldsymbol{W}\boldsymbol{x}$ subject to orthonormality constraints on $\boldsymbol{W}$. In contrast to GHA it requires the i^{th} row of $\boldsymbol{W}$ to be orthogonal to *all* other rows of $\boldsymbol{W}$, that is, that $\boldsymbol{W}$ be orthonormal. The resulting algorithm converges to an arbitrary orthonormal basis—not necessarily the eigen-basis—for the subspace spanned by the first r eigenvectors.

One can do better than PCA in minimizing the reconstruction error (1) by allowing *nonlinear* projections of the data into r dimensions. Unfortunately such approaches often pose difficult nonlinear optimization problems. Kernel methods (Schölkopf and Smola, 2002) provide a way to incorporate non-linearity without unduly complicating the optimization problem. Kernel PCA (Schölkopf et al., 1998) performs an eigendecomposition on the kernel expansion of the data, an $l \times l$ matrix. To reduce the attendant $O(l^2)$ space and $O(l^3)$ time complexity, Kim et al. (2005) introduced the *kernel Hebbian algorithm* (KHA) by kernelizing GHA.

Both GHA and KHA are examples of *stochastic approximation* algorithms, whose iterative updates employ individual observations in place of—but, in the limit, approximating—statistical properties of the entire data. By interleaving their updates with the passage through the data, stochastic approximation algorithms can greatly outperform conventional methods on large, redundant data sets, even though their convergence is comparatively slow.

Both GHA and KHA updates incorporate a scalar gain parameter η_t, which is either held fixed or annealed according to some predefined schedule. Robbins and Monro (1951) were first to establish conditions on the sequence of η_t that guarantee the convergence of many stochastic approximation algorithms on stationary input. A widely used annealing schedule (Darken and Moody, 1992) that obeys these conditions is

$$\eta_t = \frac{\tau}{t+\tau}\,\eta_0, \tag{9}$$

where t denotes the iteration number, and η_0, τ are positive tuning parameters. τ determines the length of an initial *search phase* with near-constant gain ($\eta_t \approx \eta_0$ for $t \ll \tau$), before the gain decays asymptotically as τ/t (for $t \gg \tau$) in the *annealing phase* (Darken and Moody, 1992). For non-stationary inputs (e.g., in a online setting) Kim et al. (2005) suggest a small constant gain.

Here we propose the inclusion of a gain vector in the KHA, which provides each estimated eigenvector with its individual gain parameter. In Section 3.1 we describe our KHA/et* algorithm, which sets the gain for each eigenvector inversely proportional to its estimated eigenvalue, in addition to using (9) for annealing. Our KHA/et algorithm in Section 3.3 additionally multiplies the gain vector by the length of the vector of estimated eigenvalues; this allows us to eliminate the τ tuning parameter.

We then derive and apply the *stochastic meta-descent* (SMD) gain vector adaptation technique (Schraudolph, 1999, 2002) to KHA/et* and KHA/et to further speed up their convergence. Our resulting KHA-SMD* and KHA-SMD methods (Section 4.2) adapt gains in a reproducing kernel Hilbert space (RKHS), as pioneered in the recent *Online SVMD* algorithm (Vishwanathan et al., 2006). The application of SMD to the KHA is not trivial; a naive implementation would require $O(rl^2)$ time per update. By incrementally maintaining and updating two auxiliary matrices we reduce this cost to $O(rl)$. Our experiments in Section 5 show that the combination of preconditioning by the estimated eigenvalues and SMD can yield much faster convergence than either technique applied in isolation.

The following section summarizes the KHA, before we provide our eigenvalue-based gain modifications in Section 3. Section 4 describes SMD and its application to the KHA. We report the results of our experiments with these algorithms in Section 5, then conclude with a discussion of our findings.

2. Kernel Hebbian Algorithm

Kim et al. (2005) adapt Sanger's (1989) GHA algorithm to work with data mapped into a reproducing kernel Hilbert space (RKHS) $\mathcal{H}$ via a feature map $\Phi : \mathcal{X} \to \mathcal{H}$ (Schölkopf and Smola, 2002). Here $\mathcal{X}$ is the input space, and $\mathcal{H}$ and Φ are implicitly defined by the kernel $k : \mathcal{X} \times \mathcal{X} \to \mathcal{H}$ with the property $\forall \boldsymbol{x}, \boldsymbol{x}' \in \mathcal{X}\colon\ k(\boldsymbol{x}, \boldsymbol{x}') = \langle \Phi(\boldsymbol{x}), \Phi(\boldsymbol{x}') \rangle_{\mathcal{H}}$, where $\langle \cdot, \cdot \rangle_{\mathcal{H}}$ denotes the inner product in $\mathcal{H}$. Let $\boldsymbol{\Phi}$ denote the transposed data vector in feature space:

$$\boldsymbol{\Phi} := [\Phi(\boldsymbol{x}_1), \Phi(\boldsymbol{x}_2), \ \ldots\ \Phi(\boldsymbol{x}_l)]^\top. \tag{10}$$

This assumes a fixed set of l observations whereas GHA relies on an infinite sequence of observations for convergence. Following Kim et al. (2005), we use an indexing function $p : \mathbb{N} \to \mathbb{Z}_l$ which concatenates random permutations of $\mathbb{Z}_l$ to reconcile this discrepancy. Our implementations loop through a fixed data set, permuting it anew before each pass.

PCA, GHA, and hence KHA all assume that the data is centered. Since the kernel which maps the data into feature space does not necessarily preserve such centering, we must re-center the data in feature space:

$$\boldsymbol{\Phi}' := \boldsymbol{\Phi} - \boldsymbol{M}\boldsymbol{\Phi}, \tag{11}$$

where $\boldsymbol{M}$ denotes the $l \times l$ matrix with entries all equal to $1/l$. This is achieved by replacing the *kernel matrix* $\boldsymbol{K} := \boldsymbol{\Phi}\boldsymbol{\Phi}^\top$ (that is, $[\boldsymbol{K}]_{ij} := k(\boldsymbol{x}_i, \boldsymbol{x}_j)$) by its centered version

$$\begin{aligned} \boldsymbol{K}' := \boldsymbol{\Phi}'\boldsymbol{\Phi}'^\top &= (\boldsymbol{\Phi} - \boldsymbol{M}\boldsymbol{\Phi})(\boldsymbol{\Phi} - \boldsymbol{M}\boldsymbol{\Phi})^\top \\ &= \boldsymbol{\Phi}\boldsymbol{\Phi}^\top - \boldsymbol{M}\boldsymbol{\Phi}\boldsymbol{\Phi}^\top - \boldsymbol{\Phi}\boldsymbol{\Phi}^\top\boldsymbol{M}^\top + \boldsymbol{M}\boldsymbol{\Phi}\boldsymbol{\Phi}^\top\boldsymbol{M}^\top \\ &= \boldsymbol{K} - \boldsymbol{M}\boldsymbol{K} - (\boldsymbol{M}\boldsymbol{K})^\top + \boldsymbol{M}\boldsymbol{K}\boldsymbol{M}. \end{aligned} \tag{12}$$

Since all rows of $\boldsymbol{MK}$ are identical (as are all elements of $\boldsymbol{MKM}$) we can pre-calculate each row in $O(l^2)$ time and store it in $O(l)$ space to efficiently implement operations with the centered kernel. The kernel centered on the training data is also used when testing the trained system on new data.

From kernel PCA (Schölkopf et al., 1998) it is known that the principal components must lie in the span of the centered data in feature space; we can therefore express the GHA weight matrix as $\boldsymbol{W}_t = \boldsymbol{A}_t\boldsymbol{\Phi}'$, where $\boldsymbol{A}$ is an $r \times l$ matrix of expansion coefficients, and r the desired number of principal components. The GHA weight update (7) thus becomes

$$\boldsymbol{A}_{t+1}\boldsymbol{\Phi}' = \boldsymbol{A}_t\boldsymbol{\Phi}' + \eta_t[\boldsymbol{y}_t\boldsymbol{\Phi}'(\boldsymbol{x}_{p(t)})^\top - \mathrm{lt}(\boldsymbol{y}_t\boldsymbol{y}_t^\top)\boldsymbol{A}_t\boldsymbol{\Phi}'], \tag{13}$$

where $\mathrm{lt}(\cdot)$ extracts the lower triangular part of its matrix argument (by setting all matrix elements above the diagonal to zero), and

$$\boldsymbol{y}_t := \boldsymbol{W}_t\boldsymbol{\Phi}'(\boldsymbol{x}_{p(t)}) = \boldsymbol{A}_t\boldsymbol{\Phi}'\boldsymbol{\Phi}'(\boldsymbol{x}_{p(t)}) = \boldsymbol{A}_t\boldsymbol{k}'_{p(t)}, \tag{14}$$

using $\boldsymbol{k}'_i$ to denote the i^{th} column of the centered kernel matrix $\boldsymbol{K}'$. Since we have $\boldsymbol{\Phi}'(\boldsymbol{x}_i)^\top = \boldsymbol{e}_i^\top\boldsymbol{\Phi}'$, where $\boldsymbol{e}_i$ is the unit vector in direction i, (13) can be rewritten solely in terms of expansion coefficients as

$$\boldsymbol{A}_{t+1} = \boldsymbol{A}_t + \eta_t[\boldsymbol{y}_t\boldsymbol{e}_{p(t)}^\top - \mathrm{lt}(\boldsymbol{y}_t\boldsymbol{y}_t^\top)\boldsymbol{A}_t]. \tag{15}$$

Introducing the update coefficient matrix

$$\boldsymbol{\Gamma}_t := \boldsymbol{y}_t\boldsymbol{e}_{p(t)}^\top - \mathrm{lt}(\boldsymbol{y}_t\boldsymbol{y}_t^\top)\boldsymbol{A}_t \tag{16}$$

we obtain the compact update rule

$$\boldsymbol{A}_{t+1} = \boldsymbol{A}_t + \eta_t\boldsymbol{\Gamma}_t. \tag{17}$$

In their experiments, Kim et al. (2005) employed the KHA update (17) with a constant scalar gain $\eta_t = \eta_0$; they also proposed letting the gain decay as $\eta_t = \eta_0/t$. Our implementation (which we denote KHA/t) employs the more general (9) instead, from which an $\eta_0/(t+1)$ decay is obtained by setting $\tau = 1$, and a constant gain in the limit as $\tau \to \infty$.

3. Gain Decay with Reciprocal Eigenvalues

Consider the term $\boldsymbol{y}_t \boldsymbol{x}_t^\top = \boldsymbol{W}_t \boldsymbol{x}_t \boldsymbol{x}_t^\top$ appearing on the right-hand side of the GHA update (7). At the desired solution, the rows of $\boldsymbol{W}_t$ contain the principal components, that is, the leading eigenvectors of $\boldsymbol{Q} = \boldsymbol{X}\boldsymbol{X}^\top$. The elements of $\boldsymbol{y}_t$ thus scale with the associated eigenvalues of $\boldsymbol{Q}$. Large differences in eigenvalues can therefore lead to ill-conditioning (hence slow convergence) of the GHA; the same holds for the KHA.

We counteract this problem by furnishing the KHA with a gain vector $\boldsymbol{\eta}_t \in \mathbb{R}_+^r$ that provides each eigenvector estimate with its individual gain parameter; we will discuss how to set $\boldsymbol{\eta}_t$ below. The update rule (17) thus becomes

$$\boldsymbol{A}_{t+1} = \boldsymbol{A}_t + \operatorname{diag}(\boldsymbol{\eta}_t)\,\boldsymbol{\Gamma}_t\,, \tag{18}$$

where $\operatorname{diag}(\cdot)$ maps a vector into a diagonal matrix.

3.1 The KHA/et* Algorithm

To improve the KHA's convergence, we set $\boldsymbol{\eta}_t$ proportional to the reciprocal of the estimated eigenvalues. Let $\boldsymbol{\lambda}_t \in \mathbb{R}_+^r$ denote the vector of eigenvalues associated with the current estimate of the first r eigenvectors. Our KHA/et* algorithm sets the i^{th} component of $\boldsymbol{\eta}_t$ to

$$[\boldsymbol{\eta}_t]_i = \frac{1}{[\boldsymbol{\lambda}_t]_i}\,\frac{\tau}{t+\tau}\,\eta_0, \tag{19}$$

with η_0 and τ positive tuning parameters as in (9) before. Since we do not want the annealing phase to start before we have seen all observations at least once, we tune τ in small integer multiples of the data set size l.

KHA/et* thus conditions the KHA update by proportionately decreasing (increasing) the gain (19) for rows of $\boldsymbol{A}_t$ associated with large (small) eigenvalues. A similar approach (with a simple $1/t$ gain decay) was applied by Chen and Chang (1995) to GHA for neural network feature selection.

3.2 Calculating the Eigenvalues

The above update (19) requires the first r eigenvalues of $\boldsymbol{K}'$—but the KHA is an algorithm for estimating these eigenvalues and their associated eigenvectors in the first place. The true eigenvalues are therefore not available at run-time. Instead we use the eigenvalues associated with the KHA's current eigenvector estimate in $\boldsymbol{A}_t$, computed as

$$[\boldsymbol{\lambda}_t]_i = \frac{\|\boldsymbol{K}'[\boldsymbol{A}_t]_{i*}^\top\|_2}{\|[\boldsymbol{A}_t]_{i*}^\top\|_2}, \tag{20}$$

where $[\boldsymbol{A}_t]_{i*}$ denotes the i^{th} row of $\boldsymbol{A}_t$, and $\|\cdot\|_2$ the 2-norm of a vector. This can be stated compactly as

$$\boldsymbol{\lambda}_t = \sqrt{\frac{\operatorname{diag}(\boldsymbol{A}_t \boldsymbol{K}'(\boldsymbol{A}_t \boldsymbol{K}')^\top)}{\operatorname{diag}(\boldsymbol{A}_t \boldsymbol{A}_t^\top)}}, \tag{21}$$

where the division and square root operation are performed element-wise, and $\operatorname{diag}(\cdot)$ applied to a matrix extracts the vector of elements along the matrix diagonal.

The main computational effort for calculating $\boldsymbol{\lambda}_t$ lies in computing $\boldsymbol{A}_t\boldsymbol{K}'$, which—if done naively—is quite expensive: $O(rl^2)$. Fortunately it is not necessary to do this at every iteration, since the eigenvalues evolve but gradually. We empirically found it sufficient to update $\boldsymbol{\lambda}_t$ and $\boldsymbol{\eta}_t$ only once after each pass through the data, that is, every l iterations—see Figure 4. Finally, Section 4.2 below introduces incremental updates (33) and (34) that reduce the cost of calculating $\boldsymbol{A}_t\boldsymbol{K}'$ to $O(rl)$.

3.3 The KHA/et Algorithm

The τ parameter of the KHA/et* update (19) above determines at what point in the iterative kernel PCA we gradually shift from the initial search phase (with near-constant $\boldsymbol{\eta}_t$) into the asymptotic annealing phase (with $\boldsymbol{\eta}_t$ near-proportional to $1/t$). It would be advantageous if this parameter could be determined adaptively (Darken and Moody, 1992), obviating the manual tuning required in KHA/et*.

One way to achieve this is to have some measure of progress counteract the gain decay: As long as we are making rapid progress, we are in the search phase, and do not want to decrease the gains; when progress stalls it is time to start annealing them. A suitable measure of progress is $\|\boldsymbol{\lambda}_t\|$, the length of the vector of eigenvalues associated with our current estimate of the eigenvectors, as calculated via (20) above. This quantity is maximized by the true eigenvectors; in the KHA it tends to increase rapidly early on, then approach the maximum asymptotically.

Our KHA/et algorithm fixes the gain decay schedule of KHA/et* at $\tau = l$, but multiplies the gains by $\|\boldsymbol{\lambda}_t\|$:

$$[\boldsymbol{\eta}_t]_i = \frac{\|\boldsymbol{\lambda}_t\|}{[\boldsymbol{\lambda}_t]_i} \frac{l}{t+l} \eta_0. \tag{22}$$

The rapid early growth of $\|\boldsymbol{\lambda}_t\|$ thus serves to counteract the gain decay until the leading eigenspace has been identified. Asymptotically $\|\boldsymbol{\lambda}_t\|$ approaches its (constant) maximum, and so the gain decay will ultimately dominate (22). This achieves an effect comparable to an "adaptive search then converge" (ASTC) gain schedule (Darken and Moody, 1992) while eliminating the τ tuning parameter. Since (19) and (22) can both be expressed as

$$[\boldsymbol{\eta}_t]_i = \frac{\hat{\eta}_t}{[\boldsymbol{\lambda}_t]_i}, \tag{23}$$

for particular choices of $\hat{\eta}_t$, we can compare the gain vectors used by KHA/et* and KHA/et by monitoring how they evolve the scalar $\hat{\eta}_t$; this is shown in Figure 1 for all experiments reported in Section 5. We see that although both algorithms ultimately anneal $\hat{\eta}_t$ in a similar fashion, their behavior early on is quite different: KHA/et keeps a lower initial gain roughly constant for a prolonged search phase, whereas KHA/et* (for the optimal choice of τ) starts decaying $\hat{\eta}_t$ far earlier, albeit from a higher starting value. In Section 5 we shall see how this affects the performance of the two algorithms.

4. KHA with Stochastic Meta-Descent

While KHA/et* and KHA/et make reasonable assumptions about how the gains of a KHA update should be scaled, further improvements are possible by adapting gains in response to the observed

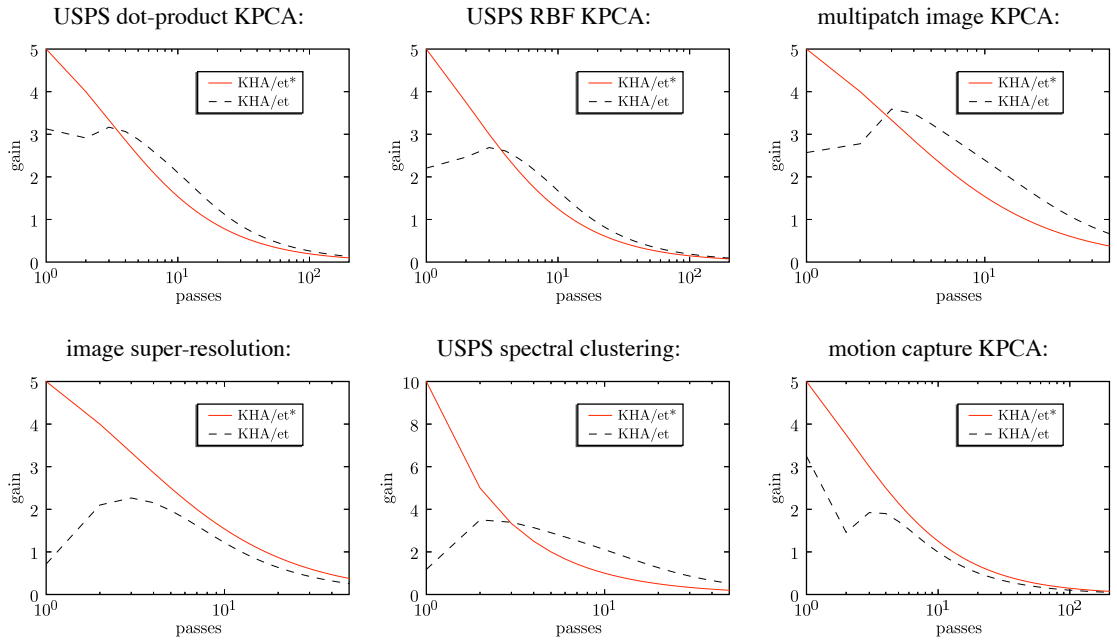


Figure 1: Comparison of gain $\hat{\eta}_t$ (23) between KHA/et* and KHA/et in all applications reported in Section 5, at individually optimal values of η_0 and (for KHA/et*) τ.

history of parameter updates so as to optimize convergence. We briefly review gradient-based gain adaptation methods, then derive and implement Schraudolph's (1999; 2002) *stochastic meta-descent* (SMD) algorithm for both KHA/et* and KHA/et, focusing on the scalar form of SMD that can be used in an RKHS.

4.1 Scalar Stochastic Meta-Descent

Let V be a vector space, $\boldsymbol{\theta} \in V$ a parameter vector, and $J : V \to \mathbb{R}$ the objective function which we would like to optimize. We assume that J is twice differentiable almost everywhere. Denote by $J_t : V \to \mathbb{R}$ the stochastic approximation of the objective function at time t. Our goal is to find $\boldsymbol{\theta}$ such that $\mathbb{E}_t[J_t(\boldsymbol{\theta})]$ is minimized. We adapt $\boldsymbol{\theta}$ via the stochastic gradient descent

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - e^{\rho_t} \boldsymbol{g}_t, \quad \text{where} \quad \boldsymbol{g}_t = \partial_{\boldsymbol{\theta}_t} J_t(\boldsymbol{\theta}_t), \tag{24}$$

using $\partial_{\boldsymbol{\theta}_t}$ as a shorthand for $\frac{\partial}{\partial \boldsymbol{\theta}}\Big|_{\boldsymbol{\theta}=\boldsymbol{\theta}_t}$. Stochastic gradient descent is sensitive to the value of the *log-gain* $\rho_t \in \mathbb{R}$: If it is too small, (24) will take many iterations to converge; if it is too large, (24) may diverge.

One solution is to adapt ρ_t by a simultaneous meta-level gradient descent. Thus we could seek to minimize the value of the objective at the next iteration by adjusting ρ_t in proportion to the gradient

$\partial_{\rho_t} J_{t+1}(\boldsymbol{\theta}_{t+1})$. Using the chain rule and (24) we find

$$\begin{aligned}\rho_{t+1} &= \rho_t - \mu\, \partial_{\rho_t} J_{t+1}(\boldsymbol{\theta}_{t+1}) \\ &= \rho_t - \mu\, [\partial_{\boldsymbol{\theta}_{t+1}} J_{t+1}(\boldsymbol{\theta}_{t+1})]^\top \partial_{\boldsymbol{\rho}_t} \boldsymbol{\theta}_{t+1} \\ &= \rho_t + \mu\, e^{\rho_t} \boldsymbol{g}_{t+1}^\top \boldsymbol{g}_t,\end{aligned} \tag{25}$$

where the *meta-gain* $\mu \geq 0$ is a scalar tuning parameter. Intuitively, the gain adaptation (25) is driven by the angle between successive gradient measurements: If it is less than 90°, then $\boldsymbol{g}_{t+1}^\top \boldsymbol{g}_t > 0$, and ρ_t will be increased. Conversely, if the angle is more than 90° (oscillating gradient), then ρ_t will be decreased because $\boldsymbol{g}_{t+1}^\top \boldsymbol{g}_t < 0$. Thus (25) serves to *decorrelate* successive gradients, which leads to improved convergence of (24).

One shortcoming of (25) is that the decorrelation occurs only across a single time step, making the gain adaptation overly sensitive to spurious short-term correlations in the data. *Stochastic meta-descent* (SMD; Schraudolph, 1999, 2002) addresses this issue by employing an exponentially decaying trace of gradients across time:

$$\begin{aligned}\rho_{t+1} &= \rho_t - \mu \sum_{i=0}^{t} \xi^i \partial_{\rho_{t-i}} J_{t+1}(\boldsymbol{\theta}_{t+1}) \\ &= \rho_t - \mu\, [\partial_{\boldsymbol{\theta}_{t+1}} J_{t+1}(\boldsymbol{\theta}_{t+1})]^\top \sum_{i=0}^{t} \xi^i \partial_{\boldsymbol{\rho}_{t-i}} \boldsymbol{\theta}_{t+1} \\ &=: \rho_t - \mu\, \boldsymbol{g}_{t+1}^\top \boldsymbol{v}_{t+1},\end{aligned} \tag{26}$$

where the vector $\boldsymbol{v}_{t+1} \in V$ characterizes the dependence of $\boldsymbol{\theta}_{t+1}$ on its gain history over a time scale governed by the decay factor $0 \leq \xi \leq 1$, a scalar tuning parameter.

To compute $\boldsymbol{v}_{t+1}$ efficiently, we expand $\boldsymbol{\theta}_{t+1}$ in terms of its recursive definition (24):

$$\begin{aligned}\boldsymbol{v}_{t+1} &:= \sum_{i=0}^{t} \xi^i \partial_{\rho_{t-i}} \boldsymbol{\theta}_{t+1} \\ &= \sum_{i=0}^{t} \xi^i \partial_{\rho_{t-i}} \boldsymbol{\theta}_t - \sum_{i=0}^{t} \xi^i \partial_{\rho_{t-i}} [e^{\rho_t} \boldsymbol{g}_t] \\ &\approx \xi \boldsymbol{v}_t - e^{\rho_t} (\boldsymbol{g}_t + \partial_{\boldsymbol{\theta}_t} \boldsymbol{g}_t \sum_{i=0}^{t} \xi^i \partial_{\rho_{t-i}} \boldsymbol{\theta}_t).\end{aligned} \tag{27}$$

Here we have used $\partial_{\rho_t} \boldsymbol{\theta}_t = \mathbf{0}$, and approximated

$$\sum_{i=1}^{t} \xi^i \partial_{\rho_{t-i}} \rho_t \approx 0, \tag{28}$$

which amounts to stating that the log-gain adaptation must be in equilibrium on the time scale determined by ξ. Noting that $\partial_{\boldsymbol{\theta}_t} \boldsymbol{g}_t$ is the Hessian $\boldsymbol{H}_t$ of $J_t(\boldsymbol{\theta}_t)$, we arrive at the simple iterative update

$$\boldsymbol{v}_{t+1} = \xi \boldsymbol{v}_t - e^{\rho_t} (\boldsymbol{g}_t + \xi \boldsymbol{H}_t \boldsymbol{v}_t). \tag{29}$$

Since the initial parameters $\boldsymbol{\theta}_0$ do not depend on any gains, $\boldsymbol{v}_0 = \mathbf{0}$. Note that for $\xi = 0$ (29) and (26) reduce to the single-step gain adaptation (25).

Computation of the Hessian-vector product $\boldsymbol{H}_t \boldsymbol{v}_t$ would be expensive if done naively. Fortunately, efficient methods exist to calculate this quantity directly without computing the Hessian (Pearlmutter, 1994; Griewank, 2000; Schraudolph, 2002). In essence, these methods work by propagating $\boldsymbol{v}$ as a differential (i.e., directional derivative) through the gradient computation:

$$\mathrm{d}\boldsymbol{\theta}_t := \boldsymbol{v}_t \quad \Rightarrow \quad \boldsymbol{H}_t \boldsymbol{v}_t := \mathrm{d}\boldsymbol{g}_t. \tag{30}$$

In other words, if we set the differential $\mathrm{d}\boldsymbol{\theta}_t$ of the parameter vector to $\boldsymbol{v}_t$, then the resulting differential of the gradient $\boldsymbol{g}_t$ (a function of $\boldsymbol{\theta}_t$) is the Hessian-vector product $\boldsymbol{H}_t \boldsymbol{v}_t$. We will see this at work for the case of the KHA in (36) below.

4.2 SMD for KHA

The KHA update (18) can be viewed as r coupled updates in RKHS, one for each row of $\boldsymbol{A}_t$, each associated with a scalar gain. To apply SMD here we introduce an additional log-gain vector $\boldsymbol{\rho}_t \in \mathbb{R}^r$

$$\boldsymbol{A}_{t+1} = \boldsymbol{A}_t + e^{\mathrm{diag}(\boldsymbol{\rho}_t)} \,\mathrm{diag}(\boldsymbol{\eta}_t)\, \boldsymbol{\Gamma}_t. \tag{31}$$

(The exponential of a diagonal matrix is obtained simply by exponentiating the individual diagonal entries.) We are thus applying SMD to KHA/et, that is, to a gradient descent *preconditioned* by the reciprocal estimated eigenvalues. SMD will happily work with such a preconditioner, and benefit from it.

In an RKHS, SMD adapts a *scalar* log-gain whose update is driven by the inner product between the gradient and a differential of the system parameters, all in the RKHS (Vishwanathan et al., 2006). In the case of KHA, $\boldsymbol{\Gamma}_t \boldsymbol{\Phi}'$ can be interpreted as the gradient in the RKHS of the merit function (2) maximized by KHA. Therefore SMD's adaptation of $\boldsymbol{\rho}_t$ in (31) is driven by the diagonal entries of $\langle \boldsymbol{\Gamma}_t \boldsymbol{\Phi}', \boldsymbol{B}_t \boldsymbol{\Phi}' \rangle_{\mathcal{H}}$, where $\boldsymbol{B}_t := \mathrm{d}\boldsymbol{A}_t$ denotes the $r \times l$ matrix of expansion coefficients for SMD's differential parameters, analogous to the $\boldsymbol{v}$ vector in Section 4.1:

$$\begin{aligned} \boldsymbol{\rho}_t &= \boldsymbol{\rho}_{t-1} + \mu \,\mathrm{diag}(\langle \boldsymbol{\Gamma}_t \boldsymbol{\Phi}', \boldsymbol{B}_t \boldsymbol{\Phi}' \rangle_{\mathcal{H}}) \\ &= \boldsymbol{\rho}_{t-1} + \mu \,\mathrm{diag}(\boldsymbol{\Gamma}_t \boldsymbol{\Phi}' \boldsymbol{\Phi}'^{\top} \boldsymbol{B}_t^{\top}) \\ &= \boldsymbol{\rho}_{t-1} + \mu \,\mathrm{diag}(\boldsymbol{\Gamma}_t \boldsymbol{K}' \boldsymbol{B}_t^{\top}). \end{aligned} \tag{32}$$

Naive computation of $\boldsymbol{\Gamma}_t \boldsymbol{K}'$ in (32) would cost $O(rl^2)$ time, which is prohibitively expensive for large l. We can, however, reduce this cost to $O(rl)$ by noting that (16) implies that

$$\begin{aligned} \boldsymbol{\Gamma}_t \boldsymbol{K}' &= \boldsymbol{y}_t \boldsymbol{e}_{p(t)}^{\top} \boldsymbol{K}' - \mathrm{lt}(\boldsymbol{y}_t \boldsymbol{y}_t^{\top}) \boldsymbol{A}_t \boldsymbol{K}' \\ &= \boldsymbol{y}_t \boldsymbol{k}'^{\top}_{p(t)} - \mathrm{lt}(\boldsymbol{y}_t \boldsymbol{y}_t^{\top}) \boldsymbol{A}_t \boldsymbol{K}', \end{aligned} \tag{33}$$

where the $r \times l$ matrix $\boldsymbol{A}_t \boldsymbol{K}'$ can be stored and updated incrementally via (31):

$$\boldsymbol{A}_{t+1} \boldsymbol{K}' = \boldsymbol{A}_t \boldsymbol{K}' + e^{\mathrm{diag}(\boldsymbol{\rho}_t)} \,\mathrm{diag}(\boldsymbol{\eta}_t)\, \boldsymbol{\Gamma}_t \boldsymbol{K}'. \tag{34}$$

The initial computation of $\boldsymbol{A}_1 \boldsymbol{K}'$ still costs $O(rl^2)$ in general but is affordable as it is performed only once. Alternatively, the time complexity of this step can easily be reduced to $O(rl)$ by making $\boldsymbol{A}_1$ suitably sparse.

Finally, we apply SMD's standard update (29) of the differential parameters:

$$\boldsymbol{B}_{t+1} = \xi \boldsymbol{B}_t + e^{\mathrm{diag}(\boldsymbol{\rho}_t)} \,\mathrm{diag}(\boldsymbol{\eta}_t)\,(\boldsymbol{\Gamma}_t + \xi \mathrm{d}\boldsymbol{\Gamma}_t). \tag{35}$$

The differential $\mathrm{d}\boldsymbol{\Gamma}_t$ of the gradient, analogous to $\mathrm{d}\boldsymbol{g}_t$ in (30), can be computed by applying the rules of calculus:

$$\begin{aligned}
\mathrm{d}\boldsymbol{\Gamma}_t &= \mathrm{d}[\boldsymbol{y}_t \boldsymbol{e}_{p(t)}^\top - \mathrm{lt}(\boldsymbol{y}_t \boldsymbol{y}_t^\top)\boldsymbol{A}_t] \\
&= (\mathrm{d}\boldsymbol{A}_t)\boldsymbol{k}'_{p(t)}\boldsymbol{e}_{p(t)}^\top - \mathrm{lt}(\boldsymbol{y}_t \boldsymbol{y}_t^\top)(\mathrm{d}\boldsymbol{A}_t) - [\mathrm{d}\,\mathrm{lt}(\boldsymbol{y}_t \boldsymbol{y}_t^\top)]\boldsymbol{A}_t \\
&= \boldsymbol{B}_t \boldsymbol{k}'_{p(t)}\boldsymbol{e}_{p(t)}^\top - \mathrm{lt}(\boldsymbol{y}_t \boldsymbol{y}_t^\top)\boldsymbol{B}_t - \mathrm{lt}(\boldsymbol{B}_t \boldsymbol{k}'_{p(t)}\boldsymbol{y}_t^\top + \boldsymbol{y}_t \boldsymbol{k}'^\top_{p(t)}\boldsymbol{B}_t^\top)\boldsymbol{A}_t,
\end{aligned} \tag{36}$$

using the fact that since $\boldsymbol{k}'$ and $\boldsymbol{e}$ are both independent of $\boldsymbol{A}$ we have $\mathrm{d}(\boldsymbol{k}'_{p(t)}\boldsymbol{e}_{p(t)}^\top) = 0$. Inserting (16) and (36) into (35) finally yields the update rule

$$\begin{aligned}
\boldsymbol{B}_{t+1} = \xi \boldsymbol{B}_t + e^{\mathrm{diag}(\boldsymbol{\rho}_t)} \,\mathrm{diag}(\boldsymbol{\eta}_t)[&(\boldsymbol{A}_t + \xi \boldsymbol{B}_t)\,\boldsymbol{k}'_{p(t)}\boldsymbol{e}_{p(t)}^\top \\
&- \mathrm{lt}(\boldsymbol{y}_t \boldsymbol{y}_t^\top)(\boldsymbol{A}_t + \xi \boldsymbol{B}_t) - \xi\,\mathrm{lt}(\boldsymbol{B}_t \boldsymbol{k}'_{p(t)}\boldsymbol{y}_t^\top + \boldsymbol{y}_t \boldsymbol{k}'^\top_{p(t)}\boldsymbol{B}_t^\top)\boldsymbol{A}_t].
\end{aligned} \tag{37}$$

In summary, our application of SMD to the KHA comprises Equations (32), (37), and (31), in that order. Our approach allows us to incorporate *a priori* knowledge about suitable gains in $\boldsymbol{\eta}_t$, which SMD will then improve upon by using empirical information gathered along the update trajectory to adaptively tune $\boldsymbol{\rho}_t$.

Algorithm 1 shows KHA-SMD, the algorithm obtained by applying SMD to KHA/et in this fashion. To obtain KHA-SMD*, the analogous algorithm applying SMD to KHA/et*, simply change step 2(b) to use (19) instead of (22). To recover KHA/et *resp*. KHA/et* from Algorithm 1, omit the steps marked with a single vertical bar. The double-barred steps do not have to be performed on every iteration; omitting them entirely, along with the single-barred steps, recovers the original KHA algorithm.

We list the worst-case time complexity of every step in terms of the number l and dimensionality n of observations, and the number r of kernel principal components to extract. For $r \ll n$ (as is typical), the most expensive step in the iteration loop will be the computation of a row of the kernel matrix in 2(c), required by all algorithms.

We initialize $\boldsymbol{\rho}_0$ to all ones, $\boldsymbol{B}_1$ to all zeroes, and $\boldsymbol{A}_1$ to an isotropic normal density with suitably small variance. The resulting time complexity of $O(rl^2)$ of step 1(c) can easily be reduced to $O(rl)$ by initializing $\boldsymbol{A}_1$ sparsely in step 1(b). This leaves the centering of the kernel in step 1(a), required by all algorithms, as the most expensive initialization step.

5. Experiments

We present two sets of experiments. In the first, we benchmark against the KHA with a conventional gain decay schedule (9), which we denote KHA/t, in a number of different settings: Performing kernel PCA and spectral clustering on the well-known USPS data set (LeCun et al., 1989), replicating image denoising and face image super-resolution experiments of Kim et al. (2005), and denoising human motion capture data. For Kim et al.'s (2005) experiments we also compare to their original KHA with the constant gain $\eta_t = \eta_0$ they employed. A common feature of all these data sets is that the kernel matrix can be stored in main memory, and the optimal reconstruction can thus be

Algorithm 1 KHA-SMD

		Eq.no.	time complexity
1.	Initialize:		
	(a) calculate $\boldsymbol{MK}$, $\boldsymbol{MKM}$		$O(l^2)$
	(b) $\boldsymbol{A}_1 \sim N(\boldsymbol{0}, (rl)^{-1}\boldsymbol{I})$		$O(rl)$
	(c) calculate $\boldsymbol{A}_1\boldsymbol{K}'$		$O(rl^2)$
	(d) $\boldsymbol{\rho}_0 := [1 \dots 1]^\top$, $\boldsymbol{B}_1 := \boldsymbol{0}$		$O(rl)$
2.	**Repeat** for $t = 1, 2, \dots$		
	(a) calculate $\boldsymbol{\lambda}_t$	(20)	$O(rl)$
	(b) calculate $\boldsymbol{\eta}_t$	(22)	$O(r)$
	(c) calculate $\boldsymbol{k}'_{p(t)}$		$O(nl)$
	(d) calculate $\boldsymbol{y}_t$	(14)	$O(rl)$
	(e) calculate $\boldsymbol{\Gamma}_t$	(16)	$O(rl)$
	(f) calculate $\boldsymbol{\Gamma}_t\boldsymbol{K}'$	(33)	$O(rl)$
	(g) update $\boldsymbol{\rho}_{t-1} \to \boldsymbol{\rho}_t$	(32)	$O(rl)$
	(h) update $\boldsymbol{B}_t \to \boldsymbol{B}_{t+1}$	(37)	$O(rl)$
	(i) update $\boldsymbol{A}_t\boldsymbol{K}' \to \boldsymbol{A}_{t+1}\boldsymbol{K}'$	(34)	$O(rl)$
	(j) update $\boldsymbol{A}_t \to \boldsymbol{A}_{t+1}$	(31)	$O(rl)$

Experiment	Section	σ	τ^1	τ^2	$\eta_0{}^1$	$\eta_0{}^2$	$\eta_0{}^3$	μ^4	μ^5	ξ
USPS (dot-prod. kernel)	5.1.1	–	$2l$	$4l$	.002	5	10^{-3}	10^{-5}	10^{-4}	0.99
USPS (RBF kernel)	5.1.1	8	l	$3l$	1	5	0.2	0.05	0.1	0.99
Lena image denoising	5.1.2	1	l	$4l$	2	5	0.1	1	2	0.99
face super-resolution	5.1.3	1	l	$4l$	0.2	5	0.02	0.2	5	0.99
USPS spectral clustering	5.1.4	8	l	l	200	10	50	20	10^3	0.99
motion capture KPCA	5.1.5	$\sqrt{1.5}$	l	$3l$	2	5	0.1	0.1	1	0.99

[1]for KHA/t [2]for KHA/et*, KHA/SMD* [3]for KHA/et, KHA/SMD [4]for KHA/SMD* [5]for KHA/SMD

Table 1: Parameter settings for our experiments. Footnotes indicate parameters which were individually tuned for each experiment and the given algorithm(s).

computed with a conventional eigensolver. In our second set of experiments we demonstrate scalability by performing kernel PCA on 60000 digits from the MNIST data set (LeCun, 1998). Here the kernel matrix cannot be stored in main memory of a standard PC, and hence one is forced to resort to iterative methods.

5.1 Experiments on Small Data Sets

In these experiments the KHA and our enhanced variants are used to find the first r eigenvectors of the centered kernel matrix $\boldsymbol{K}'$. To assess the quality of the solution, we reconstruct the kernel matrix using the eigenvectors found by the iterative algorithms, and measure the reconstruction error

$$\mathcal{E}(\boldsymbol{A}) := \|\boldsymbol{K}' - (\boldsymbol{A}\boldsymbol{K}')^\top \boldsymbol{A}\boldsymbol{K}'\|_F. \tag{38}$$

Since the kernel matrix can be stored in memory, the optimal reconstruction error from r eigenvectors, $\mathcal{E}_{\min} := \min_{\boldsymbol{A}} \mathcal{E}(\boldsymbol{A})$, is computed with a conventional eigensolver. This allows us to report reconstruction errors as excess errors relative to the optimal reconstruction, that is, $\mathcal{E}(\boldsymbol{A})/\,\mathcal{E}_{\min} - 1$.

To compare algorithms we plot the excess reconstruction error on a logarithmic scale after each pass through the entire data set. This is a fair comparison since the overhead for KHA/et*, KHA/et, and their SMD versions is negligible compared to the time required by the KHA base algorithm: The most expensive operations—the initial centering of the kernel matrix, and the repeated calculation of a row of it—are shared by all these algorithms.

Each non-SMD algorithm had η_0 and (where applicable) τ manually tuned, by iterated hill-climbing over $\eta_0 \in \{a \cdot 10^b : a \in \{1,2,5\},\, b \in \{-3,-2,-1,0,1,2\}\}$ and $\tau \in \{l, 2l, 3l, 4l, 5l, 7l, 10l, 15l, 20l, 30l, 40l, 50l\}$, for the lowest final reconstruction error in each experiment. The SMD versions used the same values of η_0 and τ as their corresponding non-SMD variant; for them we hand-tuned μ (over the same set of values as η_0), and set $\xi = 0.99$ *a priori* throughout. Thus KHA/t and KHA/et* each had two parameters tuned specifically for them, the other algorithms one. Table 1 lists the parameter settings for each experiment, with the individually tuned parameters indicated.

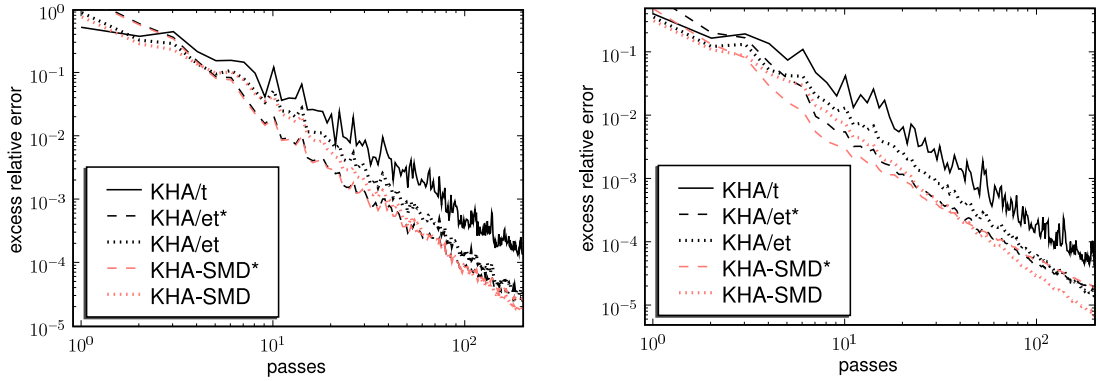


Figure 2: Excess relative reconstruction error of KHA variants for kernel PCA (16 eigenvectors) on the USPS data, using a dot-product (left) *resp*. RBF (right) kernel. (On the left, the curves for KHA/et* and KHA-SMD* virtually coincide.)

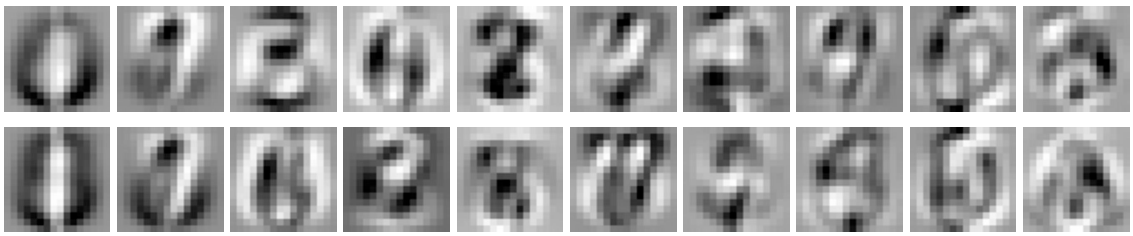


Figure 3: First ten eigenvectors (from left to right) found by KHA/et* for the dot-product (top row) *resp*. RBF kernel (bottom row).

5.1.1 USPS DIGIT KPCA

Our first benchmark is to perform iterative kernel PCA on a subset of the well-known USPS data set (LeCun et al., 1989)—namely, the first 100 samples of each digit—with two different kernel functions: the dot-product kernel[2]

$$k(\boldsymbol{x},\boldsymbol{x}') = \boldsymbol{x}^\top \boldsymbol{x}' \tag{39}$$

and the RBF kernel

$$k(\boldsymbol{x},\boldsymbol{x}') = \exp\left(\frac{(\boldsymbol{x}-\boldsymbol{x}')^\top(\boldsymbol{x}-\boldsymbol{x}')}{2\sigma^2}\right) \tag{40}$$

with $\sigma = 8$, the value used by Mika et al. (1999). We extract the first 16 eigenvectors of the kernel matrix and plot the excess relative error in Figure 2. Although KHA/et and KHA/et* differ in their transient behavior—the former performing better for the first 6 passes through the data, the latter thereafter—their error after 200 passes is quite similar; both clearly outperform KHA/t. SMD is able

2. Kernel PCA with a dot-product kernel is equivalent to ordinary PCA in the input space.

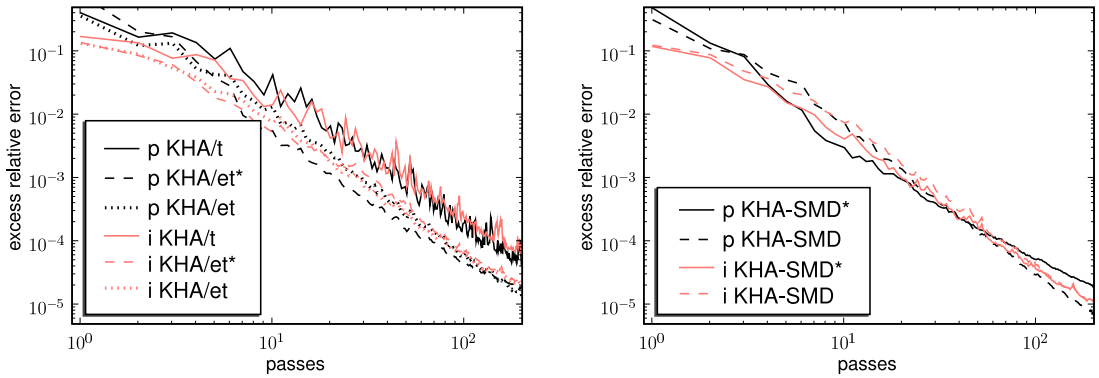


Figure 4: Comparison of excess relative reconstruction error of KHA variants estimating eigenvalues and updating gains every iteration ('i') *vs*. once every pass ('p') through the USPS data, for RBF kernel PCA extracting 16 eigenvectors.

Figure 5: Lena image—original (left), noisy (center), and denoised by KHA-SMD (right).

to significantly improve the performance of KHA/et but not KHA/et*, and so KHA-SMD achieves the best results on this task. These results hold for either choice of kernel. We show the first 10 eigenvectors obtained by KHA/et* for each kernel in Figure 3.

In Figure 4 we compare the performance of our algorithms, which estimate the eigenvalues and update the gains only once after every pass through the data ('p'), against variants ('i') which do this after every iteration. Tuning parameters were re-optimized for the new variants, though most optimal settings remained the same.[3] Updating the estimated eigenvalues after every iteration, though computationally expensive, is beneficial initially but does not seem to affect the quality of the solution much in the long run; the minor differences that can be observed are attributable to differences in parameter settings.

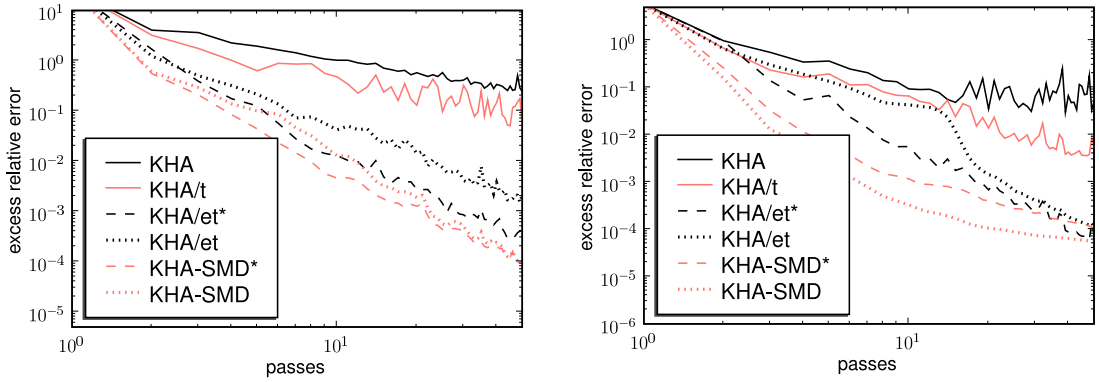


Figure 6: Excess relative reconstruction error of KHA variants in our replication of experiments due to Kim et al. (2005). Left: multipatch image kernel PCA on a noisy Lena image; Right: super-resolution of face images.

Figure 7: Reconstructed Lena image after (left to right) 1, 2, and 3 passes through the data set, for KHA with constant gain $\eta_t = 0.05$ (top row) *vs*. KHA-SMD (bottom row).

5.1.2 MULTIPATCH IMAGE DENOISING

For our second benchmark we replicate the image denoising problem of Kim et al. (2005), the idea being that noise can be removed from images by reconstructing image patches from their r leading

3. The exceptions were minor: $\tau = 4$ (instead of $\tau = 3$) for KHA/et* and KHA-SMD*, $\mu = 0.1$ (instead of $\mu = 0.05$) for KHA-SMD*, and $\mu = 0.05$ (instead of $\mu = 0.1$) for KHA-SMD.

eigenvectors. We divide the well-known Lena image (Munson, 1996) into four sub-images, from which 11×11 pixel windows are sampled on a grid with two-pixel spacing to produce 3844 vectors of 121 pixel intensity values each. Following Kim et al. (2005) we use an RBF kernel with $\sigma = 1$ to find the 20 best eigenvectors for each sub-image. Results averaged over the four sub-images are plotted in Figure 6 (left), including the KHA with constant gain of $\eta_t = 0.05$ employed by Kim et al. (2005) for comparison. The original, noisy, and denoised Lena images are shown in Figure 5.

KHA/t, while better than the conventional KHA with constant gain, is clearly not as effective as our methods. Of these, KHA/et is outperformed by KHA/et* but benefits more from the addition of SMD, so that the performance of KHA-SMD is almost comparable to KHA-SMD*. KHA-SMD and KHA-SMD* achieved an excess reconstruction error that is over three orders of magnitude better than the conventional KHA after 50 passes through the data.

Replicating Kim et al.'s (2005) 800 passes through the data with the constant-gain KHA we obtain an excess relative reconstruction error of 5.64%, 500 times that of KHA-SMD after 50 passes. The signal-to-noise ratio (SNR) of the reconstruction after 800 passes with constant gain is 13.46,[4] comparable to the SNR of 13.49 achieved by KHA/et* in 50 passes.

To illustrate the large difference in early performance between conventional KHA and KHA-SMD, we show the images reconstructed from either method after 1, 2, and 3 passes through the data set in Figure 7. KHA-SMD delivers good-quality reconstructions very quickly, while those of the conventional KHA are rather blurred.

We now investigate how the different components of KHA-SMD* affect its performance. The overall gain used by KHA-SMD* comprises three factors: the scheduled gain decay over time (9), the reciprocal of the current estimated eigenvalues, and the gain adapted by SMD. Let us denote these three factors as t, e, and s, respectively, and explore which of their combinations make sense. We clearly need either t or s to give us some form of gain decay, which e does not provide. This means that in addition to the KHA/t (using only t), KHA/et* (t and e), and KHA-SMD* (t, e, and s) algorithms, there are three more feasible variants: a) s alone, b) t and s, and c) e and s.

We compare the performance of these "anonymous" variants to that of KHA/t, KHA/et*, and KHA-SMD* on the Lena image denoising problem. Parameters were tuned for each variant individually, yielding $\eta_0 = 0.5$ and $\mu = 2$ for variant s, $\eta_0 = 1$ and $\mu = 2$ for variant es, and $\tau = l$, $\eta_0 = 2$, and $\mu = 1$ for variant ts. Figure 8 (left) shows the excess relative error as a function of the number of passes through the data. On its own, SMD (s) outperforms the scheduled gain decay (t), but combining the two (ts) is better still. Introducing the reciprocal eigenvalues (e) further improves performance in every context. In short, all three factors convey a significant benefit, both individually and in combination. The "anonymous" variants represent intermediate forms between the (poorly performing) KHA/t and KHA-SMD*, which combines all three factors to attain the best results.

Next we examine the sensitivity of the KHA with SMD to the value of the meta-gain μ by increasing $\mu \in \{a \cdot 10^b : a \in \{1,2,5\}, b \in \{-1,0,1\}\}$ until the algorithm diverges. Figure 8 (right) plots the excess relative error of the s variant (SMD alone, black) and KHA-SMD* (light red) on the Lena image denoising problem for the last three values of μ prior to divergence. In both cases the largest non-divergent meta-gain ($\mu = 2$ for s, $\mu = 1$ for KHA-SMD*) yields the fastest convergence. The differences are comparatively small though, illustrating that SMD is not overly sensitive to the

4. Kim et al. (2005) reported an SNR of 14.09; the discrepancy is due to different reconstruction methods.

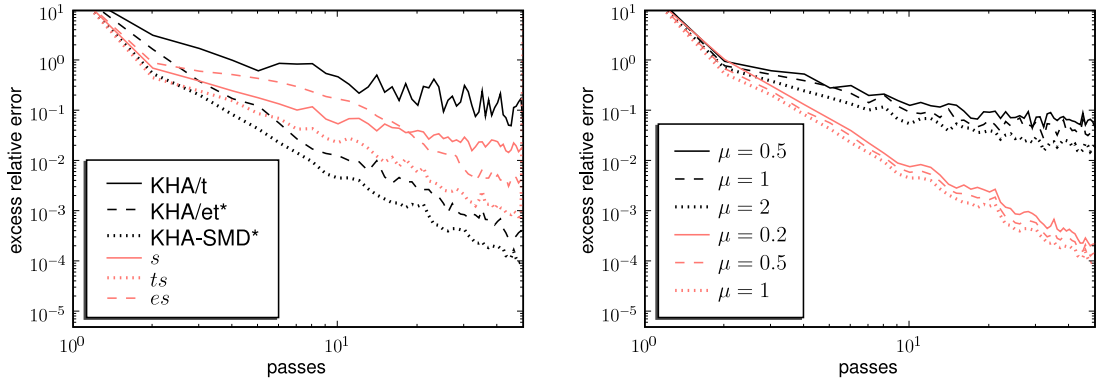


Figure 8: Excess relative reconstruction error for multipatch image PCA on a noisy Lena image. Left: comparison of original KHA variants (black) with those using other combinations (light red) of gain decay (t), reciprocal eigenvalues (e), and SMD (s). Right: effect of varying μ on the convergence of variant s (black) and KHA-SMD* (light red).

value of μ. This holds in particular for KHA-SMD*, where SMD is assisted by the other two factors, t and e.

5.1.3 FACE IMAGE SUPER-RESOLUTION

We also replicate a face image super-resolution experiment of Kim et al. (2005). Here the eigenvectors learned from a training set of high-resolution images are used to predict high-resolution detail from low-resolution test images. The training set consists of 5000 face images of 10 different people from the Yale face database B (Georghiades et al., 2001), down-sampled to 60×60 pixels. Testing is done on 10 different images from the same database; the test images are first down-sampled to 20×20 pixels, then scaled back up to 60×60 by mapping each pixel to a 3×3 block of identical pixel values. These are then projected into a 16-dimensional eigenspace learned from the training set to predict the test images at the 60×60 pixel resolution.

Figure 6 (right) plots the excess relative reconstruction error of the different algorithms on this task. KHA/t again produces better results than the KHA with constant gain but is ineffective compared to our methods. KHA/et* again does better than KHA/et but benefits less from the addition of SMD making SMD-KHA once more the best-performing method. After 50 passes through the data, all our methods achieve an excess reconstruction error about three orders of magnitude better than the conventional KHA, though KHA-SMD is substantially faster than the others at reaching this level of performance. Figure 9 illustrates that the reconstructed face images after one pass through the training data generally show better high-resolution detail for KHA-SMD than for the conventional KHA with constant gain.

5.1.4 SPECTRAL CLUSTERING OF USPS DIGITS

Our next experiment uses the spectral clustering algorithm of Ng et al. (2002):

Figure 9: Rows from top to bottom: Original face images (60×60 pixels); sub-sampled images (20×20 pixels); super-resolution images produced by KHA after one pass through the data set; likewise for KHA-SMD.

1. Define the normalized transition matrix $\boldsymbol{P} := \boldsymbol{D}^{-\frac{1}{2}}\boldsymbol{K}\boldsymbol{D}^{-\frac{1}{2}}$, where $\boldsymbol{K} \in \mathbb{R}^{l\times l}$ is the kernel matrix of the data, and $\boldsymbol{D}$ is a diagonal matrix with $[\boldsymbol{D}]_{ii} = \sum_j [\boldsymbol{K}]_{ij}$.
2. Let $\boldsymbol{A} \in \mathbb{R}^{r\times l}$ be the matrix whose rows correspond to the first r eigenvectors of $\boldsymbol{P}$.
3. Normalize the columns of $\boldsymbol{A}$ to unit length, and map each input pattern to its corresponding column in $\boldsymbol{A}$.
4. Cluster the columns of $\boldsymbol{A}$ into r clusters (using, for instance, k-means clustering), and assign each pattern to the cluster its corresponding column vector belongs to.

We can obviously employ the KHA in Step 2 above. We evaluate our results in terms of the variation of information (VI) metric (Meila, 2005): For a clustering algorithm c, let $|c|$ denote the number of clusters, and $c(\cdot)$ its cluster assignment function, that is, $c(\boldsymbol{x}_i) = j$ iff c assigns pattern $\boldsymbol{x}_i$ to cluster j. Let $P_c \in \mathbb{R}^{|c|}$ denote the probability vector whose j^{th} component denotes the fraction of points assigned to cluster j, and H_c the entropy associated with P_c:

$$H_c = -\sum_{i=1}^{|c|} [P_c]_i \ln [P_c]_i. \tag{41}$$

Given two clustering algorithms c and c' we define the confusion matrix $P_c^{c'} \in \mathbb{R}^{|c|\times|c'|}$ by

$$[P_c^{c'}]_{km} = \frac{1}{l}|\{i|(c(x_i) = k) \wedge (c'(x_i) = m)\}|, \tag{42}$$

where l is the number of patterns. The mutual information $I_c^{c'}$ associated with $P_c^{c'}$ is

$$I_c^{c'} = \sum_{i=1}^{|c|}\sum_{j=1}^{|c'|} [P_c^{c'}]_{ij} \ln \frac{[P_c^{c'}]_{ij}}{[P_c]_i [P_{c'}]_j}. \tag{43}$$

The VI metric is now defined as

$$\text{VI} = H_c + H_{c'} - 2I_c^{c'}. \tag{44}$$

Our experimental task consists of applying spectral clustering to all 7291 patterns of the USPS data (LeCun et al., 1989), using 10 kernel principal components. We used a Gaussian kernel with $\sigma = 8$ and k-means with $k = 10$ (the number of labels) for clustering the columns of $\boldsymbol{A}$. The clusterings obtained by our algorithms are compared to the clustering induced by the class labels. On the USPS data, a VI of 4.54 corresponds to random grouping, while clustering in perfect accordance with the class labels would give a VI of zero.

In Figure 10 (left) we plot the VI metric as a function of the number of passes through the data. All our accelerated KHA variants converge towards an optimal clustering in less than 10 passes—in fact, after around 7 passes their results are statistically indistinguishable from that obtained by using an exact eigensolver (labeled 'PCA' in Figure 10, left). KHA/t, by contrast, needs about 30 passes through the data to reach a similar level of performance.

The excess relative reconstruction errors—for spectral clustering, of the matrix $\boldsymbol{P}$—plotted in Figure 10 (right) confirm that our methods outperform KHA/t. They also show KHA/et* significantly outperforming KHA/et, by about an order of magnitude. Again SMD is able to substantially accelerate both KHA/et and KHA/et*. As usual the improvement is larger for the former, though in this case not by quite enough to close the performance gap to the latter.

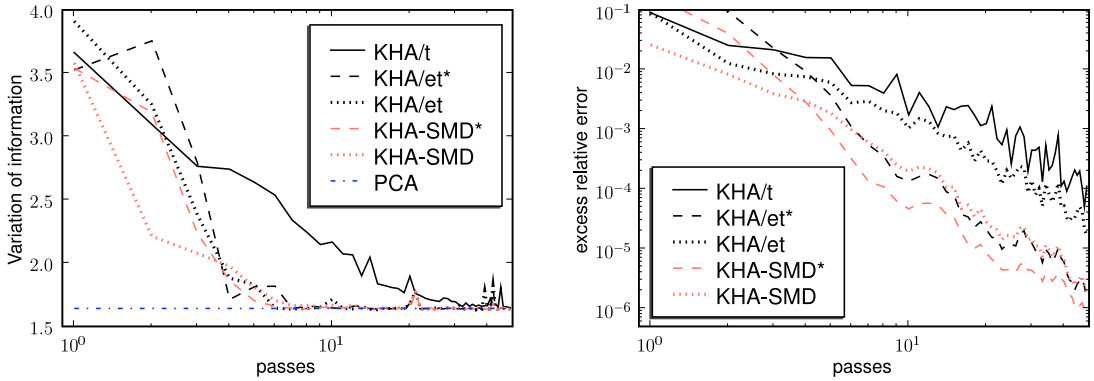


Figure 10: Quality of spectral clustering of the USPS data using an RBF kernel, as measured by variation of information (left) and excess relative reconstruction error (right). Horizontal 'PCA' line on the left marks the variation of information achieved by an exact eigensolver.

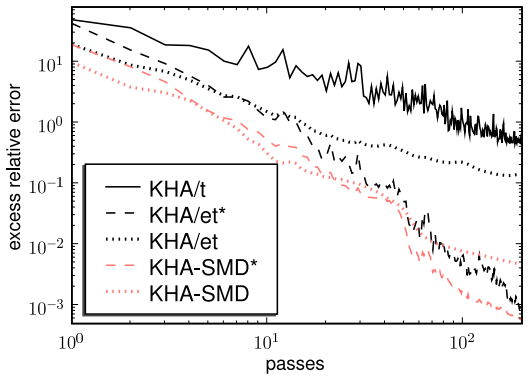


Figure 11: Excess relative reconstruction error on human motion capture data.

5.1.5 HUMAN MOTION DENOISING

For our next experiment we employ the KHA to denoise a human walking motion trajectory from the CMU motion capture database (`http://mocap.cs.cmu.edu`), converted to Cartesian coordinates via Neil Lawrence's matlab motion capture toolbox (`http://www.dcs.shef.ac.uk/~neil/mocap/`). The experimental setup is similar to that of Tangkuampien and Suter (2006): First zero-mean Gaussian noise is added to the frames of the original motion, then KHA using 25 principal components is used to denoise them. The noise is applied in "delta pose space," where each body part is represented by the normalized vector from its start to its end point, with a variance of 2 degrees for each of the two vector angles. The walking motion we consider has 343 frames, each represented by a 90-dimensional vector specifying the spatial orientation of 30 body parts. The

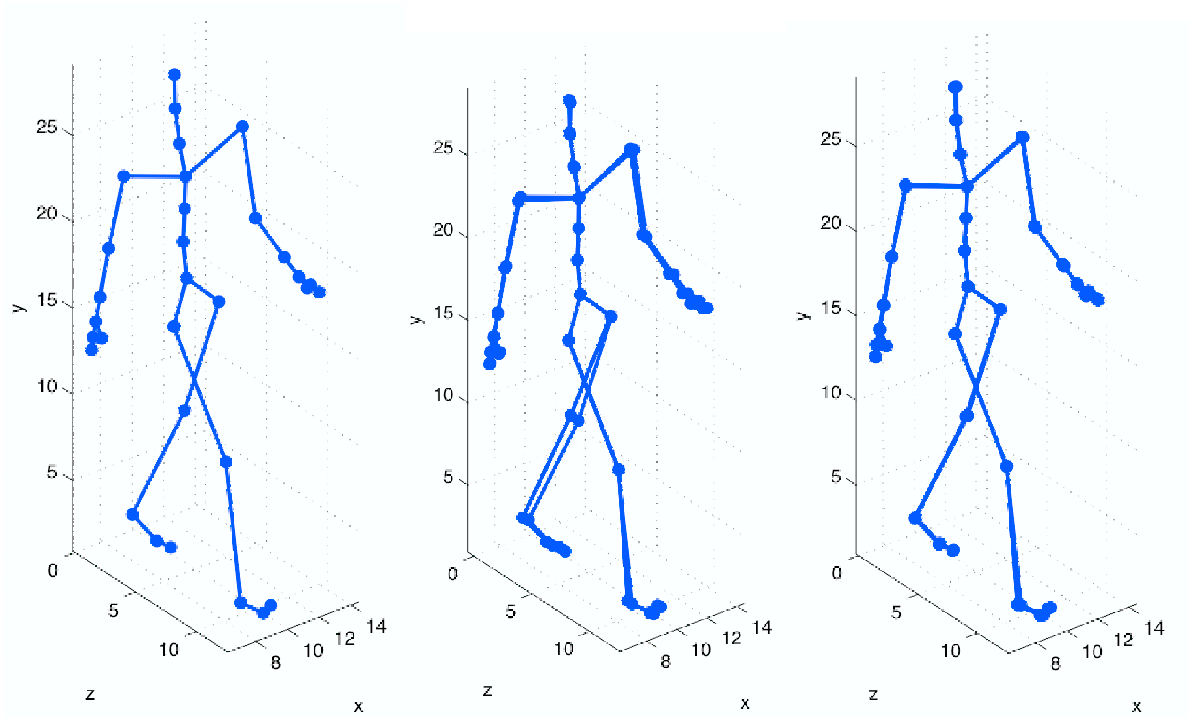


Figure 12: Reconstruction of human motion capture data: One frame of the original data (left), a superposition of this original and the noisy data (center), and a superposition of the original and reconstructed (i.e., denoised) data (right).

motion is reconstructed in $\mathbb{R}^3$ via the KHA with an RBF kernel ($\sigma = \sqrt{1.5}$); the resulting excess relative error is shown for various KHA variants in Figure 11.

As in the previous experiment, KHA/et* clearly outperforms KHA/et which in turn is better than KHA/t. Again SMD is able to improve KHA/et to a much larger extent than KHA/et*, though not enough to surpass the latter. KHA/et* reduces the noise variance by 87.5%; it is hard to visually detect any difference between the denoised frames and the original ones—see Figure 12 for an example.

5.2 Experiments on MNIST Data Set

The MNIST data set (LeCun, 1998) consists of 60000 handwritten digits, each 28×28 pixels in size. While kernel PCA has previously been applied to subsets of this data, to the best of our knowledge nobody has attempted it on the entire data set—for obvious reasons: the full kernel matrix has $3.6 \cdot 10^9$ entries, requiring over 7 GB of storage in single-precision floating-point format. Storing this matrix in main memory is already a challenge, let alone computing its eigenvalues; it thus makes sense to resort to iterative schemes.

We will perform a single pass through the MNIST data, attempting to find the first 50 eigenvalues of the centered kernel matrix. Since we run through the data just once, we will update the estimated eigenvalues after each iteration rather than after every pass. Hitherto we have used the

excess reconstruction error relative to the optimal reconstruction error to measure the performance of the KHA. For MNIST this is no longer possible since existing eigensolvers cannot handle such a large matrix. Instead we simply report the reconstruction error (38), which we can still compute—albeit with a rather high time complexity, as it requires calculating all entries of the kernel matrix.

Since our algorithms are fairly robust with respect to the value of τ, we simply set $\tau = 0.05l$ *a priori*, which corresponds to decreasing the gain by a factor of 20 during the first (and only) pass through the data. In our previous experiments we observed that the best values of η_0 and μ were usually the largest ones for which the run did not diverge. We also found that when divergence occurs, it tends to do so early and dramatically, making this event simple and inexpensive to detect. Algorithm 2 exploits this to automatically tune a gain parameter (η_0 *resp.* μ):

Algorithm 2 Auto-tune gain parameter x for KHA (any variant)

1. Compute (Algorithm 1, Step 1) and save initial KHA state;
2. $x := 500$;
3. **While** $\forall i, j$: is_finite($[\boldsymbol{A}_t]_{ij}$):
   Run KHA (Algorithm 1, Step 2) for 100 iterations;
4. $x := \max_{a,b} a \cdot 10^b : a \in \{1,2,5\}, b \in \mathbb{Z}, a \cdot 10^b < x$;
5. restore initial KHA state and **Goto** Step 3.

Algorithm 2 starts with a parameter value so large (here: 500) as to surely cause divergence (Step 2). It then runs the KHA (any variant) while testing the coefficient matrix $\boldsymbol{A}_t$ every 100 iterations for signs of divergence (Step 3). If any element of $\boldsymbol{A}_t$ becomes infinite or NaN ("not a number"), the KHA has diverged; in this case the parameter value is lowered (Step 4) and the KHA restarted (Step 5). In order to make these restarts efficient, we have precomputed and saved in Step 1 the initial state of the KHA—namely a row of $\boldsymbol{MK}$, an element of $\boldsymbol{MKM}$, the initial coefficient matrix $\boldsymbol{A}_1$, and $\boldsymbol{A}_1\boldsymbol{K}'$. Once the parameter value is low enough to avoid divergence, Algorithm 2 runs the KHA to completion in Step 3.

We use Algorithm 2 to tune η_0 for KHA/et and KHA/et*, and μ for KHA-SMD and KHA-SMD*. For η_0 the SMD variants use the same value as their respective non-SMD analogues. In our experiments, divergence always occurred within the first 600 iterations (1% of the data), or not at all. It is therefore possible to tune both η_0 and μ for the SMD variants as follows: first run Algorithm 2 to tune η_0 (with $\mu = 0$) on a small fraction of the data, then run it a second time to tune μ (with the previously obtained value for η_0) on the entire data set.

Our experiments were performed on an AMD Athlon 2.4 GHz CPU with 2 GB main memory and 512 kB cache, using a Python interface to PETSc (`http://www-unix.mcs.anl.gov/petsc/petsc-as/`). For a fair comparison, all our algorithms use the same initial random matrix $\boldsymbol{A}_1$, whose absolute reconstruction error is 33417. The reconstruction error after one pass through the data is shown in Table 2; it is evident that all our algorithms significantly improve upon the performance of KHA/t, with the SMD variants slightly ahead of their non-SMD analogues.

algorithm	parameter	rec. error	tuning	KHA time	total time
KHA/t	$\eta_0 = 5$	508.42	20'	33h 29'	57h 17'
KHA/et*	$\eta_0 = 50$	363.09	13'	41h 41'	65h 22'
KHA-SMD*	$\mu = 1$	362.44	1h 9'	53h 19'	77h 57'
KHA/et	$\eta_0 = 0.5$	415.48	47'	39h 26'	63h 42'
KHA-SMD	$\mu = 0.05$	404.06	3h 59'	64h 39'	92h 07'

Table 2: Tuned parameter values (col. 2), reconstruction errors (col. 3), and runtimes for various KHA variants on the MNIST data set. The total runtime (col. 6) is the sum of the times required to: center the kernel (11h 13'), tune the parameter (col. 4), run the KHA (col. 5), and calculate the reconstruction error (12h 16').

Table 2 also reports the time spent in parameter tuning, the resulting tuned parameter values, the time needed by each KHA variant for one pass through the data, and the total runtime (comprising kernel centering, parameter tuning, KHA proper, and computing the reconstruction error). Our KHA variants incur an overhead of 10–60% over the total runtime of KHA/t; the SMD variants are the more expensive. In all cases less than 5% of the total runtime was spent on parameter tuning.

6. Discussion and Conclusion

We modified the kernel Hebbian algorithm (KHA) of Kim et al. (2005) by providing a separate gain for each eigenvector estimate, and presented two methods, KHA/et* and KHA/et, which set those gains inversely proportional to the current estimate of the eigenvalues. KHA/et has a normalization term which allowed us to eliminate one of the free parameters of the gain decay scheme. Both methods were then enhanced by applying stochastic meta-descent (SMD) to perform gain adaptation in RKHS.

We compared our algorithms to the conventional approach of using KHA with constant gain, *resp*. with a scheduled gain decay (KHA/t), in seven different experimental settings. All our methods clearly outperformed the conventional approach in all our experiments. KHA/et* was superior to KHA/et, at the cost of having an additional free parameter τ. Its parameters, however, proved particularly easy to tune, with $\eta_0 = 5$ and $\tau = 3l$ or $4l$ optimal in all but the spectral clustering and MNIST experiments. This suggests that KHA/et* has good normalization properties and may well be preferable to KHA/et.

SMD improved the performance of both KHA/et and KHA/et*, where the improvements for the former were often larger than for the latter. This is not surprising *per se*, as it is naturally easier to improve upon a good algorithm than an excellent one. However, the fact that KHA-SMD frequently outperformed KHA-SMD* indicates that the interaction between KHA/et and SMD appears to be more effective.

Principal component analysis (PCA) is an important tool for analysis, preprocessing, and modeling of empirical data in a Euclidean space. Like other kernel methods, kernel PCA (Schölkopf et al., 1998) generalizes this to arbitrary RKHS, including those defined on structured data. Traditionally, kernel methods require computation and storage of the entire kernel matrix. As the data sets available for learning grow larger and larger, this is rapidly becoming infeasible. Recent advances eliminate this requirement by repeatedly cycling through the data set, computing kernels on demand

(e.g., Platt, 1999; Joachims, 1999; Zanni et al., 2006). This is done for kernel PCA by the KHA (Kim et al., 2005), which as originally introduced suffers from slow convergence. The acceleration techniques we have introduced here rectify this situation, and hence open the way for kernel PCA to be applied to large data sets.

Acknowledgments

We would like to thank the anonymous reviewers for their helpful comments. A short version of this paper was presented at the 2006 NIPS conference (Schraudolph et al., 2007). National ICT Australia is funded by the Australian Government's Department of Communications, Information Technology and the Arts and the Australian Research Council through Backing Australia's Ability and the ICT Center of Excellence program. This work is supported by the IST Program of the European Community, under the Pascal Network of Excellence, IST-2002-506778. Finally, we would like to acknowledge Equations (8), (10), (11), (12), (15), (21), (27), (28), (39), (40), (41), (42), (43), and (44) here, so that they are numbered.

References

Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, Cambridge, England, 2004.

Liang-Hwe Chen and Shyang Chang. An adaptive learning algorithm for principal component analysis. *IEEE Transaction on Neural Networks*, 6(5):1255–1263, 1995.

Christian Darken and John E. Moody. Towards faster stochastic gradient search. In John E. Moody, Stephen J. Hanson, and Richard Lippmann, editors, *Advances in Neural Information Processing Systems*, volume 4, pages 1009–1016. Morgan Kaufmann Publishers, 1992.

Athinodoros S. Georghiades, Peter N. Belhumeur, and David 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. ISSN 0162-8828. doi: http://doi.ieeecomputersociety.org/10.1109/34.927464.

Andreas Griewank. *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation*. Frontiers in Applied Mathematics. SIAM, Philadelphia, 2000.

Thorsten Joachims. Making large-scale SVM learning practical. In Bernhard Schölkopf, Chris J. C. Burges, and Alex J. Smola, editors, *Advances in Kernel Methods — Support Vector Learning*, pages 169–184, Cambridge, MA, 1999. MIT Press.

Juha Karhunen. Optimization criteria and nonlinear PCA neural networks. In *IEEE World Congress on Computational Intelligence*, volume 2, pages 1241–1246, 1994.

Juha Karhunen and Jyrki Joutsensalo. Representation and separation of signals using nonlinear PCA type learning. *Neural Networks*, 7(1):113–127, 1994.

Kwang In Kim, Matthias O. Franz, and Bernhard Schölkopf. Iterative kernel principal component analysis for image modeling. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 27(9): 1351–1366, 2005.

Yann LeCun. MNIST handwritten digit database, 1998. URL `http://www.research.att.com/~yann/ocr/mnist/`.

Yann LeCun, Bernhard E. Boser, John S. Denker, Donnie Henderson, R. E. Howard, Wayne E. Hubbard, and Lawrence D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1:541–551, 1989.

Marina Meila. Comparing clusterings: An axiomatic view. In *Proc. 22nd Intl. Conf. Machine Learning (ICML)*, pages 577–584, New York, NY, USA, 2005. ACM Press.

Sebastian Mika, Bernhard Schölkopf, Alex J. Smola, Klaus-Robert Müller, Matthias Scholz, and Gunnar Rätsch. Kernel PCA and de-noising in feature spaces. In Michael S. Kearns, Sara A. Solla, and David A. Cohn, editors, *Advances in Neural Information Processing Systems*, volume 11, pages 536–542. MIT Press, 1999.

David C. Munson, Jr. A note on Lena. *IEEE Trans. Image Processing*, 5(1), 1996.

Andrew Y. Ng, Michael I. Jordan, and Yair Weiss. On spectral clustering: Analysis and an algorithm. In Thomas G. Dietterich, Suzanna Becker, and Zoubin Ghahramani, editors, *Advances in Neural Information Processing Systems*, volume 14, 2002.

Erkki Oja and Juha Karhunen. On stochastic approximation of the eigenvectors and eigenvalues of the expectation of a random matrix. *Journal of Mathematical Analysis and Applications*, 106(1): 69–84, February 1985.

Barak A. Pearlmutter. Fast exact multiplication by the Hessian. *Neural Computation*, 6(1):147–160, 1994.

John Platt. Fast training of support vector machines using sequential minimal optimization. In Bernhard Schölkopf, Chris J. C. Burges, and Alex J. Smola, editors, *Advances in Kernel Methods — Support Vector Learning*, pages 185–208, Cambridge, MA, 1999. MIT Press.

Herbert Robbins and Sutton Monro. A stochastic approximation method. *Annals of Mathematical Statistics*, 22:400–407, 1951.

Terrence D. Sanger. Optimal unsupervised learning in a single-layer linear feedforward network. *Neural Networks*, 2:459–473, 1989.

Bernhard Schölkopf and Alex J. Smola. *Learning with Kernels*. MIT Press, Cambridge, MA, 2002.

Bernhard Schölkopf, Alex J. Smola, and Klaus-Robert Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 10:1299–1319, 1998.

Nicol N. Schraudolph. Fast curvature matrix-vector products for second-order gradient descent. *Neural Computation*, 14(7):1723–1738, 2002.

Nicol N. Schraudolph. Local gain adaptation in stochastic gradient descent. In *Proc. Intl. Conf. Artificial Neural Networks*, pages 569–574, Edinburgh, Scotland, 1999. IEE, London.

Nicol N. Schraudolph, Simon Günter, and S. V. N. Vishwanathan. Fast iterative kernel PCA. In Bernhard Schölkopf, John Platt, and Thomas Hofmann, editors, *Advances in Neural Information Processing Systems*, volume 19, Cambridge MA, June 2007. MIT Press.

Therdsak Tangkuampien and David Suter. Human motion de-noising via greedy kernel principal component analysis filtering. In *Proc. Intl. Conf. Pattern Recognition*, 2006.

S. V. N. Vishwanathan, Nicol N. Schraudolph, and Alex J. Smola. Step size adaptation in reproducing kernel Hilbert space. *Journal of Machine Learning Research*, 7:1107–1133, June 2006.

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, July 2006.

Journal of Machine Learning Research 8 (2007) 1919-1986 Submitted 8/06; Published 8/07

A Generalized Maximum Entropy Approach to Bregman Co-clustering and Matrix Approximation

Arindam Banerjee BANERJEE@CS.UMN.EDU
Department of Computer Science and Engineering
University of Minnesota, Twin Cities
Minneapolis, MN, USA

Inderjit Dhillon INDERJIT@CS.UTEXAS.EDU
Department of Computer Sciences
University of Texas at Austin
Austin, TX, USA

Joydeep Ghosh GHOSH@ECE.UTEXAS.EDU
Department of Electrical and Computer Engineering
University of Texas at Austin
Austin, TX, USA

Srujana Merugu SRUJANA@YAHOO-INC.COM
Yahoo! Research
Santa Clara, CA, USA

Dharmendra S. Modha DMODHA@US.IBM.COM
IBM Almaden Research Center
San Jose, CA, USA

Editor: John Lafferty

Abstract

Co-clustering, or simultaneous clustering of rows and columns of a two-dimensional data matrix, is rapidly becoming a powerful data analysis technique. Co-clustering has enjoyed wide success in varied application domains such as text clustering, gene-microarray analysis, natural language processing and image, speech and video analysis. In this paper, we introduce a partitional co-clustering formulation that is driven by the search for a good matrix approximation—every co-clustering is associated with an approximation of the original data matrix and the quality of co-clustering is determined by the approximation error. We allow the approximation error to be measured using a large class of loss functions called Bregman divergences that include squared Euclidean distance and KL-divergence as special cases. In addition, we permit multiple structurally different co-clustering schemes that preserve various linear statistics of the original data matrix. To accomplish the above tasks, we introduce a new *minimum Bregman information* (MBI) principle that simultaneously generalizes the *maximum entropy* and *standard least squares* principles, and leads to a matrix approximation that is optimal among all generalized additive models in a certain natural parameter space. Analysis based on this principle yields an elegant meta algorithm, special cases of which include most previously known alternate minimization based clustering algorithms such as kmeans and co-clustering algorithms such as information theoretic (Dhillon et al., 2003b) and minimum sum-squared residue co-clustering (Cho et al., 2004). To demonstrate the generality and flexibility of our co-clustering framework, we provide examples and empirical evidence on a vari-

ety of problem domains and also describe novel co-clustering applications such as missing value prediction and compression of categorical data matrices.

Keywords: co-clustering, matrix approximation, Bregman divergences, Bregman information, maximum entropy

1. Introduction

Data naturally arises in the form of matrices in a multitude of machine learning and data mining applications. Often, the data matrices that arise in real-world applications contain a large number of rows and columns, and may be very sparse. Understanding the natural structure of such matrices is a fundamental problem.

Clustering is an unsupervised learning technique that has been often used to discover the "latent structure" of data matrices that describe a set of objects (rows) by their feature values (columns). Typically, a clustering algorithm strives to group "similar" objects (or rows). A large number of clustering algorithms such as kmeans, agglomerative clustering, and their variants have been thoroughly studied (Jain and Dubes, 1988; Ghosh, 2003). Often, clustering is preceded by a dimensionality reduction phase, such as feature selection where only a subset of the columns is retained. As an alternative to feature selection, one can cluster the columns, and then represent each resulting group of features by a single derived feature (Dhillon et al., 2003a).

A recent paper (Dhillon and Modha, 2001) dealing with the spherical kmeans algorithm for clustering large, sparse document-term matrices arising in text mining graphically demonstrates (see Figures 13, 31, and 32 in the paper by Dhillon and Modha, 2001) that document clustering naturally brings together similar words. Intuitively, documents are similar because they use similar words. A natural question is whether it is possible to mathematically capture this relationship between rows and columns. Furthermore, is it possible to exploit this relationship to a practical advantage? This paper shows that both these questions can be answered in the affirmative in the context of clustering.

Co-clustering, also called bi-clustering (Hartigan, 1972; Cheng and Church, 2000), is the problem of simultaneously clustering rows and columns of a data matrix. Unlike clustering which seeks similar rows or columns, co-clustering seeks "blocks" (or "co-clusters") of rows and columns that are inter-related. Co-clustering has recently received a lot of attention in several practical applications such as simultaneous clustering of documents and words in text mining (Dhillon et al., 2003b; Gao et al., 2005; Takamura and Matsumoto, 2003), genes and experimental conditions in bioinformatics (Cheng and Church, 2000; Cho et al., 2004; Kluger et al., 2003), tokens and contexts in natural language processing (Freitag, 2004; Rohwer and Freitag, 2004; Li and Abe, 1998), users and movies in recommender systems (George and Merugu, 2005), etc.

Co-clustering is desirable over traditional "single-sided" clustering from a number of perspectives:

1. Simultaneous grouping of row and column clusters is more informative and digestible. Co-clustering provides compressed representations that are easily interpretable while preserving most of the information contained in the original data, which makes it valuable to a large class of statistical data analysis applications.

2. A row (or column) clustering can be thought of as dimensionality reduction along the rows (or columns). Simultaneous clustering along rows and columns reduces dimensionality along both axes, thus leading to a statistical problem with dramatically smaller number of param-

eters and hence, a much more compact representation for subsequent analysis. Since co-clustering incorporates row clustering information into column clustering and vice versa, one can think of it as a "statistical regularization" technique that can yield better quality clusters even if one is primarily interested in a single-sided clustering. The statistical regularization effect of co-clustering is extremely important when dealing with large, sparse data matrices, for example, those arising in text mining. A similar intuition can be drawn from subspace clustering methods (Parsons et al., 2004), which only use a part of the full potential of the co-clustering methodology.

3. As the size of data matrices increases, so does the need for scalable clustering algorithms. Single-sided, geometric clustering algorithms such as kmeans and its variants have computation time proportional to mnk per iteration, where m is the number of rows, n is the number of columns and k is the number of row clusters. Co-clustering algorithms based on a similar iterative process, on the other hand, involve optimizing over a smaller number of parameters, and can relax this dependence to $O(mkl + nkl)$ where m, n and k are defined as before and l is the number of column clusters. Since the number of row and column clusters is usually much smaller than the original number of rows and columns, co-clustering can lead to substantial reduction in the running time (see, for example, Dhillon et al. 2003b and Rohwer and Freitag 2004).

In summary, co-clustering is an exciting paradigm for unsupervised data analysis in that it is more informative, has less parameters, is scalable, and is able to effectively intertwine row and column information.

In this paper, we concentrate on partitional co-clustering (also called checkerboard bi-clustering by Kluger et al., 2003) where all the rows and columns are partitioned into disjoint row and column clusters respectively. We provide a general framework for addressing this problem that considerably expands the scope and applicability of the co-clustering methodology. To appreciate this generalization, it is helpful to view partitional co-clustering as a lossy data compression problem where, given a specified number of rows and column clusters, one attempts to retain as much information as possible about the original data matrix in terms of statistics based on the co-clustering (Dhillon et al., 2003b). The main idea is that a reconstruction based on co-clustering should result in the same set of user-specified statistics as the original matrix. There are two key components in formulating a co-clustering problem: (i) choosing a set of critical co-clustering-based statistics of the original data matrix that need to be preserved, and (ii) selecting an appropriate measure to quantify the information loss or the discrepancy between the original data matrix and the compressed representation provided by the co-clustering. For example, in the work of Cheng and Church (2000), the row and column averages of each co-cluster are preserved and the discrepancy between the original and the compressed representation is measured in terms of the sum of element-wise squared deviation. In contrast, information-theoretic co-clustering (ITCC) (Dhillon et al., 2003b), which is applicable to data matrices representing joint probability distributions, preserves a different set of summary statistics, that is, the row and column averages and the co-cluster averages. Further, the quality of the compressed representation is measured in terms of the sum of element-wise I-divergence. In the next subsection, we take a closer look at ITCC to provide a concrete motivating example.

1.1 ITCC: A Motivating Example

Let X and Y be discrete random variables that take values in the sets $\{x_u\}$, $[u]_1^m$, and $\{y_v\}$, $[v]_1^n$, respectively, where $[u]_1^m$ denotes an index u running over $\{1,\cdots,m\}$. Information-theoretic co-clustering provides a principled approach for simultaneously clustering the rows and columns of the joint probability distribution $p(X,Y)$. In practice, the entries of this matrix may not be known and are, instead, estimated from a contingency table or co-occurrence matrix. Let the row clusters be denoted by $\{\hat{x}_g\}$, $[g]_1^k$ and the column clusters by $\{\hat{y}_h\}$, $[h]_1^l$. Let $\hat{X}$ and $\hat{Y}$ denote the clustered random variables induced by X and Y that range over the set of row and column clusters respectively. A natural goal is to choose a co-clustering that preserves the maximum amount of "information" in the original data. In particular, since the data corresponds to the joint probability distribution of random variables X and Y, it is natural to preserve the mutual information between X and Y, or, in other words, minimize the loss in mutual information due to the compression that results from co-clustering. Thus, a suitable formulation is to solve the problem:

$$\min_{\hat{X},\hat{Y}} \quad (I(X;Y) - I(\hat{X};\hat{Y})) \ , \tag{1}$$

where $I(X;Y)$ is the mutual information between X and Y (Cover and Thomas, 1991). Dhillon et al. (2003b) showed that

$$I(X;Y) - I(\hat{X},\hat{Y}) = KL(p(X,Y)||q(X,Y)) \ , \tag{2}$$

where $q(X,Y)$ is a distribution of the form

$$q(X,Y) = p(\hat{X},\hat{Y})p(X|\hat{X})p(Y|\hat{Y}) \ , \tag{3}$$

and $KL(\cdot||\cdot)$ denotes the Kullback-Leibler(KL) divergence, also known as relative entropy. Thus, the search for the optimal co-clustering may be conducted by searching for the nearest approximation $q(X,Y)$ that has the above form. Since $p(X)$, $p(Y)$ and $p(\hat{X},\hat{Y})$ are determined by $m-1$, $n-1$ and $kl-1$ parameters respectively, with $k+l$ dependencies due to $p(\hat{X})$ and $p(\hat{Y})$, for a given co-clustering the distribution $q(X,Y)$ depends only on $(kl+m+n-k-l-3)$ independent parameters, which is much smaller than the $mn-1$ parameters that determine a general joint distribution. Hence, $q(X,Y)$ is a "low-complexity" or low-parameter matrix approximation of $p(X,Y)$.

The above viewpoint was developed by Dhillon et al. (2003b). We now present an alternate viewpoint that will enable us to generalize our approach to arbitrary data matrices and general distortion measures. The following lemma highlights a key maximum entropy property that makes $q(X,Y)$ a "low-complexity" or low-parameter approximation.

Lemma 1 *Given a fixed co-clustering, consider the set of joint distributions p' that preserve the row, column and co-cluster marginals of the input distribution p:*

$$\sum_{x\in\hat{x}}\sum_{y\in\hat{y}} p'(x,y) = p(\hat{x},\hat{y}) = \sum_{x\in\hat{x}}\sum_{y\in\hat{y}} p(x,y), \quad \forall \hat{x},\hat{y}, \tag{4}$$

$$p'(x) = p(x), \quad p'(y) = p(y), \quad \forall x,y. \tag{5}$$

Among all such distributions p', the distribution q given in (3) has the maximum entropy, that is, $H(q(X,Y)) \geq H(p'(X,Y))$.

A proof of the above lemma is presented in Appendix A. What is the significance of the above lemma? In the absence of any constraints, the uniform distribution, $p_0(X,Y) = \{\frac{1}{mn}\}$, has the maximum entropy. If only row and column marginals are to be preserved, that is, (5) holds, then the

product distribution $p(X)p(Y)$ has maximum entropy (see Cover and Thomas, 1991, Problem 5, Chapter 11). The above lemma states that among all distributions that preserve row, column, and co-cluster marginals, that is, (4) and (5) hold, the maximum entropy distribution has the form in (3). The maximum entropy characterization ensures that $q(X,Y)$ has a number of desirable properties. For instance, given the row, column and co-cluster marginals, it is the unique distribution that satisfies certain consistency criteria (Csiszár, 1991; Shore and Johnson, 1980). In Section 4, we also demonstrate that it is the optimal approximation to the original distribution p in terms of KL-divergence among all multiplicative combinations of the preserved marginals. It is important to note that the *maximum entropy characterization also implies that q is a low-complexity matrix approximation*.[1] In contrast, note that the input $p(X,Y)$ obviously satisfies the constraints in (4) and (5), but in general, is determined by $(mn-1)$ parameters and has lower entropy than q. Every co-clustering yields a unique maximum entropy distribution. Thus, by (2) and Lemma 1, the co-clustering problem (1) is equivalent to the problem of finding the nearest (in KL-divergence) maximum entropy distribution that preserves the row, column and co-cluster marginals of the original distribution. The maximum entropy property in Lemma 1 may be re-stated as $KL(q||p_0) \leq KL(p'||p_0)$, where p_0 is the uniform distribution. Thus, the maximum entropy principle is identical to the minimum relative entropy principle where the relative entropy is measured with respect to p_0.

The above formulation is applicable when the data matrix corresponds to an empirical joint distribution. However, there are important situations when the data matrix cannot be interpreted in this matter, for example the matrix may contain negative entries and/or a distortion measure other than KL-divergence, such as the squared Euclidean distance might be more appropriate.

1.2 Key Contributions

The contributions of this paper can be summarized as follows:

- We introduce a partitional co-clustering formulation driven by a matrix approximation viewpoint where the quality of co-clustering is characterized by the accuracy of an induced co-clustering-based matrix approximation, measured in terms of a suitable distortion measure. This formulation serves the dual purpose of (i) obtaining row and column clusterings that optimize a well-defined global objective function, and (ii) providing a new class of desirable matrix approximations.

- Our formulation is applicable to all Bregman divergences (Azoury and Warmuth, 2001; Banerjee et al., 2005b; Bregman, 1967; Censor and Zenios, 1998), which constitute a large class of distortion measures including the most commonly used ones such as squared Euclidean distance, KL-divergence, Itakura-Saito distance, etc. The generalization to Bregman divergences is useful due to a bijection between regular exponential families and a sub-class of Bregman divergences called regular Bregman divergences (Banerjee et al., 2005b). This bijection result enables us to choose the appropriate Bregman divergence based on the underlying data generation process or noise model. This, in turn, allows us to perform co-clustering on a wide variety of data matrices.

- Our formulation allows multiple co-clustering schemes wherein the reconstruction of the original matrix is based on different sets of linear summary statistics that one may be interested

1. The complexity here refers to the number of parameters required to construct a good approximation to the given matrix. It does not refer to the expected communication complexity, as is usual in the context of Shannon entropy.

in preserving. In particular, we focus on summary statistics that correspond to conditional expectations over partitions that result from the rows, columns and co-clusterings. We establish that there are exactly six non-trivial co-clustering schemes. Each of these schemes corresponds to a unique co-clustering basis, that is, combination of conditional expectations over various partitions. Using a formal abstraction, we explicitly enumerate and analyze the co-clustering problem for all the six bases. Existing partitional co-clustering algorithms (Cho et al., 2004; Dhillon et al., 2003b) can then be seen as special cases of the abstraction, employing one of the six co-clustering bases. Three of the six bases we discuss have not been used in the literature till date.

- Previous work on co-clustering assume that all the elements of the data matrix are equally important, that is, have uniform measure. In contrast, we associate a probability measure with the elements of the specified matrix and pose the co-clustering problem in terms of the random variable that takes values among the matrix elements following this measure. Our formulation based on random variables provides a natural mechanism for handling values with differing levels of uncertainty and in particular, missing values, while retaining both the analytical and algorithmic simplicity of the corresponding uniform-measure formulation.

- En route to formulating the Bregman co-clustering problem, we introduce the *minimum Bregman information* (MBI) principle that generalizes the well-known *maximum entropy* and *standard least-squares* principles to all Bregman loss functions. The co-clustering process is guided by the search for the matrix approximation that has the minimum Bregman information while preserving the specified co-clustering statistics.

- We provide an interpretation of the Bregman co-clustering problem in terms of minimizing the loss in Bregman information due to co-clustering, which enables us to generalize the viewpoint presented in information-theoretic co-clustering (Dhillon et al., 2003b).

- We develop an efficient meta co-clustering algorithm based on alternate minimization that is guaranteed to achieve (local) optimality for all Bregman divergences. Many previously known parametric clustering and co-clustering algorithms such as minimum sum-squared residue co-clustering (Cho et al., 2004) and information-theoretic co-clustering (Dhillon et al., 2003b) follow as special cases of our methodology.

- Lastly, we describe some novel applications of co-clustering such as predicting missing values and compression of categorical data matrices, and also provide empirical results comparing different co-clustering schemes for various application domains.

In summary, our results provide a sound theoretical framework for the analysis and design of efficient co-clustering algorithms for data approximation and compression, and considerably expand applicability of the co-clustering methodology.

1.3 Outline of the Paper and Notation

The rest of paper is organized as follows: We begin by reviewing preliminary definitions and describe the Bregman co-clustering problem at a conceptual level in Section 2. To present our co-clustering framework, we proceed as follows. First, we describe and analyze *block-average co-clustering* in Section 3, which is an important special case of our general formulation, in order

to provide intuition about the main results. Then, in Section 4, we enumerate various possible *co-clustering bases* corresponding to the summary statistics chosen to be preserved, and present a general formulation that is applicable to all these bases. In Section 5, we analyze the general Bregman co-clustering problem and propose a meta-algorithm that is applicable to all Bregman divergences and all co-clustering bases. In Appendix E, we describe how the Bregman co-clustering algorithm can be instantiated for various choices of Bregman divergence and co-clustering basis by providing the exact update steps. Readers interested in a purely computational recipe can jump to Apendix E. Empirical evidence on the benefits of co-clustering and preliminary experiments on novel co-clustering applications are presented in Section 6. We discuss related work in Section 7 and conclude in Section 8.

A brief word about the notation: Sets such as $\{x_1, \cdots, x_n\}$ are enumerated as $\{x_i\}_{i=1}^n$ and an index i running over the set $\{1, \cdots, n\}$ is denoted by $[i]_1^n$. Random variables are denoted using upper case letters, for example, Z. Matrices are denoted using upper case bold letters, for example, $\mathbf{Z}$, whereas the corresponding lower case letters z_{uv} denote the matrix elements. Transpose of a matrix $\mathbf{Z}$ is denoted by $\mathbf{Z}^T$. The effective domain of a function f is denoted by $\mathrm{dom}(f)$ and the inverse of a function f, when well defined, is denoted by $f^{(-1)}$. The relative interior and boundary of a set $\mathcal{S}$ are denoted by $\mathrm{ri}(\mathcal{S})$ and $\mathrm{bd}(\mathcal{S})$ respectively. Tables 15, 16 and 17 list the notation used in the paper.

2. Preliminaries

In this section, we discuss some important properties of Bregman divergences and also describe the basic setup of our co-clustering framework.

2.1 Bregman Divergences and Bregman Information

We start by defining Bregman divergences (Bregman, 1967; Censor and Zenios, 1998), which form a large class of well-behaved loss functions with a number of desirable properties.

Definition 1 Let ϕ be a real-valued convex function of Legendre type[2] (Rockafellar, 1970; Banerjee et al., 2005b) defined on the convex set $\mathcal{S} \equiv \mathrm{dom}(\phi)$ ($\subseteq \mathbb{R}^d$). The *Bregman divergence* $d_\phi : \mathcal{S} \times \mathrm{ri}(\mathcal{S}) \mapsto \mathbb{R}_+$ is defined as

$$d_\phi(z_1, z_2) = \phi(z_1) - \phi(z_2) - \langle z_1 - z_2, \nabla\phi(z_2)\rangle,$$

where $\nabla\phi$ is the gradient of ϕ.

Example 1.A (I-Divergence) Given $z \in \mathbb{R}_+$, let $\phi(z) = z\log z - z$. For $z_1, z_2 \in \mathbb{R}_+$, $d_\phi(z_1, z_2) = z_1 \log(z_1/z_2) - (z_1 - z_2)$.

Example 2.A (Squared Euclidean Distance) Given $z \in \mathbb{R}$, let $\phi(z) = z^2$. For $z_1, z_2 \in \mathbb{R}$, $d_\phi(z_1, z_2) = (z_1 - z_2)^2$.

Example 3.A (Itakura-Saito Distance) Given $z \in \mathbb{R}_+$, let $\phi(z) = -\log z$. For $z_1, z_2 \in \mathbb{R}_+$, $d_\phi(z_1, z_2) = \frac{z_1}{z_2} - \log\left(\frac{z_1}{z_2}\right) - 1$.

2. A proper, closed convex function ϕ is said to be of Legendre type if (i) $\mathrm{int}(\mathrm{dom}(\phi))$ is non-empty, (ii) ϕ is strictly convex and differentiable on $\mathrm{int}(\mathrm{dom}(\phi))$, and (iii) $\forall z_b \in \mathrm{bd}(\mathrm{dom}(\phi))$, $\lim_{z \to z_b} ||\nabla\phi(z)|| \to \infty$, where $z \in \mathrm{dom}(\phi)$.

Given a Bregman divergence and a random variable, the uncertainty in the random variable can be captured in terms of a useful concept called Bregman information (Banerjee et al., 2005b) defined below.

Definition 2 For any Bregman divergence $d_\phi : \mathcal{S} \times \text{int}(\mathcal{S}) \mapsto \mathbb{R}_+$ and any random variable $Z \sim w(z),\ z \in \mathcal{Z} \subseteq \mathcal{S}$, the *Bregman information* of Z is defined as the expected Bregman divergence to the expectation, that is,

$$I_\phi(Z) = E[d_\phi(Z, E[Z])] \ .$$

Intuitively, this quantity is a measure of the "spread" or the "information" in the random variable.

Example 1.B (I-Divergence) Given a random variable $Z \sim w(z),\ z \in \mathcal{Z} \subseteq \mathbb{R}_+$, the Bregman information corresponding to I-divergence is given by

$$I_\phi(Z) = E[Z \log (Z/E[Z]) - Z + E[Z]] = E[Z \log (Z/E[Z])] \ .$$

When w is the uniform measure and the support of Z (say $\mathcal{Z}$) consists of joint probability values of two other random variables X and Y, that is, $\mathcal{Z} = \{p(x_u, y_v),\ [u]_1^m, [v]_1^n\}$, then $E[Z] = \frac{1}{mn}$, that is, probability value corresponding to the uniform distribution $p_0(X,Y)$. The Bregman information in this case is given by

$$I_\phi(Z) = \frac{1}{mn} \sum_{u=1}^{m} \sum_{v=1}^{n} p(x_u, y_v) \log \left(\frac{p(x_u, y_v)}{p_0(x_u, y_v)} \right) = \frac{1}{mn} KL(p||p_0) = -\frac{1}{mn} H(p) + \text{constant},$$

where $H(\cdot)$ is the Shannon entropy.

Example 2.B (Squared Euclidean Distance) Given $Z \sim w(z),\ z \in \mathcal{Z} \subseteq \mathbb{R}$, the Bregman information corresponding to squared Euclidean distance is given by

$$I_\phi(Z) = E[Z - E[Z]]^2 \ ,$$

which is the variance of Z. When w is uniform and the support of Z, that is, $\mathcal{Z}$ consists of elements in a matrix $\mathbf{Z} \in \mathbb{R}^{m \times n}$, that is, $\mathcal{Z} = \{z_{uv},\ [u]_1^m, [v]_1^n\}$, then $E[Z] = \frac{1}{mn} \sum_{u=1}^{m} \sum_{v=1}^{n} z_{uv} \equiv \bar{z}$. The Bregman information in this case is given by

$$I_\phi(Z) = \frac{1}{mn} \sum_{u=1}^{m} \sum_{v=1}^{n} (z_{uv} - \bar{z})^2 = \frac{1}{mn} \sum_{u=1}^{m} \sum_{v=1}^{n} z_{uv}^2 - \bar{z}^2 = \frac{1}{mn} \|\mathbf{Z}\|_F^2 + \text{constant},$$

that is, a linear function of the squared Frobenius norm of $\mathbf{Z}$.

We note a useful property of Bregman information that will be extensively used in subsequent sections. The property, formally stated below, shows that the Bregman information exactly equals the difference between the two sides of Jensen's inequality (Cover and Thomas, 1991).

Lemma 2 *(Banerjee et al., 2005b) For any Bregman divergence $d_\phi : \mathcal{S} \times \text{ri}(\mathcal{S}) \mapsto \mathbb{R}_+$ and random variable $Z \sim w(z),\ z \in \mathcal{Z} \subseteq \mathcal{S}$, the Bregman information $I_\phi(Z) = E[d_\phi(Z, E[Z])] = E[\phi(Z)] - \phi(E[Z])$.*

Clearly, Bregman information is always non-negative. For a detailed list of other properties and examples of Bregman divergences and Bregman information, the reader is referred to Banerjee et al. (2005b) and Appendix B.

2.2 Data Matrix

We focus on the problem of co-clustering a specified $m \times n$ data matrix $\mathbf{Z}$. Let each entry of $\mathbf{Z} = [z_{uv}]$ take values in the convex set[3] $\mathcal{S} = \text{dom}(\phi)$, for example, $\mathcal{S} = \mathbb{R}$ for $\phi(z) = z^2$ and $\mathcal{S} = \mathbb{R}_+$ for $\phi(z) = z\log z - z$. Hence, $\mathbf{Z} \in \mathcal{S}^{m\times n}$. Observe that we are now admitting a much larger class of matrices than those used in the co-clustering formulations of Cho et al. (2004) and Dhillon et al. (2003b).

Given the data matrix $\mathbf{Z}$, we consider a random variable Z, that takes values in $\mathbf{Z}$ following a probability measure as described below. Let U be a random variable that takes values in $\{1,\cdots,m\}$, the set of row indices, and let V be a random variable that takes values in $\{1,\cdots,n\}$, the set of column indices. Let (U,V) be distributed according to a probability measure $w = \{w_{uv} : [u]_1^m, [v]_1^n\}$, which is either pre-specified or set to be the uniform distribution.[4] Let Z be a (U,V)-measurable random variable that takes values in $\mathbf{Z}$ following w, that is, $p(Z(u,v) = z_{uv}) = w_{uv}$. Clearly, for a given matrix $\mathbf{Z}$, the random variable Z is a deterministic function of the random variable (U,V). Throughout the paper, we assume the matrix $\mathbf{Z}$ and the measure w to be fixed so that taking conditional expectations of the random variable Z is well defined. In pure numeric terms, such conditional expectations are simply weighted row/column/block averages of the matrix $\mathbf{Z}$ according to the weights w. The stochastic formalization enables a succinct way to analyze such weighted averages.

Example 1.C (I-Divergence) Let $(X,Y) \sim p(X,Y)$ be jointly distributed random variables with X and Y taking values in $\{x_u\}, [u]_1^m$ and $\{y_v\}, [v]_1^n$ respectively. Then, $p(X,Y)$ can be written in the form of the matrix $\mathbf{Z} = [z_{uv}], [u]_1^m, [v]_1^n$, where $z_{uv} = p(x_u, y_v)$ is a deterministic function of u and v. This example with a uniform measure w corresponds to the setting described in Section 2, Example 1.B (originally in the work of Dhillon et al., 2003b).

Example 2.C (Squared Euclidean Distance) Let $\mathbf{Z} \in \mathbb{R}^{m\times n}$ denote a data matrix whose elements may assume positive, negative, or zero values and let w be a uniform measure. This example corresponds to the co-clustering setting described by Cheng and Church (2000) and Cho et al. (2004).

2.3 Bregman Co-clustering

We define a $k \times l$ partitional co-clustering as a pair of functions:

$$\rho : \{1,\cdots,m\} \mapsto \{1,\cdots,k\} ,$$
$$\gamma : \{1,\cdots,n\} \mapsto \{1,\cdots,l\} .$$

Let $\hat{U}$ and $\hat{V}$ be random variables that take values in $\{1,\cdots,k\}$ and $\{1,\cdots,l\}$ such that $\hat{U} = \rho(U)$ and $\hat{V} = \gamma(V)$. Let $\hat{\mathbf{Z}} = [\hat{z}_{uv}] \in \mathcal{S}^{m\times n}$ be an approximation for the data matrix $\mathbf{Z}$ such that $\hat{\mathbf{Z}}$ *depends only upon* a given co-clustering (ρ,γ) and certain summary statistics derived from the co-clustering. Let $\hat{Z}$ be a (U,V)-measurable random variable that takes values in this approximate matrix $\hat{\mathbf{Z}}$ following

3. $\mathcal{S}$ need not necessarily be a subset of $\mathbb{R}$. It is convenient to assume this for ease of exposition. In general, the elements of the matrix $\mathbf{Z}$ can take values over any convex domain with a well-defined Bregman divergence. We give examples of such settings in Section 6.
4. Associating a measure with the elements of a matrix is not common, but this construct allows us to deal with a wider variety of situations including the modeling of matrices with missing values. Further, several quantities of interest, such as row/column/block averages, can now be succinctly described in terms of conditional expectations.

w, that is, $p(\hat{Z}(U,V) = \hat{z}_{uv}) = w_{uv}$. Then the goodness of the underlying co-clustering can be measured in terms of the expected distortion between Z and $\hat{Z}$, that is,

$$E[d_\phi(Z,\hat{Z})] = \sum_{u=1}^{m}\sum_{v=1}^{n} w_{uv} d_\phi(z_{uv},\hat{z}_{uv}) = d_{\Phi_w}(\mathbf{Z},\hat{\mathbf{Z}}), \tag{6}$$

where $\Phi_w : \mathcal{S}^{m\times n} \mapsto \mathbb{R}$ is a separable convex function induced on the matrices such that the Bregman divergence between any pair of matrices is the weighted sum of the element-wise Bregman divergences corresponding to the convex function ϕ. From the matrix approximation viewpoint, the above quantity is simply the weighted element-wise distortion between the given matrix $\mathbf{Z}$ and the approximation $\hat{\mathbf{Z}}$. The co-clustering problem is then to find (ρ,γ) such that (6) is minimized. To carry out this plan, we need to make precise the connection between (ρ,γ) and $\hat{Z}$.

Example 1.D (I-Divergence) The Bregman co-clustering objective function (6) in this case is given by $E[d_\phi(Z,\hat{Z})] = E[Z\log(Z/\hat{Z}) - Z + \hat{Z}]$.

Example 2.D (Squared Euclidean Distance) The Bregman co-clustering objective function (6) in this case is given by $E[d_\phi(Z,\hat{Z})] = E[(Z-\hat{Z})^2]$.

The goodness of a co-clustering (ρ,γ) is determined by how well $\hat{Z}$ (or the matrix $\hat{\mathbf{Z}}$) approximates Z (or the matrix $\mathbf{Z}$). The crucial thing to note is that the construction of the approximation $\hat{Z}$ is based on the co-clustering (ρ,γ) and certain summary statistics of the original random variable Z that one wants to preserve in the approximation. The summary statistics may be properties of the co-clusters themselves, such as co-cluster marginals as in (4), and/or some other important statistics of the data, such as row and column marginals as in (5). Note that Z is not accessible while constructing $\hat{Z}$, since otherwise one could just set $\hat{Z} = Z$ and get perfect reconstruction. The special case when $\hat{Z}$ is constructed only using the co-clustering (ρ,γ) and the co-cluster marginals is important and easy to understand. Moreover, it is a straightforward generalization of one-sided clustering schemes such as kmeans. Hence, we first investigate this special case in detail in the next section. The general case, where additional summary information such as row/column marginals of the original matrix are available, will be analyzed in Sections 4 and 5.

3. Block Average Co-clustering: A Special Case

In this section, we discuss the important special case of Bregman co-clustering where the summary statistics are derived by aggregating along the co-clusters, that is, the summary statistics preserved are just the co-cluster means. Hence, in this case, for a given co-clustering (ρ,γ), $\hat{Z}$ has to be reconstructed based only on the co-cluster means, or equivalently, the conditional expectation random variable $E[Z|\hat{U},\hat{V}]$ where expectation is taken with respect to the measure w.[5] The quality of the co-clustering (ρ,γ) is determined by the approximation error between Z and $\hat{Z}$.

3.1 Minimum Bregman Information (MBI) Principle

In order to analyze the block co-clustering problem, we first focus on characterizing the approximation random variable $\hat{Z}$ given a fixed co-clustering (ρ,γ) and the resulting co-cluster means

5. Unless otherwise mentioned, the expectations in the rest of the paper are with respect to the probability measure w.

$\{E[Z|\hat{u},\hat{v}]\}$. While there can be many different ways to get an approximation $\hat{Z}$ from the available information, we consider a principled characterization based on the Bregman information of the reconstruction $\hat{Z}$. In particular, we propose and use the *minimum Bregman information principle* that can be shown to be a direct generalization of the maximum entropy as well as the least squares principles.

In order to get the "best" approximation, we consider a special class of approximating random variables Z' based on the given co-clustering and the available information $E[Z|\hat{U},\hat{V}]$. Let $\mathcal{S}_A$ be defined as

$$\mathcal{S}_A = \{Z' | E[Z'|\hat{u},\hat{v}] = E[Z|\hat{u},\hat{v}],\ \forall [\hat{u}]_1^k, [\hat{v}]_1^l\ \}\ . \tag{7}$$

It is reasonable to search for the best approximation in $\mathcal{S}_A$ since any random variable Z' in this class has the same co-cluster statistics as the original random variable Z. In other words, the corresponding reconstructed matrices preserve the co-cluster statistics of the original matrix, which is desirable. Then, with respect to the set $\mathcal{S}_A$, we ask: What is the "best" random variable to select from this set? We propose a new *minimum Bregman information principle* that recommends selecting a random variable that has the minimum Bregman information subject to the linear constraints (7):

$$\hat{Z}_A\ \equiv\ \hat{Z}_A(\rho,\gamma) = \underset{Z' \in \mathcal{S}_A}{\operatorname{argmin}}\ I_\phi(Z'). \tag{8}$$

The basic philosophy behind the minimum Bregman information principle is that the "best" approximation given certain information is one that does not make any extra assumptions over the available information. Mathematically, the notion of *no extra assumptions* or maximal uncertainty translates to *minimum Bregman information* while the available information is provided by the linear constraints that preserve the specified statistics.

As the following examples show, the widely used *maximum entropy principle* (Jaynes, 1957; Cover and Thomas, 1991) and *standard least squares principles* (Csiszár, 1991) can be obtained as special cases of the MBI principle.

Example 1.E From Example 1.B, we observe that the Bregman information of a random variable Z following a uniform distribution over the joint probability values of two other random variables X and Y is given by $-\frac{1}{mn}H(p(X,Y))$ upto an additive constant, that is, it is negatively related to entropy of the joint distribution of X and Y. Hence, minimizing the Bregman information is equivalent to maximizing the entropy demonstrating that the *maximum entropy principle* is a special case of the MBI principle corresponding to I-divergence.

Example 2.E From Example 2.B, we observe that the Bregman information of a random variable Z following a uniform distribution over the elements of a matrix $\mathbf{Z}$ is given by $\frac{1}{mn}\|\mathbf{Z}\|_F^2$ upto an additive constant. Hence, minimizing the Bregman information in this case is equivalent to minimizing the Frobenius norm of the matrix (L_2 norm for a vector), which in turn implies that the *standard least squares principle* is a special case of the MBI principle corresponding to squared error.

Now, we focus on getting a closed form solution of the minimum Bregman information problem. In the absence of any constraints, the minimum Bregman information solution corresponds to a constant random variable. For the current situation, where we are constrained to preserve the co-cluster means $\{E[Z|\hat{u},\hat{v}]\}$, the following theorem shows that the best approximation $\hat{Z}_A$ simply equals $E[Z|\hat{U},\hat{V}]$.

Theorem 1 *The solution to (8) is unique and is given by*

$$\hat{Z}_A = E[Z|\hat{U},\hat{V}].$$

Proof Let Z' be any random variable in $\mathcal{S}_A$ and let $\hat{Z}_A$ denote $E[Z|\hat{U},\hat{V}]$. By definition,

$$\begin{aligned}
I_\phi(Z') &\overset{(a)}{=} E[\phi(Z')] - \phi(E[Z']) \\
&= E[\phi(Z')] - E_{\hat{U},\hat{V}}[\phi(E[Z'|\hat{U},\hat{V}])] + E_{\hat{U},\hat{V}}[\phi(E[Z'|\hat{U},\hat{V}])] - \phi(E[Z']) \\
&\overset{(b)}{=} E[\phi(Z')] - E_{\hat{U},\hat{V}}[\phi(E[Z'|\hat{U},\hat{V}])] + E_{\hat{U},\hat{V}}[\phi(\hat{Z}_A)] - \phi(E_{\hat{U},\hat{V}}[\hat{Z}_A]) \\
&\overset{(c)}{=} E_{\hat{U},\hat{V}}\left[E_{Z'|\hat{U},\hat{V}}[\phi(Z')] - \phi(E[Z'|\hat{U},\hat{V}])\right] + I_\phi(\hat{Z}_A) \\
&\overset{(d)}{\geq} I_\phi(\hat{Z}_A),
\end{aligned}$$

where (a) and (c) follow from Lemma 2; (b) follows from the fact that $E[Z'|\hat{U},\hat{V}] = E[Z|\hat{U},\hat{V}] = \hat{Z}_A$ and $E_{\hat{U},\hat{V}}[E[Z|\hat{U},\hat{V}]] = E_{\hat{U},\hat{V}}[\hat{Z}_A] = E[Z] = E[Z']$; and (d) follows from conditional Jensen's inequality. In particular, since ϕ is convex, we have $E_{Z'|\hat{U},\hat{V}}[\phi(Z')] \geq \phi(E[Z'|\hat{U},\hat{V}])$.

Hence, $\hat{Z}_A$ has lower Bregman information than any random variable in $\mathcal{S}_A$. Further, $\hat{Z}_A \in \mathcal{S}_A$, that is, $E[\hat{Z}_A|\hat{U},\hat{V}] = \hat{Z}_A = E[Z|\hat{U},\hat{V}]$. Along with the strict convexity of ϕ, this ensures that $\hat{Z}_A = E[Z|\hat{U},\hat{V}]$ is the unique solution to (8). ■

For an alternative constructive proof of Theorem 1, please see Appendix C.

Besides being the MBI solution, $\hat{Z}_A$ has an additional important property that makes it the "best" reconstruction. Although we focused on the set $\mathcal{S}_A$ that contains all Z' that preserve the known co-cluster statistics, an alternative could have been to investigate the set $\mathcal{S}_B$ that contains all deterministic functions of the available information $E[Z|\hat{U},\hat{V}]$, that is,

$$\mathcal{S}_B = \{Z''|Z'' = f(E[Z|\hat{U},\hat{V}])\}\ , \tag{9}$$

where f is an arbitrary $(\hat{U},\hat{V})$-measurable function. In $\mathcal{S}_B$, the optimal approximation $\hat{Z}_B$ is the one that is closest to the true Z:

$$\hat{Z}_B \equiv \operatorname*{argmin}_{Z'' \in \mathcal{S}_B} E[d_\phi(Z,Z'')]\ . \tag{10}$$

In order to show a relationship between $\hat{Z}_A$ and $\hat{Z}_B$, we start with the following lemma (Lemma 3), which establishes the fact that the MBI solution $\hat{Z}_A$ allows a Pythagorean decomposition of the expected divergence between any $Z' \in \mathcal{S}_A$ and any $Z'' \in \mathcal{S}_B$.[6] Recall that $\mathcal{S}_A$ consists of all random variables that have the same co-cluster statistics as Z and $\mathcal{S}_B$ consists of all measurable functions of $E[Z|\hat{U},\hat{V}]$.

Lemma 3 *For any $Z' \in \mathcal{S}_A$ as in (7), any $Z'' \in \mathcal{S}_B$ as in (9), and $\hat{Z}_A$ as in (8),*

$$E[d_\phi(Z',Z'')] = E[d_\phi(Z',\hat{Z}_A)] + E[d_\phi(\hat{Z}_A,Z'')].$$

6. The analysis using Pythagorean decomposition of Bregman divergences can be viewed as a special case of Bregman duality analysis of Della Pietra et al. (2001). The advantage of our special case analysis is that it has rich semantics relevant to the co-clustering setting, and the proofs are simpler than the general case proofs in Della Pietra et al. (2001). See Section 4.4 for more details.

A proof of the lemma is presented in Appendix C. Now, since $\hat{Z}_A = E[Z|\hat{U},\hat{V}]$, and is hence a function of $E[Z|\hat{U},\hat{V}]$, we have $\hat{Z}_A \in \mathcal{S}_B$. As a result, from Lemma 3, we get the following projection theorem, which states that the MBI solution $\hat{Z}_A$ is the "forward" Bregman projection of any element of $\mathcal{S}_A$ onto the set $\mathcal{S}_B$ as well as the "backward" Bregman projection of any element of $\mathcal{S}_B$ onto the set $\mathcal{S}_A$.

Theorem 2 (Projection Theorem) *For any $Z' \in \mathcal{S}_A$ as in (7), any $Z'' \in \mathcal{S}_B$ as in (9), and $\hat{Z}_A$ as in (8), we have,*

(a) $\hat{Z}_A = \underset{Z' \in \mathcal{S}_A}{\operatorname{argmin}} E[d_\phi(Z', Z'')]$,

(b) $\hat{Z}_A = \underset{Z'' \in \mathcal{S}_B}{\operatorname{argmin}} E[d_\phi(Z', Z'')]$.

A proof of the theorem is presented in Appendix C. Since the original $Z \in \mathcal{S}_A$, we observe that $\hat{Z}_A$ is the best approximation (by a backward Bregman projection) to Z in $\mathcal{S}_B$, implying $\hat{Z}_B = \hat{Z}_A$ as formally stated below.

Corollary 1 *For $\hat{Z}_A$ and $\hat{Z}_B$ given by (8) and (10) respectively, we have*

$$\hat{Z} \equiv \hat{Z}_A = \hat{Z}_B. \tag{11}$$

The equivalence result is a precise mathematical quantification of the optimal approximation property of the MBI solution for the special case where only $E[Z|\hat{U},\hat{V}]$ is available during reconstruction. It shows that the best approximation in terms of expected Bregman divergence given the co-cluster statistics is indeed the MBI solution that preserves those statistics.

3.2 Co-clustering Problem Formulation

Now that we have associated an approximation $\hat{Z}$ with a given co-clustering (ρ,γ), we return to the original Bregman co-clustering problem in (6). The goal is to obtain a co-clustering (ρ,γ) such that the expected Bregman divergence between Z and the approximation $\hat{Z}$ is minimized. So far, we know that the best reconstruction $\hat{Z}$ is the MBI solution and is expressed in closed form by Theorem 1. The following lemma presents an alternative characterization of the co-clustering objective function (6). It shows that the expected Bregman divergence to the approximation $\hat{Z}$ is exactly equal to the loss in Bregman information due to co-clustering.

Lemma 4 *For any random variable Z and $\hat{Z}$ as in (11),*

$$E[d_\phi(Z,\hat{Z})] = I_\phi(Z) - I_\phi(\hat{Z}) \ .$$

Proof By definition,

$$\begin{aligned} E[d_\phi(Z,\hat{Z})] &= E[\phi(Z)-\phi(\hat{Z})-\langle Z-\hat{Z},\nabla\phi(\hat{Z})\rangle] \\ &\overset{(a)}{=} E[\phi(Z)]-E[\phi(\hat{Z})]-E_{\hat{U},\hat{V}}[\langle E[Z|\hat{U},\hat{V}]-E[\hat{Z}|\hat{U},\hat{V}],\nabla\phi(\hat{Z})\rangle] \\ &\overset{(b)}{=} E[\phi(Z)]-E[\phi(\hat{Z})] \\ &\overset{(c)}{=} E[\phi(Z)]-\phi(E[Z])-E[\phi(\hat{Z})]+\phi(E[\hat{Z}]) \\ &\overset{(d)}{=} I_\phi(Z)-I_\phi(\hat{Z}), \end{aligned}$$

where (a) follows from the fact that $\hat{Z}$ and hence, $\nabla\phi(\hat{Z})$ is constant for fixed $(\hat{U},\hat{V})$, (b) follows since $\hat{Z}\in\mathcal{S}_A$, (c) follows since $E[Z]=E[\hat{Z}]$ and (d) follows from Lemma 2. ■

Using Lemma 4, the original Bregman clustering problem in (6) can be posed as one of finding the optimal co-clustering (ρ^*,γ^*) defined as follows:

$$(\rho^*,\gamma^*) \;=\; \operatorname*{argmin}_{(\rho,\gamma)} E[d_\phi(Z,\hat{Z})] \;=\; \operatorname*{argmin}_{(\rho,\gamma)} [I_\phi(Z)-I_\phi(\hat{Z})] \;=\; \operatorname*{argmax}_{(\rho,\gamma)} I_\phi(\hat{Z})\;, \tag{12}$$

since $I_\phi(Z)$ is a constant. Further, using the fact that $\hat{Z}$ is the solution to the MBI problem, we have

$$(\rho^*,\gamma^*) \;=\; \operatorname*{argmax}_{(\rho,\gamma)} \min_{Z'\in\mathcal{S}_A} I_\phi(Z')\;. \tag{13}$$

Hence, the best co-clustering (ρ^*,γ^*) is the one that results in the matrix reconstruction corresponding to the minimum approximation error, or equivalently, the one that solves the max-min problem in (13).

3.3 Block Average Co-clustering Algorithm

In this section, we present an algorithm for block average co-clustering based on a useful decomposition of the objective function (12), which gives a better insight on how to update either the row clustering ρ or the column clustering γ.

3.3.1 A USEFUL DECOMPOSITION

From Theorem 1, it follows that for a given co-clustering (ρ,γ), the approximation $\hat{Z}$ that achieves the minimum Bregman information is given by $\hat{z}_{uv}=E[Z|\hat{u},\hat{v}]$, where $\hat{u}=\rho(u),\hat{v}=\gamma(v)$. We denote the co-cluster means corresponding to (ρ,γ) as $\mu_{\hat{u}\hat{v}}$, that is, $\mu_{\hat{u}\hat{v}}=E[Z|\hat{u},\hat{v}]$. Hence, the optimal approximation $\hat{Z}$ corresponding to (ρ,γ) is given by

$$\hat{z}_{uv}=\mu_{\hat{u}\hat{v}}=\mu_{\rho(u)\gamma(v)}.$$

With this closed form for $\hat{Z}$, we have

$$\begin{aligned} E[d_\phi(Z,\hat{Z})] &= \sum_{u,v} w_{uv}\, d_\phi(z_{uv},\mu_{\rho(u)\gamma(v)}) \\ &= \sum_{g=1}^{k}\sum_{h=1}^{l}\sum_{u:\rho(u)=g}\sum_{v:\gamma(v)=h} w_{uv}\, d_\phi(z_{uv},\mu_{gh})\;. \end{aligned} \tag{14}$$

Note that (14) decomposes the objective function in terms of the row cluster assignment $\rho(u)$ of each row u and column cluster assignment $\gamma(v)$ of each column v.

3.3.2 UPDATING ROW AND COLUMN CLUSTERS

Since the decomposition (14) is additive over all the rows (or columns), we can update the current cluster assignment of each row (or column) in order to decrease the objective function. For any particular row u, the contribution to the overall objective function is determined by its current assignment $\rho(u)$. Assuming $\rho(u) = g$, we can express the objective function (14) as the sum of row contributions of the form

$$J_u(g) = \sum_{h=1}^{l} \sum_{v:\gamma(v)=h} w_{uv} d_\phi(z_{uv}, \mu_{gh}) \ . \tag{15}$$

Note that the co-cluster means μ_{gh} remain unchanged during the update of the row (or column) clustering.

The choice of row cluster assignment g exactly determines what set of l co-cluster means μ_{gh} occur in (15). Hence, the best possible choice for the new row cluster assignment $\rho^{new}(u)$ is to pick the value of g that has the minimum cost, that is,

$$\rho^{new}(u) = \operatorname*{argmin}_g J_u(g) = \operatorname*{argmin}_g \sum_{h=1}^{l} \sum_{v:\gamma(v)=h} w_{uv} d_\phi(z_{uv}, \mu_{gh}) \ . \tag{16}$$

Since the terms corresponding to each row are additive in (14), the row assignment update in (16) can be applied simultaneously to all rows to get the new row assignments $\rho^{new}(u), [u]_1^n$. The new row assignments effectively change the current approximation matrix $\hat{Z}$ to a new matrix $\tilde{Z}_{\rho_1\gamma_0}$, which is just a row-permuted version of $\hat{Z}$ that achieves a lower cost, that is,

$$E[d_\phi(Z, \tilde{Z}_{\rho_1\gamma_0})] \leq E[d_\phi(Z, \hat{Z})] \ .$$

The decrease in the objective function value is due to the optimal greedy update in the row cluster assignments. A similar approach can be applied to update the column cluster assignments in order to obtain an even better approximation $\tilde{Z}_{\rho_1\gamma_1}$. Note that the current approximation can possibly be further improved by another round of row clustering updates to get an approximation $\tilde{Z}_{\rho_2\gamma_1}$, where the subscript in ρ (or γ) denotes the number of times the row (column) cluster assignment has been updated. The same process can be repeated multiple times. For simplicity, we denote the final assignments by $(\rho^{new}, \gamma^{new})$ and the approximation obtained from such reassignments as $\tilde{Z}$.

Once all row and column assignments have been updated, the new approximation matrix $\tilde{Z}$ need not be the minimum Bregman information solution for the new co-clustering $(\rho^{new}, \gamma^{new})$. Hence, one needs to recompute the new minimum Bregman solution $\hat{Z}^{new}$ corresponding to $(\rho^{new}, \gamma^{new})$. The following lemma, proved in Appendix C, establishes that the updated $\hat{Z}^{new}$ is guaranteed to either decrease the objective, or keep it unchanged. In fact, $\hat{Z}^{new}$ is the best approximation possible based on the co-clustering $(\rho^{new}, \gamma^{new})$.

Lemma 5 *Let $\hat{Z}^{new}$ be the minimum Bregman information solution corresponding to $(\rho^{new}, \gamma^{new})$. Then,*

$$E[d_\phi(Z, \hat{Z}^{new})] \leq E[d_\phi(Z, \tilde{Z})] \ .$$

Algorithm 1 Bregman Block Average Co-clustering (BBAC) Algorithm

Input: Matrix $\mathbf{Z} \subseteq \mathcal{S}^{m\times n}$, probability measure w, Bregman divergence $d_\phi : \mathcal{S} \times \mathrm{ri}(\mathcal{S}) \mapsto \mathbb{R}_+$, num. of row clusters l, num. of column clusters k.
Output: Block Co-clustering (ρ^*, γ^*) that (locally) optimizes the objective function in (12).
Method:

```
{Initialize ρ, γ }
Start with an arbitrary co-clustering (ρ, γ)
repeat
  {Step A: Update Co-cluster Means}
  for g = 1 to k do
    for h = 1 to l do
      μ_gh = (Σ_{u:ρ(u)=g} Σ_{v:γ(v)=h} w_uv z_uv) / (Σ_{u:ρ(u)=g} Σ_{v:γ(v)=h} w_uv)
    end for
  end for
  {Step B: Update Row Clusters (ρ)}
  for u = 1 to m do
    ρ(u) = argmin_{g∈{1,...,k}} Σ_{h=1}^{l} Σ_{v:γ(v)=h} w_uv d_φ(z_uv, μ_gh)
  end for
  {Step C: Update Column Clusters (γ)}
  for v = 1 to n do
    γ(v) = argmin_{h∈{1,...,l}} Σ_{g=1}^{k} Σ_{u:ρ(u)=g} w_uv d_φ(z_uv, μ_gh)
  end for
until convergence
return (ρ, γ)
```

3.3.3 THE ALGORITHM

The above analysis leads to a simple iterative algorithm for Bregman block average co-clustering (BBAC in Algorithm 1). The algorithm starts with an arbitrary choice of co-clustering (ρ, γ). At every iteration, either the row clustering ρ or the column clustering γ is updated in order to decrease the objective function value in (12). In practice, one could run multiple iterations of such updates. After the assignments have been updated for all rows and columns, the co-clustering means are updated, which further decreases the objective. The process is repeated till convergence. Since the objective decreases at every iteration, and the objective is lower bounded, the algorithm is guaranteed to converge to a (local) minimum of the objective.

3.4 Block Average Co-clustering as Matrix Factorization

Since the MBI solution is always the co-cluster means, and the BBAC algorithm essentially alternates between updating the row and column assignments, and updating the co-cluster means, the BBAC algorithm is a direct generalization of the Bregman clustering algorithm (Banerjee et al., 2005b). As we show below, the BBAC algorithm can also be viewed as solving a matrix factorization problem.

Let $\mathbf{Z}$ be the $m \times n$ matrix corresponding to the random variable Z and $\mathbf{W} \in \mathbb{R}_+^{m\times n}$ denote the matrix corresponding to a probability measure over the matrix elements. Let $\mathbf{R} \in \{0,1\}^{m\times k}$ and

$\mathbf{C} \in \{0,1\}^{n\times l}$ denote the row and column cluster membership matrices, that is,

$$r_{ug} = \begin{cases} 1 & g = \rho(u), \\ 0 & \text{otherwise}, \end{cases} \qquad c_{vh} = \begin{cases} 1 & h = \gamma(v), \\ 0 & \text{otherwise}. \end{cases}$$

Further, let $\mathbf{M}$ be a $k \times l$ matrix corresponding to the co-cluster means, that is, expectations or weighted averages of the matrix values over the co-clusters. Since the minimum Bregman information solution for the block co-clustering case are the co-cluster averages, the reconstructed matrix $\hat{\mathbf{Z}}$ can be expressed as the product $\mathbf{RMC}^T$. Therefore, the co-clustering problem is essentially reduces to finding row assignment matrix $\mathbf{R}$, column assignment matrix $\mathbf{C}$ such that the approximation error $d_{\Phi_w}(\mathbf{Z}, \hat{\mathbf{Z}})$ is minimized where $\hat{\mathbf{Z}} = \mathbf{RMC}^T$. The BBAC algorithm returns matrices $\mathbf{R}, \mathbf{M}$ and $\mathbf{C}$ that achieves a local minimum of the above objective function. When $l = n$, the BBAC algorithm reduces to the Bregman clustering algorithm (Banerjee et al., 2005b) applied to rows of $\mathbf{Z}$. In particular, when the Bregman divergence is the squared Euclidean distance, we obtain the classical kmeans algorithm.

3.5 General Formulation and Analysis: Warm Up

So far, we have studied in detail the important special case of block average co-clustering. In the next section, we will formulate and analyze a more general class of co-clustering problems.

The differences between the various formulations will stem from the different summary statistics used in the approximation $\hat{Z}$. For the block co-clustering case, $\hat{Z}$ depended only on the co-cluster means $\{E[Z|\hat{u},\hat{v}]\}$. In Section 4, we shall consider the exhaustive list of summary statistics based on which $\hat{Z}$ can be reconstructed, and go on to propose a general case meta-algorithm with provable properties in Section 5. The BBAC algorithm can then be seen as a special case of this meta-algorithm obtained for a particular choice of summary statistics.

Before going into the formulation and analysis of the general case, we want to highlight the results that are specific to block average co-clustering as well as the results that continue to hold in the general case for any choice of summary statistics. We start with the results that hold only for block average co-clustering and do not carry over to the general case.

1. For block average co-clustering, the MBI solution is the *same* for all Bregman divergences (Theorem 1). However, in the general case, the solution generally depends on the choice of the Bregman divergence. In fact, block average co-clustering is the only case when the solution is independent of this choice.

2. In the general case, it is not possible to get a closed form MBI solution. In general, a convex optimization problem has to be solved to find the solution; see Section 5.5 for some iterative approaches for computing the MBI solution. We also provide exact solutions for the important special cases where closed form solutions do exist.

3. For block co-clustering, the reconstruction from the minimum Bregman information solution is also the best approximation of the original Z among all functions of the co-cluster means (Corollary 1). This result holds only when the reconstruction is based on one set of summary statistics, which was the co-cluster means in the block co-clustering case. More formally, the result holds when the random variable is approximated based on a single sub-σ-algebra (see Section 4.1). In general, multiple sets of summary statistics may need to be preserved and the reconstruction will be based on multiple sub-σ-algebras.

4. The matrix approximation obtained in the general case need not be expressible as a matrix factorization in terms of the cluster membership matrices **R** and **C**. In fact, block average co-clustering is the only formulation where such an interpretation is possible for all Bregman divergences.

Finally, we focus on the results that continue to hold in the general case for arbitrary choices of summary statistics:

1. Although there need not be a closed form solution to the minimum Bregman information problem and the solution may depend on the choice of the Bregman divergence, some important properties of the solution remain unchanged in the general case. In particular, the form of the solution in terms of the Lagrange multipliers (see the constructive proof of Theorem 1 in Appendix C) remains unchanged.

2. The Pythagorean decomposition (Lemma 3) and the projection theorem (Theorem 2) associated with the sets $\mathcal{S}_A$ and $\mathcal{S}_B$ continues to hold for the general case, with $\mathcal{S}_B$ defined as the set of all generalized additive models of the various summary statistics in a transformed space (see Section 4.4). For the block average co-clustering case, since we only preserve the co-cluster means, the set $\mathcal{S}_B$ turns out to be set of all functions of the co-cluster means. Further, the MBI solution can be shown to be the best approximation to the original Z among this special class of functions of the summary statistics $\mathcal{S}_B$ generalizing the equivalence in Corollary 1. The general result that we discuss in Section 4.4 provides an axiomatic justification of the minimum Bregman information principle (Csiszár, 1991).

3. The loss in Bregman information result (Lemma 4) continues to hold.

4. Similar to Algorithm 1, we obtain an iterative algorithm for the general case where we alternately optimize over the row cluster assignments, column cluster assignments and the MBI solution. As in the block-average case, the co-clustering objective function allows an additive decomposition over the rows and columns and the resulting meta-algorithm (Algorithm 2) guarantees monotonic decrease of the objective function at every iteration.

4. Bregman Co-clustering: Formulation and Analysis

In this section, we formulate a general version of the Bregman co-clustering problem by abstracting out the commonalities between various possible co-clustering schemes that arise due to constraints that preserve different choices of summary statistics. To achieve this, we first define the notion of a co-clustering basis in terms of the conditional expectation-based statistics that one might want to preserve, and then enumerate all the possible co-clustering bases that may be of interest.

4.1 Co-clustering Bases

Let us fix a co-clustering (ρ, γ). Given the co-clustering, there are essentially four random variables of interest: U, V, $\hat{U}$, and $\hat{V}$. To these, we add two random variables $U_\emptyset$ and $V_\emptyset$ corresponding to the constant random variables over the rows and columns respectively, for easy enumeration. Let Γ_1 denote the set $\{U_\emptyset, V_\emptyset, \hat{U}, \hat{V}, U, V\}$. Our goal is to approximate the random variable Z using (possibly multiple) conditional expectations of Z where the conditioning is done on one or more of

the random variables in Γ_1. Observe that choosing one or more random variables to condition on is equivalent to choosing a sub-σ-algebra[7] $\mathcal{G}$ of Z. We focus on approximating Z using conditional expectations $E[Z|\mathcal{G}]$ since the conditional expectation $E[Z|\mathcal{G}]$ is the optimal approximation of the true Z with respect to any Bregman divergence among all $\mathcal{G}$-measurable functions (Banerjee et al., 2005a).

Since duplication of information in the preserved conditional expectations does not lead to a different approximation, we only focus on combinations of random variables from Γ_1 that will lead to a unique set of summary statistics. First, we observe that some of the random variables in Γ_1 are measurable with respect to some others. In other words, some random variables are just "high resolution" versions of some others so that conditioning on certain sets of members of Γ_1 is equivalent to conditioning on the subset with respect to which the rest are measurable. For example, $E[Z|U,\hat{U},V_\emptyset,\hat{V}] = E[Z|U,\hat{V}]$, since $\hat{U}$ is U-measurable, and $V_\emptyset$ is $\hat{V}$-measurable. In fact, due to the natural ordering of the random variables $\{U_\emptyset,\hat{U},U\}$ and $\{V_\emptyset,\hat{V},V\}$ in terms of measurability, only the row and column random variables of the highest granularity matter. Hence, there are only 9 unique sub-σ-algebras of Z based on which conditional expectations may be taken. We denote this set by Γ_2:

$$\Gamma_2 = \{\{U_\emptyset,V_\emptyset\},\{U_\emptyset,\hat{V}\},\{U_\emptyset,V\},\{\hat{U},V_\emptyset\},\{\hat{U},\hat{V}\},\{\hat{U},V\},\{U,V_\emptyset\},\{U,\hat{V}\},\{U,V\}\}.$$

Γ_2 determines the set of all summary statistics that one maybe interested in preserving. A particular choice of an element of Γ_2, such as $\{\hat{U},\hat{V}\}$, leads to an approximation scheme where the reconstruction matrix preserves the corresponding summary statistics. For the choice of $\{\hat{U},\hat{V}\}$, we get the block average co-clustering discussed in Section 3 where the matrix approximation preserves all co-cluster means.

Now, we focus on a much more general scenario where one may want to preserve possibly more than one summary statistic. In fact, one could consider all possible subsets of Γ_2. Of these, some combinations of summary statistics are effectively equivalent, for example, $\{\{\hat{U},V_\emptyset\},\{U_\emptyset,\hat{V}\},\{\hat{U},\hat{V}\}\}$ and $\{\{\hat{U},\hat{V}\}\}$, whereas some others are trivial and even independent of the co-clustering, for example, $\{\{U_\emptyset,V_\emptyset\}\}$ and $\{\{U_\emptyset,V\},\{U,V_\emptyset\}\}$. In this paper, we focus only on unique and non-trivial combinations of elements of Γ_2, that we call *co-clustering bases* and define them as follows:

Definition 3 [8] A *co-clustering basis* $\mathcal{C}$ is a set of elements of Γ_2, that is, an element of the power set 2^{Γ_2}, which satisfies the following two conditions:

(a) There exist $\mathcal{G}_1,\mathcal{G}_2 \in \mathcal{C}$ (with $\mathcal{G}_1$ possibly the same as $\mathcal{G}_2$) such that $\hat{U} \in \mathcal{G}_1$ and $\hat{V} \in \mathcal{G}_2$.

(b) There do not exist $\mathcal{G}_1,\mathcal{G}_2 \in \mathcal{C},\ \mathcal{G}_1 \neq \mathcal{G}_2$ such that $\mathcal{G}_2$ is a sub-σ-algebra of $\mathcal{G}_1$.

In the above definition, condition (a) ensures that the approximation depends on the co-clustering while condition (b) ensures that for any pair $\mathcal{G}_1,\mathcal{G}_2$, the conditional expectation $E[Z|\mathcal{G}_2]$ cannot be obtained from $E[Z|\mathcal{G}_1]$. The latter ensures that the approximation obtained using the basis $\mathcal{C}$ is not identical to that obtained using $\mathcal{C} \setminus \mathcal{G}_2$.

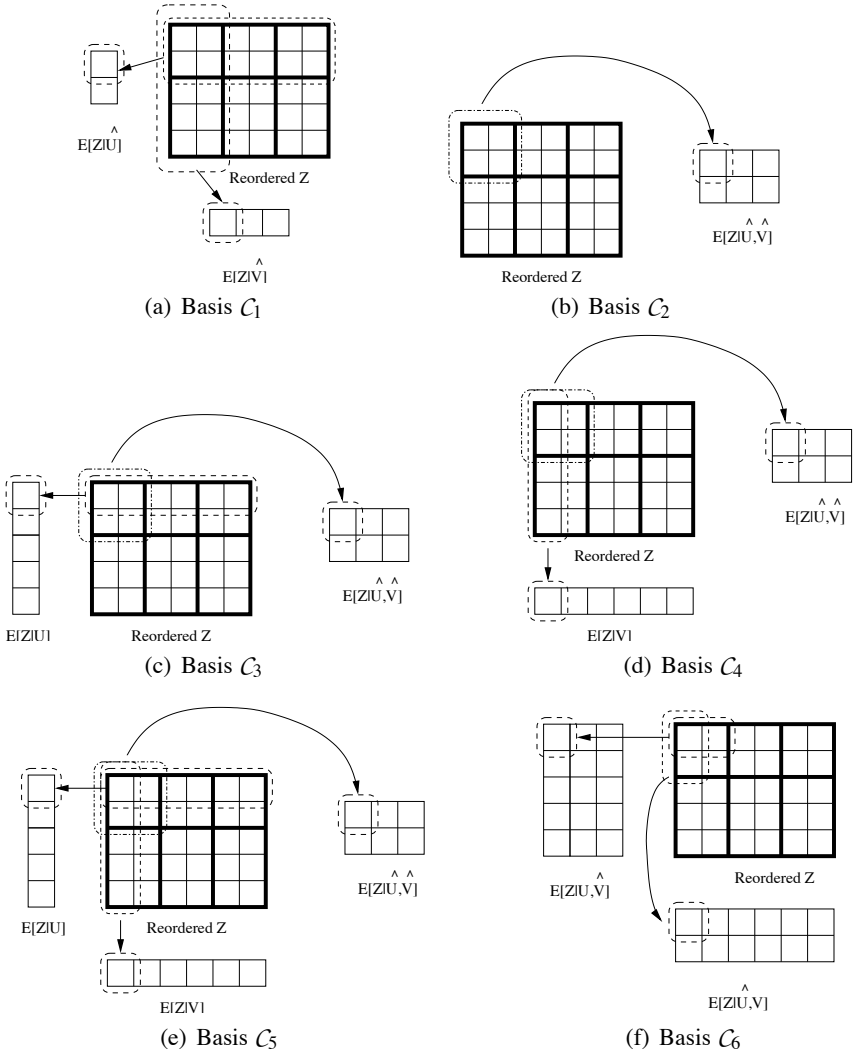
7. A σ-algebra is a collection of sets that includes the empty-set and is closed w.r.t. complements, countable unions and intersections. Further, $\mathcal{G}_1$ is a sub-σ-algebra of a σ-algebra $\mathcal{G}$ (or a $\mathcal{G}$-measurable random variable) if $\mathcal{G}_1$ is itself a σ-algebra and $\mathcal{G}_1 \subseteq \mathcal{G}$.

8. Note that each element of Γ_2 corresponds to a unique sub-σ-algebra of Z, and hence, we use identical notation for the elements of the co-clustering bases and the corresponding sub-σ-algebras.

The following theorem shows that there are only six possible co-clustering bases, each of which leads to a distinct matrix approximation scheme.

Theorem 3 *Given the random variable Z, there are only six distinct co-clustering bases that approximate Z using conditional expectations of Z given combinations of the row and column random variables $\{U, V, \hat{U}, \hat{V}\}$. The six bases correspond to the sets*

$$\begin{aligned} \mathcal{C}_1 &= \{\{\hat{U}\},\{\hat{V}\}\}, & \mathcal{C}_2 &= \{\{\hat{U},\hat{V}\}\}, \\ \mathcal{C}_3 &= \{\{\hat{U},\hat{V}\},\{U\}\}, & \mathcal{C}_4 &= \{\{\hat{U},\hat{V}\},\{V\}\}, \\ \mathcal{C}_5 &= \{\{\hat{U},\hat{V}\},\{U\},\{V\}\}, & \mathcal{C}_6 &= \{\{U,\hat{V}\},\{\hat{U},V\}\}. \end{aligned}$$



(a) Basis $\mathcal{C}_1$ (b) Basis $\mathcal{C}_2$

(c) Basis $\mathcal{C}_3$ (d) Basis $\mathcal{C}_4$

(e) Basis $\mathcal{C}_5$ (f) Basis $\mathcal{C}_6$

Figure 1: Schematic diagram of the six co-clustering bases. In each case, the summary statistics used for reconstruction (e.g., $E[Z|\hat{U}]$ and $E[Z|\hat{V}]$) are expectations taken over the corresponding dotted regions (e.g., over all the columns and all the rows in the row cluster determined by $\hat{U}$ in case of $E[Z|\hat{U}]$).

Figure 1 contains a graphical representation of the various co-clustering bases. For example, in Figure 1(a), the expectations along the row clusters ($E[Z|\hat{U}]$) and the column clusters ($E[Z|\hat{V}]$) are the statistics used for reconstructing the original Z. From the table, we can see that the various conditional expectations correspond to matrices of different sizes. We make use of this observation later in Appendix E to obtain a computational recipe for the Bregman co-clustering problem. The sets $\mathcal{C}_1, \mathcal{C}_2, \mathcal{C}_5$ and $\mathcal{C}_6$ are symmetric in the row and column random variables whereas $\mathcal{C}_3$ and $\mathcal{C}_4$ are not. Further, if we have access to $\{E[Z|\mathcal{G}] : \mathcal{G} \in \mathcal{C}_i\}$, for some $1 \leq i \leq 6$, then we can compute $\{E[Z|\mathcal{G}] : \mathcal{G} \in \mathcal{C}_j\}$ for all $1 \leq j \leq i,\ i \neq 4, j \neq 3$. In this sense, we say that the constraint set $\mathcal{C}_i$ is more complex than $\mathcal{C}_j$ for all $j < i \leq 6, i \neq 4, j \neq 3$ as illustrated in Figure 2. From a practical perspective, a more complex set of constraints allows us to retain more information about Z, but obviously requires an increased number of parameters.

Our abstraction allows us to handle all the above schemes in a systematic way. Now, consider a co-clustering basis $\mathcal{C} \in \{\mathcal{C}_i\}_{i=1}^{6}$ as the pertinent one. Given the choice of a particular basis, we need to decide on the "best" reconstruction $\hat{Z}$ for a given co-clustering (ρ, γ). Then the general co-clustering problem will effectively reduce to one of finding an optimal co-clustering (ρ^*, γ^*) whose reconstruction has the lowest approximation error with respect to the original Z.

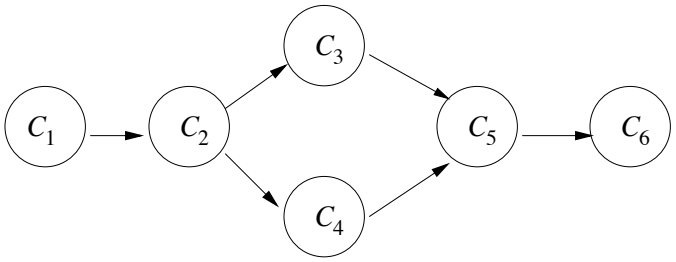


Figure 2: Relative complexity of the 6 co-clustering bases.

4.2 Minimum Bregman Information (MBI) Approximation

As in Section 3.1, for a given co-clustering (ρ, γ) and a given co-clustering basis $\mathcal{C}$, we use the MBI principle to obtain the "best" approximation $\hat{Z}$. Recall that for block average co-clustering, the search for the MBI solution was restricted to all Z' that preserved the co-cluster means. For a general co-clustering basis $\mathcal{C}$, the search space has to be appropriately generalized (or restricted) such that Z' preserves all the summary statistics relevant to $\mathcal{C}$. Let $\mathcal{S}_A$ denote a class of random variables such that every Z' in the class satisfies the following *linear constraints*, that is,

$$\mathcal{S}_A = \{Z' | E[Z|\mathcal{G}] = E[Z'|\mathcal{G}],\ \forall \mathcal{G} \in \mathcal{C}\}. \tag{17}$$

The reader may wish to compare the above definition (17) to the more specific definition (7) that is applicable in the case of block co-clustering. It can be readily seen that (7) follows by assuming that the co-clustering basis $\mathcal{C} = \{\{\hat{U}, \hat{V}\}\}$.

We now select the random variable $\hat{Z}_A \in \mathcal{S}_A$ that has the minimum Bregman information as the "best" approximation, that is,

$$\hat{Z}_A \equiv \operatorname*{argmin}_{Z' \in \mathcal{S}_A} I_\phi(Z'). \tag{18}$$

Coclustering basis $\mathcal{C}$	Lagrange multipliers	Approximation $\hat{Z}_A$
$\mathcal{C}_1$	$\Lambda^*_{\hat{U}} = -w_{\hat{U}} \log\left(\frac{E[Z\|\hat{U}]}{E[Z]}\right)$, $\Lambda^*_{\hat{V}} = -w_{\hat{V}} \log\left(\frac{E[Z\|\hat{V}]}{E[Z]}\right)$	$\frac{E[Z\|\hat{U}]\times E[Z\|\hat{V}]}{E[Z]}$
$\mathcal{C}_2$	$\Lambda^*_{\hat{U},\hat{V}} = -w_{\hat{U},\hat{V}} \log\left(\frac{E[Z\|\hat{U},\hat{V}]}{E[Z]}\right)$	$E[Z\|\hat{U},\hat{V}]$
$\mathcal{C}_3$	$\Lambda^*_{\hat{U},\hat{V}} = -w_{\hat{U},\hat{V}} \log\left(\frac{E[Z\|\hat{U},\hat{V}]}{E[Z]}\right)$, $\Lambda^*_{U} = -w_{U} \log\left(\frac{E[Z\|U]}{E[Z\|\hat{U}]}\right)$	$\frac{E[Z\|\hat{U},\hat{V}]\times E[Z\|U]}{E[Z\|\hat{U}]}$
$\mathcal{C}_4$	$\Lambda^*_{\hat{U},\hat{V}} = -w_{\hat{U},\hat{V}} \log\left(\frac{E[Z\|\hat{U},\hat{V}]}{E[Z]}\right)$, $\Lambda^*_{V} = -w_{V} \log\left(\frac{E[Z\|V]}{E[Z\|\hat{V}]}\right)$	$\frac{E[Z\|\hat{U},\hat{V}]\times E[Z\|V]}{E[Z\|\hat{V}]}$
$\mathcal{C}_5$	$\Lambda^*_{\hat{U},\hat{V}} = -w_{\hat{U},\hat{V}} \log\left(\frac{E[Z\|\hat{U},\hat{V}]}{E[Z]}\right)$ $\Lambda^*_{U} = -w_{U} \log\left(\frac{E[Z\|U]}{E[Z\|\hat{U}]}\right)$, $\Lambda^*_{V} = -w_{V} \log\left(\frac{E[Z\|V]}{E[Z\|\hat{V}]}\right)$	$\frac{E[Z\|\hat{U},\hat{V}]\times E[Z\|U]\times E[Z\|V]}{E[Z\|\hat{U}]\times E[Z\|\hat{V}]}$
$\mathcal{C}_6$	$\Lambda^*_{\hat{U},V} = -w_{\hat{U},V} \log\left(\frac{E[Z\|\hat{U},V]}{(E[Z]E[Z\|\hat{U},\hat{V}])^{1/2}}\right)$ $\Lambda^*_{U,\hat{V}} = -w_{U,\hat{V}} \log\left(\frac{E[Z\|U,\hat{V}]}{(E[Z]E[Z\|\hat{U},\hat{V}])^{1/2}}\right)$	$\frac{E[Z\|U,\hat{V}]\times E[Z\|\hat{U},V]}{E[Z\|\hat{U},\hat{V}]}$

Table 1: MBI solution and optimal Lagrange multipliers for I-Divergence.

The following theorem characterizes the solution to the MBI problem (18).

Theorem 4 *For any random variable Z and a specified co-clustering basis $\mathcal{C} = \{\mathcal{G}_r\}_{r=1}^s$, the solution $\hat{Z}_A$ to (18) is given by*

$$\nabla\phi(\hat{Z}_A) = \nabla\phi(E[Z]) - \sum_{r=1}^{s} \frac{\Lambda^*_{\mathcal{G}_r}}{w_{\mathcal{G}_r}}, \tag{19}$$

where $w_{\mathcal{G}_r}$ is the measure corresponding to $\mathcal{G}_r$ and $\{\Lambda^*_{\mathcal{G}_r}\}_{r=1}^s$ are the optimal Lagrange multipliers corresponding to the set of linear constraints:

$$E[Z'|\mathcal{G}_r] = E[Z|\mathcal{G}_r], \;\; [r]_1^s.$$

In the above theorem, note that every instantiation of the random variables $\{\mathcal{G}_r\}_{r=1}^s$ determines a single linear constraint and corresponds to uniquely determined scalar values for the optimal Lagrange multipliers $\{\Lambda^*_{\mathcal{G}_r}\}_{r=1}^s$, that is, $\Lambda^*_{\mathcal{G}_r}$ is a deterministic function of $\mathcal{G}_r$. Similarly, for each instantiation of $\mathcal{G}_r$, $w_{\mathcal{G}_r}$ equals the total measure associated with that particular instantiation, for example, $w_{\hat{u},\hat{v}} = \sum_{u:\rho(u)=\hat{u},v:\gamma(v)=\hat{v}} w_{uv}$. Further, the fact that ϕ is a strictly convex function ensures that $\nabla\phi$ is a one-to-one function so that (19) uniquely determines the approximation $\hat{Z}_A$. A proof of the above theorem is given in Appendix D.

For easy reference, in Tables 1-2, we present the optimal Lagrange multipliers[9] and the MBI solutions for I-divergence and squared Euclidean distance for each of the six co-clustering bases. Note that the approximation $\hat{Z}_A$ is itself a (U,V) measurable random variable and the elements of the corresponding matrix approximation $\hat{\mathbf{Z}}_A$ can be obtained by instantiating $\hat{Z}_A$ for specific choices of U and V. From Table 1, we observe that in case of I-divergence and original Z taking values over the probabilities of a joint distribution $p(X,Y)$, the approximation $\hat{Z}_A$ for the co-clustering basis $\mathcal{C}_5$ is given by $\frac{E[Z|\hat{U},\hat{V}]E[Z|U]E[Z|V]}{E[Z|\hat{U}]E[Z|\hat{V}]}$ which reduces to $q(X,Y) = \frac{p(X)p(Y)p(\hat{X},\hat{Y})}{p(\hat{X})p(\hat{Y})}$ (same as (3)) since the marginal over the various row, column and co-cluster partitions are directly proportional to the

9. The Lagrange dual $L(\Lambda)$ of Bregman information is concave in Λ for all bases, but strictly concave only for $\mathcal{C}_2$. Hence, the multipliers shown in Tables 1 and 2 are only one of the possible maximizers of $L(\Lambda)$ (for all the cases except $\mathcal{C}_2$).

Coclustering basis $\mathcal{C}$	Lagrange multipliers	Approximation $\hat{Z}_A$
$\mathcal{C}_1$	$\Lambda^*_{\hat{U}} = -2w_{\hat{U}}(E[Z\|\hat{U}] - E[Z])$, $\Lambda^*_{\hat{V}} = -2w_{\hat{U}}(E[Z\|\hat{V}] - E[Z])$	$E[Z\|\hat{U}] + E[Z\|\hat{V}] - E[Z]$
$\mathcal{C}_2$	$\Lambda^*_{\hat{U},\hat{V}} = -2w_{\hat{U},\hat{V}}(E[Z\|\hat{U},\hat{V}] - E[Z])$	$E[Z\|\hat{U},\hat{V}]$
$\mathcal{C}_3$	$\Lambda^*_{\hat{U},\hat{V}} = -2w_{\hat{U},\hat{V}}(E[Z\|\hat{U},\hat{V}] - E[Z])$, $\Lambda^*_U = -2w_U(E[Z\|U] - E[Z\|\hat{U}])$	$E[Z\|\hat{U},\hat{V}] + E[Z\|U] - E[Z\|\hat{U}]$
$\mathcal{C}_4$	$\Lambda^*_{\hat{U},\hat{V}} = -2w_{\hat{U},\hat{V}}(E[Z\|\hat{U},\hat{V}] - E[Z])$, $\Lambda^*_V = -2w_V(E[Z\|V] - E[Z\|\hat{V}])$	$E[Z\|\hat{U},\hat{V}] + E[Z\|V] - E[Z\|\hat{V}]$
$\mathcal{C}_5$	$\Lambda^*_{\hat{U},\hat{V}} = -2w_{\hat{U},\hat{V}}(E[Z\|\hat{U},\hat{V}] - E[Z])$ $\Lambda^*_U = -2w_U(E[Z\|U] - E[Z\|\hat{U}])$ $\Lambda^*_V = -2w_V(E[Z\|V] - E[Z\|\hat{V}])$	$E[Z\|\hat{U},\hat{V}] + E[Z\|U] + E[Z\|V]$ $-E[Z\|\hat{U}] - E[Z\|\hat{V}]$
$\mathcal{C}_6$	$\Lambda^*_{\hat{U},V} = -2w_{\hat{U},V}(E[Z\|\hat{U},V] - \frac{E[Z]}{2} - \frac{E[Z\|\hat{U},\hat{V}]}{2})$ $\Lambda^*_{U,\hat{V}} = -2w_{U,\hat{V}}(E[Z\|U,\hat{V}] - \frac{E[Z]}{2} - \frac{E[Z\|\hat{U},\hat{V}]}{2})$	$E[Z\|U,\hat{V}] + E[Z\|\hat{U},V] - E[Z\|\hat{U},\hat{V}]$

Table 2: MBI solution and optimal Lagrange multipliers for Squared Euclidean distance.

corresponding conditional expectations of Z. Further, the fact that $q(X,Y)$ is the minimum Bregman information solution for KL-divergence under certain constraints is equivalent to Lemma 1, which shows that it is the maximum entropy distribution under those constraints.

4.3 Co-clustering Problem Formulation

The expected Bregman divergence between the given random variable Z and the minimum Bregman information solution $\hat{Z}$ provides us with an elegant way to quantify the goodness of a co-clustering. This expected Bregman divergence is also exactly equal to the loss in Bregman information due to co-clustering as the following lemma shows. This equivalence provides another nice interpretation for the Bregman co-clustering formulation while generalizing the viewpoint presented in the information-theoretic co-clustering formulation (1) (originally Lemma 2.1 of Dhillon et al., 2003b).

Lemma 6 *For any random variable Z,*

$$E[d_\phi(Z,\hat{Z})] = I_\phi(Z) - I_\phi(\hat{Z}),$$

where $\hat{Z} = \hat{Z}_A$ defined in (18).

We are now ready to define the generalized co-clustering problem.

Definition 4 Given k, l, a Bregman divergence d_ϕ, a random variable Z following a non-negative measure w over the data matrix $\mathbf{Z} \in \mathcal{S}^{m\times n}$, and a co-clustering basis $\mathcal{C}$, we wish to find a co-clustering $(\rho^\star,\gamma^\star)$ that minimizes:

$$(\rho^\star,\gamma^\star) = \operatorname*{argmin}_{(\rho,\gamma)} E[d_\phi(Z,\hat{Z})] = \operatorname*{argmin}_{(\rho,\gamma)} (I_\phi(Z) - I_\phi(\hat{Z})) = \operatorname*{argmax}_{(\rho,\gamma)} I_\phi(\hat{Z}) \ , \tag{20}$$

where $\hat{Z} = \operatorname*{argmin}_{Z'\in\mathcal{S}_A} I_\phi(Z')$ as defined in (18).

The general problem is NP-hard by a reduction from the kmeans problem. Hence, it is difficult to obtain a globally optimal solution efficiently. However, in Section 5, we prove that it is possible to come up with an iterative update scheme that (a) monotonically decreases the objective function, and (b) converges to a local minimum of the problem.

Example 1.F (I-Divergence) Continuing from Example 1.C, the Bregman co-clustering objective function is given by $E[Z\log(Z/\hat{Z}) - Z + \hat{Z}] = E[Z\log(Z/\hat{Z})]$ since $E[Z] = E[\hat{Z}]$ where $\hat{Z}$ is the minimum Bregman information solution from Table 1. Note that for the co-clustering basis $\mathcal{C}_5$ and Z based on a joint distribution $p(X,Y)$, this reduces to $KL(p||q)$ where q is the joint distribution corresponding to the minimum Bregman solution indicating that (1) follows as a special case of (20).

Example 2.F (Squared Euclidean Distance) Continuing from Example 2.C, the Bregman co-clustering objective function is $E[(Z-\hat{Z})^2]$ where $\hat{Z}$ is the minimum Bregman information solution from Table 2. Note that for the co-clustering basis $\mathcal{C}_6$, this reduces to $E[(Z - E[Z|U,\hat{V}] - E[Z|\hat{U},V] + E[Z|\hat{U},\hat{V}])^2]$, which is equivalent to the squared residue objective function used in Cho et al. (2004) and Cheng and Church (2000).

4.4 Optimality of the MBI Solution

We now present an analysis of the optimality of the MBI solution as the "best" reconstruction of the original matrix given the row and column clustering and the summary statistics corresponding to any of the co-clustering bases. In Section 3, we showed that the minimum Bregman information solution is the best reconstruction among all measurable functions of the preserved summary statistics, that is, conditional expectations with respect to the co-clusters (Theorem 2). In this section, we present a generalization of that result, applicable to all the co-clustering bases discussed above.

Ideally, we would like to demonstrate that the MBI solution minimizes the approximation error with respect to the original matrix among all reconstructions that correspond to measurable functions of the available summary statistics. However, this property is not true for a general co-clustering basis since the optimal reconstruction depends on the structure of the original matrix, which is not available during the reconstruction process. For example, if the original matrix admits a perfect additive decomposition with respect to some coclustering basis, for example, $Z = E[Z|\hat{U}] + E[Z|\hat{V}] - E[Z]$ for basis $\mathcal{C}_1$, then the "best" reconstruction among all measurable functions of the conditional expectation statistics is given by this additive decomposition itself irrespective of the choice of the Bregman divergence. From Table 1, one can readily see that this solution is different from the MBI solution for I-divergence and basis $\mathcal{C}_1$ and in fact, it is different from the MBI solution for all Bregman divergences other than squared Euclidean distance. Therefore, instead of seeking the optimal reconstruction from the class of all measurable functions of the available summary statistics, we focus on a special class of approximations that correspond to additive models over the summary statistics.

Let $\mathcal{S}_B$ denote the set of all matrices Z'' whose inverse image under $\nabla\phi$ can be written as an additive model over the summary statistics, that is,

$$\mathcal{S}_B = \left\{ Z'' \,\middle|\, Z'' = (\nabla\phi)^{-1}\left(\sum_{r=1}^{s} g_r(E[Z|\mathcal{G}_r])\right)\right\}, \tag{21}$$

where $\{g_r\}_{r=1}^{s}$ are arbitrary functions measurable with respect to $\{\mathcal{G}_r\}_{r=1}^{s}$. Note that unlike $\mathcal{S}_A$, the set $\mathcal{S}_B$ explicitly depends on the choice of the convex function ϕ. The reader may wish to compare (9) for block average co-clustering with (21). Since $\mathcal{S}_B$ is defined in terms of arbitrary measurable functions, when there is only one conditional expectation to be preserved as in the case of block average co-clustering, $\mathcal{S}_B$ turns out to be the set of all possible measurable functions of that conditional expectation.

Interestingly, S_B can be alternatively understood from the perspective of Lagrange duality for Bregman divergences. Following Della Pietra et al. (2001), consider the Bregman projection problem of minimizing $d_\phi(p, q_0)$ over $p \in \mathbb{R}^d$ such that p lies in a linear subspace determined by π and a set of features $F = \{f_j, [j]_1^J\}$, that is, $\langle p, f_j \rangle = \langle \pi, f_j \rangle, [j]_1^J$. The dual of the problem turns out to be one of minimizing $d_\phi(\pi, q)$ over q, where q belongs to the dual space determined by q_0, feature set F, and Lagrange multipliers λ. Della Pietra et al. (2001) give a complete characterization of the dual space as a Legendre-Bregman projection family $Q(q_0, F)$ of approximations q. By generalizing their analysis, one can show that S_B is the Legendre-Bregman projection family corresponding to the set of linear constraints determined by S_A. Therefore, the Bregman duality and projection results of Della Pietra et al. (2001) also apply to our setting. Related analyses have appeared in the literature in the context of incremental learning of generalized entropy functionals (Lafferty, 1999), convergence analysis of boosting algorithms (Collins et al., 2000), and game theoretic interpretation of Bayesian decision theory (Grünwald and Dawid, 2004). However, we present our analysis using co-clustering semantics for ease of exposition. Further, our analysis leads to simpler proofs compared to the general setting of Della Pietra et al. (2001).

Example 1.G (I-Divergence) When $\phi(z) = z \log z - z$, the Legendre transformation or the gradient mapping turns out to be log-transformation, that is, $\nabla\phi(z) = \log z$ so that addition in the natural parameter space corresponds to multiplication in the original expectation parameter space and generalized additive models in the natural parameter space lead to generalized multiplicative models. The set $\mathcal{S}_B$ in this case can, therefore, be characterized as the set of all reconstructions corresponding to generalized multiplicative models, or in other words, products of arbitrary functions of the conditional expectations, that is,

$$\mathcal{S}_B = \left\{ Z'' \,\middle|\, Z'' = \prod_{r=1}^{s} g_r(E[Z|\,\mathcal{G}_r]) \right\},$$

where $\{g_r(\cdot)\}_{r=1}^{s}$ are arbitrary functions measurable with respect to $\{\mathcal{G}_r\}_{r=1}^{s}$.

Example 2.G (Squared Euclidean Distance) When $\phi(z) = z^2$, the Legendre transformation or the gradient mapping is the identity transformation, that is, $\nabla\phi(z) = z$ so that natural parameter space is identical to the original space. Therefore, $\mathcal{S}_B$ is just the set of all reconstructions corresponding to generalized additive models, or in other words, additive combinations of arbitrary functions of the conditional expectations, that is,

$$\mathcal{S}_B = \left\{ Z'' \,\middle|\, Z'' = \sum_{r=1}^{s} g_r(E[Z|\,\mathcal{G}_r]) \right\},$$

where $\{g_r(\cdot)\}_{r=1}^{s}$ are arbitrary functions measurable with respect to $\{\mathcal{G}_r\}_{r=1}^{s}$.

Among this class of reconstructions $\mathcal{S}_B$, let $\hat{Z}_B$ be the best approximation to Z in terms of Bregman divergence, that is,

$$\hat{Z}_B = \underset{Z'' \in \mathcal{S}_B}{\operatorname{argmin}} \; E[d_\phi(Z, Z'')] \; . \tag{22}$$

As Corollary 2 below shows, the best reconstruction $\hat{Z}_B$ among all elements of $\mathcal{S}_B$ is exactly identical to $\hat{Z}_A$, that is, the MBI solution among all elements of $\mathcal{S}_A$, which preserve the relevant conditional expectations. In order to arrive at this result, we make use of a projection theorem (Theorem 5) that characterizes the backward and forward Bregman projections of elements of $\mathcal{S}_B$ onto the set $\mathcal{S}_A$ and vice versa. This projection theorem, in turn, readily follows from the observation (Lemma 7) that the expected Bregman divergence between any $Z' \in \mathcal{S}_A$ and any $Z'' \in \mathcal{S}_B$ follows a Pythagorean decomposition involving the MBI solution $\hat{Z}_A$, that is, it can be expressed as the sum of expected Bregman divergences between the pairs $(Z', \hat{Z}_A)$, and $(\hat{Z}_A, Z'')$.

Lemma 7 *For any $Z' \in \mathcal{S}_A$ as in (17) and any $Z'' \in \mathcal{S}_B$ as in (21) and $\hat{Z}_A$ as in (18)*

$$E[d_\phi(Z', Z'')] = E[d_\phi(Z', \hat{Z}_A)] + E[d_\phi(\hat{Z}_A, Z'')].$$

A proof of the above lemma is given in Appendix D.[10] Using Lemma 7, we can now obtain the following projection theorem, which states that the MBI solution is the forward Bregman projection of any element of $\mathcal{S}_A$ onto the set $\mathcal{S}_B$ and the backward Bregman projection of any element of $\mathcal{S}_B$ onto the set $\mathcal{S}_A$.

Theorem 5 (Projection Theorem) *For any $Z' \in \mathcal{S}_A$ as in (17) and any $Z'' \in \mathcal{S}_B$ as in (21) and $\hat{Z}_A$ as in (18), the following two statements hold true:*

(a) $\hat{Z}_A = \underset{Z' \in \mathcal{S}_A}{\operatorname{argmin}} \; E[d_\phi(Z', Z'')], \;\; \forall Z'' \in \mathcal{S}_B,$

(b) $\hat{Z}_A = \underset{Z'' \in \mathcal{S}_B}{\operatorname{argmin}} \; E[d_\phi(Z', Z'')], \;\; \forall Z' \in \mathcal{S}_A.$

Since the original Z is also an element of $\mathcal{S}_A$, we observe that $\hat{Z}_A$ is the forward Bregman projection of Z onto $\mathcal{S}_B$, which leads to the equivalence between $\hat{Z}_A$ and $\hat{Z}_B$, which is the best reconstruction in $\mathcal{S}_B$.

Corollary 2 *For $\hat{Z}_A$ and $\hat{Z}_B$ given by (18) and (22), we have*

$$\hat{Z}_A = \hat{Z}_B \equiv \hat{Z} \; .$$

Proof Follows from the definition of $\hat{Z}_B$ and the projection theorem (Theorem 5). ∎

Corollary 2 gives a concrete justification for the use of the minimum Bregman information solution as the best matrix reconstruction for a given co-clustering since it is the optimum approximation among a large class of possible reconstructions obtained from the summary statistics. It is

10. The result can be derived by an application of Della Pietra et al. (2001, Proposition 3.2). We give a different proof, appropriate for the co-clustering setting.

straightforward to see that the corresponding result for block average co-clustering (Corollary 1) is a special case of this result. This equivalence result is also closely related to Csiszar's axiomatic justification (Csiszár, 1991) of the least squares and maximum entropy principles for linear inverse problems based on *sum consistency* and *product consistency* respectively. More specifically, the sum and product consistency conditions along with certain regularity, locality and fixed point assumptions[11] restrict the best reconstruction $\hat{Z}$ to generalized additive and multiplicative combinations of the observed linear functionals (i.e., conditional expectations in our case) respectively. Hence, the best approximation $\hat{Z} \in \mathcal{S}_B$ where $\mathcal{S}_B$ is defined as in Examples 1.G and 2.G. On the other hand, the constraint of preserving the observed linear functionals (i.e., conditional expectations) ensures that $\hat{Z} \in \mathcal{S}_A$ as well. Since $\mathcal{S}_A \bigcap \mathcal{S}_B = \{\hat{Z}\}$, it follows that $\hat{Z}$ is the MBI solution itself. In particular, the best reconstruction satisfying sum consistency is the least squares solution while the one satisfying product consistency is the maximum entropy solution.

Example 1.H (I-divergence) From Example 1.E, we observe that when $\phi(z) = z\log z - z$, the MBI solution $\hat{Z}_A$ is identical to the maximum entropy solution that preserves the conditional expectations. Further from Example 1.G, we note that the set $\mathcal{S}_B$ consists of generalized multiplicative combinations of the conditional expectations. Hence, from the projection theorem, it follows that the maximum entropy solution is the only generalized multiplicative solution that preserves the relevant conditional expectations. It is also the best reconstruction of Z (or any other $Z' \in \mathcal{S}_A$) among all multiplicative combinations of arbitrary functions of the conditional expectations.

Example 2.H (Squared Euclidean Distance) From Example 2.E, we observe that when $\phi(z) = z^2$, the MBI solution $\hat{Z}_A$ is identical to the standard least squares solution that preserves the conditional expectations. Further from Example 2.G, we note that the set $\mathcal{S}_B$ consists of generalized additive combinations of the conditional expectations. Hence, from the projection theorem, it follows that the least squares solution is the only generalized additive solution that preserves the relevant conditional expectations. It is also the best reconstruction of Z (or any other $Z' \in \mathcal{S}_A$) among all additive combinations of arbitrary functions of the conditional expectations.

5. A Meta Algorithm

In this section, we shall develop an alternating minimization scheme for the general Bregman co-clustering problem (20). Our scheme shall serve as a *meta algorithm* from which a number of special cases (both previously known and unknown) can be derived.

Throughout this section, let us suppose that the underlying measure w, the Bregman divergence d_ϕ, the data matrix $\mathbf{Z}$, number of row clusters k, number of column clusters l, and the co-clustering basis $\mathcal{C}$ are specified and fixed.

5.1 Intuition and Plan of Attack

We first outline the essence of our meta algorithm.

Step 1: Start with an arbitrary row and column clustering, say, (ρ^0, γ^0). Set $t = 0$.

Step 2: Repeat either of the following steps till convergence:

11. Please refer to Csiszár (1991) for details.

Step 2A: With respect to co-clustering (ρ^t, γ^t), compute the matrix approximation $\hat{Z}^t$ by solving the MBI problem (18).

Step 2B: Hold the column clustering γ^t fixed, and find a better row co-clustering, say, ρ^{t+1}. Set $\gamma^{t+1} = \gamma^t$. Set $t = t+1$.

Step 2C: Hold the row clustering ρ^{t+1} fixed, and find a better column co-clustering, say, γ^{t+1}. Set $\rho^{t+1} = \rho^t$. Set $t = t+1$.

We shall prove that this meta algorithm converges in a finite number of steps to a local minima.[12] As is clear from the outline above, a key step in our algorithm will involve finding a solution of the MBI problem (18). Further, since the number of possible row (or column) clusterings is exponential in the number of rows (or columns), it is also essential to have an efficient means for determining the best row (or column) clustering for a fixed choice of the column (or row) clustering and the MBI solution. Fortunately for the co-clustering problem, the expected distortion measure that quantifies the quality of a row (or column) clustering admits a separability property that allows independent optimal updates of the cluster assignments of every row (or column). We discuss this property in more detail below.

5.2 A Separability Property

We begin by considering the quality of a candidate row (or column) clustering ρ in Step 2B (or step 2C) for a fixed choice of column (or row) clustering and MBI solution parameters. Since our objective is to obtain an accurate reconstruction of the original matrix, a natural choice is to consider the expected Bregman distortion between the original Z and a reconstruction $\tilde{Z}$ based on the row (or column) clustering ρ while keeping everything else fixed. To characterize this reconstruction, we employ the functional form for the MBI solution $\hat{Z}$ given in Theorem 4. In general, the formula in Theorem 4 provides a unique reconstruction $\tilde{Z}$ for any set of Lagrange multipliers Λ (not necessarily optimal) and (ρ, γ), since $\nabla\phi(\cdot)$ is a monotonic, hence invertible, function (Azoury and Warmuth, 2001; Banerjee et al., 2005b). To underscore the dependence of $\tilde{Z}$ on the Lagrange multipliers, we shall use the notation $\tilde{Z} = \zeta(\rho, \gamma, \Lambda) = (\nabla\phi)^{-1}(\nabla\phi(E[Z]) - \sum_{r=1}^{s} \Lambda_{\mathcal{G}_r}/w_{\mathcal{G}_r})$. The quality of a candidate row (or column) clustering can now be quantified in terms of the accuracy of the corresponding $\tilde{Z}$ where the other two arguments, that is, the column (or row) clustering and Lagrange multipliers are fixed. In particular, $\hat{Z} = \zeta(\rho, \gamma, \Lambda^*)$ is the approximation corresponding to the optimal Lagrange multipliers Λ^*.

Given a set of (not necessarily optimal) Lagrange multipliers Λ, we now consider updating the current co-cluster assignments (ρ, γ) in order to improve the current approximation $\tilde{Z} = \zeta(\rho, \gamma, \Lambda)$. Although $\tilde{Z}$ looks complex, the fact that $\nabla\phi$ is a one-one invertible function ensures that each element $\tilde{z}_{uv}$ in the matrix $\tilde{\mathbf{Z}}$ corresponding to $\tilde{Z}$ depends only on $(u, \rho(u), v, \gamma(v))$ for a given Λ. Hence, for any given Λ, there exists a function ξ such that the point-wise distortion $d_\phi(z_{uv}, \tilde{z}_{uv})$ can be expressed as $\xi(u, \rho(u), v, \gamma(v))$, that is, it depends only on the corresponding row/column and cluster assignments. Since the expected distortion $E[d_\phi(Z, \tilde{Z})]$ is weighted sum of the point wise distortions, it satisfies a nice separability property that allows the current row (or column) assignments to be efficiently updated. In particular, for any given Λ, the expected distortion $E[d_\phi(Z, \tilde{Z})]$ can be expressed

12. In fact, any ordering of Steps 2B and 2C gives the same guarantees. Alternatively, one can run Steps 2A and 2C for some iterations followed by Steps 2A and 2B. We will establish that each step can only improve the quality of the current approximation. Hence, any ordering is sufficient to reach a local minimum.

as the sum of contributions from the rows (or columns) where each row (or column) contribution only depends on the row and its current cluster assignment. Note that this separability property is similar to that of the kmeans objective function, which can be also be expressed as the sum of terms corresponding to each point and its cluster assignment. As in the case of kmeans, the separability property allows independent updates of the cluster assignments of every row (or column). Further, for a fixed Λ and γ, since the total approximation error is the sum over the approximation errors due to each row (or column) and its cluster assignment, greedy cluster assignments of the individual rows result in a globally optimal row clustering ρ for the given Λ and γ. An equivalent statement is true for column assignments for a given Λ and ρ. The following lemma formally states this separability property. The proof simply follows from definitions, and is hence omitted.

Lemma 8 *For a fixed co-clustering (ρ,γ) and a fixed set of (not necessarily optimal) Lagrange multipliers Λ, and $\tilde{Z} = \zeta(\rho,\gamma,\Lambda)$, we can write:*

$$E[d_\phi(Z,\tilde{Z})] = E_U[E_{V|U}[\xi(U,\rho(U),V,\gamma(V))]] = E_V[E_{U|V}[\xi(U,\rho(U),V,\gamma(V))]] \ ,$$

where $\xi(U,\rho(U),V,\gamma(V)) = d_\phi(Z,\tilde{Z})$.

5.3 Updating Row and Column Clusters

We will now present the details of our plan in Section 5.1. First, we will demonstrate how to update row clustering (or column clustering) with respect to a fixed column clustering (or row clustering) and a fixed set of Lagrange multipliers. Then, we will find the optimal Lagrange multipliers corresponding to the minimum Bregman solution of the updated co-clustering.

Suppose we are in Step 2A outlined in Section 5.1. Updating the row clustering keeping the column clustering and the Lagrange multipliers fixed leads to a new value for the Bregman co-clustering objective function. Now making use of the separability property in Lemma 8, we can efficiently optimize the contribution of each row assignment to the overall objective function to obtain the following row cluster update step.

Lemma 9 *Let ρ^{t+1} be defined as*

$$\rho^{t+1}(u) = \operatorname*{argmin}_{g:[g]_1^k} E_{V|u}[\xi(u,g,V,\gamma^t(V))], \ \ [u]_1^m,$$

and let $\tilde{Z}^t = \zeta(\rho^{t+1},\gamma^t,\Lambda^{*t})$. Then,

$$E[d_\phi(Z,\tilde{Z}^t]) \leq E[d_\phi(Z,\hat{Z}^t)].$$

where $\hat{Z}^t = \zeta(\rho^t,\gamma^t,\Lambda^{*t})$.

A similar argument applies to step 2B where we seek to update the column clustering keeping the row clustering fixed.

Lemma 10 *Let γ^{t+1} be defined as*

$$\gamma^{t+1}(v) = \operatorname*{argmin}_{h:[h]_1^l} E_{U|v}[\xi(U,\rho^t(U),v,h)] \ \ [v]_1^n,$$

and let $\tilde{Z}^t = \zeta(\rho^t, \gamma^{t+1}, \Lambda^{*t})$. *Then,*

$$E[d_\phi(Z, \tilde{Z}^t)] \leq E[d_\phi(Z, \hat{Z}^t)].$$

where $\hat{Z}^t = \zeta(\rho^t, \gamma^t, \Lambda^{*t})$.

We now consider step 2C. So far, we have only considered updating the row (or column) assignments keeping the Lagrange multipliers fixed. After such updates, the approximation $\tilde{Z}^t = \zeta(\rho^{t+1}, \gamma^{t+1}, \Lambda^{*t})$ is closer to the original matrix Z than the earlier minimum Bregman information solution $\hat{Z}^t$, but the Lagrange multipliers Λ^{*t} are not optimal, in general. In other words, the approximation $\tilde{Z}^t$ is not a minimum Bregman information solution. For the given co-clustering $(\rho^{t+1}, \gamma^{t+1})$, let Λ^{*t+1} be the optimal Lagrange multipliers for the corresponding minimum Bregman information problem in (18). The corresponding matrix approximation $\hat{Z}^{t+1} = \zeta(\rho^{t+1}, \gamma^{t+1}, \Lambda^{*t+1})$ is a minimum Bregman information solution. As the following lemma shows, this approximation is better than the current approximation $\tilde{Z}^t$.

Lemma 11 *Let* $\hat{Z}^{t+1} = \zeta(\rho^{t+1}, \gamma^{t+1}, \Lambda^{*t+1})$ *be the minimum Bregman information solution corresponding to* $(\rho^{t+1}, \gamma^{t+1})$ *with* Λ^{*t+1} *being the optimal Lagrange multipliers for (18). Then,*

$$E[d_\phi(Z, \hat{Z}^{t+1}) \leq E[d_\phi(Z, \tilde{Z}^t)],$$

where $\tilde{Z}^t = \zeta(\rho^{t+1}, \gamma^{t+1}, \Lambda^{*t})$.

Proof By definition,

$$\begin{aligned}
E[d_\phi(Z, \hat{Z}^{t+1})] &= E[\phi(Z) - \phi(\hat{Z}^{t+1}) - \langle Z - \hat{Z}^{t+1}, \nabla\phi(\hat{Z}^{t+1})\rangle] \\
&\overset{(a)}{=} E[\phi(Z) - \phi(\hat{Z}^{t+1})] \\
&= E[d_\phi(Z, \tilde{Z}^t)] - E[d_\phi(\hat{Z}^{t+1}, \tilde{Z}^t)] - E[\langle Z - \hat{Z}^{t+1}, \nabla\phi(\tilde{Z}^t)\rangle] \\
&\overset{(b)}{=} E[d_\phi(Z, \tilde{Z}^t)] - E[d_\phi(\hat{Z}^{t+1}, \tilde{Z}^t)] \\
&\leq E[d_\phi(Z, \tilde{Z}^t)] \,,
\end{aligned}$$

where (a) follows since $\hat{Z}^{t+1}$ belongs to both Γ_A and Γ_B so that taking conditional expectations over $E[Z|\mathcal{G}], \mathcal{G} \in \mathcal{C}$ makes the last term zero and (b) follows since $\nabla\phi(\tilde{Z}^t)$ is a summation of terms involving $E[Z]$ and $\Lambda_{\mathcal{G}_r}, [r]_1^s$, and $E[\hat{Z}^{t+1}|\mathcal{G}_r] = E[Z|\mathcal{G}_r]$, thus making the last term vanish. ∎

5.4 The Algorithm

The meta algorithm for generalized Bregman co-clustering (see Algorithm 2) is a concrete "implementation" of our plan in Section 5.1. Comparing this algorithm with the solution for block average co-clustering (Algorithm 1), one can readily see that both the algorithms are based on an identical alternate minimization strategy and Algorithm 1 is in fact a special case of Algorithm 2 when the MBI solution corresponds to the co-cluster means. We now establish that our algorithm is guaranteed to achieve local optimality.

Algorithm 2 Bregman Co-clustering Algorithm

Input: Matrix $\mathbf{Z} \subseteq S^{m\times n}$, probability measure w, Bregman divergence $d_\phi : S \times \text{int}(S) \mapsto \mathbb{R}_+$, num. of row clusters l, num. of column clusters k, co-clustering basis $\mathcal{C}$.
Output: Co-clustering (ρ^*, γ^*) that (locally) optimize the objective function in (20).
Method:

{**Initialize** ρ, γ }
Start with an arbitrary co-clustering (ρ, γ)
repeat
{**Step A: Update Minimum Bregman Information Solution** (Λ^*)}
$\Lambda^* \leftarrow \underset{\Lambda}{\text{argmax}}\ L(\Lambda)$ where $L(\cdot)$ is Lagrange dual of the MBI problem (18).
{**Step B: Update Row Clusters** (ρ)}
for $u = 1$ to m **do**
$\rho(u) \leftarrow \underset{g:[g]_1^k}{\text{argmin}}\ E_{V|u}[\xi(u, g, V, \gamma(V))]$
where $\xi(U, \rho'(U), V, \gamma(V)) = d_\phi(Z, \tilde{Z}),\ \ \tilde{Z} = \zeta(\rho', \gamma, \Lambda^*)$ for any ρ'
end for
{**Step C: Update Column Clusters** (γ)}
for $v = 1$ to n **do**
$\gamma(v) \leftarrow \underset{h:[h]_1^l}{\text{argmin}}\ E_{U|v}[\xi(U, \rho(U), v, h)]$
where $\xi(U, \rho(U), V, \gamma'(V)) = d_\phi(Z, \tilde{Z}),\ \ \tilde{Z} = \zeta(\rho, \gamma', \Lambda^*)$ for any γ'
end for
until *convergence*
return (ρ, γ)

Theorem 6 *The general Bregman co-clustering algorithm (Algorithm 2) converges to a solution that is locally optimal for the Bregman co-clustering problem (20), that is, the objective function cannot be improved by changing either the row clustering, the column clustering or the Lagrange multipliers.*

Proof From Lemmas 9, 10, and 11, it follows that updating the row clustering ρ, the column clustering γ and the Lagrange multipliers Λ one at a time decreases the objective function of the Bregman co-clustering problem. Hence, the Bregman co-clustering algorithm (Algorithm 2) which proceeds by alternately updating $\rho \rightarrow \gamma \rightarrow \Lambda$ monotonically decreases the Bregman co-clustering objective function. Since the number of distinct co-clusterings is finite, the algorithm is guaranteed to converge to a locally optimal solution. ∎

Note that updating Λ is the same as obtaining the MBI solution. When the Bregman divergence is I-divergence or squared deviation, the minimum Bregman information problem has an analytic closed form solution as shown in Tables 1 and 2. Hence, it is straightforward to obtain the row and column cluster update steps and implement these Bregman co-clustering algorithms. The resulting algorithms involve computational effort that is linear per iteration in the size of the data and are hence scalable. In general, the MBI problem has a unique solution since it involves a strictly convex objective function and linear constraints. However, the solution need not have a closed form and has to be obtained numerically using iterative projection algorithms, which in turn involves solving non-

linear systems of equations. In the general case, the Bregman co-clustering algorithm will include such iterative projection procedures as a sub-routine.

Details on several different instantiations of the meta-algorithm (using matrix notation) for certain specific choices of Bregman divergences and co-clustering bases are given in Appendix E. In particular, exact algorithms for (i) basis $\mathcal{C}_2$ and all Bregman divergences, (ii) Euclidean distance and all co-clustering bases, and (iii) I-divergence and all co-clustering bases have been worked out. The MBI problem has a closed form solution in all of the above three cases. Further, as a representative of the general case of arbitrary Bregman divergences and co-clustering bases, we show an instantiation of the meta-algorithm to Itakura-Saito distance, for which the MBI problem does not have a closed form solution.

5.5 Iterative Algorithms for the Minimum Bregman Information Problem

An important part of the Bregman co-clustering algorithm involves solving the MBI problem. While there are closed form solutions for some important choices of Bregman divergences and summary statistics, the general case leads to a convex programming problem and does not have a closed form solution. In this section, we discuss two simple iterative algorithms to solve the MBI problem. The first one is *Bregman's algorithm* (Bregman, 1967; Censor and Zenios, 1998) and the second is an *iterative scaling* method (Della Pietra et al., 2001).

Recall that the MBI solution $\hat{Z}$ for a co-clustering basis $\mathcal{C}$ is given by

$$\hat{Z} = \operatorname*{argmin}_{Z' | E[Z'|C]=E[Z|C],\ \forall C \in \mathcal{C}} E[d_\phi(Z', E[Z'])] \ .$$

For notational convenience, let $\mathbf{z}$, $\mathbf{z}'$ and $\bar{\mathbf{z}}$ denote vectorized versions of the original matrix $\mathbf{Z}$, the tentative solution matrix $\mathbf{Z}'$, and a constant matrix consisting of the expectation $E[Z]$ respectively. Then $\mathbf{z}, \mathbf{z}'$ and $\bar{\mathbf{z}}$ are all vectors of dimension mn. Let $\mathbf{A}$ denote the $c \times mn$ matrix corresponding to the linear constraints $E[Z'|\mathcal{G}] = E[Z|\mathcal{G}]$, $\forall \mathcal{G} \in \mathcal{C}$, where c is the total number of constraints, so that the constraints can be written as $\mathbf{A}\mathbf{z}' = \mathbf{A}\mathbf{z}$. The vectorized version $\hat{\mathbf{z}}$ of the MBI solution can now be written as

$$\hat{z} = \operatorname*{argmin}_{\mathbf{z}' | \mathbf{A}\mathbf{z}' = \mathbf{A}\mathbf{z}} \sum_{\iota=1}^{mn} w_\iota d_\phi(z'_\iota, \bar{z}_\iota). \tag{23}$$

Since a convex combination of Bregman divergences is again a Bregman divergence, the objective function in (23) can be readily expressed as the Bregman divergence between the vectors $\mathbf{z}'$ and $\bar{\mathbf{z}}$ derived from the convex function $\phi_w(\mathbf{z}') = \sum_{\iota=1}^{mn} w_\iota \phi(z'_\iota)$, that is,

$$\hat{z} = \operatorname*{argmin}_{\mathbf{z}' | \mathbf{A}\mathbf{z}' = \mathbf{A}\mathbf{z}} d_{\phi_w}(\mathbf{z}', \bar{\mathbf{z}}).$$

Since ϕ_w is the convex function induced on the vectorized matrices by the original convex function ϕ, we ignore this distinction and use ϕ to denote ϕ_w as well when it is clear that the function is being applied to matrices.

5.5.1 BREGMAN'S ALGORITHM (BREGMAN, 1967)

Bregman's algorithm requires that the initial guess $\mathbf{z}'_0$ belong to the set $\{\mathbf{z}' | \mathbf{z}' \in \text{int}(\text{dom}(\phi)),\ \nabla\phi(\mathbf{z}') = \mathbf{A}^T\mathbf{x}, \mathbf{x} \in \mathbb{R}^c\}$. The unconstrained global optimum $\mathbf{z}_*$ belongs to this set since $\nabla\phi(\mathbf{z}_*) = \mathbf{0}$ which is

$\mathbf{A}^T\mathbf{x}$ for $\mathbf{x} = \mathbf{0} \in \mathbb{R}^c$. Hence, we use $\mathbf{z}_*$ as the initial guess, that is,

$$\mathbf{z}'_0 \;=\; \mathbf{z}_* \;. \tag{24}$$

Subsequent iterative updates are obtained by solving the following set of equations:

$$\nabla\phi(\mathbf{z}'^{t+1}) \;=\; \nabla\phi(\mathbf{z}'^{t}) + \lambda A_i^T \;, \tag{25}$$

$$A_i\,\mathbf{z}'^{t+1} \;=\; A_i\,\mathbf{z} \;, \tag{26}$$

where A_i is the i^{th} row of $\mathbf{A}$ and $\lambda \in \mathbb{R}$. The solution to the above set of equations can be considered as the Bregman projection of the current tentative solution $\mathbf{z}'^t$ onto the hyperplane $\{\mathbf{z}' | A_i\mathbf{z}' = A_i\mathbf{z}\}$. Due to the strict convexity of ϕ, the update equations, under proper regularity conditions (Bregman, 1967), uniquely determine $\mathbf{z}'^{t+1}$ and λ. However, the equations are non-linear and one needs to use appropriate numerical techniques to solve for $\mathbf{z}'^{t+1}$.

The update equations (25) and (26) are based on only one linear constraint. For convergence to the optimum, the updates must touch upon all the constraints following a schedule known as relaxation control (Bregman, 1967; Bauschke and Borowein, 1997). For simplicity, we consider updates based on a cyclic ordering of the constraints, where all constraints are considered one after the other. The cyclic ordering schedule is sufficient to guarantee convergence to the optimum solution, although more general schedules are admissible (Bauschke and Borowein, 1997).

5.5.2 ITERATIVE SCALING (DELLA PIETRA ET AL., 2001)

We now discuss an auxiliary function-based iterative scaling method to solve the problem. The method makes use of the Legendre-Bregman projection $\mathcal{L}_\phi(\mathbf{z}'^t, \mathbf{A}^T\lambda)$, which is the "backward" Bregman projection of $\mathbf{z}'^t$ onto the hyperplane determined by $\{\mathbf{z}' | \mathbf{z}'^T\mathbf{A}^T\lambda = \mathbf{z}^T\mathbf{A}^T\lambda\}$, so that

$$\begin{aligned} \mathbf{z}'^{t+1} \;&=\; \mathcal{L}_\phi(\mathbf{z}', \mathbf{A}^T\lambda) \;=\; (\nabla\phi)^{-1}(\nabla\phi(\mathbf{z}') + \mathbf{A}^T\lambda) \\ \Rightarrow \nabla\phi(\mathbf{z}'^{t+1}) \;&=\; \nabla\phi(\mathbf{z}') + \mathbf{A}^T\lambda \;. \end{aligned} \tag{27}$$

The similarity between the Legendre-Bregman projection as in (27) and the first update equation (25) is due to the fact that both are Bregman projections of a point onto a hyperplane. However, Bregman's algorithm considers one constraint at a time, whereas iterative scaling works with all the constraints simultaneously.

As before, we set the initial guess $\mathbf{z}'_0 = \mathbf{z}_*$. Using the constraint matrix $\mathbf{A}$, we select $N_j \geq \sum_{i=1}^{c} A_{ij}$ for $j = 1, \ldots, mn$. Then, the iterative update of the tentative solution is given by (27), where $\lambda \in \mathbb{R}^c$ and each component λ_i satisfies

$$\sum_{j=1}^{mn} A_{ij}\; \mathcal{L}_\phi(z'^t{}_j,\; s_{ij}N_j\lambda_i) \;=\; A_i\mathbf{z} \;, \tag{28}$$

where $s_{ij} = \text{sign}(A_{ij})$ and ϕ operates on the matrix elements.

As before, the system of equations (27) and (28) is non-linear and one needs to use proper numerical methods to obtain the updates. However, there is an important difference between the iterative scaling updates and the updates of Bregman's algorithm. Since (28) is in terms of each component of λ, one can obtain λ entirely from (28). This λ can then be used in (27) to get $\mathbf{z}'^{t+1}$. In other words, analogous to the EM algorithm, iterative scaling allows one to alternate updates to λ

and $\mathbf{z}'$ till convergence. This is not possible in case of Bregman's algorithm where both the equations (25) and (26) have to be solved simultaneously. Note that both the algorithms require regularity conditions to guarantee convergence. The reader is referred to the original papers (Bregman, 1967; Della Pietra et al., 2001) for details.

6. Experiments

In recent years, co-clustering has been successfully applied to a number of application domains such as text mining (Dhillon et al., 2003b; Gao et al., 2005; Takamura and Matsumoto, 2003), image and video analysis (Zhong et al., 2004; Qiu, 2004; Guan et al., 2005; Cai et al., 2005), natural language processing (Freitag, 2004; Rohwer and Freitag, 2004; Li and Abe, 1998), bio-informatics (Cheng and Church, 2000; Cho et al., 2004; Kluger et al., 2003) as well as other applications (Carrasco et al., 2003). In particular, there exist a number of empirical studies that illustrate the usefulness of particular instances of the Bregman co-clustering framework that we describe in this paper. Hence, instead of extensively evaluating our methodology on various application domains, we present a brief summary of existing experimental results. Further, we present a comparative empirical study of the different co-clustering bases as well as the divergences discussed in this paper. Finally, we highlight new applications such as missing value prediction and co-clustering of matrices with categorical elements.

6.1 Existing Applications and Results

In this section, we present a brief overview of some of the existing applications of co-clustering.

6.1.1 TEXT CLUSTERING

Text clustering is one of the first domains where a special case of the Bregman co-clustering algorithm, namely the information-theoretic co-clustering algorithm based on I-divergence and basis $\mathcal{C}_5$, has been successfully applied. The key task in text clustering is to identify document clusters. Since most of the information in a document can be captured using a *bag-of-words* model, a convenient vector-space representation is in the form of word-document co-occurrence matrices with documents corresponding to rows and words corresponding to columns. However, it is often difficult to obtain good document clusters by directly clustering the matrix rows due to the inherent sparsity and high dimensionality (i.e., large number of words). Co-clustering, on the other hand, performs an implicit dimensionality reduction by clustering the words and hence, is more effective and efficient for identifying document clusters. Since word-document co-occurrence matrices can be interpreted as estimates of unnormalized joint distribution, an appropriate choice for the loss function is the I-divergence cost used by Dhillon et al. (2003b) and Takamura and Matsumoto (2003). Previous empirical evaluations on some of the popular text data sets (*NG20* and *CLASSIC3*) (Dhillon et al., 2003b) reveal that this choice of co-clustering algorithm provides performance comparable to the best text-clustering algorithms while yielding superior results than single-sided information-theoretic clustering. In particular, there is a significant improvement in the micro-averaged precision values with respect to single-sided clustering; See Dhillon et al. (2003b) for more details.

6.1.2 NATURAL LANGUAGE PROCESSING

Natural language processing is yet another domain where co-clustering has been widely employed as a key intermediate technique for obtaining an informative partitioning of both the language tokens and contexts, which in turn facilitates improved performance on various tasks such as named-entity recognition (Freitag, 2004), automatic construction of lexicon (Rohwer and Freitag, 2004) and prepositional phrase attachment disambiguation (Li and Abe, 1998). In all these applications, the relevant structural information in an unlabeled text corpus can be effectively captured in terms of the distributional properties of appropriately defined language tokens with respect to the contexts in which they occur, for example, k-neighborhood of tokens on either side, verb preceding the token, etc. Hence, one could expect improved performance by leveraging the token-context co-occurrence matrices. However, for most natural language processing applications, the number of tokens and contexts is extremely large, making it infeasible to directly employ computationally intensive learning algorithms. Co-clustering alleviates this problem by producing a highly informative, but reduced cluster-based representation for both tokens and contexts, thus making it possible to incorporate additional information from unlabeled text. As in the case of text clustering, the normalized token-context co-occurrence matrices can be interpreted as a joint distribution and hence, most of the co-clustering methods employed in natural processing applications are based on the KL-divergence loss function, or equivalently, the loss in mutual information using co-clustering basis $\mathcal{C}_5$. Empirical studies (Freitag, 2004; Rohwer and Freitag, 2004; Li and Abe, 1998) demonstrate that the use of co-clustering as an intermediate step makes it straightforward to leverage the additional information in unlabeled repositories and leads to substantial performance improvement for a number of natural language processing applications with negligible manual supervision. In particular, Freitag (2004) shows that including additional features based on co-clustering resulted in better entity recognition accuracy (statistically significant for certain entity types) on the MUC 6 named entity data set, while Li and Abe (1998) demonstrate that predictive methods based on the conditional probabilities derived from co-clustering noun and verb phrases provide better accuracy than state-of-the-art rule-based methods on the prepositional phrase attachment task.

6.1.3 BIO-INFORMATICS

In recent years, co-clustering methods are being increasingly employed for analyzing biological data as well, in particular for studying microarray data consisting of gene expression matrices where rows corresponds to genes and columns correspond to experimental conditions. The fundamental problem in this setting is to identify groups of similar genes and similar conditions based on their expression levels. To address this problem, a number of co-clustering configurations (e.g., overlapping, partitional) and loss functions based on additive and multiplicative models have been proposed (Madeira and Oliveira, 2004). These methods have been shown to be quite effective for identifying highly correlated genes and conditions. In particular, a special case of the Bregman co-clustering (Cheng and Church, 2000; Cho et al., 2004) corresponding to squared loss function and basis $\mathcal{C}_6$ has been shown to provide high quality co-clusters on biological data sets involving a variety of human cancer data sets.

6.1.4 VIDEO/IMAGE/SPEECH CONTENT ANALYSIS

There have also been a number of interesting applications of co-clustering in areas such as video, image and speech content analysis for performing unsupervised categorization of video segments

(Zhong et al., 2004), images (Qiu, 2004; Guan et al., 2005) and auditory scenes (Cai et al., 2005). Each of these settings involves two large sets of entities related to each other through co-occurrence matrices—(i) auditory scenes and audio effects in case of speech content analysis, (ii) fixed length video segments and prototype images for video content recognition, and (iii) images and low level features in case of image recognition. Further, as in the case of text clustering, information-theoretic co-clustering methods based on preserving mutual information effectively handle the sparsity and high dimensionality problems to provide high quality categorization of the dual sets of entities. Empirical results on auditory scene and image categorization show improved classification accuracy as compared to single-sided clustering methods.

6.2 Choice of Bregman Divergence and Co-clustering Basis

We now empirically study the appropriateness of the choice of the Bregman divergence and the co-clustering basis for specific tasks. When the choice of the Bregman divergence and the specified statistics capture the natural structure of the data, it is possible to obtain a more accurate low parameter representation of the original data. To illustrate this idea, we perform co-clustering on synthetic data matrices produced using certain generative models as well as on real-life matrices—(i) word-document matrices encountered in text analysis, and (ii) user-movie rating matrices for recommender systems.

6.2.1 SYNTHETIC DATA MATRICES

First, to study the dependence on the Bregman divergence, we generated multiple (10) sets of three classes of artificial 50×50 matrices $\mathbf{M}_{Euc}, \mathbf{M}_{Idiv}$, and $\mathbf{M}_{IS}$, using generative models corresponding to three different choices of Bregman divergences—squared Euclidean distance, I-divergence, and Itakura-Saito distance. It can be shown that the appropriate generative models in this case respectively correspond to mixtures of Gaussian, Poisson and exponential distributions centered at the co-cluster means.[13] In the generative model, we used 5 row clusters and 5 column clusters. The means of each of the co-clusters were chosen to be identical (all positive values) for all the three classes of matrices. Table 3 shows the results (averaged over 10 sets) of co-clustering these matrices using the Bregman co-clustering algorithms corresponding to the basis $\mathcal{C}_2$ and the three choices of Bregman divergence with $k = l = 5$. In each case, the co-clustering algorithms were run 10 times and the reported quality corresponds to the best run in terms of the objective function. Since the co-clustering objective functions based on the different divergences are not comparable and sometimes not even well-defined,[14] we measure the co-clustering quality in terms of the average of the normalized mutual information (Strehl and Ghosh, 2002) between the clustering and true class labels over both the rows and the columns. The standard-deviations reported in the table correspond to the deviations over multiple sets of matrices. From the table, it is clear that the co-clustering quality (i.e., row and column clustering), as indicated by the normalized mutual information with true labels, is better when the Bregman divergence used in the co-clustering algorithm matches that of the generative model.

13. The reader is referred to Banerjee et al. (2005b) for a connection between Bregman divergences and exponential family distributions. The data sets were generated based on extensions of the results obtained by Banerjee et al. (2005b).

14. For example, I-divergence and Itakura-Saito costs are not defined for approximation matrices with negative values.

	NMI for Co-clustering		
Matrix	**Squared Euclidean Distance**	**I-divergence**	**Itakura-Saito distance**
$\mathbf{M}_{Euc}$	$\mathbf{0.812 \pm 0.029}$	0.685 ± 0.041	0.637 ± 0.044
$\mathbf{M}_{Idiv}$	0.645 ± 0.037	$\mathbf{0.689 \pm 0.035}$	0.621 ± 0.042
$\mathbf{M}_{IS}$	0.586 ± 0.082	0.622 ± 0.047	$\mathbf{0.636 \pm 0.039}$

Table 3: Normalized mutual information (NMI) between the true labels and the clusters obtained using different Bregman divergences, basis $\mathcal{C}_2$ and $k = l = 5$. Results indicate better performance when the Bregman divergence matches the generative model.

Matrix	$\mathcal{C}_1$	$\mathcal{C}_2$	$\mathcal{C}_3$	$\mathcal{C}_4$	$\mathcal{C}_5$	$\mathcal{C}_6$
$\mathbf{M}_1$	6.10 ± 0.13	6.02 ± 0.13	5.80 ± 0.15	5.69 ± 0.14	5.40 ± 0.12	4.89 ± 0.10
$\mathbf{M}_2$	22.62 ± 1.81	6.32 ± 0.94	6.15 ± 0.91	6.16 ± 0.95	5.99 ± 0.89	5.12 ± 0.23
$\mathbf{M}_3$	22.39 ± 1.87	12.84 ± 1.06	6.76 ± 1.24	8.82 ± 1.15	6.57 ± 1.03	5.04 ± 0.29
$\mathbf{M}_4$	23.28 ± 1.93	12.98 ± 1.11	8.87 ± 1.04	6.19 ± 0.98	6.42 ± 0.96	5.08 ± 0.31
$\mathbf{M}_5$	24.53 ± 2.08	14.19 ± 1.28	10.31 ± 1.22	11.96 ± 1.18	6.14 ± 0.99	5.29 ± 0.25
$\mathbf{M}_6$	44.41 ± 2.75	33.34 ± 1.79	29.18 ± 2.05	31.26 ± 1.99	25.74 ± 1.26	5.01 ± 0.33

Table 4: Approximation errors on synthetic matrices for different co-clustering bases using squared Euclidean distance and $k = l = 5$. The results indicate that the performance saturates when the complexity of the co-clustering basis matches that of the generative model.

In order to study how the approximation error depends on the choice of co-clustering basis, we created multiple (10) sets of six 50×50 data matrices, $\mathbf{M}_1, \mathbf{M}_2, \ldots,$ and $\mathbf{M}_6$ using generative models based on the Gaussian family, but with increasing levels of complexity corresponding to the various co-clustering bases. This was done by first obtaining the minimum Bregman information approximations of an arbitrary 50×50 matrix corresponding to the various co-clustering bases and then adding Gaussian noise to each of the approximations. We perform Bregman co-clustering on each of these matrices using squared Euclidean distance and $k = l = 5$. Table 4 presents the approximation error obtained for each of these matrices using the various co-clustering bases. From the table, it is clear that for relatively simple matrices such as $\mathbf{M}_1$ and $\mathbf{M}_2$, reasonably low parameter bases such as $\mathcal{C}_1$ or $\mathcal{C}_2$ suffice, whereas for more complex matrices such as $\mathbf{M}_6$, high parameter co-clustering bases such as $\mathcal{C}_6$ are necessary. Figures 3 and 4 show images of the original data matrix $\mathbf{M}_2$ and $\mathbf{M}_6$, and the reconstructions obtained using the different co-clustering bases. The figures reinforce the observation we make from the table. In particular, in Figure 3, one can visually infer that the reconstruction of the matrix $\mathbf{M}_2$ obtained using $\mathcal{C}_2$ is reasonably accurate and cannot be improved much using more complex co-clustering bases whereas, in Figure 4, the reconstruction of $\mathbf{M}_6$ obtained using $\mathcal{C}_6$ is significantly better than that obtained using the other co-clustering bases, thus clearly demonstrating that the choice of co-clustering basis should match the generative model in order to obtain an accurate approximation.

6.2.2 WORD-DOCUMENT MATRICES

As mentioned earlier, co-clustering has been successfully applied to text analysis (Dhillon et al., 2003b). Since several results comparing specific co-clustering schemes to alternative text clustering approaches have already been studied, we focus on the relative performance of the different co-clustering bases introduced in this paper. We use the *CLASSIC3* data set with 3891 documents represented in the bag-of-words model with 4666 words. We fix the number of document clusters to be three, which is the number of document classes in the data set. Figure 5 shows the relative performance (averaged over 10 runs) of all the six co-clustering schemes for a varying number

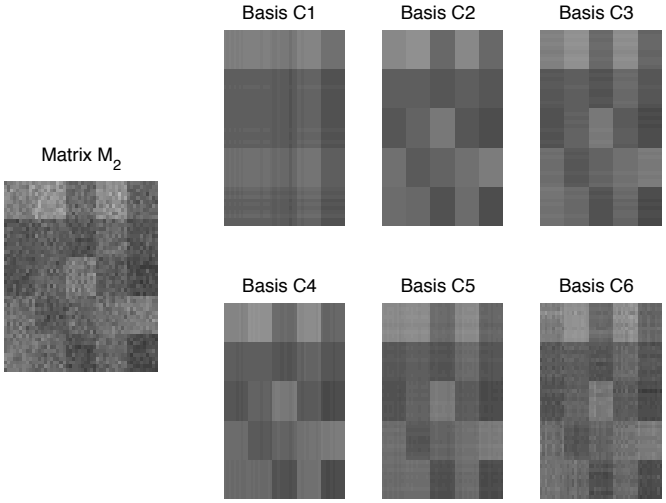


Figure 3: Co-clustering-based approximation of a simple 50×50 matrix $\mathbf{M}_2$ using various co-clustering bases, squared distortion and $k = l = 5$. While the matrix is too complex for $\mathcal{C}_1$, all bases from $\mathcal{C}_2$ onwards get an accurate approximation. Note that all matrices are shown with a consistent permutation (which the co-clustering finds) for easy visual comparison.

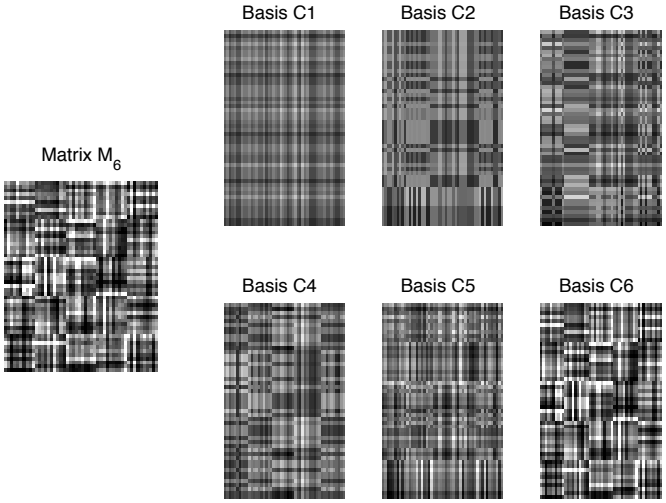


Figure 4: Co-clustering-based approximation of a 50×50 matrix $\mathbf{M}_6$ using various co-clustering bases, squared distortion and $k = l = 5$. Since the given matrix has a fairly complicated structure, only $\mathcal{C}_6$ gets an accurate approximation. All other schemes have more errors, with the simple bases ($\mathcal{C}_1$ and $\mathcal{C}_2$) having high errors. As before, the matrices are consistently permuted for visualization. The co-clustering algorithm also finds this permutation.

of word clusters and for two Bregman divergences—squared Euclidean distance and I-divergence. Performance is evaluated by the normalized mutual information of the document clusters with the true labels of the documents (Strehl and Ghosh, 2002). As in many of the other experiments, we note that co-clustering bases $\mathcal{C}_2$ and $\mathcal{C}_5$ are suitable for both divergences. In Figure 6, we compare the performances of $\mathcal{C}_2$ and $\mathcal{C}_5$ for both divergences, using the spherical k-means (SPKmeans) algorithm

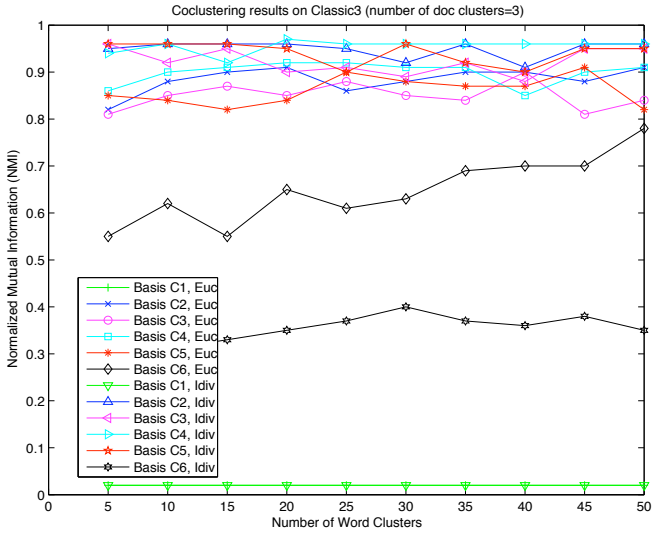


Figure 5: Co-clustering results from *CLASSIC3*—6 bases and 2 divergences. Bases $\mathcal{C}_2 - \mathcal{C}_5$ perform very well in getting back the hidden true labels. Basis $\mathcal{C}_1$ performs the worst as it has access to minimal amount of information. Interestingly, basis $\mathcal{C}_6$, in spite of having the maximal information, performs poorly according to NMI. Possibly $\mathcal{C}_6$ is overfitting, that is, finding some additional structure in the data that goes beyond what is needed to get the labels right. There is no significant difference between the two loss functions used.

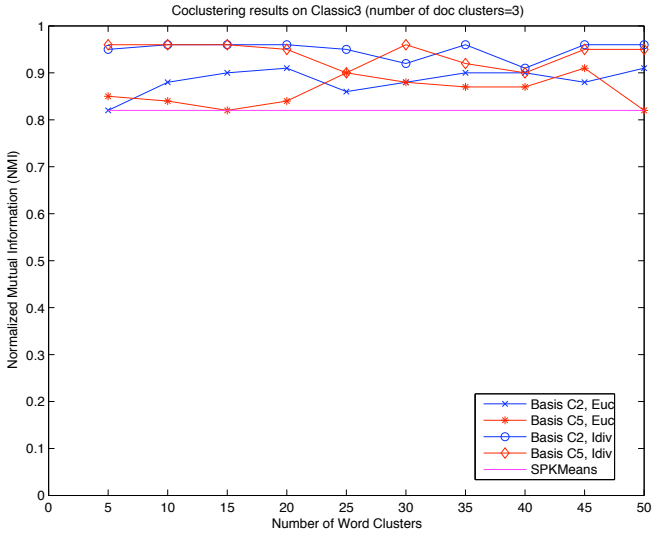


Figure 6: Co-clustering on *CLASSIC3*—Bases $\mathcal{C}_2$ and $\mathcal{C}_5$ using squared Euclidean distance and I-divergence compared with SPKmeans. The co-clustering results compare favorably to SPKmeans.

(Dhillon and Modha, 2001) as a benchmark. We note that the co-clustering algorithms, in particular the ones based on I-divergence, have very good performance for the entire range of word clusters. Our results are in agreement with similar results reported in the literature (Dhillon et al., 2003b).

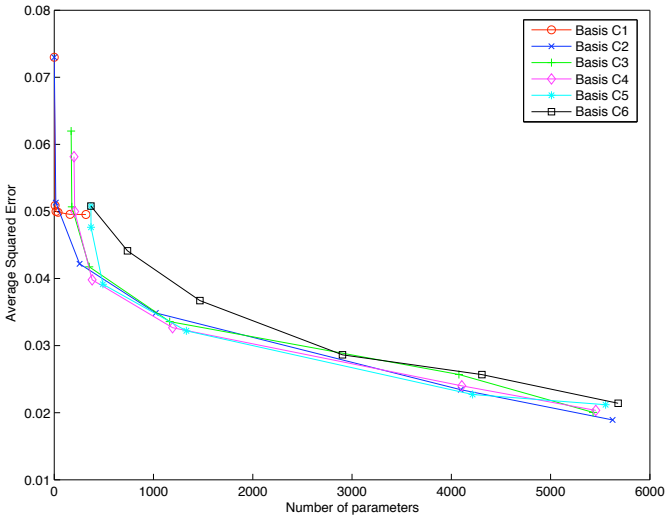


Figure 7: Approximation error (average squared error) on MovieLens data using squared Euclidean distance-based co-clustering. As expected, the error decreases with increasing number of parameters for all bases. For each basis, the number of parameters varies as a function of the number of row and column clusters that the co-clustering algorithm uses.

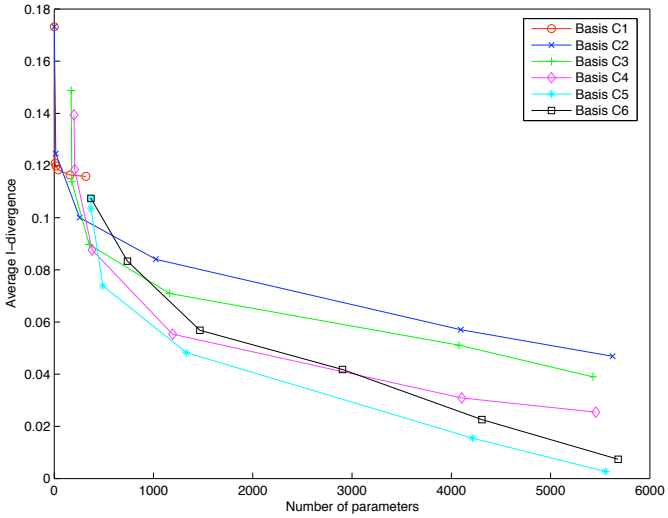


Figure 8: Approximation error (average I-divergence) on MovieLens data using I-divergence-based co-clustering. The error decreases with increasing number of parameters.

Bregman divergence	$k=l=1$	$k=l=2$	$k=l=12$	$k=l=32$	$k=l=64$	$k=l=75$
Squared Euclidean distance	0.7004	0.6816	0.6048	0.5547	0.4451	0.4052
I-divergence	0.7006	0.6824	0.6029	0.5573	0.4492	0.4080

Table 5: Mean absolute error (MAE) for reconstructing MovieLens data (all values) using co-clustering methods based on squared Euclidean distance and I-divergence and co-clustering basis $\mathcal{C}_5$.

6.2.3 USER-MOVIE RATING MATRICES

The other real-life data domain that we studied is that of movie recommender systems. The data matrices in this case consist of user ratings for various movies. For our experiments, we used the MovieLens data set (GroupLens) consisting of 100,000 ratings in the range 0-5 corresponding to 943 users and 1682 movies. To figure out the appropriate divergence and co-clustering basis for this data, we performed experiments using both squared Euclidean distance and I-divergence and various co-clustering bases with varying number of row and column clusters. For each case, the co-clustering was performed assuming uniform weights on the known ratings and zero weights for the unknown ones. The known ratings were then reconstructed using the MBI principle. Figures 7 and 8 show how the approximation error varies with the number of parameters for different co-clustering bases using squared Euclidean distance and I-divergence cost functions respectively. In the case of squared Euclidean distance-based co-clustering, we observe that $\mathcal{C}_2$ provides the best accuracy when an extremely low parameter approximation is required while $\mathcal{C}_2$-$\mathcal{C}_5$ are more suitable for moderately low parameter sizes. In the case of I-divergence-based co-clustering, $\mathcal{C}_5$ is better than the other bases over a wide range of parameter sizes. Further as Table 5 shows, both choices of Bregman divergence, that is, squared Euclidean distance and I-divergence, seem to provide similar performance in terms of the mean absolute error for $\mathcal{C}_5$.

6.3 Novel Applications of Bregman Co-clustering

We now briefly describe two novel applications of our Bregman co-clustering framework and illustrate these with specific real-life examples.

6.3.1 MISSING VALUE PREDICTION

Prediction of missing values is an important task encountered in a number of real-world domains such as recommender systems, bioinformatics, etc. For our experiments, we consider a collaborative filtering-based recommender system where the main task is to predict the preference of a given user for a given item using known preferences of the other users. One of the earliest and most popular approaches to solve this problem is by computing the Pearson correlation of each user with all other users based on the known preferences and predict the unknown rating by proportionately combining all the users' ratings. Based on the observation that the known ratings correspond to elements in a matrix and the missing ratings can be predicted using suitable low parameter approximations of the ratings matrix, a number of other collaborative filtering approaches based on matrix approximation methods such as SVD (Sarwar et al., 2000), and PLSI (Hofmann, 2004) have been proposed in recent years.

Following the same general intuition, we propose a mathematically well-motivated solution based on co-clustering. The main idea is to (i) assume that the ratings matrix has a low parameter structure involving properties of user and item clusters, (ii) deduce the relevant parameters using the

SVD	NNMF	CORR	$\mathcal{C}_2$	$\mathcal{C}_5$
0.7721 ± 0.0164	0.7636 ± 0.0186	0.8214 ± 0.0201	0.8733 ± 0.197	0.7608 ± 0.0211

Table 6: Mean absolute error on MovieLens data set for various collaborative filtering approaches. Number of row and column clusters for co-clustering (based on squared Euclidean distance and basis $\mathcal{C}_5$) and rank of SVD and NNMF is set to 5 and the number of neighbors in the correlation method was set to 50.

available ratings so that the desired loss function is minimized, and (iii) use a matrix reconstruction based on this structure for predicting the missing values. More specifically, in our co-clustering approach, we assume a low parameter structure by using the MBI principle so that the parameter learning can be readily performed using the Bregman co-clustering algorithm with a suitably weighted loss function (weight is uniform for known ratings, 0 otherwise). The missing values are then predicted using the reconstructed approximate matrix. Based on the results in Section 7.2.3, we consider low parameter structures corresponding to the bases $\mathcal{C}_2$ and $\mathcal{C}_5$. In case of $\mathcal{C}_2$, the use of the MBI principle implies that the user-item rating depends equals the average rating in the co-cluster whereas in $\mathcal{C}_5$, the user-item rating is a combination of the user-bias, item-bias and the average rating in the co-cluster.

For our experiments, we used the MovieLens data set (GroupLens) described earlier and the results reported are averaged over multiple runs of five-fold cross-validation with 80% of ratings as the training data and 20% of the ratings as the test data in each run.

Table 6 shows the mean absolute error (MAE) obtained using various existing collaborative filtering approaches (Sarwar et al., 2000; Hofmann, 2004; Resnick et al., 1994) as well as the co-clustering approach based on squared Euclidean distance. From the table, we note that the co-clustering method based on $\mathcal{C}_5$ provides accuracy comparable to that of the SVD and NNMF-based methods. The co-clustering approach also has significant benefits in terms of computational effort as the training time is linear in the number of known ratings and the missing value prediction is a constant time operation unlike in other approaches. The number of parameters in the compressed representation is also much lower in the case of co-clustering as compared to SVD, NNMF and correlation methods when the rank or neighborhood size is of the same order as the number of row and column clusters.

6.3.2 CO-CLUSTERING CATEGORICAL DATA MATRICES

The second data analysis task we consider involves co-clustering data matrices consisting of categorical values from a finite set. Examples of such data include (i) market-basket data matrices with users as rows and products as columns and the entries corresponding to preferred brands, and (ii) genomic data matrices with rows corresponding to patients and columns corresponding to various positions/loci of gene sequences (also referred to as single nucleotide polymorphisms) and matrix entries indicating the occupying allele (usually only 4 possible alleles for each location) (Lin and Altman, 2004). Though the matrix elements take a finite number of values, there is no natural ordering, which makes it impossible to directly map them to the set of reals $\mathbb{R}$ (except in the case of binary valued data) in order to perform co-clustering as in the case of co-occurrence matrices. However, it is straightforward to represent each of these categorical values using discrete distributions over the set of all possible values. For example, when the matrix elements take values in $\{A, B, C, D\}$, then A can be represented as the distribution $[1,0,0,0]$ while B can be represented as $[0,1,0,0]$ and so on. With this representation, each element of the data matrix is a member of the r-simplex where

Num. clusters	Co-clustering Code	Summary Statistics Code	Matrix Code (Co-clustering Cost)	Total
$k = l = 1$	0	32.4	4973.3 ± 30.8	5005.7 ± 30.8
$k = l = 5$	232.2	425.8 ± 4.7	2695.4 ± 47.6	3353.4 ± 52.3
$k = l = 50$	564.4	5000	0	5564.4

Table 7: Description length (in bits) for encoding matrix information. Summary statistic code is the number of bits for encoding the counts of the four possible values in each co-cluster given the co-clustering whereas the matrix code is description length of the actual matrix given the summary statistics and the co-clustering. Co-clustering was performed using relative entropy cost function and basis $\mathcal{C}_2$.

r denotes the number of possible categorical values. Defining the domain $\mathcal{S}$ of the matrix elements to be the r-simplex, we can now proceed to perform co-clustering on the categorical data matrix by choosing an appropriate Bregman divergence over $\mathcal{S}$ and a suitable co-clustering basis. Since elements of $\mathcal{S}$ correspond to probability distributions, a natural choice of distortion measure is the relative entropy (or KL-divergence) over the r-simplex. The co-clustering objective function in this case is given by

$$J(\rho,\gamma) = \sum_{u=1}^{m}\sum_{v=1}^{n} KL(\mathbf{z}_{uv}||\hat{\mathbf{z}}_{uv})$$

where $\mathbf{Z} = [\mathbf{z}_{uv}]$ is the original matrix, $\hat{\mathbf{Z}} = [\hat{\mathbf{z}}_{\mathbf{uv}}]$ is the MBI solution based on the co-clustering, and the elements $\mathbf{z}_{uv}$ and $\hat{\mathbf{z}}_{\mathbf{uv}}$ belong to the r-simplex. This co-clustering objective function is also exactly equal to the minimum achievable description length (in bits) required for a lossless encoding of the original matrix $\mathbf{Z}$ given the MBI solution $\hat{\mathbf{Z}}$. Hence, assuming that the cost of describing the co-clustering and the summary statistics depends only on the pre-specified number of row and column clusters, the Bregman co-clustering algorithm corresponding to the relative entropy-based cost function automatically seeks to find an optimal (minimum length) lossless code for the matrix. A recent paper (Chakrabarti et al., 2004) follows a similar co-clustering based approach using binary relative entropy and basis $\mathcal{C}_2$ for performing lossless coding of binary valued matrices.

To demonstrate the effectiveness of the co-clustering approach described above, we generated 10 artificial 50×50 matrices consisting of four categorical values $\{A,B,C,D\}$. For all the matrices, we assumed generative models corresponding to multinomial distributions over $\{A,B,C,D\}$ and co-clustering basis $\mathcal{C}_2$ with $k = l = 5$. The elements in each co-cluster were generated using a single multinomial distribution with a purity of about 0.8, that is, the most likely categorical value had a probability of 0.8 with the rest all being equally likely with probability 0.067. Each of these matrices was then co-clustered using the relative entropy-based cost function on a 4-simplex with $k = l = 5$. Table 7 shows a comparison of the description lengths for various choices of k and l using a three-step encoding protocol where we first encode the co-clustering, then the summary statistics, that is, counts of $\{A,B,C,D\}$ in each co-cluster, and finally the original matrix given the summary statistics and the co-clustering.

For encoding the co-clustering, we employ a naive scheme that involves specifying the row and column clusters for each row and column respectively. Since there are k row clusters and l column clusters, the total number of bits required is given by $m\log_2 k + n\log_2 l$, as shown in the second column of Table 7. Given this co-clustering, we then proceed to encode the summary statistics, that is, counts of $\{A,B,C,D\}$, corresponding to each co-cluster. First, we observe that for each co-cluster, the four counts have to be non-negative integers that sum up to the total size of the

particular co-cluster. Since the co-clustering already specifies the total size of all the co-clusters, it is sufficient to specify any three of the four counts. Further, information about the count of a particular categorical value reduces the number of possible choices for the rest of the counts. In particular, if $m_{\hat{u}}$ and $n_{\hat{v}}$ denote the number of rows and columns in row cluster $\hat{u}$ and column cluster $\hat{v}$ respectively, then the number of bits for encoding the first count (say that of A) is given by $\log_2(1+m_{\hat{u}}n_{\hat{v}})$ while the cost for the second count (say that of B) is given by $\log_2(1+m_{\hat{u}}n_{\hat{v}}-N_A)$ where N_A is the count of A. Similarly, the encoding cost for the third count is given by $\log_2(1+m_{\hat{u}}n_{\hat{v}}-N_A-N_B)$ where N_B denotes the count of B. Thus, the total number of bits for encoding the summary statistics in this case is given by[15]

$$\sum_{\hat{u}=1}^{k}\sum_{\hat{v}=1}^{l}\left(\log_2(1+m_{\hat{u}}n_{\hat{v}})+\log_2(1+m_{\hat{u}}n_{\hat{v}}-N_A)+\log_2(1+m_{\hat{u}}n_{\hat{v}}-N_A-N_B)\right).$$

The third column in Table 7 shows the above encoding cost for different choices of k and l. When $k=l=50$, the co-clusters are all singleton sets so that it is sufficient to specify the single categorical value in each co-cluster. Since there are 4 possible values and mn co-clusters, the encoding cost in this case equals $2mn=5000$ bits.

The final step is to specify the original matrix given the summary statistics and the co-clustering and as mentioned earlier, the description length in this case is identical to the co-clustering objective function, which is shown in the fourth column of Table 7. When $k=l=50$, the description length is zero since the summary statistics fully specify the original matrix. From the table, we observe that with an optimal choice of row and column clusters, one can obtain an efficient lossless compression of matrix consisting of finite categorical values. On examining the resulting co-clusters, we find that most of them are quite homogeneous as well.

7. Related Work

We have discussed several related methods that have appeared in the literature throughout the paper. We have also discussed existing as well as novel applications of co-clustering in Section 6. In this section, we briefly review further connections and contrast our work to the existing literature. Our current work is related to several active areas of research, namely co-clustering, matrix approximation, learning based on Bregman divergences and convex optimization. In particular, our formulation of a general co-clustering problem was motivated by earlier work on co-clustering and matrix approximation (Dhillon et al., 2003b).

Co-clustering has been a topic of much interest in the recent years because of its applications to problems such as microarray analysis (Cheng and Church, 2000; Cho et al., 2004), natural language processing (Li and Abe, 1998; Freitag, 2004; Rohwer and Freitag, 2004), recommender systems (Hofmann, 2004) and text, image and speech analysis (Dhillon et al., 2003b; Takamura and Matsumoto, 2003; Qiu, 2004; Cai et al., 2005). Currently, there exist many formulations of the co-clustering problem such as the hierarchical co-clustering model (Hartigan, 1972), the sequential bi-clustering model (Cheng and Church, 2000) that involves finding the best co-clusters one at a time, and the spectral co-clustering model (Dhillon, 2001; Kluger et al., 2003) that involves partitioning a bipartite graph with vertices corresponding to the rows and columns. The reader

15. It is possible to have a more efficient encoding scheme by choosing an ordering of the categorical values $\{A,B,C,D\}$ that is likely to lead to the lowest number of bits, but does not make a significant difference in the current experiment as all the categorical values have nearly equal counts over the entire matrix.

should refer to Madeira and Oliveira (2004) for an extensive survey on various co-clustering models proposed in literature and their applications. Recently, there have also been other clustering formulations (Bekkerman et al., 2005; Gao et al., 2005) that are closely connected to co-clustering, but involve simultaneous clustering of multiple sets of related entities. In our current work, we focus on the partitional co-clustering formulation, first introduced by Hartigan (1972), where the objective is to partition the data matrix into $k \times l$ non-overlapping co-clusters where the quality of co-clusters is determined in terms of an appropriate cost function. Recently, quite a few algorithms (Cho et al., 2004; Dhillon et al., 2003b; Li and Abe, 1998; Li, 2005) have been proposed to address the above partitional problem for various cost functions based on squared Euclidean distance and I-divergence. One of the objectives of the current work is to generalize these algorithms to a large set of loss functions based on Bregman divergences.

Partitional co-clustering can also be readily viewed as an efficient low parameter matrix approximation technique as each homogeneous co-cluster can be accurately approximated by a small number of parameters. In fact, the flexibility to approximate a given data matrix in terms of a wide range of loss functions subject to a large class of constraints makes the co-clustering methods more widely applicable than traditional matrix approximation methods based on singular value decomposition. In particular, classical singular value decomposition (SVD) (Papadimitriou et al., 1998) based approaches to matrix approximation are quite often inappropriate for certain data matrices such as co-occurrence and contingency tables as singular vectors can have negative entries and the contributions of the component vectors in the approximation matrix are not localized. Both these issues make the interpretation of SVD-based approximations difficult, which is necessary for data mining purposes. To address these and related issues, techniques involving non-negativity constraints (Lee and Seung, 2001) using KL-divergence as the approximation loss function (Hofmann and Puzicha, 1998; Lee and Seung, 2001) have been proposed. However, these approaches apply to special types of matrices such as doubly stochastic and fully non-negative matrices. A general formulation that is both interpretable and applicable to various classes of matrices is often necessary for a number of real-life applications and the proposed Bregman co-clustering formulation attempts to address this requirement.

Co-clustering involving constraints on conditional expectations gives rise to theoretically elegant models with wide range of practical applicability since key summary statistics can be naturally preserved. Several co-clustering algorithms (Dhillon et al., 2003b; Cho et al., 2004) that have been proposed in the recent years can be derived from conditional expectation-based constraints. Conditional expectation constrained co-clustering, along with its demonstrated connection to the widely used maximum entropy principle (Jaynes, 1957; Cover and Thomas, 1991) and conditional independence based models (Hofmann and Puzicha, 1998), provides a strong foundation for a unified analysis and design of unsupervised learning algorithms.

Recent research (Azoury and Warmuth, 2001; Banerjee et al., 2005b) has shown that several results involving the KL-divergence and the squared Euclidean distance are in fact based on certain convexity properties and hence, generalize to all Bregman divergences. This intuition motivated us to consider co-clustering based on Bregman divergences. Further, the similarities between the maximum entropy and the least squares principles (Csiszár, 1991) prompted us to explore a more general minimum Bregman information principle for all Bregman divergences.

It is important to note that most clustering and co-clustering techniques based on the alternate minimization scheme can be obtained as special cases of the Bregman co-clustering algorithm. For example, information-theoretic co-clustering (Dhillon et al., 2003b) corresponds to the case where

the constraint set is $\mathcal{C}_5$ and the Bregman divergence is KL-divergence. Similarly, the minimum sum-squared residue co-clustering algorithms (Cho et al., 2004) correspond to the cases where the constraint sets are $\mathcal{C}_2$ and $\mathcal{C}_6$ respectively while the Bregman divergence is the squared Euclidean distance. The one-sided Bregman clustering algorithms (Banerjee et al., 2005b) are also a special case with $l = n$.

8. Discussion

In this paper, we have presented a general theory of partitional Bregman co-clustering. Our analysis leads to a unified treatment of several known co-clustering methods that are being successfully used in the literature. Further, the analysis gives rise to an entire class of new co-clustering algorithms based on particular choices of the Bregman divergence and the set of summary statistics to be preserved. We have provided a meta-algorithm for the general case, and have demonstrated how to instantiate the algorithm for specific choices of divergences and statistics. There are several potential benefits to our formulation and analysis:

- Since our co-clustering formulation allows loss functions corresponding to all Bregman divergences, the technique now becomes applicable to practically all types of data matrices. The particular choice of the divergence function can be determined by (i) the data type, for example, if the data corresponds to joint probability distributions, relative entropy is an appropriate choice as the divergence function; (ii) the appropriate noise model, for example, Euclidean distance is appropriate for Gaussian noise, Itakura-Saito is appropriate for Poisson noise, etc.; or (iii) domain knowledge/requirements, for example, sparsity of the original matrix can be preserved using I-divergence.

- Our formulation allows approximation models of various complexities depending on the statistics that are constrained to be preserved. There are two key advantages to this flexibility. First, preserving summary statistics of the data may be crucial for some applications as well as important for subsequent analysis. Since the statistics preserving property is intrinsic to our approach, it is readily applicable to domains where summary statistics are important. Second, the multiple sets of preserved statistics may enable discovery of different structural patterns in the data.

- We have proposed and extensively used the minimum Bregman information (MBI) principle as a generalization of the maximum entropy principle. Since the approximations obtained from the MBI principle extend some of the desirable properties of the maximum entropy models to settings where a Bregman divergence other than the relative entropy is more appropriate, we get a new class of statistical modeling techniques that are applicable to more general settings. The MBI principle has potential applications beyond the co-clustering approximations considered in this paper.

- While the central focus of this paper has been to obtain good co-clusterings using matrix approximation error to evaluate goodness, as a by-product, we have obtained a general class of fast matrix approximation techniques with several desirable properties. In particular, the approximation techniques can work with general divergence functions and preserve desirable statistical properties of the original data. The approximations are based on co-clustering, and are expected to have different behavior from the spectral methods typically employed for

matrix approximations. Further, since the methods are iterative and do not involve eigenvalue computations, they are significantly faster than existing methods and hence, more appropriate for large data matrices.

In this paper, our analysis of co-clustering has focused on data matrices that represent the relationship between two entities. Many emerging application domains collect data on relationships between multiple entities, which can be represented as a tensor. Our proposed co-clustering technique can be extended to this general setting involving tensors unlike other methods that are specific to matrices. It will be worthwhile to investigate how the extensions of co-clustering to tensor data perform compared to existing techniques. In particular, several practical problem domains have known statistical dependency relationships between the several entities of interest. One of the key challenges of an extension of co-clustering to such multi-entity relational domains is to come up with efficient algorithms that take advantage of the statistical relationships and maintain succinct representations of the entities and their relationships.

Acknowledgments

We would like to thank Hyuk Cho and the reviewers for their detailed comments and suggestions that significantly improved the paper. We would like to thank John Lafferty and an anonymous reviewer for pointing out the connection of our projection results to existing literature on Bregman duality. The research was partly supported by NSF grants IIS-0307792, CCF-0431257, III-0713142, NSF Career Award ACI-0093404, and NSF ITR award IIS-0325116.

Appendix A. Information Theoretic Co-clustering

Proof of Lemma 1 Let p' be any distribution that satisfies (4) and (5), and let q be as in (3). Consider

$$\begin{aligned}
KL(p'||q) &= \sum_{\hat{x}}\sum_{\hat{y}}\sum_{x\in\hat{x}}\sum_{y\in\hat{y}} p'(x,y)\log\frac{p'(x,y)}{q(x,y)} \\
&= -H(p') - \sum_{\hat{x}}\sum_{\hat{y}}\sum_{x\in\hat{x}}\sum_{y\in\hat{y}} p'(x,y)\left(\log p(\hat{x},\hat{y}) + \log p(x|\hat{x}) + \log p(y|\hat{y})\right) \\
&= -H(p') - \sum_{\hat{x}}\sum_{\hat{y}} p(\hat{x},\hat{y})\log p(\hat{x},\hat{y}) - \sum_{\hat{x}}\sum_{x\in\hat{x}} p(x)\log p(x|\hat{x}) - \sum_{\hat{y}}\sum_{y\in\hat{y}} p(y)\log p(y|\hat{y}) \\
&= -H(p') - \sum_{\hat{x}}\sum_{\hat{y}} p(\hat{x},\hat{y})\left(\sum_{x\in\hat{x}} p(x|\hat{x})\right)\left(\sum_{y\in\hat{y}} p(y|\hat{y})\right)\log p(\hat{x},\hat{y}) \\
&\qquad - \sum_{\hat{x}}\sum_{x\in\hat{x}} q(x)\log p(x|\hat{x}) - \sum_{\hat{y}}\sum_{y\in cy} q(y)\log p(y|\hat{y})
\end{aligned}$$

$$
\begin{aligned}
&= -H(p') - \sum_{\hat{x}}\sum_{\hat{y}}\sum_{x\in\hat{x}}\sum_{y\in\hat{y}} p(x|\hat{x})p(\hat{x},\hat{y})p(y|\hat{y})\log p(\hat{x},\hat{y}) \\
&\qquad - \sum_{\hat{x}}\sum_{x\in\hat{x}}\left(\sum_{\hat{y}}\sum_{y\in\hat{y}} q(x,y)\right)\log p(x|\hat{x}) - \sum_{\hat{y}}\sum_{y\in\hat{y}}\left(\sum_{\hat{x}}\sum_{x\in\hat{x}} q(x,y)\right)\log p(y|\hat{y}) \\
&= -H(p') - \sum_{\hat{x}}\sum_{\hat{y}}\sum_{x\in\hat{x}}\sum_{y\in\hat{y}} q(x,y)\log(p(\hat{x},\hat{y})p(x|\hat{x})p(y|\hat{y})) \\
&= -H(p') + H(q).
\end{aligned}
$$

Since $KL(p'||q) \geq 0$, we have $H(q) \geq H(p')$. ∎

Appendix B. Some Properties of Bregman Divergences

We present some useful properties of Bregman divergences and Bregman information that we use in our analysis in the paper.

Lemma 12 (Bregman 1967; Censor and Zenios 1998) *For any Bregman divergence $d_\phi : S \times \mathrm{int}(S) \mapsto \mathbb{R}_+$ and $z_1 \in \mathcal{S}$ and $z_2, z_3 \in \mathrm{ri}(\mathcal{S})$, the following three-point property holds:*

$$d_\phi(z_1, z_3) = d_\phi(z_1, z_2) + d_\phi(z_2, z_3) + \langle z_1 - z_2, \nabla\phi(z_2) - \nabla\phi(z_3)\rangle \ .$$

Theorem 7 (Banerjee et al. 2005a) *For any Bregman divergence $d_\phi : S \times \mathrm{ri}(S) \mapsto \mathbb{R}_+$, random variable $Z \sim w(z),\ z \in \mathcal{Z} \subseteq S$ and sub-σ algebra $\mathcal{G}$ for Z, the conditional expectation $E[Z|\mathcal{G}]$ is the optimal predictor of Z among all $\mathcal{G}$ measurable random variables in terms of Bregman divergence, that is,*

$$E[Z|\mathcal{G}] = \operatorname*{argmin}_{Z' \in \mathcal{G}} d_\phi(Z, Z') \ .$$

Lemma 13 (Banerjee et al. 2005b) *For any Bregman divergence $d_\phi : S \times \mathrm{ri}(S) \mapsto \mathbb{R}_+$, random variable $Z \sim w(z),\ z \in \mathcal{Z} \subseteq S$ and any constant $c \in \mathrm{int}(S)$, the following decomposition holds:*

$$E[d_\phi(Z, c)] = E[d_\phi(Z, E[Z])] + d_\phi(E[Z], c) \ .$$

Lemma 14 (Banerjee et al. 2005b) *For any Bregman divergence $d_\phi : S \times \mathrm{ri}(S) \mapsto \mathbb{R}_+$ and random variable $Z \sim w(z),\ z \in \mathcal{Z} \subseteq S$, the optimal constant predictor of Z in terms of Bregman divergence is its expectation, that is,*

$$E[Z] = \operatorname*{argmin}_{c} E[d_\phi(Z, c)] \ .$$

Appendix C. Block Average Co-clustering

Proof of Theorem 1 Consider the Lagrangian $J(Z', \Lambda)$ of the MBI problem:

$$\begin{aligned} J(Z', \Lambda) &= I_\phi(Z') + \sum_{\hat{u},\hat{v}} \lambda_{\hat{u}\hat{v}} (E[Z'|\hat{u},\hat{v}] - E[Z|\hat{u},\hat{v}]) \\ &\overset{(a)}{=} E[\phi(Z')] - \phi(E[Z']) + \sum_{\hat{u},\hat{v}} \lambda_{\hat{u}\hat{v}} (E[Z'|\hat{u},\hat{v}] - E[Z|\hat{u},\hat{v}]) \\ &\overset{(b)}{=} E[\phi(Z')] - \phi(E[Z]) + \sum_{\hat{u},\hat{v}} \lambda_{\hat{u}\hat{v}} (E[Z'|\hat{u},\hat{v}] - E[Z|\hat{u},\hat{v}]) \ , \end{aligned}$$

where $\lambda_{\hat{u}\hat{v}}$ is the Lagrange multiplier corresponding to the constraint $E[Z'|\hat{u},\hat{v}] - E[Z|\hat{u},\hat{v}] = 0$ Further, (a) follows from Lemma 2 and (b) follows since $E[Z'] = E_{\hat{U},\hat{V}}[E[Z'|\hat{U},\hat{V}]] = E[Z]$.

Rewriting the Lagrangian in terms of matrix elements $\{\{z'_{uv}\}_{u=1}^m\}_{v=1}^n$ corresponding to Z', we obtain

$$J(Z', \Lambda) = \sum_{u=1}^{m} \sum_{v=1}^{n} w_{uv} (\phi(z'_{uv}) - \phi(\bar{z})) + \sum_{\hat{u},\hat{v}} \lambda_{\hat{u}\hat{v}} \frac{1}{w_{\hat{u}\hat{v}}} \sum_{\substack{u:\rho(u)=\hat{u} \\ \gamma(v)=\hat{v}}} w_{uv} (z'_{uv} - z_{uv}) \ , \tag{29}$$

where $w_{\hat{u}\hat{v}} = \sum_{u:\rho(u)=\hat{u}, v:\gamma(v)=\hat{v}} w_{uv}$ and $\bar{z} = \sum_{u=1}^m \sum_{v=1}^n w_{uv} z'_{uv} = \sum_{u=1}^m \sum_{v=1}^n w_{uv} z_{uv}$.

To obtain the optimal solution $\hat{Z}_A$, we consider the first order necessary conditions, that is, set the partial derivatives with respect to the matrix elements and the Lagrange multipliers. Taking partial derivatives with respect to $\lambda_{\hat{u},\hat{v}}$, we obtain

$$\frac{1}{w_{\hat{u}\hat{v}}} \sum_{\substack{u:\rho(u)=\hat{u} \\ \gamma(v)=\hat{v}}} w_{uv} (z'_{uv} - z_{uv}) = 0 \qquad \forall \hat{u}, \hat{v}, \tag{30}$$

that is, $E[Z|\hat{u},\hat{v}] = E[Z'|\hat{u},\hat{v}]$ for all $[\hat{u}]_1^k$ and $[\hat{v}]_1^l$.

Now, setting partial derivatives of (29) with respect to z'_{uv} equal to 0, we get

$$w_{uv} \nabla\phi(z'_{uv}) - w_{uv} \nabla\phi(\bar{z}) + \lambda_{\hat{u}\hat{v}} \frac{w_{uv}}{w_{\hat{u}\hat{v}}} = 0 \ ,$$

where $\hat{u} = \rho(u)$ and $\hat{v} = \gamma(v)$. Since $w_{uv} \in \mathbb{R}_+$ and $\bar{z} = E[Z] = E[Z']$, the optimal solution $\hat{Z} = \hat{Z}_A$ has the form

$$\hat{z}_{uv} = \nabla\phi^{(-1)} \left(\nabla\phi(E[Z]) - \frac{\lambda^*_{\hat{u}\hat{v}}}{w_{\hat{u}\hat{v}}} \right), \ \ \hat{u} = \rho(u), \hat{v} = \gamma(v), \tag{31}$$

where $\lambda^*_{\hat{u}\hat{v}}$ corresponds to the optimal Lagrange multiplier. Note that the right hand side is constant for a given $(\hat{u},\hat{v})$. Substituting (31) into (30) gives us

$$E[Z|\hat{u},\hat{v}] = \nabla\phi^{(-1)} \left(\nabla\phi(E[Z]) - \frac{\lambda^*_{\hat{u}\hat{v}}}{w_{\hat{u}\hat{v}}} \right).$$

Hence, the only solution satisfying the first order necessary conditions is $\hat{z}_{uv} = E[Z|\hat{u},\hat{v}], \ \forall u, v$, that is, $\hat{Z}_A = E[Z|\hat{U},\hat{V}]$. The existence and uniqueness of $\hat{Z}_A$ follow from the strict convexity of ϕ. ∎

Proof of Lemma 3 Using the three point property (Lemma 12) and taking expectations, for any $Z' \in \mathcal{S}_A$ and $Z'' \in \mathcal{S}_B$, we have

$$E[d_\phi(Z', Z'')] = E[d_\phi(Z', \hat{Z}_A)] + E[d_\phi(\hat{Z}_A, Z'') + E[\langle Z' - \hat{Z}_A, \nabla\phi(\hat{Z}_A)\rangle] - E[\langle Z' - \hat{Z}_A, \nabla\phi(Z'')\rangle] \ .$$

We now argue that the last two terms in the above expression vanish to give the desired result. From Theorem 1, we note that $\hat{Z}_A = E[Z|\hat{U}, \hat{V}]$ so that

$$E[\langle Z' - \hat{Z}_A, \nabla\phi(\hat{Z}_A)] = E_{\hat{U},\hat{V}}[\langle E[Z'|\hat{U}, \hat{V}] - E[\hat{Z}_A|\hat{U}, \hat{V}], \nabla\phi(\hat{Z}_A)\rangle] = 0 \ ,$$

since $\hat{Z}_A$ is a constant given $(\hat{U}, \hat{V})$ and has the same co-cluster means as $Z' \in \mathcal{S}_A$.

To show that the last term $E[\langle Z' - \hat{Z}_A, \nabla\phi(Z'')\rangle]$ also vanishes, we note that for any $Z'' \in S_B$, $\nabla\phi(Z'') = g(E[Z|\hat{U}, \hat{V}])$ for some deterministic function g so that

$$\begin{aligned} E[\langle Z' - \hat{Z}_A, \nabla\phi(Z'')\rangle] &= E[\langle Z' - \hat{Z}_A, g(E[Z|\hat{U}, \hat{V}])\rangle] \\ &= E_{\hat{U},\hat{V}}[\langle (E[Z'|\hat{U}, \hat{V}] - E[\hat{Z}_A|\hat{U}, \hat{V}]), g(E[Z|\hat{U}, \hat{V}])\rangle] \ = \ 0 \ , \end{aligned}$$

since $\hat{Z}_A$ and Z' both belong to $\mathcal{S}_A$ and hence, have the same co-cluster means. ∎

Proof of Theorem 2 From Lemma 7, we observe that for any $Z' \in \mathcal{S}_A$ and $Z'' \in \mathcal{S}_B$,

$$E[d_\phi(Z', Z'')] = E[d_\phi(Z', \hat{Z}_A)] + E[d_\phi(\hat{Z}_A, Z'') \ .$$

Hence, for a given $Z'' \in \mathcal{S}_B$ and any $Z' \in \mathcal{S}_A$, $E[d_\phi(Z', Z'')] \geq E[d_\phi(\hat{Z}_A, Z'')]$, with equality only when $Z' = \hat{Z}_A$. Since $\hat{Z}_A \in \mathcal{S}_A$, this implies that

$$\hat{Z}_A = \operatorname*{argmin}_{Z' \in \mathcal{S}_A} E[d_\phi(Z', Z'')], \ \ \forall Z'' \in \mathcal{S}_B \ .$$

Similarly, for a given $Z' \in \mathcal{S}_A$ and any $Z'' \in \mathcal{S}_A$, $E[d_\phi(Z', Z'')] \geq E[d_\phi(Z', \hat{Z}_A)]$ with equality only when $Z'' = \hat{Z}_A$. Since $\hat{Z}_A \in \mathcal{S}_B$ as well, we obtain the second part of the result, that is,

$$\hat{Z}_A = \operatorname*{argmin}_{Z'' \in \mathcal{S}_B} E[d_\phi(Z', Z'')], \ \ \forall Z' \in \mathcal{S}_A \ . \ ∎$$

Proof of Lemma 5 By definition,

$$\begin{aligned} E[d_\phi(Z, \hat{Z}^{new})] &= E[\phi(Z) - \phi(\hat{Z}^{new}) - \langle Z - \hat{Z}^{new}, \nabla\phi(\hat{Z}^{new})\rangle] \\ &\overset{(a)}{=} E[\phi(Z) - \phi(\hat{Z}^{new})] \\ &= E[\phi(Z) - \phi(\tilde{Z}) - \langle Z - \tilde{Z}, \nabla\phi(\tilde{Z})\rangle] - E[\phi(\hat{Z}^{new}) - \phi(\tilde{Z}) - \langle Z - \tilde{Z}, \nabla\phi(\tilde{Z})\rangle] \\ &= E[d_\phi(Z, \tilde{Z})] - E[d_\phi(\hat{Z}^{new}, \tilde{Z})] + E[\langle Z - \hat{Z}^{new}, \nabla\phi(\tilde{Z})\rangle] \\ &\overset{(b)}{=} E[d_\phi(Z, \tilde{Z})] - E[d_\phi(\hat{Z}^{new}, \tilde{Z})] \\ &\leq E[d_\phi(Z, \tilde{Z})] \ , \end{aligned}$$

where (a) follows since $\hat{Z}^{new} \in S_A$ and $\hat{Z}^{new} \in S_B$ so that taking conditional expectations over $E[Z|\hat{U}, \hat{V}]$ makes the last term zero and (b) follows since $\nabla\phi(\tilde{Z})$ remains unchanged given $(\hat{U}, \hat{V})$ corresponding to $\hat{Z}^{new}$, and $E[\hat{Z}^{new}|\hat{U}, \hat{V}] = E[Z|\hat{U}, \hat{V}]$, thus making the last term vanish. ∎

Appendix D. Analysis of the General Case

Proof of Theorem 3 In order to identify the various matrix approximation schemes, we determine the subsets $\mathcal{C} \subseteq \Gamma_2$ that satisfy conditions (a) and (b). First, observe that $E[Z|U_\emptyset, V_\emptyset] = E[Z]$ and $E[Z|U,V] = Z$. Since $E[Z] = E[Z|U_\emptyset, V_\emptyset]$ can be obtained from every other conditional expectation $E[Z|C], C \in \Gamma_2$, and $Z = E[Z|U,V]$ determines every other conditional expectation, condition (b) implies that the pairs $\{U_\emptyset, V_\emptyset\}$ and $\{U,V\}$ cannot occur in combination with any other. As these pairs do not contain $\hat{U}$ or $\hat{V}$, we only need to consider combinations of the remaining members of Γ_2.

Further, we note that if there are two pairs $\mathcal{G}_1, \mathcal{G}_2 \in \mathcal{C},\ \mathcal{G}_1 \neq \mathcal{G}_2$ such that $\hat{U} \in \mathcal{G}_1$ and $\hat{U} \in \mathcal{G}_2$, then either $E[Z|\mathcal{G}_1]$ subsumes $E[Z|\mathcal{G}_2]$ or vice versa depending on the granularity of the column random variables in $\mathcal{G}_1$ and $\mathcal{G}_2$. A similar observation holds for $\hat{V}$. Hence, condition (b) implies that each non-trivial combination $\mathcal{C} \subseteq \Gamma_2$ should contain *exactly one* pair (possibly the same) that contains $\hat{U}$ and $\hat{V}$. Using the above observation, we enumerate the various possible cases as follows:

case 1: $\{\hat{U}, V_\emptyset\} \in \mathcal{C}$. $\mathcal{C}$ should also contain a pair containing $\hat{V}$, which can only be $\{U_\emptyset, \hat{V}\}$ since every other eligible pair $\in \Gamma_2$ uniquely determines $\{\hat{U}, V_\emptyset\}$ so that inclusion of any other pair leads to a violation of condition (b). Therefore, the only possible combination in this case is $\{\{\hat{U}, V_\emptyset\}, \{U_\emptyset, \hat{V}\}\}$.

case 2: $\{\hat{U}, \hat{V}\} \in \mathcal{C}$. Since condition (a) is already satisfied, we only need to identify the pairs in Γ_2 that can be included in $\mathcal{C}$ without violating condition (b), that is, pairs for which the row random variable is of higher granularity than $\hat{U}$ and the column random variable is of lower granularity than $\hat{V}$ or vice versa, which leads to two possibilities—$\{U, V_\emptyset\}$ and $\{U_\emptyset, V\}$. Hence, there are four combinations corresponding to the cases where we include neither of the pairs, exactly one of the pairs and both of them, that is,

$$\{\{\hat{U}, \hat{V}\}\}, \quad \{\{\hat{U}, \hat{V}\}, \{U\}, \{V\}\}, \quad \{\{\hat{U}, \hat{V}\}, \{U, V_\emptyset\}\}, \quad \text{and} \quad \{\{\hat{U}, \hat{V}\}, \{U_\emptyset, V\}\}.$$

case 3: $\{\hat{U}, V\} \in \mathcal{C}$. $\mathcal{C}$ should also contain a pair containing $\hat{V}$, which can only be $\{U, \hat{V}\}$ since every other eligible pair $\in \Gamma_2$ is subsumed by $\{\hat{U}, V\}$. Therefore, the only possible combination in this case is $\{\{\hat{U}, V\}, \{U, \hat{V}\}\}$.

Ignoring $U_\emptyset$ and $V_\emptyset$ since they are constant random variables and putting together all the different possible bases, we obtain the desired result. ∎

Proof of Theorem 4 Consider the Lagrangian $J(Z', \Lambda)$ of the MBI problem:

$$\begin{aligned} J(Z', \Lambda) &= I_\phi(Z') + \sum_{r=1}^{s} E_{\mathcal{G}_r}\left[\frac{\Lambda_{\mathcal{G}_r}}{w_{G_r}}(E[Z'|\mathcal{G}_r] - E[Z|\mathcal{G}_r])\right] \\ &= E[\phi(Z')] - \phi(E[Z']) + \sum_{r=1}^{s} E_{\mathcal{G}_r}\left[\frac{\Lambda_{\mathcal{G}_r}}{w_{G_r}}(E[Z'|\mathcal{G}_r] - E[Z|\mathcal{G}_r])\right], \end{aligned}$$

where $\Lambda_{\mathcal{G}_r}$ is a deterministic function of the random variable $\mathcal{G}_r$ and equals the appropriate Lagrange multiplier when $\mathcal{G}_r$ is specified. The Lagrange dual, $L(\Lambda) = \inf_{Z'} J(Z', \Lambda)$, is concave in Λ. By maximizing the Lagrange dual, we get the optimal Lagrange multipliers, that is, $\Lambda^* = \{\Lambda^*_{\mathcal{G}_r}\} =$

$\text{argmax}_\Lambda L(\Lambda)$. Substituting Λ^* into the first order necessary conditions corresponding to the minimizer $\hat{Z}_A$, we get

$$\begin{aligned} \nabla\left(E[\phi(\hat{Z}_A)] - \phi(E[\hat{Z}_A]) + \sum_{r=1}^{s} E_{G_r}\left[\frac{\Lambda^*_{\mathcal{G}_r}}{w_{\mathcal{G}_r}}(E[\hat{Z}_A|\mathcal{G}_r] - E[Z|\mathcal{G}_r])\right]\right) = 0, \\ \implies \quad \nabla\phi(\hat{Z}_A) \;=\; \nabla\phi(E[Z]) - \sum_{r=1}^{s}\frac{\Lambda^*_{\mathcal{G}_r}}{w_{\mathcal{G}_r}}, \end{aligned} \tag{32}$$

where $w_{\mathcal{G}_r}$ is the measure corresponding to $\mathcal{G}_r$ and $E[\hat{Z}_A] = E[Z]$. Rearranging terms proves the first part of the theorem. The existence and uniqueness of $\hat{Z}_A$ follow from the strict convexity of ϕ. ■

Proof of Lemma 6 From Theorem 4, we note that

$$\hat{Z} = (\nabla\phi)^{(-1)}\left(\nabla\phi(E[Z]) - \sum_{r=1}^{s}\frac{\Lambda^*_{\mathcal{G}_r}}{w_{\mathcal{G}_r}}\right),$$

where $\mathcal{C} = \{\mathcal{G}_r\}_{r=1}^{s}$ and $\Lambda^*_{\mathcal{G}_r}$ are the optimal Lagrange multipliers corresponding to the constraints $E[Z|\mathcal{G}_r] = E[\hat{Z}|\mathcal{G}_r]$. Now, by definition,

$$\begin{aligned} E[d_\phi(Z,\hat{Z})] &= E[\phi(Z) - \phi(\hat{Z}) - \langle Z-\hat{Z}, \nabla\phi(\hat{Z})\rangle] \\ &= E[\phi(Z) - \phi(\hat{Z})] - E[\langle Z-\hat{Z}, (\nabla\phi(E[Z]) - \sum_{r=1}^{s}\frac{\Lambda^*_{\mathcal{G}_r}}{w_{\mathcal{G}_r}})\rangle] \\ &= E[\phi(Z) - \phi(\hat{Z})] - E[\langle Z-\hat{Z}, \nabla\phi(E[Z])\rangle] + \sum_{r=1}^{s} E[\langle Z-\hat{Z}, \frac{\Lambda^*_{\mathcal{G}_r}}{w_{\mathcal{G}_r}}\rangle] \\ &\overset{(a)}{=} E[\phi(Z) - \phi(\hat{Z})] + \sum_{r=1}^{s} E_{\mathcal{G}_r}[\langle E[Z|\mathcal{G}_r] - E[\hat{Z}|\mathcal{G}_r], \frac{\Lambda^*_{\mathcal{G}_r}}{w_{\mathcal{G}_r}}\rangle] \\ &\overset{(b)}{=} E[\phi(Z) - \phi(\hat{Z})] \\ &\overset{(c)}{=} E[\phi(Z) - \phi(E[Z])] - E[\phi(\hat{Z}) - \phi(E[\hat{Z}])] \\ &\overset{(d)}{=} E[\phi(Z) - \phi(E[Z]) - \langle Z - E[Z], \nabla\phi(E[Z])\rangle] \\ &\quad -E[\phi(\hat{Z}) - \phi(E[\hat{Z}]) - \langle \hat{Z} - E[\hat{Z}], \nabla\phi(E[\hat{Z}])\rangle] \\ &= I_\phi(Z) - I_\phi(\hat{Z}), \end{aligned}$$

where (a) and (c) follow since $E[Z] = E[\hat{Z}]$, (b) follows since $E[Z|\mathcal{G}_r] = E[\hat{Z}|\mathcal{G}_r]$, $\forall \mathcal{G}_r \in \mathcal{C}$, and (d) follows since $E[\langle Z - E[Z], \nabla\phi(E[Z])\rangle] = 0$ and $E[\langle \hat{Z} - E[\hat{Z}], \nabla\phi(E[\hat{Z}])] = 0$. ■

Proof of Lemma 7 Using the three point property (Lemma 12) and taking expectations, for any $Z' \in \mathcal{S}_A$ and $Z'' \in \mathcal{S}_B$, we have

$$E[d_\phi(Z',Z'')] = E[d_\phi(Z',\hat{Z}_A)] + E[d_\phi(\hat{Z}_A,Z'') + E[\langle Z'-\hat{Z}_A, \nabla\phi(\hat{Z}_A)\rangle] - E[\langle Z'-\hat{Z}_A, \nabla\phi(Z'')\rangle]\,.$$

We now argue that the last two terms in the expression vanish to give the desired result. Note that since $\hat{Z}_A$ and $Z' \in \mathcal{S}_A$, we have $E[Z|\mathcal{G}_r] = E[\hat{Z}_A|\mathcal{G}_r] = E[Z'|\mathcal{G}_r],\ \forall \mathcal{G}_r \in \mathcal{C}$. By (32),

$$\begin{aligned} E[\langle Z' - \hat{Z}_A, \nabla\phi(\hat{Z}_A)\rangle] &= E[\langle Z' - \hat{Z}_A, (\nabla\phi(E[Z]) - \sum_{r=1}^{s} \frac{\Lambda^*_{\mathcal{G}_r}}{w_{\mathcal{G}_r}})\rangle] \\ &= \langle E[Z' - \hat{Z}_A], \nabla\phi(E[Z])\rangle - \sum_{r=1}^{s} E[\langle Z' - \hat{Z}_A, \frac{\Lambda^*_{\mathcal{G}_r}}{w_{\mathcal{G}_r}}\rangle] \\ &\overset{(a)}{=} -\sum_{r=1}^{s} E_{\mathcal{G}_r}[\langle E[Z'|\mathcal{G}_r] - E[\hat{Z}_A|\mathcal{G}_r], \frac{\Lambda^*_{\mathcal{G}_r}}{w_{\mathcal{G}_r}}\rangle] \overset{(b)}{=} 0\ , \end{aligned}$$

where (a) follows since $E[Z] = E[Z_A] = E[Z']$, and (b) follows since both Z' and $\hat{Z}_A$ satisfy the constraints, $E[Z|\mathcal{G}_r] = E[\hat{Z}_A|\mathcal{G}_r],\ \forall \mathcal{G}_r \in \mathcal{C}$.

To show that the last term $E[\langle Z' - \hat{Z}_A, \nabla\phi(Z'')]$ also vanishes, we use the fact that by definition $\nabla\phi(Z'') = \sum_{r=1}^{s} g_r(E[Z|\mathcal{G}_r])$. Hence,

$$\begin{aligned} E[\langle Z' - \hat{Z}_A, \nabla\phi(Z'')\rangle] &= E[\langle Z' - \hat{Z}_A, \sum_{r=1}^{s} g_r(E[Z|\mathcal{G}_r])\rangle] \\ &= \sum_{r=1}^{s} E[\langle Z' - \hat{Z}_A, g_r(E[Z|\mathcal{G}_r])\rangle] \\ &= \sum_{r=1}^{s} E_{\mathcal{G}_r}[\langle E[Z'|\mathcal{G}_r] - E[\hat{Z}_A|\mathcal{G}_r], g_r(E[Z|\mathcal{G}_r])\rangle] = 0\ , \end{aligned}$$

since $E[Z|\mathcal{G}_r] = E[\hat{Z}_A|\mathcal{G}_r],\ \forall \mathcal{G}_r \in \mathcal{C}$. That completes the proof. ∎

Proof of Theorem 5 From Lemma 7, we observe that for any $Z' \in \mathcal{S}_A$ and $Z'' \in \mathcal{S}_B$,

$$E[d_\phi(Z', Z'')] = E[d_\phi(Z', \hat{Z}_A)] + E[d_\phi(\hat{Z}_A, Z'')]\ .$$

that is, it is additive in functions of the conditional expectations, that is, $\frac{\Lambda^*_{\mathcal{G}_r}}{w_{\mathcal{G}_r}}$ in the natural parameter space, which implies that $\hat{Z}_A \in \mathcal{S}_B$ as well. For a given $Z'' \in \mathcal{S}_B$ and any $Z' \in \mathcal{S}_A$, $E[d_\phi(Z', Z'')] \geq E[d_\phi(\hat{Z}_A, Z'')]$, with equality only when $Z' = \hat{Z}_A$, due to strict convexity of ϕ . Since $\hat{Z}_A \in \mathcal{S}_A$, this implies that

$$\hat{Z}_A = \operatorname*{argmin}_{Z' \in \mathcal{S}_A} E[d_\phi(Z', Z'')],\ \ \forall Z'' \in \mathcal{S}_B\ .$$

Similarly, for a given $Z' \in \mathcal{S}_A$ and any $Z'' \in \mathcal{S}_A$, $E[d_\phi(Z', Z'')] \geq E[d_\phi(Z', \hat{Z}_A)]$ with equality only when $Z'' = \hat{Z}_A$. By (32), we observe that $\nabla\phi(\hat{Z}_A)$ is an additive function of the conditional expectations, which implies that $\hat{Z}_A \in \mathcal{S}_B$. Thus, we obtain the second part of the result, that is,

$$\hat{Z}_A = \operatorname*{argmin}_{Z'' \in \mathcal{S}_B} d_\phi(Z', Z''),\ \ \forall Z' \in \mathcal{S}_A\ .\ \blacksquare$$

Proof of Lemma 9 From Lemma 8, we have

$$\begin{aligned} E[d_\phi(Z, \tilde{Z}^t)] &= E_U[E_{V|U}[\xi(U, \rho^{t+1}(U), V, \gamma^t(V))]] \\ &= E_U[\min_{g:[g]_1^k} E_{V|U}[\xi(U, g, V, \gamma^t(V))]] \\ &\leq E_U[E_{V|U}[\xi(U, \rho^t(U), V, \gamma(V))]] \\ &= E[d_\phi(Z, \hat{Z}^t)]\ . \ \blacksquare \end{aligned}$$

Appendix E. A Recipe for Implementation

To instantiate the Bregman co-clustering meta-algorithm, two key ingredients need to be selected: (a) the Bregman divergence suitable for a given data matrix, and (b) a co-clustering basis. The goal of this section is to show how to translate the abstract meta algorithm in Section 5 into a concrete and operational co-clustering recipe that is customized for the selected ingredients. We discuss four such concrete recipes. The first three cases concern special cases that admit significant structural and computational simplifications in the meta-algorithm and the last case concerns an example that requires us to use the full power of the abstract framework.

The Bregman co-clustering algorithm (Algorithm 2) involves three main steps—(i) obtaining the MBI solution (Section 5.5) or the optimal Lagrange multipliers, (ii) row assignment, and (iii) column assignment. Of these three steps, the last two involve conceptually straightforward comparisons to determine the optimal row and column assignments at each stage whereas the first step usually involves non-linear optimization and can be computationally expensive. Nonetheless, it is possible to implement these steps in a computationally economical fashion. For certain special cases, the MBI problem has a closed form solution, which eliminates the need for the MBI routine and allows significant simplification of the overall co-clustering algorithm. In particular, there are three cases for which such closed form exists:

Case A: When the co-clustering basis $\mathcal{C}$ is $\mathcal{C}_2$ and d_ϕ is any Bregman divergence. Conceptually, this case was covered in complete detail in Section 3, but we present additional operational details in this section.

Case B: When d_ϕ is squared Euclidean distance and $\mathcal{C}$ is any co-clustering basis in the set $\{\mathcal{C}_i\}_{i=1}^{6}$,

Case C: When d_ϕ is I-divergence and $\mathcal{C}$ is any co-clustering basis in the set $\{\mathcal{C}_i\}_{i=1}^{6}$.

For these special cases, the cost functions that determine the row and column assignments in steps 2B and 2C of the co-clustering algorithm (Algorithm 2) can be directly expressed in terms of the co-clustering (ρ,γ) and the input matrix $\mathbf{Z}$ without any Lagrange multipliers and the computational effort required to evaluate the cost is linear in the size of $\mathbf{Z}$ (i.e., number of non-zeros). For the general case, the computation time per iteration for the co-clustering algorithm is still linear in the size of $\mathbf{Z}$, but the total time taken will depend on the number of iterations required to obtain the MBI solution.

In order to describe the Bregman co-clustering algorithm for the special cases mentioned above, we use a matrix notation that is more suitable for computation and exposition. From Theorem 1 and Tables 1 and 2, we observe that the MBI solution for the three special cases mentioned above can be expressed as a combination of conditional expectations of the random variable Z corresponding to the input matrix. Since the computation of the MBI solution is an important task in the co-clustering algorithm, we proceed by first expressing the various conditional expectations in matrix notation. We use the symbols $\otimes$ and $\oslash$ respectively to denote element-wise multiplication (i.e., the Hadamard product) and element-wise division between two matrices of the same size.

E.1 Matrix Representation of Conditional Expectations

Let $\mathbf{Z} \in S^{m\times n}$ denote the data matrix and $\mathbf{W} \in \mathbb{R}_+^{m\times n}$ denote the matrix corresponding to a probability measure over the matrix elements. Let $\mathbf{R} \in \{0,1\}^{m\times k}$ and $\mathbf{C} \in \{0,1\}^{n\times l}$ denote the row and column

cluster membership matrices, that is,

$$r_{ug} = \begin{cases} 1 & g = \rho(u), \\ 0 & \text{otherwise}, \end{cases} \qquad c_{vh} = \begin{cases} 1 & h = \gamma(v), \\ 0 & \text{otherwise}. \end{cases}$$

In other words, the entry $r_{ug} = 1$ iff row u belongs to row cluster g and the entry $c_{vh} = 1$ iff column v belongs to column cluster h. Further, let $\mathbf{E}_m$ and $\mathbf{E}_n$ denote $m \times 1$ and $n \times 1$ vectors consisting of all ones. It should now be straightforward to see that elements in different partitions (e.g., rows or row clusters) of the input matrix $\mathbf{Z}$ can be aggregated using the appropriate matrix multiplication operations, from the ones listed below:

(a) Left multiplication by $\mathbf{R}^T$—Aggregation of the rows into row clusters

(b) Right multiplication by $\mathbf{C}$—Aggregation of the columns into column clusters

(c) Left multiplication by $\mathbf{E}_m^T$—Aggregation of all the rows into a single group

(d) Right multiplication by $\mathbf{E}_n$—Aggregation of all the columns into a single group

To obtain the expected values along the various partitions instead of the sums, we need to perform an element-wise multiplication with the measure matrix $\mathbf{W}$ before aggregation and later follow it up with an appropriate element-wise division. It is important to note here that the size of matrix containing the expected values is equal to the number of corresponding partitions, which is usually smaller than that of the original $\mathbf{Z}$. Therefore, to create a $m \times n$ matrix such that the uv^{th} element reflects the expectation along the partition containing z_{uv}, we need to replicate the expected values for all members of the corresponding partitions, which can be achieved using the following matrix multiplications:

(a) Left multiplication by $\mathbf{R}$—Replication of the given (row) vectors corresponding to each row cluster along all the constituent rows.

(b) Right multiplication by $\mathbf{C}^T$—Replication of the given (column) vectors corresponding to each column cluster along all the constituent columns.

(c) Left multiplication by $\mathbf{E}_m$—Replication of a given (row) vector along all the rows.

(d) Right multiplication by $\mathbf{E}_n^T$—Replication of a given (column) vector along all the columns

For example, the conditional expectation $E[Z|\hat{U},\hat{V}]$ involves partitioning along $(\hat{U},\hat{V})$, that is, both row and column clusters. Since there are k row clusters and l column clusters, there are kl partitions (or co-clusters) and a conditional expectation value corresponding to each of these partitions. To obtain these expectation values, we need to aggregate the rows into the row clusters as well as the columns into column clusters. In particular, the conditional expectation values are given by

$$E[Z|\hat{u},\hat{v}] = \bar{z}_{\hat{u},\hat{v}} \text{ where } \bar{\mathbf{Z}}_{\hat{U},\hat{V}} = \left(\mathbf{R}^T(\mathbf{W} \otimes \mathbf{Z})\mathbf{C}\right) \oslash \left(\mathbf{R}^T\mathbf{W}\mathbf{C}\right) .$$

Though seemingly complicated, the above expression has a simple interpretation in terms of the aggregation and replication operators described earlier. Operation $\mathbf{W} \otimes \mathbf{Z}$ has the effect of attenuating each element z_{uv} by its corresponding weight w_{uv}. Left multiplication by $\mathbf{R}^T$ aggregates the matrix

$E[Z\|\mathcal{G}]$	$\bar{\mathbf{Z}}_{\mathcal{G}}$	size($\bar{\mathbf{Z}}_{\mathcal{G}}$)	$\mathbf{Z}^f_{\mathcal{G}}$ $(m \times n)$
$E[Z]$	$(\mathbf{E}_m^T(\mathbf{W} \otimes \mathbf{Z})\mathbf{E}_n) \oslash (\mathbf{E}_m^T\mathbf{W}\mathbf{E}_n)$	1×1	$\mathbf{E}_m\bar{\mathbf{Z}}\mathbf{E}_n^T$
$E[Z\|U]$	$((\mathbf{W} \otimes \mathbf{Z})\mathbf{E}_n) \oslash (\mathbf{W}\mathbf{E}_n)$	$m \times 1$	$\bar{\mathbf{Z}}_U\mathbf{E}_n^T$
$E[Z\|V]$	$(\mathbf{E}_m^T(\mathbf{W} \otimes \mathbf{Z})) \oslash (\mathbf{E}_m^T\mathbf{W})$	$1 \times n$	$\mathbf{E}_m\bar{\mathbf{Z}}_V$
$E[Z\|\hat{U}]$	$(\mathbf{R}^T(\mathbf{W} \otimes \mathbf{Z})\mathbf{E}_n) \oslash (\mathbf{R}^T\mathbf{W}\mathbf{E}_n)$	$k \times 1$	$\mathbf{R}\bar{\mathbf{Z}}_{\hat{U}}\mathbf{E}_n^T$
$E[Z\|\hat{V}]$	$(\mathbf{E}_m^T(\mathbf{W} \otimes \mathbf{Z})\mathbf{C}) \oslash (\mathbf{E}_m^T\mathbf{W}\mathbf{C})$	$1 \times l$	$\mathbf{E}_m\bar{\mathbf{Z}}_{\hat{V}}\mathbf{C}^T$
$E[Z\|U,\hat{V}]$	$((\mathbf{W} \otimes \mathbf{Z})\mathbf{C}) \oslash (\mathbf{W}\mathbf{C})$	$m \times l$	$\bar{\mathbf{Z}}_{U,\hat{V}}\mathbf{C}^T$
$E[Z\|\hat{U},V]$	$(\mathbf{R}^T(\mathbf{W} \otimes \mathbf{Z})) \oslash (\mathbf{R}^T\mathbf{W})$	$k \times n$	$\mathbf{R}\bar{\mathbf{Z}}_{\hat{U},V}$
$E[Z\|\hat{U},\hat{V}]$	$(\mathbf{R}^T(\mathbf{W} \otimes \mathbf{Z})\mathbf{C}) \oslash (\mathbf{R}^T\mathbf{W}\mathbf{C})$	$k \times l$	$\mathbf{R}\bar{\mathbf{Z}}_{\hat{U},\hat{V}}\mathbf{C}^T$
$E[Z\|U,V]$	$(\mathbf{W} \otimes \mathbf{Z}) \oslash \mathbf{W}$	$m \times n$	$\bar{\mathbf{Z}}_{U,V}$

Table 8: Conditional expectations in matrix notation.

along rows in the same row cluster across each column, and then right multiplication by $\mathbf{C}$ aggregates this reduced matrix consisting of row cluster sums along columns in the same column cluster. Thus, each element of $\mathbf{R}^T(\mathbf{W} \otimes \mathbf{Z})\mathbf{C}$ represents the sum along each co-cluster of the attenuated $\mathbf{Z}$. Similarly, the matrix $\mathbf{R}^T\mathbf{W}\mathbf{C}$ contains the probability mass assigned to the different co-cluster by $\mathbf{W}$ and the element-wise division results in $k \times l$ matrix whose $\hat{u}\hat{v}^{th}$ entry is the expected value in $\hat{u}\hat{v}^{th}$ co-cluster.

To obtain a $m \times n$ full matrix $\mathbf{Z}^f_{\hat{U},\hat{V}}$ such that $z^f_{uv} = E[Z|\rho(u),\gamma(v)]$, we need to replicate the co-cluster values along the rows and columns corresponding to the row and column clusters respectively. Hence, the reconstructed matrix

$$\mathbf{Z}^f_{\hat{U},\hat{V}} = \mathbf{R}\bar{\mathbf{Z}}_{\hat{U},\hat{V}}\mathbf{C}^T = \mathbf{R}\left(\left(\mathbf{R}^T(\mathbf{W} \otimes \mathbf{Z})\mathbf{C}\right) \oslash \left(\mathbf{R}^T\mathbf{W}\mathbf{C}\right)\right)\mathbf{C}^T.$$

Table 8 shows the matrices corresponding to the various conditional expectations. Note that the number of independent parameters in $\mathbf{Z}^f_{\mathcal{G}}$ (in Table 8) is equal to that in $\bar{\mathbf{Z}}_{\mathcal{G}}$ in spite of the difference in the matrix sizes.

E.2 Bregman Co-clustering Algorithm for Special Cases

We will now consider the three special cases mentioned above and illustrate how the various steps in the Bregman co-clustering algorithm can be instantiated.

E.2.1 CASE A: BASIS $\mathcal{C}_2$ AND ANY BREGMAN DIVERGENCE

1. **Obtaining the MBI Solution.** From Theorem 1, we note that the MBI solution for case A is $\hat{Z} = E[Z|\hat{U},\hat{V}]$. From Table 8, the corresponding MBI matrix $\hat{\mathbf{Z}}$ is given by $\mathbf{Z}^f_{\hat{U},\hat{V}} = \mathbf{R}\bar{\mathbf{Z}}_{\hat{U},\hat{V}}\mathbf{C}^T$ where $\bar{\mathbf{Z}}_{\hat{U},\hat{V}}$ is computed as $(\mathbf{R}^T(\mathbf{W} \otimes \mathbf{Z})\mathbf{C}) \oslash (\mathbf{R}^T\mathbf{W}\mathbf{C})$. Since $\hat{\mathbf{Z}}$ is completely determined by the smaller $k \times l$ matrix $\bar{\mathbf{Z}}_{\hat{U},\hat{V}}$, we only compute and store the reduced matrix. Using the fact that $\mathbf{R}$ and $\mathbf{C}$ are binary matrices, this computation can be performed efficiently using $O(mn)$ operations.

2. **Row Cluster Assignment Step.** Given the parameters of the MBI solution and a fixed column clustering determined by $\mathbf{C}$, we want to find for each row, the row cluster assignment that leads to the best approximation to the original matrix. In other words, we are searching for a

row cluster membership matrix $\mathbf{R}'$ that results in the most accurate reconstruction $\tilde{\mathbf{Z}}$. For the current case, this reconstructed matrix $\tilde{\mathbf{Z}}$ takes the same functional form as the MBI solution and is given by $\mathbf{R}'\bar{\mathbf{Z}}_{\hat{U},\hat{V}}\mathbf{C}^T$ where $\bar{\mathbf{Z}}_{\hat{U},\hat{V}}$ is based on the previous row clustering. From step 2B of the Bregman co-clustering algorithm (Algorithm 2), the optimal row assignment for each row u is given by

$$\begin{aligned}
& \rho^*(u) = \operatorname*{argmin}_{g\in\{1,\cdots,k\}} E_{V|u}[d_\phi(Z,\tilde{Z})] = \operatorname*{argmin}_{g\in\{1,\cdots,k\}} \sum_{v=1}^{n} w_{uv} d_\phi(z_{uv},\tilde{z}_{uv}),\ [u]_1^m, \\
\overset{(a)}{\Rightarrow}\ & \mathbf{R}^* = \operatorname*{argmin}_{\mathbf{R}'} d_{\Phi_w}(\mathbf{Z},\tilde{\mathbf{Z}}) = \operatorname*{argmin}_{\mathbf{R}'} d_{\Phi_w}(\mathbf{Z},\mathbf{R}'\bar{\mathbf{Z}}_{\hat{U},\hat{V}}\mathbf{C}^T), \\
\overset{(b)}{\Rightarrow}\ & \mathbf{R}^* = \operatorname*{argmin}_{\mathbf{R}'} d_{\Phi_w}(\mathbf{Z}^{rowRed},\mathbf{R}'\bar{\mathbf{Z}}_{\hat{U},\hat{V}}), \\
\overset{(c)}{\Rightarrow}\ & \rho^*(u) = \operatorname*{argmin}_{g\in\{1,\cdots,k\}} \sum_{h=1}^{l} w_{uh} d_\phi(z_{uh}^{rowRed},\bar{z}_{gh}),\ [u]_1^m,
\end{aligned}$$

where $\mathbf{Z}^{rowRed} \equiv ((\mathbf{W}\otimes\mathbf{Z})\mathbf{C}) \oslash (\mathbf{W}\mathbf{C})$ and d_{Φ_w} is the induced Bregman divergence that applies to matrices in $\mathcal{S}^{k\times n}$.[16] In the above, (a) and (c) follow from the definition of the row cluster membership matrix, and (b) follows from the fact that minimizing the (weighted) average Bregman divergence from a set $\{\mathbf{x}_i\}_{i=1}^n$ to a fixed point $\mathbf{a}$ is equivalent to minimizing the Bregman divergence between the (weighted) average of the set and $\mathbf{a}$ (Banerjee et al., 2005b). Assuming the matrix $\mathbf{Z}^{rowRed}$ is computed apriori, the row clustering only requires $O(mkl)$ operations as opposed to $O(mkn)$ since for each row, we only compare the reduced rows (of size $1\times l$) in $\mathbf{Z}^{rowRed}$ with the k possible row cluster representatives.

3. **Column Cluster Assignment Step.** The column cluster assignment step is similar to that of the previous row cluster assignment step and involves finding a column cluster membership matrix $\mathbf{C}'$ that results in the most accurate reconstruction $\tilde{\mathbf{Z}} = \mathbf{R}\bar{\mathbf{Z}}_{\hat{U},\hat{V}}\mathbf{C}'^T$. From step 2C of the Bregman co-clustering algorithm (Algorithm 2), the optimal column assignment for each column v is given by

$$\begin{aligned}
& \gamma^*(v) = \operatorname*{argmin}_{h\in\{1,\cdots,l\}} E_{U|v}[d_\phi(Z,\tilde{Z})] = \operatorname*{argmin}_{h\in\{1,\cdots,l\}} \sum_{u=1}^{m} w_{uv} d_\phi(z_{uv},\tilde{z}_{uv}),\ [v]_1^n, \\
\overset{(a)}{\Rightarrow}\ & \mathbf{C}^* = \operatorname*{argmin}_{\mathbf{C}'} d_{\Phi_w}(\mathbf{Z},\tilde{\mathbf{Z}}) = \operatorname*{argmin}_{\mathbf{C}'} d_{\Phi_w}(\mathbf{Z},\mathbf{R}\bar{\mathbf{Z}}_{\hat{U},\hat{V}}\mathbf{C}'^T), \\
\overset{(b)}{\Rightarrow}\ & \mathbf{C}^* = \operatorname*{argmin}_{\mathbf{C}'} d_{\Phi_w}(\mathbf{Z}^{colRed},\bar{\mathbf{Z}}_{\hat{U},\hat{V}}\mathbf{C}'^T), \\
\overset{(c)}{\Rightarrow}\ & \gamma^*(v) = \operatorname*{argmin}_{h\in\{1,\cdots,l\}} \sum_{g=1}^{k} w_{gv} d_\phi(z_{gv}^{colRed},\bar{z}_{gh}),\ [v]_1^n,
\end{aligned}$$

where $\mathbf{Z}^{colRed} \equiv (\mathbf{R}^T(\mathbf{W}\otimes\mathbf{Z})) \oslash (\mathbf{R}^T\mathbf{W})$, (a) and (c) follow from the definition of the column cluster membership matrix, and (b) follows from the same reduction (Banerjee et al., 2005b)

16. Note that d_{Φ_w} has been overloaded to denote the separable Bregman divergences induced from the original d_ϕ and the measure w that apply to matrices in $\mathcal{S}^{m\times n}$, $\mathcal{S}^{k\times n}$ and $\mathcal{S}^{m\times l}$.

employed in the row cluster assignment step. As in the previous case, the column clustering involves a reduced number of distance computations and comparisons and in particular, requires $O(nkl)$ operations.

E.2.2 CASE B: SQUARED EUCLIDEAN DISTANCE

1. **Obtaining the MBI Solution.** For this case, the MBI solution $\hat{Z}$ has a closed form for all the six co-clustering bases in terms of the appropriate conditional expectations as shown in Table 2. Using Table 8, we can exactly compute each of the relevant conditional expectations, which requires $O(mn)$ operations. Though we do not explicitly compute it, the MBI matrix $\hat{\mathbf{Z}}$ (shown in Table 9) can be expressed in terms of the row clustering $\mathbf{R}$, column clustering $\mathbf{C}$ and these conditional expectations for any co-clustering basis.

2. **Row Cluster Assignment Step.** To obtain the row cluster assignment step, we observe that the reconstructed matrix $\tilde{\mathbf{Z}}$, which has the same form as $\hat{\mathbf{Z}}$ can be split into two additive terms of which only one depends on the candidate row clustering. In particular, for the row assignment step, the reconstructed matrix $\tilde{\mathbf{Z}}$ based on a candidate row clustering $\mathbf{R}'$ can be written as

$$\tilde{\mathbf{Z}} = \tilde{\mathbf{Z}}^{rowConst} + \mathbf{R}'\tilde{\mathbf{Z}}^{rowVar}, \tag{33}$$

where $\tilde{\mathbf{Z}}^{rowConst}$ is an $m \times n$ matrix corresponding to the constant part of $\tilde{\mathbf{Z}}$ and $\tilde{\mathbf{Z}}^{rowVar}$ is a $k \times n$ matrix corresponding to the variable part of $\tilde{\mathbf{Z}}$. Table 10 provides the $\tilde{\mathbf{Z}}^{rowConst}$ and $\tilde{\mathbf{Z}}^{rowVar}$ for the different co-clustering bases. From step 2B of Algorithm 2 and (33), the row cluster update step for squared Euclidean distance is given by

$$\begin{aligned}
\rho^*(u) &= \operatorname*{argmin}_{g\in\{1,\cdots,k\}} E_{V|u}[(Z-\tilde{Z})^2] = \operatorname*{argmin}_{g\in\{1,\cdots,k\}} \sum_{v=1}^{n} w_{uv}(z_{uv}-\tilde{z}_{uv})^2,\ [u]_1^m, \\
\Rightarrow \quad \mathbf{R}^* &= \operatorname*{argmin}_{\mathbf{R}'} ||\mathbf{Z}-\tilde{\mathbf{Z}}||_w^2 = \operatorname*{argmin}_{\mathbf{R}'} ||\mathbf{Z}-\tilde{\mathbf{Z}}^{rowConst}-\mathbf{R}'\tilde{\mathbf{Z}}^{rowVar}||_w^2, \\
\Rightarrow \quad \mathbf{R}^* &= \operatorname*{argmin}_{\mathbf{R}'} ||\mathbf{Z}^{row}-\mathbf{R}'\tilde{\mathbf{Z}}^{rowVar}||_w^2, \\
\Rightarrow \quad \rho^*(u) &= \operatorname*{argmin}_{g\in\{1,\cdots,k\}} \sum_{v=1}^{n} w_{uv}(z_{uv}^{row}-z_{gv}^{rowVar})^2,\ [u]_1^m,
\end{aligned}$$

where $\mathbf{Z}^{row} = \mathbf{Z}-\tilde{\mathbf{Z}}^{rowConst}$ and $||\cdot||_w$ is the weighted squared Euclidean distance. The optimal row assignments can, therefore, be determined by finding the *nearest* row (among k possible ones) in $\tilde{\mathbf{Z}}^{rowVar}$ for each of the m rows in $\mathbf{Z}^{row}$. The above row assignment step can be readily instantiated for any specified co-clustering basis by choosing the appropriate matrices $\tilde{\mathbf{Z}}^{rowConst}$ and $\tilde{\mathbf{Z}}^{rowVar}$ from Table 10.

For co-clustering bases $\{C_i\}_{i=1}^5$, it is possible to further optimize the above update step using the same observation as in case A, that is, minimizing the row update cost function $||\mathbf{Z}^{row}-\mathbf{R}'\tilde{\mathbf{Z}}^{rowVar}||_w^2$ is equivalent to minimizing the distortion between reduced versions of these matrices, that is, $||\mathbf{Z}^{rowRed}-\mathbf{R}'\tilde{\mathbf{Z}}^{rowVRed}||_w^2$ where $\mathbf{Z}^{rowRed} \equiv ((\mathbf{W}\otimes\mathbf{Z}^{row})\mathbf{C})\oslash(\mathbf{WC})$ and $\mathbf{R}'\tilde{\mathbf{Z}}^{rowVRed} \equiv ((\mathbf{W}\otimes(\mathbf{R}'\tilde{\mathbf{Z}}^{rowVar}))\mathbf{C})\oslash(\mathbf{WC})$. Though the expression for $\tilde{\mathbf{Z}}^{rowVRed}$ looks complicated, it can be simplified using the fact that $\tilde{\mathbf{Z}}^{rowVar}$ can always be written as $\mathbf{AC}^T+\mathbf{BE}_n^T$ for some matrices $\mathbf{A}$ and $\mathbf{B}$, which ensures that $\tilde{\mathbf{Z}}^{rowVRed} = \mathbf{A}+\mathbf{BE}_l^T$, that is, independent of $\mathbf{R}'$. For all the five co-clustering bases, $\tilde{\mathbf{Z}}^{rowVRed}$ is determined by the relevant conditional

Co-clustering basis $\mathcal{C}$	$\hat{\mathbf{Z}}$ $(m \times n)$
$\mathcal{C}_1$	$\mathbf{R}\bar{\mathbf{Z}}_{\hat{U}}\mathbf{E}_n^T + \mathbf{E}_m\bar{\mathbf{Z}}_{\hat{V}}\mathbf{C}^T - \mathbf{E}_m\bar{\mathbf{Z}}\mathbf{E}_n^T$
$\mathcal{C}_2$	$\mathbf{R}\bar{\mathbf{Z}}_{\hat{U},\hat{V}}\mathbf{C}^T$
$\mathcal{C}_3$	$\mathbf{R}\bar{\mathbf{Z}}_{\hat{U},\hat{V}}\mathbf{C}^T + \bar{\mathbf{Z}}_U\mathbf{E}_n^T - \mathbf{R}\bar{\mathbf{Z}}_{\hat{U}}\mathbf{E}_n^T$
$\mathcal{C}_4$	$\mathbf{R}\bar{\mathbf{Z}}_{\hat{U},\hat{V}}\mathbf{C}^T + \mathbf{E}_m\bar{\mathbf{Z}}_V - \mathbf{E}_m\bar{\mathbf{Z}}_{\hat{V}}\mathbf{C}^T$
$\mathcal{C}_5$	$\mathbf{R}\bar{\mathbf{Z}}_{\hat{U},\hat{V}}\mathbf{C}^T + \bar{\mathbf{Z}}_U\mathbf{E}_n^T - \mathbf{R}\bar{\mathbf{Z}}_{\hat{U}}\mathbf{E}_n^T + \mathbf{E}_m\bar{\mathbf{Z}}_V - \mathbf{E}_m\bar{\mathbf{Z}}_{\hat{V}}\mathbf{C}^T$
$\mathcal{C}_6$	$\bar{\mathbf{Z}}_{U,\hat{V}}\mathbf{C}^T + \mathbf{R}\bar{\mathbf{Z}}_{\hat{U},V} - \mathbf{R}\bar{\mathbf{Z}}_{\hat{U},\hat{V}}\mathbf{C}^T$

Table 9: MBI matrix for squared Euclidean distance.

Co-clustering basis $\mathcal{C}$	$\tilde{\mathbf{Z}}^{rowConst}$ $(m \times n)$	$\tilde{\mathbf{Z}}^{rowVar}$ $(k \times n)$	$\tilde{\mathbf{Z}}^{rowVRed}$ $(k \times l)$
$\mathcal{C}_1$	$\mathbf{E}_m\bar{\mathbf{Z}}_{\hat{V}}\mathbf{C}^T - \mathbf{E}_m\bar{\mathbf{Z}}\mathbf{E}_n^T$	$\bar{\mathbf{Z}}_{\hat{U}}\mathbf{E}_n^T$	$\bar{\mathbf{Z}}_{\hat{U}}\mathbf{E}_l^T$
$\mathcal{C}_2$	$\mathbf{0}$	$\bar{\mathbf{Z}}_{\hat{U},\hat{V}}\mathbf{C}^T$	$\bar{\mathbf{Z}}_{\hat{U},\hat{V}}$
$\mathcal{C}_3$	$\bar{\mathbf{Z}}_U\mathbf{E}_n^T$	$\bar{\mathbf{Z}}_{\hat{U},\hat{V}}\mathbf{C}^T - \bar{\mathbf{Z}}_{\hat{U}}\mathbf{E}_n^T$	$\bar{\mathbf{Z}}_{\hat{U},\hat{V}} - \bar{\mathbf{Z}}_{\hat{U}}\mathbf{E}_l^T$
$\mathcal{C}_4$	$\mathbf{E}_m\bar{\mathbf{Z}}_V - \mathbf{E}_m\bar{\mathbf{Z}}_{\hat{V}}\mathbf{C}^T$	$\bar{\mathbf{Z}}_{\hat{U},\hat{V}}\mathbf{C}^T$	$\bar{\mathbf{Z}}_{\hat{U},\hat{V}}$
$\mathcal{C}_5$	$\bar{\mathbf{Z}}_U\mathbf{E}_n^T + \mathbf{E}_m\bar{\mathbf{Z}}_V - \mathbf{E}_m\bar{\mathbf{Z}}_{\hat{V}}\mathbf{C}^T$	$\bar{\mathbf{Z}}_{\hat{U},\hat{V}}\mathbf{C}^T - \bar{\mathbf{Z}}_{\hat{U}}\mathbf{E}_n^T$	$\bar{\mathbf{Z}}_{\hat{U},\hat{V}} - \bar{\mathbf{Z}}_{\hat{U}}\mathbf{E}_l^T$
$\mathcal{C}_6$	$\bar{\mathbf{Z}}_{U,\hat{V}}\mathbf{C}^T$	$\bar{\mathbf{Z}}_{\hat{U},V} - \bar{\mathbf{Z}}_{\hat{U},\hat{V}}\mathbf{C}^T$	n/a

Table 10: Row assignment update matrices for squared Euclidean distance.

expectations and can be looked up from Table 10. As a result of this optimization, the row clustering step involves comparisons between smaller matrices and requires only $O(mkl)$ operations.

3. **Column Cluster Assignment Step.** The column assignment step employs a similar decomposition of $\tilde{\mathbf{Z}}$ in terms of the column clustering, that is, $\tilde{\mathbf{Z}} = \tilde{\mathbf{Z}}^{colConst} + \tilde{\mathbf{Z}}^{colVar}\mathbf{C}'^T$ and the optimal assignments are determined by

$$\gamma(v) = \operatorname*{argmin}_{h\in\{1,\cdots,l\}} \sum_{u=1}^{m} w_{uv}(z_{uv}^{col} - z_{uh}^{colVar})^2,\ [v]_1^n,$$

where $\mathbf{Z}^{col} \equiv \mathbf{Z} - \tilde{\mathbf{Z}}^{colConst}$ and the matrices $\tilde{\mathbf{Z}}^{colConst}$ and $\tilde{\mathbf{Z}}^{colVar}$ can be obtained from Table 11. As in the case of row clustering, it is possible to further optimize the above update step for co-clustering bases $\{C_i\}_{i=1}^5$ by computing $\mathbf{Z}^{colRed} \equiv (\mathbf{R}^T(\mathbf{W}\otimes\mathbf{Z}^{col}))\oslash(\mathbf{R}^T\mathbf{W})$ and comparing it with $\tilde{\mathbf{Z}}^{colVRed}\mathbf{C}^T \equiv (\mathbf{R}^T(\mathbf{W}\otimes(\tilde{\mathbf{Z}}^{colVar}\mathbf{C}^T)))\oslash(\mathbf{R}^T\mathbf{W})$, which can be obtained from Table 11. Further, as in the previous step, the column clustering step only requires $O(nkl)$ operations similar to that in case A.

E.2.3 CASE C: I-DIVERGENCE

1. **Obtaining the MBI Solution.** As in the previous case, the MBI solution for case C has a closed form for all the six co-clustering bases in terms of the appropriate conditional expectations as shown in Table 1. Using Table 8, one can exactly compute each of the relevant conditional expectations, which completely determine the MBI matrix $\hat{\mathbf{Z}}$ (shown in Table 12) for a given row clustering $\mathbf{R}$ and column clustering $\mathbf{C}$.

Co-clustering basis $\mathcal{C}$	$\tilde{\mathbf{Z}}^{colConst}$ $(m \times n)$	$\tilde{\mathbf{Z}}^{colVar}$ $(m \times l)$	$\tilde{\mathbf{Z}}^{colVRed}$ $(k \times l)$
$\mathcal{C}_1$	$\mathbf{R}\bar{\mathbf{Z}}_{\hat{U}}\mathbf{E}_n^T - \mathbf{E}_m\bar{\mathbf{Z}}\mathbf{E}_n^T$	$\mathbf{E}_m\bar{\mathbf{Z}}_{\hat{V}}$	$\mathbf{E}_k\bar{\mathbf{Z}}_{\hat{V}}$
$\mathcal{C}_2$	$\mathbf{0}$	$\mathbf{R}\bar{\mathbf{Z}}_{\hat{U},\hat{V}}$	$\bar{\mathbf{Z}}_{\hat{U},\hat{V}}$
$\mathcal{C}_3$	$\bar{\mathbf{Z}}_U\mathbf{E}_n^T - \mathbf{R}\bar{\mathbf{Z}}_{\hat{U}}\mathbf{E}_n^T$	$\mathbf{R}\bar{\mathbf{Z}}_{\hat{U},\hat{V}}$	$\bar{\mathbf{Z}}_{\hat{U},\hat{V}}$
$\mathcal{C}_4$	$\mathbf{E}_m\bar{\mathbf{Z}}_V$	$\mathbf{R}\bar{\mathbf{Z}}_{\hat{U},\hat{V}} - \mathbf{E}_m\bar{\mathbf{Z}}_{\hat{V}}$	$\bar{\mathbf{Z}}_{\hat{U},\hat{V}} - \mathbf{E}_k\bar{\mathbf{Z}}_{\hat{V}}$
$\mathcal{C}_5$	$\bar{\mathbf{Z}}_U\mathbf{E}_n^T + \mathbf{E}_m\bar{\mathbf{Z}}_V - \mathbf{R}\bar{\mathbf{Z}}_{\hat{U}}\mathbf{E}_n^T$	$\mathbf{R}\bar{\mathbf{Z}}_{\hat{U},\hat{V}} - \mathbf{E}_m\bar{\mathbf{Z}}_{\hat{V}}$	$\bar{\mathbf{Z}}_{\hat{U},\hat{V}} - \mathbf{E}_k\bar{\mathbf{Z}}_{\hat{V}}$
$\mathcal{C}_6$	$\mathbf{R}\bar{\mathbf{Z}}_{\hat{U},V}$	$\bar{\mathbf{Z}}_{U,\hat{V}} - \mathbf{R}\bar{\mathbf{Z}}_{\hat{U},\hat{V}}$	n/a

Table 11: Column assignment update matrices for squared Euclidean distance.

2. **Row Cluster Assignment Step.** To obtain the row assignment steps for I-divergence, we make use of the fact that the reconstructed matrix $\tilde{\mathbf{Z}}$, can be decomposed as the Hadamard product of two terms of which only one depends on the candidate row or column clustering. In particular, the reconstructed matrix $\tilde{\mathbf{Z}}$ can be expressed as

$$\tilde{\mathbf{Z}} = (\tilde{\mathbf{Z}}^{rowConst}) \otimes (\mathbf{R}'\tilde{\mathbf{Z}}^{rowVar}),$$

where $\tilde{\mathbf{Z}}^{rowConst}$ is the constant factor and $\tilde{\mathbf{Z}}^{rowVar}$ is the variable factor that depends on $\mathbf{R}'$, both of which can be looked up from Table 13.

From step 2B of Algorithm 2 and (33), the row cluster update step for I-divergence for $[u]_1^m$ is given by

$$\begin{aligned}
\rho^*(u) &= \operatorname*{argmin}_{g\in\{1,\cdots,k\}} E_{V|u}\left[Z\log\left(\frac{Z}{\tilde{Z}}\right) - Z + \tilde{Z}\right], \\
&= \operatorname*{argmin}_{g\in\{1,\cdots,k\}} \sum_{v=1}^{n} w_{uv}\left(z_{uv}\log\left(\frac{z_{uv}}{\tilde{z}_{uv}}\right) - z_{uv} + \tilde{z}_{uv}\right), \\
&= \operatorname*{argmin}_{g\in\{1,\cdots,k\}} \sum_{v=1}^{n} w_{uv}\left(z_{uv}\log\left(\frac{z_{uv}}{\tilde{z}_{uv}^{rowConst}}\right) - z_{uv}\right) \\
&\qquad + \sum_{v=1}^{n} w_{uv}(\tilde{z}_{uv}^{rowConst}\tilde{z}_{\rho'(u)v}^{rowVar} - z_{uv}\log(\tilde{z}_{\rho'(u)v}^{rowVar})), \\
&\overset{(a)}{=} \operatorname*{argmin}_{g\in\{1,\cdots,k\}} \sum_{v=1}^{n} w_{uv}\left(\tilde{z}_{uv}^{rowConst}\tilde{z}_{\rho'(u)v}^{rowVar} - z_{uv}\log(\tilde{z}_{\rho'(u)v}^{rowVar})\right),
\end{aligned}$$

where (a) follows since the first term in the cost function is independent of the row clustering.

As in case B, it is possible to optimize the row assignment step for the co-clustering bases $\{C_i\}_{i=1}^5$ by minimizing a simplified row update cost function $d_{\Phi_w}(\mathbf{Z}^{rowRed}, \tilde{\mathbf{Z}}^{rowCRed} \otimes \mathbf{R}'\tilde{\mathbf{Z}}^{rowVRed})$ based on equivalent reduced matrices instead of the original cost function $d_{\Phi_w}(\mathbf{Z}, \tilde{\mathbf{Z}}^{rowConst} \otimes \mathbf{R}'\tilde{\mathbf{Z}}^{rowVar})$ where $\mathbf{Z}^{rowRed} \equiv ((\mathbf{W}\otimes\mathbf{Z})\mathbf{C}) \oslash (\mathbf{W}\mathbf{C})$, $\mathbf{Z}^{rowCRed} \equiv ((\mathbf{W}\otimes\mathbf{Z}^{rowConst})\mathbf{C}) \oslash (\mathbf{W}\mathbf{C})$, and $\mathbf{R}'\tilde{\mathbf{Z}}^{rowVRed} \equiv ((\mathbf{W}\otimes(\mathbf{R}'\tilde{\mathbf{Z}}^{rowVar}))\mathbf{C}) \oslash (\mathbf{W}\mathbf{C})$. Further as in the previous case, $\tilde{\mathbf{Z}}^{rowVRed}$ can be simplified by noticing that $\tilde{\mathbf{Z}}^{rowVar}$ in this case can be written as $\mathbf{A}\mathbf{C}^T \otimes \mathbf{B}\mathbf{E}_n^T$ for some matrices $\mathbf{A}$ and $\mathbf{B}$, ensuring that $\tilde{\mathbf{Z}}^{rowVRed} = \mathbf{A} \otimes (\mathbf{B}\mathbf{E}_l^T)$, that is, independent of $\mathbf{R}'$. Table 13 shows the matrix $\tilde{\mathbf{Z}}^{rowVRed}$ for the different co-clustering bases.

Co-clustering basis $\mathcal{C}$	$\hat{\mathbf{Z}}$ $(m \times n)$
$\mathcal{C}_1$	$((\mathbf{R}\bar{\mathbf{Z}}_{\hat{U}}\mathbf{E}_n^T) \otimes (\mathbf{E}_m\bar{\mathbf{Z}}_{\hat{V}}\mathbf{C}^T)) \oslash (\mathbf{E}_m\bar{\mathbf{Z}}\mathbf{E}_n^T)$
$\mathcal{C}_2$	$\mathbf{R}\bar{\mathbf{Z}}_{\hat{U},\hat{V}}\mathbf{C}^T$
$\mathcal{C}_3$	$((\mathbf{R}\bar{\mathbf{Z}}_{\hat{U},\hat{V}}\mathbf{C}^T) \otimes (\bar{\mathbf{Z}}_U\mathbf{E}_n^T)) \oslash (\mathbf{R}\bar{\mathbf{Z}}_{\hat{U}}\mathbf{E}_n^T)$
$\mathcal{C}_4$	$((\mathbf{R}\bar{\mathbf{Z}}_{\hat{U},\hat{V}}\mathbf{C}^T) \otimes (\mathbf{E}_m\bar{\mathbf{Z}}_V)) \oslash (\mathbf{E}_m\bar{\mathbf{Z}}_{\hat{V}}\mathbf{C}^T)$
$\mathcal{C}_5$	$((\mathbf{R}\bar{\mathbf{Z}}_{\hat{U},\hat{V}}\mathbf{C}^T) \otimes (\bar{\mathbf{Z}}_U\mathbf{E}_n^T) \otimes (\mathbf{E}_m\bar{\mathbf{Z}}_V)) \oslash ((\mathbf{R}\bar{\mathbf{Z}}_{\hat{U}}\mathbf{E}_n^T) \otimes (\mathbf{E}_m\bar{\mathbf{Z}}_{\hat{V}}\mathbf{C}^T))$
$\mathcal{C}_6$	$((\bar{\mathbf{Z}}_{U,\hat{V}}\mathbf{C}^T) \otimes (\mathbf{R}\bar{\mathbf{Z}}_{\hat{U},V})) \oslash (\mathbf{R}\bar{\mathbf{Z}}_{\hat{U},\hat{V}}\mathbf{C}^T)$

Table 12: MBI matrix for I-divergence.

Co-clustering basis $\mathcal{C}$	$\tilde{\mathbf{Z}}^{rowConst}$ $(m \times n)$	$\tilde{\mathbf{Z}}^{rowVar}$ $(k \times n)$	$\tilde{\mathbf{Z}}^{rowVRed}$ $(k \times l)$
$\mathcal{C}_1$	$(\mathbf{E}_m\bar{\mathbf{Z}}_{\hat{V}}\mathbf{C}^T) \oslash (\mathbf{E}_m\bar{\mathbf{Z}}\mathbf{E}_n^T)$	$\bar{\mathbf{Z}}_{\hat{U}}\mathbf{E}_n^T$	$\bar{\mathbf{Z}}_{\hat{U}}\mathbf{E}_l^T$
$\mathcal{C}_2$	$\mathbf{E}_{mn}$	$\bar{\mathbf{Z}}_{\hat{U},\hat{V}}\mathbf{C}^T$	$\bar{\mathbf{Z}}_{\hat{U},\hat{V}}$
$\mathcal{C}_3$	$\bar{\mathbf{Z}}_U\mathbf{E}_n^T$	$(\bar{\mathbf{Z}}_{\hat{U},\hat{V}}\mathbf{C}^T) \oslash (\bar{\mathbf{Z}}_{\hat{U}}\mathbf{E}_n^T)$	$(\bar{\mathbf{Z}}_{\hat{U},\hat{V}}) \oslash (\bar{\mathbf{Z}}_{\hat{U}}\mathbf{E}_l^T)$
$\mathcal{C}_4$	$(\mathbf{E}_m\bar{\mathbf{Z}}_V) \oslash (\mathbf{E}_m\bar{\mathbf{Z}}_{\hat{V}}\mathbf{C}^T)$	$\bar{\mathbf{Z}}_{\hat{U},\hat{V}}\mathbf{C}^T$	$\bar{\mathbf{Z}}_{\hat{U},\hat{V}}$
$\mathcal{C}_5$	$((\bar{\mathbf{Z}}_U\mathbf{E}_n^T) \otimes (\mathbf{E}_m\bar{\mathbf{Z}}_V)) \oslash (\mathbf{E}_m\bar{\mathbf{Z}}_{\hat{V}}\mathbf{C}^T)$	$(\bar{\mathbf{Z}}_{\hat{U},\hat{V}}\mathbf{C}^T) \oslash (\bar{\mathbf{Z}}_{\hat{U}}\mathbf{E}_n^T)$	$(\bar{\mathbf{Z}}_{\hat{U},\hat{V}}) \oslash (\bar{\mathbf{Z}}_{\hat{U}}\mathbf{E}_l^T)$
$\mathcal{C}_6$	$\bar{\mathbf{Z}}_{U,\hat{V}}\mathbf{C}^T$	$(\bar{\mathbf{Z}}_{\hat{U},V}) \oslash (\bar{\mathbf{Z}}_{\hat{U},\hat{V}}\mathbf{C}^T)$	n/a

Table 13: Row assignment update matrices for I-divergence.

3. **Column Cluster Assignment Step.** The optimal column assignments can be obtained in similar fashion by computing the matrices $\tilde{\mathbf{Z}}^{colConst}$ and $\tilde{\mathbf{Z}}^{colVar}$ shown in Table 14 and optimizing the part of the column update cost function that depends on the column clustering, that is,

$$\gamma(v) = \underset{h\in\{1,\cdots,l\}}{\operatorname{argmin}} \sum_{u=1}^{n} w_{uv} \left(z_{uv}^{colConst} z_{uh}^{colVar} - z_{uv} \log(z_{uh}^{colVar}) \right), \ [v]_1^n.$$

Further, as in the row clustering case, the column assignment step can be optimized further for co-clustering bases $\{C_i\}_{i=1}^5$ by computing $\mathbf{Z}^{colRed} \equiv (\mathbf{R}^T(\mathbf{W} \otimes \mathbf{Z})) \oslash (\mathbf{R}^T\mathbf{W})$, $\mathbf{Z}^{colCRed} \equiv (\mathbf{R}^T(\mathbf{W} \otimes \tilde{\mathbf{Z}}^{colConst})) \oslash (\mathbf{R}^T\mathbf{W})$ and $\tilde{\mathbf{Z}}^{colVRed}C^T \equiv (\mathbf{R}^T(\mathbf{W} \otimes (\tilde{\mathbf{Z}}^{colVar}C^T))) \oslash (\mathbf{R}^T\mathbf{W})$, using Table 14 and finding the column clustering $\mathbf{C}'$ that optimizes the cost $d_{\Phi_w}(\mathbf{Z}^{colRed}, \tilde{\mathbf{Z}}^{colCRed} \otimes \tilde{\mathbf{Z}}^{colVRed}\mathbf{C})$. The computational time for these update steps is same as in the cases A and B.

E.2.4 CASE D: ANY BREGMAN DIVERGENCE AND CO-CLUSTERING BASIS

The proposed meta-algorithm can be instantiated for any Bregman divergence and co-clustering basis. We now consider a particular example of the general case corresponding to Itakura-Saito distance, which is the Bregman divergence corresponding to the convex function $\phi(z) = -\log(z)$, a uniform measure and the co-clustering basis $\mathcal{C}_1$. The example is a representative of the general case, since no divergence/basis specific optimizations are possible in this case.

1. **Obtaining the MBI Solution.** For the general case involving a Bregman divergence other than squared Euclidean distance and I-divergence and a co-clustering basis different from $\mathcal{C}_2$, the MBI solution does not have a closed form, which makes it necessary to use a convex

Co-clustering basis $\mathcal{C}$	$\tilde{\mathbf{Z}}^{colConst}$ $(m \times n)$	$\tilde{\mathbf{Z}}^{colVar}$ $(m \times l)$	$\tilde{\mathbf{Z}}^{colVRed}$ $(k \times l)$
$\mathcal{C}_1$	$(\mathbf{R}\bar{\mathbf{Z}}_{\hat{U}}\mathbf{E}_n^T) \oslash (\mathbf{E}_m\bar{\mathbf{Z}}\mathbf{E}_n^T)$	$\mathbf{E}_m\bar{\mathbf{Z}}_{\hat{V}}$	$\mathbf{E}_k\bar{\mathbf{Z}}_{\hat{V}}$
$\mathcal{C}_2$	$\mathbf{E}_{mn}$	$\mathbf{R}\bar{\mathbf{Z}}_{\hat{U},\hat{V}}$	$\bar{\mathbf{Z}}_{\hat{U},\hat{V}}$
$\mathcal{C}_3$	$(\bar{\mathbf{Z}}_U\mathbf{E}_n^T) \oslash (\mathbf{R}\bar{\mathbf{Z}}_{\hat{U}}\mathbf{E}_n^T)$	$\mathbf{R}\bar{\mathbf{Z}}_{\hat{U},\hat{V}}$	$\bar{\mathbf{Z}}_{\hat{U},\hat{V}}$
$\mathcal{C}_4$	$\mathbf{E}_m\bar{\mathbf{Z}}_V$	$(\mathbf{R}\bar{\mathbf{Z}}_{\hat{U},\hat{V}}) \oslash (\mathbf{E}_m\bar{\mathbf{Z}}_{\hat{V}})$	$(\bar{\mathbf{Z}}_{\hat{U},\hat{V}}) \oslash (\mathbf{E}_k\bar{\mathbf{Z}}_{\hat{V}})$
$\mathcal{C}_5$	$(\bar{\mathbf{Z}}_U\mathbf{E}_n^T) \otimes (\mathbf{E}_m\bar{\mathbf{Z}}_V) \oslash (\mathbf{R}\bar{\mathbf{Z}}_{\hat{U}}\mathbf{E}_n^T)$	$(\mathbf{R}\bar{\mathbf{Z}}_{\hat{U},\hat{V}}) \oslash (\mathbf{E}_m\bar{\mathbf{Z}}_{\hat{V}})$	$(\bar{\mathbf{Z}}_{\hat{U},\hat{V}}) \oslash (\mathbf{E}_k\bar{\mathbf{Z}}_{\hat{V}})$
$\mathcal{C}_6$	$\mathbf{R}\bar{\mathbf{Z}}_{\hat{U},V}$	$(\bar{\mathbf{Z}}_{U,\hat{V}}) \oslash (\mathbf{R}\bar{\mathbf{Z}}_{\hat{U},\hat{V}})$	n/a

Table 14: Column assignment update matrices for I-divergence.

optimization algorithm (e.g., Bregman's algorithm or Iterative Scaling algorithm). Further, since the reconstructed $\tilde{Z}$ is defined in terms of the optimal Lagrange multipliers, we also need to compute these Lagrange parameters from the MBI solution. For the example under consideration, $\nabla\phi(z) = -\frac{1}{z}$. Hence, using the notation in Section 5.5, the matrix $\mathbf{A}$ for co-clustering basis $\mathcal{C}_1$ corresponds to a $(k+l) \times mn$ membership matrix where the rows correspond to the clusters (first k rows to row clusters and the next l rows to the column clusters) and the columns correspond to the elements of the matrix $\mathbf{Z}$ (or the corresponding $mn \times 1$ vector $\mathbf{z}$). Assuming $\mathbf{E}_{mn}$ is $mn \times 1$ vector consisting of all ones, the update steps in Bregman's algorithm (Section 5.5.1) are, therefore, given by

$$\begin{aligned} \mathbf{E}_{mn} \oslash \mathbf{z}'^{t+1} &= \mathbf{E}_{mn} \oslash \mathbf{z}'^{t} + \lambda_i A_i^T \\ A_i\mathbf{z}^{t+1} &= A_i\mathbf{z}, \end{aligned}$$

where A_i is the i^{th} row in $\mathbf{A}$ and $\lambda_i \in \mathbb{R}$. These updates are cyclically repeated over all the $k+l$ rows in $\mathbf{A}$. On convergence, we get the MBI solution $\hat{\mathbf{Z}}$ ($m \times n$ matrix) as well as the $k \times 1$ and $1 \times l$ matrices $\Lambda_{\hat{U}}, \Lambda_{\hat{V}}$ containing the optimal Lagrange multipliers.

2. **Row Cluster Assignment Step.** To obtain the row cluster assignment step, we first reconstruct $\tilde{\mathbf{Z}}$ for a candidate co-clustering $\mathbf{R}'$ using the Lagrange multipliers $\Lambda_{\hat{U}}$ and $\Lambda_{\hat{V}}$ computed in the previous step. More specifically, the reconstruction $\tilde{\mathbf{Z}}$ is given by

$$\tilde{\mathbf{Z}} = \mathbf{E}_{mn} \oslash (\bar{\mathbf{Z}} - \mathbf{R}'\Lambda_{\hat{U}}\mathbf{E}_n^T - \mathbf{E}_m\Lambda_{\hat{V}}\mathbf{C}^T), \tag{34}$$

that is, $\tilde{z}_{uv} = 1/(\bar{z} - \lambda_{\rho'(u)} - \lambda_{\gamma(v)})$.

Using (34) the row update cost function reduces to

$$\begin{aligned} & E_{V|u}[d_\phi(Z,\tilde{Z}] \\ &= E_{V|u}[Z/\tilde{Z} - \log(Z/\tilde{Z}) - 1] = \sum_{v=1}^{n} w_{uv}(z_{uv}/\tilde{z}_{uv} - \log(z_{uv}/\tilde{z}_{uv}) - 1) \\ &= \sum_{v=1}^{n} w_{uv}(z_{uv}(\bar{z} - \lambda_{\rho'(u)} - \lambda_{\gamma(v)}) - \log(z_{uv}) + \log(\bar{z} - \lambda_{\rho'(u)} - \lambda_{\gamma(v)}) - 1) \\ &= \sum_{v=1}^{n} w_{uv}(z_{uv}(\bar{z} - \lambda_{\gamma(v)}) - \log(z_{uv}) - 1) + \sum_{v=1}^{n} w_{uv}(-z_{uv}\lambda_{\rho'(u)} + \log(\bar{z} - \lambda_{\rho'(u)} - \lambda_{\gamma(v)})). \end{aligned}$$

Since the first term is independent of the row clustering, it is sufficient to optimize only the second term. Hence, the row assignment step is given by

$$\rho(u) = \operatorname*{argmin}_{g \in \{1,\cdots,k\}} \sum_{v=1}^{n} w_{uv}(-z_{uv}\lambda_g + \log(\bar{z} - \lambda_g - \lambda_{\gamma(v)})),\ [u]_1^m.$$

3. **Column Cluster Assignment Step.** The column assignment step can be similarly obtained by substituting the appropriate reconstructed matrix $\tilde{\mathbf{Z}}$ into the column update cost function and optimizing the part that depends on the column clustering, that is,

$$\gamma(v) = \operatorname*{argmin}_{h \in \{1,\cdots,l\}} \sum_{u=1}^{m} w_{uv}(-z_{uv}\lambda_h + \log(\bar{z} - \lambda_{\rho(u)} - \lambda_h)),\ [v]_1^n.$$

Appendix F. Notation

Notation	Usage	Introduced in
X, Y	Random variables over $\{x_1, \ldots, x_m\}$ and $\{y_1, \ldots, y_n\}$	Sec 1.1
m, n	Cardinality of support sets of X and Y	Sec 1.1
u, v	Indices over the sets $\{1, \cdots, m\}$ and $\{1, \cdots, n\}$	Sec 1.1
$\hat{X}, \hat{Y}$	Compressed/clustered versions of random variables X and Y	Sec 1.1
k, l	Number of row and column clusters	Sec 1.1
g, h	Indices over the sets $\{1, \cdots k\}$ and $\{1, \cdots, l\}$	Sec 1.1
$p(\cdot)$	Given joint (and induced) distributions over $X, Y, \hat{X}$ and $\hat{Y}$	Sec 1.1
$p'(\cdot)$	Candidate joint (and induced) distributions over $X, Y, \hat{X}$ and $\hat{Y}$	Sec 1.1
$q(\cdot)$	Max. entropy joint (and induced) distributions over $X, Y, \hat{X}$ and $\hat{Y}$	Sec 1.1
$p_0(\cdot)$	Uniform joint (and induced) distributions over $X, Y, \hat{X}$ and $\hat{Y}$	Sec 1.1
$\phi(\cdot)$	Strictly convex, differentiable function of Legendre type	Sec 2.1
$d_\phi(\cdot)$	Bregman divergence derived from ϕ	Sec 2.1
S	Effective domain of ϕ	Sec 2.1
z, z_i	Elements of S	Sec 2.1
Z	Random variable taking values in S	Sec 2.1
$\mathcal{Z}$	Support of Z	Sec 2.1
w	Probability measure associated with random variable Z	Sec 2.1
$\mathbf{Z}$	Matrix $\in S^{m \times n}$	Sec 2.1
U, V	Random variables over $\{1, \ldots, m\}$ and $\{1, \ldots, n\}$	Sec 2.2
ρ, γ	Row and column cluster mapping	Sec 2.3
$\hat{U}, \hat{V}$	Cluster random variables $\rho(U)$ and $\gamma(V)$	Sec 2.3
$\hat{\mathbf{Z}}$	Matrix approximation of $\mathbf{Z}$ (size $m \times n$)	Sec 2.3
$\hat{Z}$	Random variable approximating Z	Sec 2.3
Φ_w	Convex function induced on matrix by ϕ	Sec 2.3

Table 15: Notation used in the paper

Notation	Usage	Introduced in
$\hat{u}, \hat{v}$	Indices representing $\rho(u)$ and $\gamma(v)$	Sec 3.1
$\mathcal{S}_A$	Set of random variables preserving co-cluster means	Sec 3.1
$\hat{Z}_A$	Minimum Bregman information solution	Sec 3.1
Z'	Element of $\mathcal{S}_A$	Sec 3.1
$\mathcal{S}_B$	Set of random variables that are functions of co-cluster means	Sec 3.1
$\hat{Z}_B$	Best approximation to Z in $\mathcal{S}_B$	Sec 3.1
Z''	Element of $\mathcal{S}_B$	Sec 3.1
$\hat{Z}$	Same as $\hat{Z}_A$ and $\hat{Z}_B$	Sec 3.1
(ρ^*, γ^*)	Optimal row and column clustering	Sec 3.2
$\mu_{\hat{u},\hat{v}}$	co-cluster mean $E[Z\|\hat{u}, \hat{v}]$	Sec 3.3
$J_u(\cdot)$	Contribution of u^{th} row to the objective function	Sec 3.3
ρ^t	Row clustering in the t^{th} iteration	Sec 3.3
γ^t	Column clustering in the t^{th} iteration	Sec 3.3
$\hat{Z}^t$	MBI solution corresponding to (ρ^t, γ^t)	Sec 3.3
$\tilde{Z}^t$	Row permuted version of $\hat{Z}^t$ according to ρ^t	Sec 3.3
$\mathbf{R}$	Row assignment matrix (size $m \times k$)	Sec 3.4
$\mathbf{C}$	Column assignment matrix (size $n \times l$)	Sec 3.4
$\mathbf{M}$	Co-cluster mean matrix (size $k \times l$)	Sec 3.4
$U_\emptyset$	Constant random variable over rows	Sec 4.1
$V_\emptyset$	Constant random variable over columns	Sec 4.1
Γ_1	Set of index random variables	Sec 4.1
Γ_2	Unique sub-σ-algebra of Z	Sec 4.1
$\mathcal{C}, \mathcal{C}_i$	Co-clustering basis	Sec 4.1
$\mathcal{G}, \mathcal{G}_i$	Sub-σ algebra corresponding to co-clustering basis	Sec 4.1
s	Total number of constraints in a co-clustering basis	Sec 4.2
r	Index over the set $\{1, \cdots s\}$	Sec 4.2
$\Lambda^*_{\mathcal{G}_r}, \Lambda_{\mathcal{G}_r}$	(Optimal) Lagrange multipliers associated with $\mathcal{G}_r$	Sec 4.2
$w_{\mathcal{G}_r}$	Induced measure on $\mathcal{G}_r$	Sec 4.2
$J(\cdot)$	Lagrangian for the minimum Bregman information problem	Sec 4.2
$L(\cdot)$	Lagrange dual of the Bregman information	Sec 4.2
$\mathcal{S}_A$	Set of random variables preserving summary statistics	Sec 4.2
$\hat{Z}_A$	MBI solution in $\mathcal{S}_A$	Sec 4.2
Z'	Element of $\mathcal{S}_A$	Sec 4.2
ψ	Legendre conjugate of ϕ	Sec 4.4
Θ	Domain of ψ	Sec 4.4
$\theta_{\mathcal{G}_r}$	Random variables corresponding to $E[Z\|\mathcal{G}_r]$ in Θ	Sec 4.4
Θ_B	Set of generalized additive models of $\theta_{\mathcal{G}_r}$ in Θ space	Sec 4.4
θ''	Element of Θ_B	Sec 4.4
$\mathcal{S}_B$	Set of generalized additive models of summary statistics in Θ space	Sec 4.4
$\hat{Z}_B$	Best approximation to Z in $\mathcal{S}_B$	Sec 4.4
Z''	Element of $\mathcal{S}_B$	Sec 4.4
$g_r(\cdot)$	Arbitrary function of $E[Z\|\mathcal{G}_r]$ and $\theta_{\mathcal{G}_r}$	Sec 4.4

Table 16: Notation used in the paper

Notation	Usage	Introduced in
$\zeta(\rho,\gamma,\Lambda)$	Functional form of the min. Bregman information solution for (ρ,γ) with Lagrange multipliers Λ possibly instead of optimal Λ^*	Sec 5.2
$\xi(U,\rho(U),V,\gamma(V))$	Objective function $E[d_\phi(Z,\tilde{Z})]$	Sec 5.2
$\mathbf{z},\hat{\mathbf{z}},\mathbf{z}'$	Vectorized versions of $\mathbf{Z}$, $\hat{\mathbf{Z}}$ and Z' respectively	Sec 5.5
$\bar{\mathbf{z}}$	$mn \times 1$ vector with all values $= E[Z]$	Sec 5.5
$\mathbf{A}$	Matrix corresponding to the linear conditional expectation constraints	Sec 5.5
c	Number of linear constraints (rows in $\mathbf{A}$)	Sec 5.5
$\mathcal{L}_\phi$	Legendre-Bregman projection derived from ϕ	Sec 5.5
λ_i,λ	Lagrange multipliers corresponding to A_i and $\mathbf{A}$ resp.	Sec 5.5
$\mathbf{z}'_0$	Initial choice of $\mathbf{z}'$	Sec 5.5
s_{ij}	Sign of A_{ij}	Sec 5.5
N_j	Upper bound on L_1 norm of j^{th} column of $\mathbf{A}$	Sec 5.5
$\mathbf{W}$	$m \times n$ matrix corresponding to the measure w	Sec E.1
$\mathbf{E}_m$ ($\mathbf{E}_n$)	constant $m \times 1$ ($n \times 1$) vector consisting of all ones	Sec E.1
$\bar{\mathbf{Z}}_\mathcal{G}$	Matrix of conditional expectations over $\mathcal{G}$	Sec E.1
$\mathbf{Z}^f_\mathcal{G}$	$m \times n$ matrix expansion of $\bar{\mathbf{Z}}_\mathcal{G}$	Sec E.1
$\tilde{\mathbf{Z}}$	Matrix corresponding to $\tilde{Z}$	Sec E.2
ρ',γ'	Candidate row and column clustering	Sec E.2
$\mathbf{R}',\mathbf{C}'$	Candidate row and column membership matrices	Sec E.2
$\tilde{\mathbf{Z}}^{rowVar}$	Variable part of $\tilde{\mathbf{Z}}$ during row clustering (size $k \times n$)	Sec E.2
$\tilde{\mathbf{Z}}^{rowConst}$	Constant part of $\tilde{\mathbf{Z}}$ during row clustering (size $m \times n$)	Sec E.2
$\mathbf{Z}^{row}$	Constant matrix determining row-clustering (size $m \times n$)	Sec E.2
$\mathbf{Z}^{rowRed}$	Reduced representation of $\mathbf{Z}^{row}$ (size $m \times l$)	Sec E.2
$\tilde{\mathbf{Z}}^{rowVRed}$	Reduced representation of $\tilde{\mathbf{Z}}^{rowVar}$ (size $k \times l$)	Sec E.2
$\tilde{\mathbf{Z}}^{colVar}$	Variable part of $\tilde{\mathbf{Z}}$ during column clustering (size $m \times l$)	Sec E.2
$\tilde{\mathbf{Z}}^{colConst}$	Constant part of $\tilde{\mathbf{Z}}$ during column clustering (size $m \times n$)	Sec E.2
$\mathbf{Z}^{col}$	Constant matrix determining column clustering (size $m \times n$)	Sec E.2
$\mathbf{Z}^{colRed}$	Reduced representation of $\mathbf{Z}^{col}$ (size $k \times n$)	Sec E.2
$\tilde{\mathbf{Z}}^{colVRed}$	Reduced representation of $\tilde{\mathbf{Z}}^{colVar}$ (size $k \times l$)	Sec E.2
$\mathbf{E}_{m\times n}$	$m \times n$ matrix consisting of all ones	Sec E.2

Table 17: Notation used in the paper

References

K. S. Azoury and M. K. Warmuth. Relative loss bounds for on-line density estimation with the exponential family of distributions. *Machine Learning*, 43(3):211–246, 2001.

A. Banerjee, X. Guo, and H. Wang. On the optimality of conditional expectation as a Bregman predictor. *IEEE Transactions on Information Theory*, 51(7):2664–2669, July 2005a.

A. Banerjee, S. Merugu, I. Dhillon, and J. Ghosh. Clustering with Bregman divergences. *Journal of Machine Learning Research*, 6:1705–1749, 2005b.

H. H. Bauschke and J. M. Borowein. Legendre functions and the method of random Bregman projections. *Journal of Convex Analysis*, 4(1):27–67, 1997.

R. Bekkerman, R. El-Yaniv, and A. McCallum. Multi-way distributional clustering via pairwise interactions. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 41–48, 2005.

L. M. Bregman. The relaxation method of finding the common point of convex sets and its application to the solution of problems in convex programming. *USSR Computational Mathematics and Physics*, 7:200–217, 1967.

R. Cai, L. Lu, and L. Cai. Unsupervised auditory scene categorization via key audio effects and information-theoretic co-clustering. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP05)*, pages 1073–1076, 2005.

J. J. M. Carrasco, D. Fain, K. Lang, and L. Zhukov. Clustering of bipartite advertiser-keyword graph. In *Proceedings of the Workshop on Large Scale Clustering, ICDM*, 2003.

Y. Censor and S. Zenios. *Parallel Optimization: Theory, Algorithms, and Applications*. Oxford University Press, 1998.

D. Chakrabarti, S. Papadimitriou, D. S. Modha, and C. Faloutsos. Fully automatic cross-associations. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 79–88, 2004.

Y. Cheng and G. M. Church. Biclustering of expression data. In *Proceedings of the 8th International Conference on Intelligent Systems for Molecular Biology (ISMB)*, pages 93–103, 2000.

H. Cho, I. S. Dhillon, Y. Guan, and S. Sra. Minimum sum-squared residue co-clustering of gene expression data. In *Proceedings of the 4th SIAM International Conference on Data Mining (SDM)*, pages 114–125, 2004.

M. Collins, R. E. Schapire, and Y. Singer. Logistic regression, adaboost and bregman distances. In *Proceedings of the 13th Annual Conference on Computational Learing Theory (COLT)*, pages 158–169, 2000.

T. M. Cover and J. A. Thomas. *Elements of Information Theory*. Wiley-Interscience, 1991.

I. Csiszár. Why least squares and maximum entropy? An axiomatic approach to inference for linear inverse problems. *The Annals of Statistics*, 19:2032–2066, 1991.

S. Della Pietra, V. Della Pietra, and J. Lafferty. Duality and auxiliary functions for Bregman distances. Technical Report CMU-CS-01-109, School of Computer Science, Carnegie Mellon University, 2001.

I. Dhillon, S. Mallela, and R. Kumar. A divisive information-theoretic feature clustering algorithm for text classification. *Journal of Machine Learning Research*, 3(4):1265–1287, 2003a.

I. Dhillon, S. Mallela, and D. Modha. Information-theoretic co-clustering. In *Proceedings of the 9th International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 89–98, 2003b.

I. S. Dhillon. Co-clustering documents and words using bipartite spectral graph partitioning. In *Proceedings of the 7th International Conference on Knowledge Discovery and Data Mining*, pages 269–274, 2001.

I. S. Dhillon and D. S. Modha. Concept decompositions for large sparse text data using clustering. *Machine Learning*, 42(1):143–175, January 2001.

D. Freitag. Trained named entity recognition using distributional clusters. In *EMNLP*, pages 262–269, 2004.

B. Gao, T. Liu, X. Zheng, Q. Cheng, and W. Ma. Consistent bipartite graph co-partitioning for star-structured high-order heterogeneous data co-clustering. In *Proceedings of the 11th International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 41–50, 2005.

T. George and S. Merugu. A scalable collaborative filtering framework based on co-clustering. In *Proceedings of the IEEE Conference on Data Mining*, pages 625–628, 2005.

J. Ghosh. Scalable clustering. In Nong Ye, editor, *The Handbook of Data Mining*, pages 247–277. Lawrence Erlbaum Assoc., 2003.

GroupLens. Movielens data set. http://www.cs.umn.edu/Research/GroupLens/data/ml-data.tar.gz.

P. D. Grünwald and A. Dawid. Game theory, maximum entropy, minimum discrepancy, and robust Bayesian decision theory. *Annals of Statistics*, 32(4), 2004.

J. Guan, G. Qiu, and X. Y. Xue. Spectral images and features co-clustering with application to content-based image retrieval. In *IEEE Workshop on Multimedia Signal Processing*, 2005.

J. A. Hartigan. Direct clustering of a data matrix. *Journal of the American Statistical Association*, 67(337):123–129, 1972.

T. Hofmann. Latent semantic models for collaborative filtering. *ACM Transactions on Information Systems*, 22(1):89–115, 2004.

T. Hofmann and J. Puzicha. Unsupervised learning from dyadic data. Technical Report ICSI TR-98-042, International Computer Science Institute (ICSI), Berkeley, 1998.

A. K. Jain and R. C. Dubes. *Algorithms for Clustering Data*. Prentice Hall, New Jersey, 1988.

E. T. Jaynes. Information theory and statistical mechanics. *Physical Reviews*, 106:620–630, 1957.

Y. Kluger, R. Basri, J. T. Chang, and M. Gerstein. Spectral biclustering of microarray data: Coclustering genes and conditions. *Genome Research*, 13(4):703–716, 2003.

J. Lafferty. Additive models, boosting, and inference for generalized divergences. In *Proceedings of the 13th Annual Conference on Computational Learing Theory (COLT)*, 1999.

D. L. Lee and S. Seung. Algorithms for non-negative matrix factorization. In *Proceedings of the 14th Annual Conference on Neural Information Processing Systems (NIPS)*, pages 556–562, 2001.

H. Li and N. Abe. Word clustering and disambiguation based on co-occurence data. In *COLING-ACL*, pages 749–755, 1998.

T. Li. A general model for clustering binary data. In *Proceedings of the 11th International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 188–197, 2005.

Z. Lin and R.B. Altman. Finding haplotype tagging snps by use of principal components analysis. *The American Journal of Human Genetics*, 75:850–861, 2004.

S. C. Madeira and A. L. Oliveira. Biclustering algorithms for biological data analysis: A survey. *IEEE Trans. Computational Biology and Bioinformatics*, 1(1):24–45, 2004.

C. H. Papadimitriou, P. Raghavan, H. Tamaki, and S. Vempala. Latent semantic indexing: A probabilistic analysis. In *Proceedings of the 16th Annual ACM Symposium on Principles of Distributed Computing (PODC)*, pages 159–168, 1998.

L. Parsons, E. Haque, and H. Liu. Subspace clustering for high dimensinal data: A review. *ACM SIGKDD Explorations*, 6(1):90–105, 2004.

G. Qiu. Image and feature co-clustering. In *Proceedings of the International Conference on Pattern Recognition*, pages 991–994, 2004.

P. Resnick, N. Iacovou, M. Suchak, P. Bergstorm, and J. Riedl. GroupLens: An Open Architecture for Collaborative Filtering of Netnews. In *Proceedings of the ACM Conference on CSCW*, pages 175–186, 1994.

R. T. Rockafellar. *Convex Analysis*. Princeton Landmarks in Mathematics. Princeton University Press, 1970.

R. Rohwer and D. Freitag. Towards full automation of lexicon construction. In Dan Moldovan and Roxana Girju, editors, *HLT-NAACL 2004: Workshop on Computational Lexical Semantics*, pages 9–16, 2004.

B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Application of dimensionality reduction in recommender systems–a case study. In *WebKDD Workshop*., 2000.

J. Shore and R. Johnson. Axiomatic derivation of the principle of maximum entropy and the principle of minimum cross entropy. *IEEE Transactions on Information Theory*, 26(1):26–37, 1980.

A. Strehl and J. Ghosh. Cluster ensembles – a knowledge reuse framework for combining partitionings. *Journal of Machine Learning Research*, 3(3):583–617, 2002.

H. Takamura and Y. Matsumoto. Co-clustering for text categorization. *Information Processing Society of Japan Journal*, 2003.

H. Zhong, J. Shi, and M. Visontai. Detecting unusual activity in video. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition*, pages 819–826, 2004.

Journal of Machine Learning Research 8 (2007) 1987-2016 Submitted 12/06; Revised 7/07; Published 9/07

Truncating the Loop Series Expansion for Belief Propagation

Vicenç Gómez* VGOMEZ@IUA.UPF.EDU
Departament de Tecnologies de la Informació i les Comunicacions
Universitat Pompeu Fabra
Passeig de Circumval·lació 8, 08003 Barcelona, Spain

Joris M. Mooij J.MOOIJ@SCIENCE.RU.NL
Hilbert J. Kappen B.KAPPEN@SCIENCE.RU.NL
Department of Biophysics
Radboud University Nijmegen
6525 EZ Nijmegen, The Netherlands

Editor: Michael Jordan

Abstract

Recently, Chertkov and Chernyak (2006b) derived an exact expression for the partition sum (normalization constant) corresponding to a graphical model, which is an expansion around the belief propagation (BP) solution. By adding correction terms to the BP free energy, one for each "generalized loop" in the factor graph, the exact partition sum is obtained. However, the usually enormous number of generalized loops generally prohibits summation over *all* correction terms. In this article we introduce truncated loop series BP (TLSBP), a particular way of truncating the loop series of Chertkov & Chernyak by considering generalized loops as compositions of simple loops. We analyze the performance of TLSBP in different scenarios, including the Ising model on square grids and regular random graphs, and on PROMEDAS, a large probabilistic medical diagnostic system. We show that TLSBP often improves upon the accuracy of the BP solution, at the expense of increased computation time. We also show that the performance of TLSBP strongly depends on the degree of interaction between the variables. For weak interactions, truncating the series leads to significant improvements, whereas for strong interactions it can be ineffective, even if a high number of terms is considered.

Keywords: belief propagation, loop calculus, approximate inference, partition function, Ising grid, random regular graphs, medical diagnosis

1. Introduction

Belief propagation (Pearl, 1988; Murphy et al., 1999) is a popular inference method that yields exact marginal probabilities on graphs without loops and can yield surprisingly accurate results on graphs with loops. BP has been shown to outperform other methods in rather diverse and competitive application areas, such as error correcting codes (Gallagher, 1963; McEliece et al., 1998), low level vision (Freeman et al., 2000), combinatorial optimization (Mézard et al., 2002) and stereo vision (Sun et al., 2005).

Associated to a probabilistic model is the partition sum, or normalization constant, from which marginal probabilities can be obtained. Exact calculation of the partition function is only feasible

*. Visiting Radboud University Nijmegen

for small problems, and there is considerable statistical physics literature devoted to the approximation of this quantity. Existing methods include stochastic Monte Carlo techniques (Potamianos and Goutsias, 1997) or deterministic algorithms which provide lower bounds (Jordan et al., 1999; Leisink and Kappen, 2001), upper bounds (Wainwright et al., 2005), or approximations (Yedidia et al., 2005).

Recently, Chertkov and Chernyak (2006b) have presented a loop series expansion formula that computes correction terms to the belief propagation approximation of the partition sum. The series consists of a sum over all so-called generalized loops in the graph. When all loops are taken into account, Chertkov & Chernyak show that the exact result is recovered. Since the number of generalized loops in a graphical model easily exceeds the number of configurations of the model, one could argue that the method is of little practical value. However, if one could truncate the expansion in some principled way, the method could provide an efficient improvement to BP.[1]

Most inference algorithms on loopy graphs can be viewed as generalizations of BP, where messages are propagated between regions of variables. For instance, the junction-tree algorithm (Lauritzen and Spiegelhalter, 1988) which transforms the original graph in a region tree such that the influence of all loops in the original graph is implicitly captured, and the exact result is obtained. However, the complexity of this algorithm is exponential in time and space on the size of the largest clique of the resulting join tree, or equivalently, on the tree-width of the original graph, a parameter which measures the network complexity. Therefore, for graphs with high tree-width one is resorted to approximate methods such as Monte Carlo sampling or generalized belief propagation (GBP) (Yedidia et al., 2005), which captures the influence of short loops using regions which contain them. One way to select valid regions is the cluster variation method (CVM) (Pelizzola, 2005). In general, selecting a good set of regions is not an easy task, as described by Welling et al. (2005). Alternatively, double-loop methods can be used (Heskes et al., 2003; Yuille, 2002) which are guaranteed to converge, often at the cost of more computation time.

In this work we propose TLSBP, an algorithm to compute generalized loops in a graph which are then used for the approximate computation of the partition sum and the single-node marginals. The proposed algorithm is parametrized by two arguments which are used to prune the search for generalized loops. For large enough values of these parameters, all generalized loops present in a graph are retrieved and the exact result is obtained. One can then study how the error is progressively corrected as more terms are considered in the series. For cases were exhaustive computation of all loops is not feasible, the search can be pruned, and the result is a truncated approximation of the exact solution. We focus mainly on problems where BP converges easily, without the need of damping or double loop alternatives (Heskes et al., 2003; Yuille, 2002) to force convergence. It is known that accuracy of the BP solution and convergence rate are negatively correlated. Throughout the paper we show evidence that for those cases where BP has difficulties to converge, loop corrections are of little use, since loops of all lengths tend to have contributions of similar order of magnitude.

The paper is organized as follows. In Section 2 we briefly summarize the series expansion method of Chertkov and Chernyak (2006b). In Section 3 we provide a formal characterization of the different types of generalized loops that can be present in an arbitrary graph. This description is relevant to understand the proposed algorithm described in Section 4. We present experimental results in Section 5 for the Ising model on grids, regular random graphs and medical diagnosis.

1. Note that the number of generalized loops in a finite graph is finite, and strictly speaking, the term *series* denotes an *infinite* sequence of terms. For clarity, we prefer to use the original terminology.

Concerning grids and regular graphs, we show that the success of restricting the loop series expansion to a reduced quantity of loops depends on the type of interactions between the variables in the network. For weak interactions, the largest correction terms come from the small elementary loops and therefore truncation of the series at some maximal loop length can be effective. For strong interactions, loops of all lengths contribute significantly and truncation is of limited use. We numerically show that when more loops are taken into account, the error of the partition sum decreases and when all loops are taken into account the method is correct up to machine precision. We also apply the truncated loop expansion to a large probabilistic medical diagnostic decision support system (Wiegerinck et al., 1999). The network has 2000 diagnoses and about 1000 findings and is intractable for computation. However, for each patient case unobserved findings and irrelevant diagnoses can be pruned from the network. This leaves a much smaller network that may or may not be tractable depending on the set of clamped findings. For a number of patient cases, we compare the BP approximation and the truncated loop correction. We show results and characterize when the loop corrections significantly improve the accuracy of the BP solution. Finally, in Section 6 we provide some concluding remarks.

2. BP and the Loop Series Expansion

Consider a probability model on a set of binary variables $x_i = \pm 1, i = 1, \ldots, n$:

$$P(x) = \frac{1}{Z} \prod_{\alpha=1}^{m} f_\alpha(x_\alpha), \qquad Z = \sum_x \prod_{\alpha=1}^{m} f_\alpha(x_\alpha), \tag{1}$$

where $\alpha = 1, \ldots, m$ labels interactions (factors) on subsets of variables x_α, and Z is the partition function, which sums over all possible states or variable configurations. Note that the only restriction here is that variables are binary, since arbitrary factor nodes are allowed, as in Chertkov and Chernyak (2006b).

The probability distribution in (1) can be directly expressed by means of a factor graph (Kschischang et al., 2001), a bipartite graph where variable nodes i are connected to factor nodes α if and only if x_i is an argument of f_α. Figure 3 (left) on page 1999 shows an example of a graph where variable and factor nodes are indicated by circles and squares respectively.

For completeness, we now briefly summarize Pearl's belief propagation (BP) (Pearl, 1988) and define the Bethe free energy. If the graph is acyclic, BP iterates the following message update equations, until a fixed point is reached:

$$\textbf{variable } i \textbf{ to factor } \alpha\textbf{:} \qquad \mu_{i\to\alpha}(x_i) = \prod_{\beta \ni i \setminus \{\alpha\}} \mu_{\beta\to i}(x_i),$$

$$\textbf{factor } \alpha \textbf{ to variable } i\textbf{:} \qquad \mu_{\alpha\to i}(x_i) = \sum_{x_{\alpha\setminus\{i\}}} f_\alpha(x_\alpha) \prod_{j\in\alpha\setminus\{i\}} \mu_{j\to\alpha}(x_j),$$

where $i \in \alpha$ denotes variables included in factor α, and $\alpha \ni i$ denotes factor indices α which have i as argument. After the fixed point is reached, exact marginals and correlations associated with the factors ("beliefs") can be computed using:

$$b_i(x_i) \propto \prod_{\alpha\ni i} \mu_{\alpha\to i}(x_i),$$

$$b_\alpha(x_\alpha) \propto f_\alpha(x_\alpha) \prod_{i\in\alpha} \mu_{i\to\alpha}(x_i),$$

where $\propto$ indicates normalization so that beliefs sum to one.

For graphs with cycles the same update equations can be iterated (the algorithm is then called loopy, or iterative, belief propagation), and one can still obtain very accurate approximations of the beliefs. However, convergence is not guaranteed in these cases. For example, BP can get stuck in limit cycles. An important step towards the understanding and characterization of the convergence properties of BP came from the observation that fixed points of this algorithm correspond to stationary points of a particular function of the beliefs, known as the Bethe free energy (Yedidia et al., 2000), which is defined as:

$$F_{BP} = U_{BP} - H_{BP}, \tag{2}$$

where U_{BP} is the Bethe average energy:

$$U_{BP} = -\sum_{\alpha=1}^{m} \sum_{x_\alpha} b_\alpha(x_\alpha) \log f_\alpha(x_\alpha),$$

and H_{BP} is the Bethe approximate entropy:

$$H_{BP} = -\sum_{\alpha=1}^{m} \sum_{x_\alpha} b_\alpha(x_\alpha) \log b_\alpha(x_\alpha) + \sum_{i=1}^{n} (d_i - 1) \sum_{x_i} b_i(x_i) \log b_i(x_i), \tag{3}$$

where d_i is the number of neighboring factor nodes of variable node i. The second term in (3) ensures that every node in the graph is counted once (see Yedidia et al., 2005, for details). The BP algorithm tries to minimize (2) and, for trees, the exact partition function can be obtained after the fixed point has been reached, $Z = \exp(-F_{BP})$. However, for graphs with loops, F_{BP} provides just an approximation.

If one can calculate the exact partition function Z defined in Equation (1), one can also calculate any marginal in the network. For instance, the marginal

$$P_i(x_i) = \left. \frac{\partial \log Z(\theta_i)}{\partial \theta_i(x_i)} \right|_{\theta_i \to 0}, \qquad \text{where} \qquad Z(\theta_i) := \sum_x e^{\theta_i x_i} \prod_{\alpha=1}^{m} f_\alpha(x_\alpha)$$

is the partition sum of the network, perturbed by an additional local field potential θ_i on variable x_i.

Alternatively, one can compute different partition functions for different settings of the variables, and derive the marginals from ratios of them:

$$P_i(x_i) = \frac{Z^{x_i}}{\sum_{x_i'} Z^{x_i'}}, \tag{4}$$

where Z^{x_i} indicates the partition function calculated from the same model conditioning on variable i, that is, with variable i fixed (clamped) to value x_i. Therefore, approximation errors in the computation of any marginal can be related to approximation errors in the computation of Z. We will thus focus on the approximation of Z mainly, although marginal probabilities will be computed as well.

Of central interest in this work is the concept of generalized loop, which is defined in the following way:

Definition 1 *A **generalized loop** in a graph $G = \langle V, E \rangle$ is any subgraph $C = \langle V', E' \rangle$, $V' \subseteq V, E' \subseteq (V' \times V') \cap E$ such that each node in V' has degree two or larger. The length (size) of a generalized loop is its number of edges.*

For the rest of the paper, the terms loop and generalized loop are used interchangeably. The main result of Chertkov and Chernyak (2006b) is the following. Let $b_\alpha(x_\alpha), b_i(x_i)$ denote the beliefs after the BP algorithm has been converged, and let $Z_{\mathrm{BP}} = \exp(-F_{\mathrm{BP}})$ denote the corresponding approximation to the partition sum, with F_{BP} the value of the Bethe free energy evaluated at the BP solution. Then Z_{BP} is related to the exact partition sum Z as:

$$Z = Z_{\mathrm{BP}} \left(1 + \sum_{C \in \mathcal{C}} r(C) \right), \qquad r(C) = \prod_{i \in C} \mu_i(C) \prod_{\alpha \in C} \mu_\alpha(C), \tag{5}$$

where summation is over the set $\mathcal{C}$ of all generalized loops in the factor graph. Any term $r(C)$ in the series corresponds to a product with as many factors as nodes present in the loop. Each factor is related to the beliefs at each variable node or factor node according to the following formulas:

$$\mu_i(C) = \frac{(1-m_i)^{q_i(C)-1} + (-1)^{q_i(C)}(1+m_i)^{q_i(C)-1}}{2(1-m_i^2)^{q_i(C)-1}}, \qquad q_i(C) = \sum_{\alpha \in C, \alpha \ni i} 1, \tag{6}$$

$$\mu_\alpha(C) = \sum_{x_\alpha} b_\alpha(x_\alpha) \prod_{i \in C, i \in \alpha} (x_i - m_i), \tag{7}$$

where $m_i = \sum_{x_i} b_i(x_i) x_i = b_i(+) - b_i(-)$ is the expected value of x_i computed in the BP approximation. Generally, terms $r(C)$ can take positive or negative values. Even the same variable i may have positive or negative subterms μ_i depending on the structure of the particular loop.

Expression (5) represents an exact and finite decomposition of the partition function with the first term of the series being exactly represented by the BP solution. Note that, although the series is finite, the number of generalized loops in the factor graph can be enormous and easily exceed the number of configurations 2^n. In these cases the loop series is less efficient than the most naive way to compute Z exactly, namely by summing the contributions of all 2^n configurations one by one.

On the other hand, it may be that restricting the sum in (5) to a subset of the total generalized loops captures the most important corrections and may yield a significant improvement in comparison to the BP estimate. We therefore define the truncated form of the loop corrected partition function as:

$$Z_{TLSBP} = Z_{\mathrm{BP}} \left(1 + \sum_{C \in \mathcal{C}'} r(C) \right), \tag{8}$$

where summation is over the subset $\mathcal{C}' \subseteq \mathcal{C}$ obtained by Algorithm 2, which we will discuss in Section 4. Approximations for the single-node marginals can then be obtained from (8), using the method proposed in Equation (4):

$$b_i'(x_i) = \frac{Z_{TLSBP}^{x_i}}{\sum_{x_i'} Z_{TLSBP}^{x_i'}}. \tag{9}$$

Because the the terms $r(C)$ can have different signs, the approximation Z_{TLSBP} is in general not a bound of the exact Z, but just an approximation.

3. Loop Characterization

In this section we characterize different types of generalized loops that can be present in a graph. This classification is the basis of the algorithm described in the next section and also exemplifies the different shapes a generalized loop can take. For clarity, we illustrate them by means of a factor graph arranged in a square lattice with only pairwise interactions. However, definitions are not restricted to this particular model and can be applied generally to any factor graph.

Definition 2 *A simple (elementary) generalized loop (from now on **simple loop**) is defined as a connected subgraph of the original graph where all nodes have exactly degree two.*

This type of generalized loop coincides with the concept of simple circuit or simple cycle in graph theory: a path which starts and ends at the same node with no repeated vertices except for the start and end vertex. Figure 1a shows an example of a simple loop of size 8. On the contrary, in Figure 1b we show an example of generalized loop which is not a simple loop, because three nodes have degree larger than two.

We now define the union of two generalized loops, $l_1 = \langle V_1, E_1 \rangle$ and $l_2 = \langle V_2, E_2 \rangle$, as the generalized loop which results from taking the union of the vertices and the edges of l_1 and l_2, that is, $l' = l_1 \cup l_2 = \langle V_1 \cup V_2, E_1 \cup E_2 \rangle$. Note that the union of two simple loops is never a simple loop except for the trivial case in which both loops are equal. Figure 1b shows an example of a generalized loop which can be described as the union of three simple loops, each of size 8. The same example can be also defined as the union of two overlapping simple loops, each of size 12.

Definition 3 *A disconnected generalized loop, **disconnected loop**, is defined as a generalized loop with more than one connected component.*

Figure 1c shows an example of a disconnected loop composed of three simple loops. Note that components are not restricted to be simple loops. Figure 1d illustrates this fact using an example where one connected component (the left one) is not a simple loop.

Definition 4 *A complex generalized loop, **complex loop**, is defined as a generalized loop which cannot be expressed as the union of two or more different simple loops.*

Figures 1e and 1f are examples of complex loops. Intuitively, they result after the connection of two or more connected components of a disconnected loop.

Any generalized loop can be categorized according to these three different categories: a simple loop cannot be a disconnected loop, neither a complex loop. On the other hand, since Definitions 3 and 4 are not mutually exclusive, a disconnected loop can be a complex loop and vice-versa, and also there are generalized loops which are neither disconnected nor complex, for instance the example of Figure 1b. An example of a disconnected loop which is not a complex loop is shown in Figure 1c. An example of a complex loop which is not a disconnected loop is shown in Figure 1e. Finally, an example of a complex loop which is also a disconnected loop is shown in Figure 1f.

We finish this characterization using a diagrammatic representation in Figure 2 which illustrates the definitions. Usually, the smallest subset contains the simple loops and both disconnected loops and complex loops have nonempty intersection. There is another subset of all generalized loops which are neither simple, disconnected, nor complex.

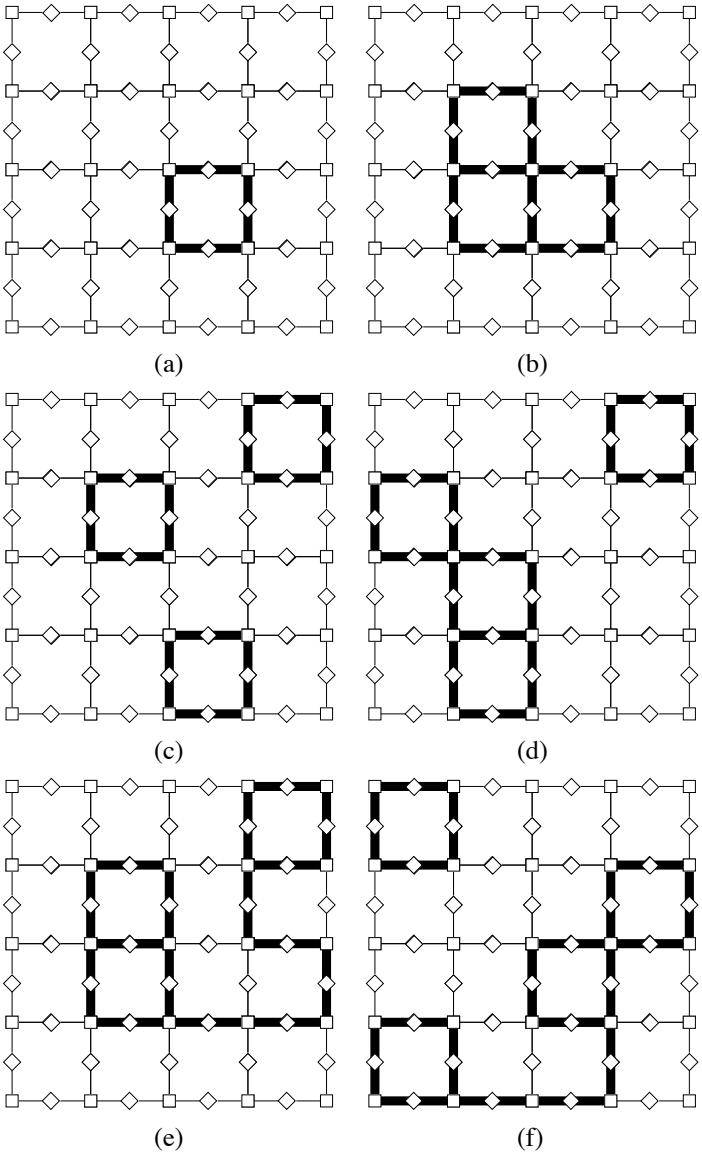


Figure 1: Examples of generalized loops in a factor graph with lattice structure. Variable nodes and factor nodes are represented as squares and rhombus respectively. Generalized loops are indicated using bold edges underlying the factor graph. **(a)** A simple loop. **(b)** A non-simple loop which is neither a disconnected loop nor a complex loop. **(c)** A disconnected loop of three components, each a simple loop. **(d)** A disconnected loop of two components, the left one a non-simple loop. **(e)** A complex loop which is not a disconnected loop. **(f)** A complex loop which is also a disconnected loop. *(See text for definitions)*.

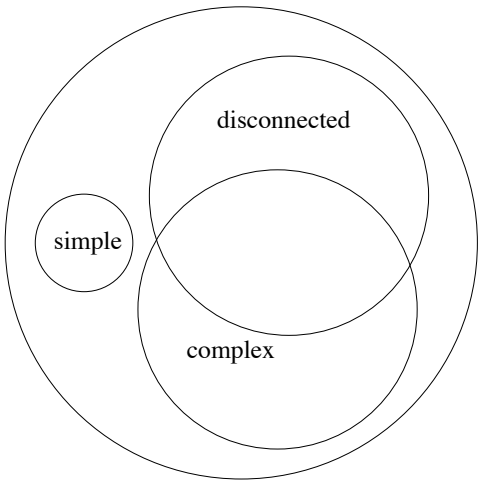


Figure 2: Diagrammatic representation of the different types of generalized loops present in any graph. Sizes of the sets are just indicative and depend on the particular instance.

4. The Truncated Loop Series Algorithm

In this section we describe the TLSBP algorithm to compute generalized loops in a factor graph, and use them to correct the BP solution. The algorithm is based on the principle that every generalized loop can be decomposed in smaller loops. The general idea is to search first for a subset of the simple loops and, after that, merge all of them iteratively until no new loops are produced. As expected, a brute force search algorithm will only work for small instances. We therefore prune the search using two different bounds as input arguments. Eventually, a high number of generalized loops which presumably will account for the major contributions in the loops series expansion will be obtained. We show that the algorithm is complete, or equivalently, that all generalized loops are obtained by the proposed approach when the constraints expressed by the two arguments are relaxed. Although exhaustive enumeration is of little interest for complex instances, it allows to check the validity of (5) and to study the loop series expansion for simpler instances. The algorithm is composed of three steps:

1. First, we remove recursively all the leaves of the original graph, until its 2-core is obtained. This initial step has two main advantages. On the one hand, since some nodes are deleted, the complexity of the problem is reduced. On the other hand, we can use the resulting graph as a test for any possible improvement to the BP solution. Indeed, if the original graph did not contain any loop then the null graph is obtained, the BP solution is exact on the original graph, and the series expansion has only one term. On the other hand, if a nonempty graph remains after this preprocessing, it will have loops and the BP solution can be improved using the proposed approach.

2. After the graph is preprocessed, the second step searches for simple loops. The result of this search will be the initial set of loops for the next step and will also provide a bound b which will be used to truncate the search for new generalized loops. Finding circuits in a

graph is a problem addressed for long (Tiernan, 1970; Tarjan, 1973; Johnson, 1975) whose computational complexity grows exponentially with the length of the cycle (Johnson, 1975). Nevertheless, we do not count all the simple loops but only a subset. Actually, to avoid dependence on particular instances, we parametrize this search by a size S, which limits the number of shortest simple loops to be considered. Once S simple loops have been found in order of increasing length, the length of the largest simple loop is used as the bound b for the remaining steps.

3. The third step of the algorithm consists of obtaining all non-simple loops that the set of S simple loops can"generate".

According to definition 4, complex loops can not be expressed as union of simple loops. To develop a complete method, in the sense that all existing loops can be obtained, we define the operation *merge loops*, which extends the simple union in such a way that complex loops are retrieved as well. Given two generalized loops, l_1, l_2, *merge loops* returns a *set* of generalized loops. One can observe that for each disconnected loop, a set of complex loops can be generated by connecting two (or more) components of the disconnected loop. In other words, complex loops can be expressed as the union of disjoint loops with a path connecting two vertices of different components. Therefore the set computed by *merge loops* will have only one element $l' = \{l_1 \cup l_2\}$ if $l_1 \cup l_2$ is not disconnected. Otherwise, all the possible complex loops in which $l_1 \cup l_2$ appears are included in the resulting set.

We use the following procedure to compute all complex loops associated to the disconnected loop l': we start at a vertex of a connected component of l' and perform depth-first-search (DFS) until a vertex of a different component has been reached. At this point, the connecting path and the reached component are added to the first component. Now the generalized loop has one less connected component. This procedure is repeated again until the resulting generalized loop is not disconnected, or equivalently, until all its vertices are members of the first connected component. Iterating this search for each vertex every time two components are connected, and also for each initial connected component, one obtains all the required complex loops.

Note that deciding whether $l_1 \cup l_2$ is disconnected or not requires finding all connected components of the resulting loop. Moreover, given a disconnected loop, the number of associated complex loops can be enormous. In practice, the bound b obtained previously is used to reduce the number of calculations. First, testing if the length of $l_1 \cup l_2$ is larger than b can be done without computing the connected components. Second, the DFS search for complex loops is limited using b, so very large complex loops will not be retrieved.

However, restricting the DFS search for complex loops using the bound b could result in too deep searches. Consider the worst case of merging the two shortest, non-overlapping, simple loops which have size L_s. The maximum depth of the DFS search for complex loops is $d = b - 2L_s$. Then the computational complexity of the merge loops operation depends exponentially on d. This dependence is especially relevant when $b >> L_s$, for instance in cases where loops of many different lengths exist. To overcome this problem we define another parameter M, the maximum depth of the DFS search in the merge loops operation. For small values of M, the operation *merge loops* will be fast but a few (if any) complex loops will be obtained. Conversely, for higher values of M the operation *merge loops* will find more complex loops at the cost of more time.

Algorithm 1 in the previous page describes briefly the operation *merge loops*. It receives two loops l_1 and l_2, and bounds b and M as arguments, and returns the set *newloops* which contains the

Algorithm 1 merge loops

Arguments:

$l_1 = \langle V_1, E_1 \rangle$ loop,
$l_2 = \langle V_2, E_2 \rangle$ loop,
b maximal length of a loop,
M maximal depth of complex loops search,
G preprocessed factor graph

1: $newloops \leftarrow \emptyset$
2: **if** $(|E_1 \cup E_2| \leq b)$ **then**
3: $C \leftarrow$ Find connected components$(l_1 \cup l_2)$
4: $newloops \leftarrow \{l_1 \cup l_2\}$
5: **for all** $(c_i \in C)$ **do**
6: **for all** $(v_i \in c_i)$ **do**
7: $newloops \leftarrow newloops \cup$ Find complex loopsDFS(v_i, c_i, C, M, b, G)
8: **end for**
9: **end for**
10: **end if**
11: **return** *newloops*

loop resulting of the union of l_1 and l_2 plus all complex loops obtained in the DFS search bounded by b and M.

Once the problem of expressing all generalized loops as compositions of simple loops has been solved using the *merge loops* operation, we need to define an efficient procedure to merge them. Note that, given S simple loops, a brute force approach tries all combinations of two, three, ... up to $S-1$ simple loops. Hence the total number is:

$$\binom{S}{2} + \binom{S}{3} + \ldots + \binom{S}{S-1} = O(2^S),$$

which is prohibitive. Nevertheless, we can avoid redundant combinations by merging pairs of loops iteratively: in a first iteration, all pairs of simple loops are merged, which produces new generalized loops. In a next iteration i, instead of performing all $\binom{S}{i}$ mergings, only the new generalized loops obtained in iteration $i-1$ are merged with the initial set of simple loops. The process ends when no new loops are found. Using this merging procedure, although the asymptotic cost is still exponential in S, many redundant mergings are not considered.

Summarizing, the third step applies iteratively the *merge loops* operation until no new generalized loops are obtained. After this step has finished, the final step computes the truncated loop corrected partition function defined in Equation (8) using all the obtained generalized loops. We describe the full procedure in Algorithm 2. Lines 2 and 4 correspond to the first and second steps and lines $5-13$ correspond to the third step.

To show that this process produces all the generalized loops we first assume that S is sufficiently large to account for all the simple loops in the graph, and that M is larger or equal than the number of edges of the graph. Now let C be a generalized loop. According to the definitions of Section 3, either C can be expressed as a union of s simple loops, or C is a complex loop. In the first case, C is clearly produced in the sth iteration. In the second case, let s' denote the number of simple loops

Algorithm 2 Algorithm TLSBP

Arguments:

S maximal number of simple loops,
M maximal depth of complex loops search,
G original factor graph

Run belief propagation algorithm over G
$G' \leftarrow$ Obtain the 2-core(G)
$\mathcal{C}' \leftarrow \emptyset$
if ($\neg$empty(G')) **then**
  $\langle sloops, b \rangle \leftarrow$ Compute first S simple loops(G')
  $\langle oldloops, newloops \rangle \leftarrow \langle sloops, \emptyset \rangle$
  $\mathcal{C}' \leftarrow sloops$
  while ($\neg$empty($oldloops$)) **do**
    for all ($l_1 \in sloops$) **do**
      for all ($l_2 \in oldloops$) **do**
        $newloops \leftarrow newloops \cup$ mergeLoops(l_1, l_2, b, M, G')
      end for
    end for
    $oldloops \leftarrow newloops$
    $\mathcal{C}' \leftarrow \mathcal{C}' \cup newloops$
  end while
end if
return the result of expression (8) using $\mathcal{C}'$

which appear in C. Then C is produced in iteration s', during the DFS for complex loops within the merging of one of the simple loops contained in C.

The obtained collection of loops can be used for the approximation of the singe node marginals as well, as described in Equation (9). The method consists of clamping one variable i to all its possible values (± 1) and computing the corresponding approximations of the partition functions: $Z^{x_i=+1}_{TLSBP}$ and $Z^{x_i=-1}_{TLSBP}$. This requires to run BP in each clamped network, and reuse the set of loops replacing with zero those terms where the clamped variable appears. The computational complexity of approximating all marginals using this approach is in general $O(N \cdot L \cdot d \cdot T_{BP})$, where L is the number of found loops, d is the cardinality of the variables (two in our case), and T_{BP} the average time of BP to converge after clamping one variable. Usually, this task requires less computation time than the search for loops.

As a final remark, we want to stress a more technical aspect related to the implementation. Note that generalized loops can be expressed as the composition of other loops in many different ways. In consequence, they all must be stored incrementally and the operation of checking if a loop has been previously counted or not should be done efficiently. An appropriate way to implement this fast look-up/insertion is to encode all loops in a string composed by the edge identifiers in some order with a separator character between them. This identifier is used as a key to index an ordered tree, or hash structure. In practice, a hash structure is only necessary if large amounts of loops need to be stored. For the cases analyzed here, choosing a balanced tree instead of a hash table resulted in a more efficient data structure.

5. Experiments

In this section we show the performance of TLSBP in three different scenarios. First, we focus on square lattices and study how loop corrections improve the BP solution as a function of the interaction between variables and the size of the problem. Second, we study the performance of the method in random regular graphs as a function of the degree between the nodes. Finally, we apply the algorithm on a medical diagnosis bayesian network.

In all the experiments we show results for tractable instances, where the exact solution using the junction tree (Lauritzen and Spiegelhalter, 1988) can be computed. Performance is evaluated comparing the TLSBP error against the BP solution, and also against the cluster variation method (CVM). Instead of using a generalized belief propagation algorithm (GBP) which usually requires several trials to find the proper damping factor to converge, we use a double-loop implementation which has convergence guarantees (Heskes et al., 2003). For this study we select as outer regions of the CVM method all maximal factors together with all loops that consist up to four different variables. This choice represents a good trade-off between computation time required for convergence and accuracy of the solution.

We report two different error measures. Concerning the partition function Z we compute:

$$\text{Error}_{Z'} = \left| \frac{\log Z'}{\log Z} \right|, \tag{10}$$

where Z' is the partition function corresponding to the method used: BP, TLSBP, or CVM. Error of single-node marginals is measured using the maximum ℓ_∞ error, which is a reasonable quantity if one is interested in worst-case scenarios:

$$\text{Error}_b = \max_{\substack{i=1,\ldots,n \\ x_i=\pm 1}} |P_i(x_i) - b_i(x_i)|, \tag{11}$$

were again $b_i(x_i)$ are the single-node marginal approximations corresponding to the method used.

We use four different schemas for belief-updating of BP: *(i)* fixed and *(ii)* random sequential updates, *(iii)* parallel (or synchronous) updates, and *(iv)* residual belief propagation (RBP), a recent method proposed by Elidan et al. (2006). The latter method schedules the updates of the BP messages heuristically by selecting the next message to be updated which has maximum *residual*, a quantity defined as an upper bound on the distance of the current messages from the fixed point. In general, we experienced that for some instances where the RBP method converged, the other update schemas (fixed, random sequential and parallel updates) failed to converge.

In all schemas we interpret that a fixed point is reached at iteration t when the maximum absolute value of the updates of all the messages from iteration $t-1$ to t is smaller than a threshold ϑ. We notice a large correlation between the order of magnitude of ϑ and the ratio between the BP and the TLSBP errors. For this reason we used a very small value of the threshold, $\vartheta = 10^{-15}$.

5.1 Ising Grids

This model is defined on a grid where each variable, also called spin, takes binary values $x_i = \pm 1$. A spin is coupled with its direct neighbors only, so that pairwise interactions $f_{ij}(x_i, x_j) = \exp(\theta_{ij} x_i x_j)$ are considered, parametrized by θ_{ij}. Every spin can be exposed to an external field $f_i(x_i) = \exp(\theta_i x_i)$, or single-node potential, parametrized by θ_i. Figure 3 (left) shows the factor

graph associated to the 4x4 Ising grid, composed of 16 variables. The Ising grid model is often used as a test-bed for inference algorithms. It is of great relevance in statistical physics, and has applications in different areas such as image processing. In our context it also represents a challenge since it has many loops. Good results in this model will likely translate into good results for less loopy graphs.

Usually, two cases are differentiated according to the sign of the θ_{ij} parameters. For $\theta_{ij} > 0$ coupled spins tend to be in the same state. This is known as the attractive, or "ferromagnetic" setting. On the other hand, for mixed interactions, θ_{ij} can be either positive or negative, and this setting is called "spin-glass" configuration. Concerning the external field, one can distinguish two cases. For the case of nonzero fields, larger values of θ_i imply easier inference problems in general. On the other hand, for $\theta_i = 0$, there exist two phase transitions from easy inference problems (small θ_{ij}) to more difficult ones (large θ_{ij}) depending on the type of pairwise couplings (see Mooij and Kappen, 2005, for more details).

This experimental subsection is structured in three parts: First, we study a small 4x4 grid. We then study the performance of the algorithm in a 10x10 grid, where complete enumeration of all generalized loops is not feasible. Finally, we analyze the scalability of the method with problem size.

The 4x4 Ising grid is complex enough to account for all types of generalized loops. It is the smallest size where complex loops are present. At the same time, the problem is still tractable and exhaustive enumeration of all the loops can be done.

We ran the TLSBP algorithm in this model with arguments S and M large enough to retrieve all the loops. Also, the maximum length b was constrained to be 48, the total number of edges for this model. After 4 iterations all generalized loops were obtained. The total number is 16371 from which 213 are simple loops. The rest of generalized loops are classified as follows: 174 complex and

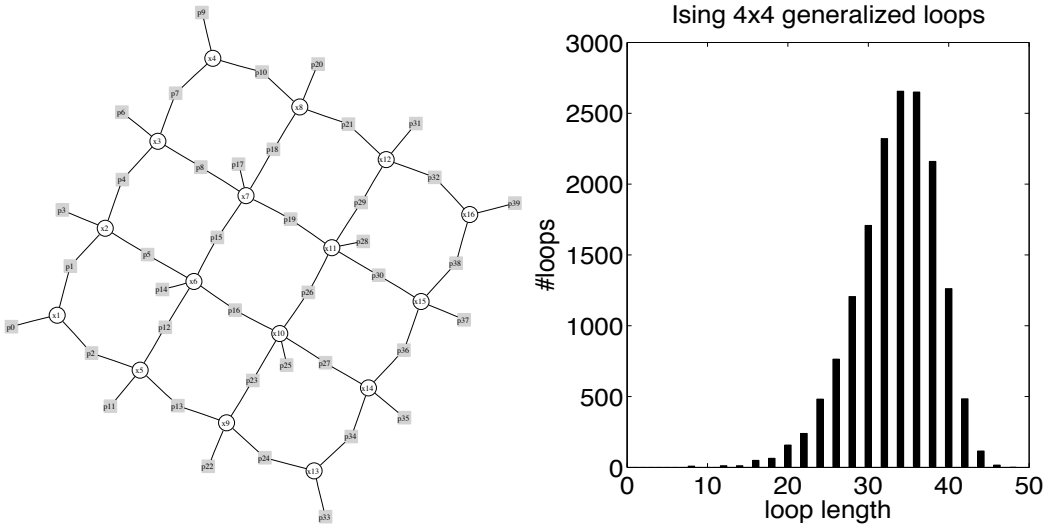


Figure 3: **(left)** A factor graph representing the 4x4 Ising grid. **(right)** Number of generalized loops as a function of the length using the factor graph representation.

disconnected loops, 1646 complex but non-disconnected loops, 604 non-complex but disconnected loops, and 13734 neither complex nor disconnected loops.

Figure 3 (right) shows the histogram of all generalized loops for this small grid. Since we use the factor graph representation the smallest loop has length 8. The largest generalized loop includes all nodes and all edges of the *preprocessed* graph, and has length 48. The Poisson-like shape of the histogram is a characteristic of this model and for larger instances we observed the same tendency. Thus the analysis for this small model can be extrapolated to some extent to grids with more variables.

To analyze how the error changes as more loops are considered it is useful to sort all the terms $r(C)$ by their absolute value in descending order such that $|r(C_i)| \geq |r(C_{i+1})|$. We then compute, for each number of loops $l = 1 \ldots 16371$, the approximated partition function which accounts for the l most important loops:

$$Z_{TLSBP}(l) = Z_{BP}\left(1 + \sum_{i=1\ldots l} r(C_i)\right). \tag{12}$$

From these values of the partition function we calculate the error measure indicated in Equation (11). Estimations of the single-node marginals were obtained using the clamping method, and their corresponding error was calculated using Equation (10).

We now study how loop contributions change as a function of the coupling strength between the variables. We ran several experiments using mixed interactions with $\theta_{ij} \sim \mathcal{N}(0, \sigma^2)$ independently for each factor node, and σ varying between 0.1 and 2. Single-node potentials were drawn according to $\theta_i \sim \mathcal{N}(0, 0.05^2)$. For small values of σ, interactions are weak and BP converges easily, whereas for high values of σ variables are strongly coupled and BP has more difficulties, or does not converge at all.

Figure 4 shows results of representative instances of three different interaction strengths. For each instance we plot the partition function error (left column) together with errors of the single-node marginals (middle column) and loop contributions as a function of the length (right column). First, we can see that improvements of the partition sum correspond to improvements of the estimates of marginal probabilities as well. Second, for weak couplings ($\sigma = 0.1$, first row) we can see that truncating the series until a small number of loops (around 10) is enough to achieve machine precision. In this case the errors of BP are most prominently due to small simple loops. As the right column illustrates, loop contributions decrease exponentially with the size, and loops with the same length correspond to very similar contributions. Larger loops give negligible contributions and can thus be ignored by truncating the series. As interactions are strengthened, however, more loops have to be considered to achieve maximum accuracy, and contributions show more variability for a given length (see middle row). Also, oscillations of the error due to the different signs in loop terms (caused by the mixed interactions) of the same order of magnitude become more frequent. Eventually, for large couplings ($\sigma \geq 2$), where BP often fails to converge, loops of all lengths give significant contributions. In the bottom panels of Figure 4 we show an example of a 'difficult' case for which the BP result is not improved until more than 10^3 loop terms are summed. The observed discontinuities in the error of the partition sum are caused by the fact that oscillations become more pronounced, and corrections composed of negative terms $r_i(C_i)$ can result in negative values of the partially corrected partition function, see Equation (12). This occurs for very strong interactions only, and when a small fraction of the total number of loops is considered. In addition, as the right column indicates, there is a shift of the main contributions towards the largest loops.

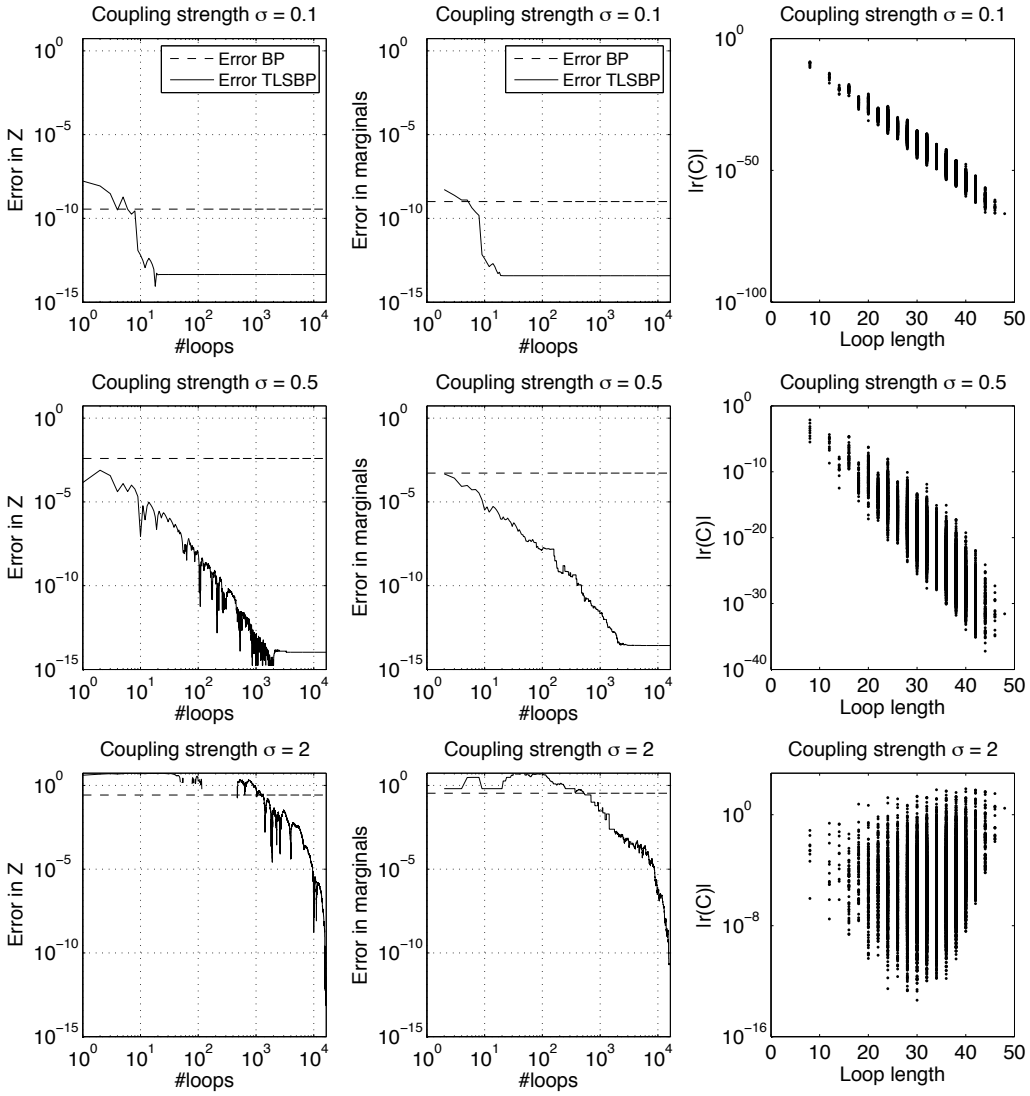


Figure 4: Cumulative error for the spin-glass 4x4 Ising grid for different interaction strengths, see Equation (12). **(left column)** Error of Z. **(middle column)** Error of single-node marginals. Dashed lines correspond to the BP error, and solid lines correspond to the loop-corrected (TLSBP) error. **(right column)** Absolute values of all loop terms as a function of the length of the corresponding loop.

After analyzing a small grid, we now address the case of the 10x10 Ising grid, where exhaustive enumeration of all the loops is not computationally feasible. We test the algorithm in two scenarios: for attractive interactions (ferromagnetic model) where pairwise interactions are parametrized as $\theta_{ij} = |\theta'_{ij}|, \theta'_{ij} \sim \mathcal{N}(0, \sigma^2)$, and also for the previous case of mixed interactions (spin-glass model). Single-node potentials were chosen $\theta_i \sim \mathcal{N}(0.1, 0.05^2)$ in both cases.

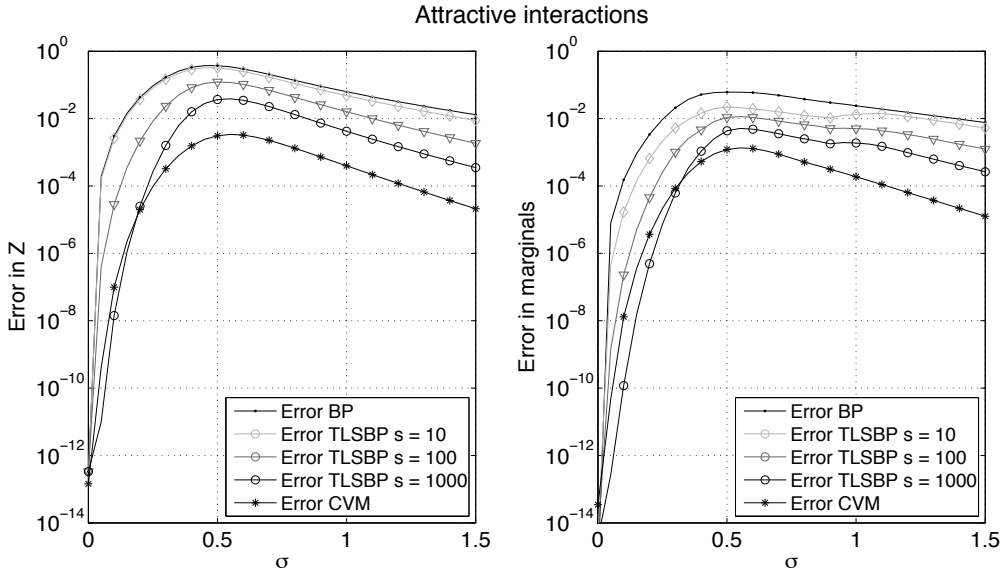


Figure 5: TLSBP error for the 10x10 Ising grid with attractive interactions for different values of the parameter S. **(left)** Error of the partition function. **(right)** Error of single-node marginals.

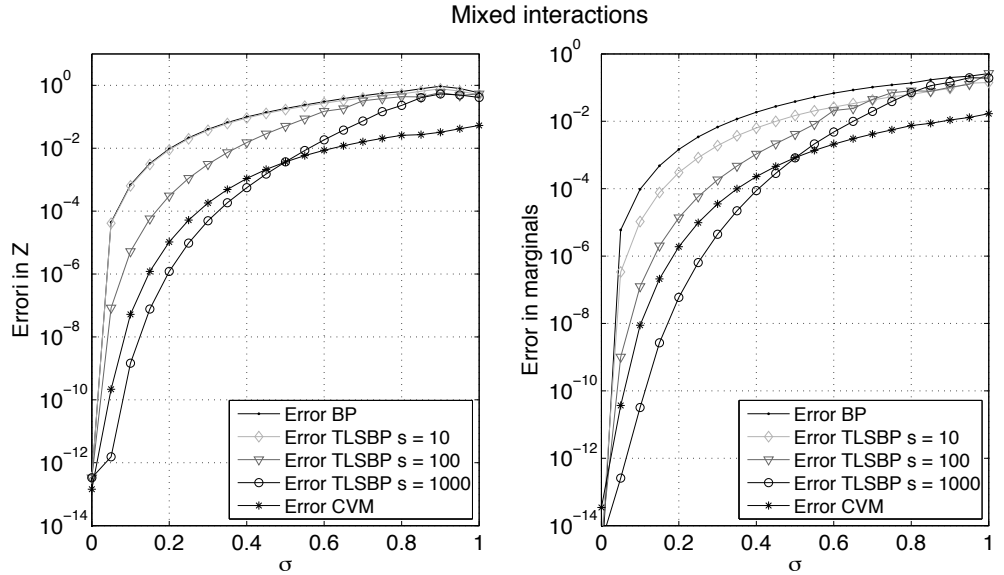


Figure 6: TLSBP error of the 10x10 Ising grid with mixed interactions for different values of the parameter S. **(left)** Error of the partition function. **(right)** Error of single-node marginals.

We show results in Figures 5 and 6 for three values of the parameter $S = \{10, 100, 1000\}$ and a fixed value of $M = 10$. For $S = 10$ and $S = 100$, only simple loops were obtained whereas for $S = 1000$, a total of 44590 generalized loops was used to compute the truncated partition sum. Results are averaged errors over 50 random instances. The selected loops were the same in all instances. Although in both types of interactions the BP error (solid line with dots) is progressively reduced as more loops are considered, the picture differs significantly between the two cases.

For the ferromagnetic case shown in Figure 5, we noticed that all loops have positive contributions, $r(C) > 0$. This is a consequence of this particular type of interactions, since all magnetizations have the same sign at the BP fixed point, and also all nodes have an even number of neighbors. Consequently, improvements in the BP result are monotonic as more loops are considered, and in this case, the TLSBP can be considered as a lower bound of the exact solution. For the case of $S = 1000$, the BP error is reduced substantially at nonzero σ, but around $\sigma \sim 0.5$, where the BP error reaches a maximum, also the TLSBP improvement is minimal. From this maximum, the BP error decreases again, and loop corrections tend to improve progressively the BP solution again as the coupling is strengthened. We remark that improvements were obtained for all instances in the three cases.

Comparing with CVM, TLSBP is better for weak couplings and for $S = 1000$ only. This indicates that for intermediate and strong couplings one would need more than the selected 44590 generalized loops to improve on the CVM result.

For the case of spin-glass interactions we report different behavior. From Figure 6 we see again that for weak couplings the BP error is corrected substantially, but the improvement decreases as the coupling strength is increased. Eventually, for $\sigma \sim 1$ BP fails to converge in most of the cases and also gives poor results. In these cases loop corrections are of little use, and there is no actual difference in considering $S = 1000$ or $S = 10$. Moreover, because loop terms $r(C)$ now can have different signs, truncating the series can lead to worse results for $S = 1000$ than for $S = 10$. Interestingly, the range where TLSBP performs better than CVM is slightly larger in this type of interactions, TLSBP being better for $\sigma < 0.5$.

To end this subsection, we study how loop corrections scale with the number of nodes in the graph. We only use spin-glass interactions, since it is a more difficult configuration than the ferromagnetic case, as previous experiments suggest. We compare the performance for weak couplings ($\sigma = 0.1$), and strong couplings ($\sigma = 0.5$), where BP has difficulties to converge in large instances. The number of variables N^2 is increased for grids of size $N \times N$ until exact computation using the junction tree algorithm is not feasible anymore.

Since the number of generalized loops grows very fast with the size of the grid, we choose increasing values of S as well. We use values of S proportional to the number of variable nodes N^2 such that $S = 10N^2$. This simple linear increment in S means that as N is increased, the proportion of simple loops captured by TLSBP over the total existing number of simple loops decreases. It is interesting to see how this affects the performance of TLSBP in terms of time complexity and accuracy of the solution. For simplicity, M is fixed to zero, so no complex loops are considered. Moreover, to facilitate the computational cost comparison, we only compute mergings of pairs of simple loops. Actually, for large instances the latter choice does not modify the final set of loops, since generalized loops which can only be expressed as compositions of three or more simple loops are pruned using the bound b.

In Figure 7, the top panels show averaged results of the computational cost. The left plot indicates the relation between the number of loops computed by TLSBP and the time required to compute them. This relation can be fit accurately using a line which means that for this choice of

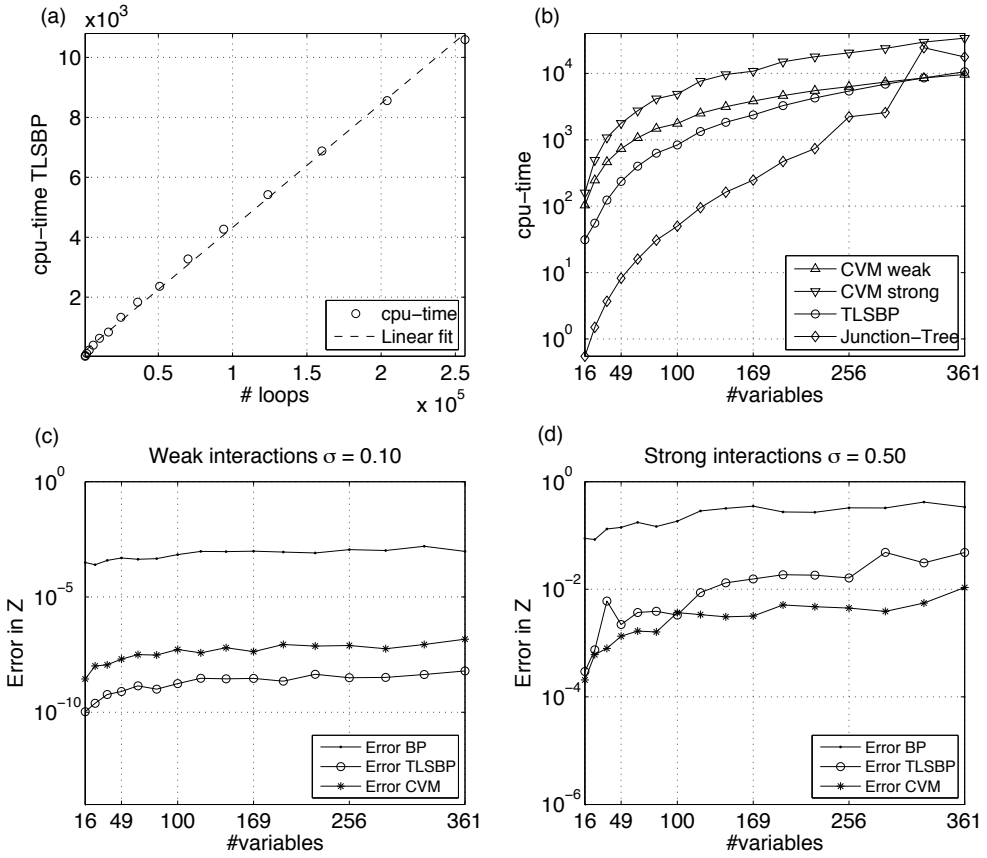


Figure 7: Scalability of the method in the Ising model. **(a)** Time complexity as a function of the produced number of generalized loops. **(b)** Relation between the time complexity of TLSBP and CVM. Comparison of the error of the partition function between BP, TLSBP and CVM as a function of the graph size for **(c)** weak interactions and **(d)** strong interactions.

parameters S and M, and considering only mergings of simple loops, the computational complexity of the algorithm grows just linearly with the found loops. One has to keep in mind that the number of loops obtained using the TLSBP algorithm grows much faster, but much less than the total number of existing loops in the model.

Figure 7b shows the CPU time consumed by CVM, TLSBP, and the junction tree algorithm. In this case, since we only compute the partition function Z, the CPU time of TLSBP is constant for both weak and strong couplings. On the contrary, CVM depends on the type of interactions. For weak interactions, TLSBP is in general more efficient than CVM, although the scaling trend is slightly better for CVM. For $N = 19$, CVM starts to be more efficient than TLSBP. For strong interactions, CVM needs significantly more time to converge in all cases. If we compare the computational cost of the exact method against TLSBP, we can see that the junction tree is very efficient for networks with small N, and the best option in those cases. However, for $N > 17$, the junction tree needs more computation time, and for $N > 19$, the tree-width of the resulting grids is too large.

TLSBP memory requirements were considerably less in these cases, since loops can be stored efficiently using sets of chars. Also, we can see that the TLSBP scaling is better for this choice of parameters than that of the exact method.

Bottom panels show the accuracy of the TLSBP solution. For weak couplings (bottom-left) the BP error is always decreased significantly for this choice of parameters and the improvement remains almost constant as N increases, meaning that, in this case the number of loops which contribute most to the series expansion does not grow significantly with N. Interestingly, results are always better than CVM for this regime.

For strong couplings (bottom-right) the picture changes. First, results differ more between instances causing a less smooth curve. Second, the TLSBP error also increases with the problem size, so improvements tend to decrease with N, even faster than the BP error increase. Eventually, for the largest tractable instance the TLSBP improvement is still significant, about one order of magnitude. Comparing against CVM, unlike in the weak coupling scenario, the TLSBP method does not seem to perform better, and only for some cases TLSBP error is comparable to the CVM error on average. The accuracy of the TLSBP solution for these instances can be increased by considering larger values of S and M, at the cost of more time.

5.2 Random Graphs

The previous experimental results were focused on the Ising grid which only considers pairwise and singleton interactions in such a way that each node in the graph is at most linked with four neighbors. Here we briefly analyze the performance of TLSBP applied on a more general case, where interactions are less restricted.

We perform experiments on random graphs with regular topology, where each variable is coupled randomly with d other variables using pairwise interactions parametrized by $\theta_{ij} \sim \mathcal{N}(0, \sigma^2)$. Single-node potentials were parametrized in this case by $\theta_i \sim \mathcal{N}(0, 0.05^2)$. We study how loop corrections improve the BP solution as a function of the degree d, and compare improvements against the CVM. As in the previous subsection, for CVM we select the loops of four variables and all maximal factors as outer clusters.

Note that the rate of increase in the number of loops with the degree d is even higher than with the number of variables in the Ising model. Adding one more link to all the variables means adding N more factor nodes to the factor graph. This raises the number of loops dramatically.

For this scenario, we use $N = 20$ variables and also increase S every time d is increased. We simply start with $S = 10$ and use increments of 250 for each new d. M was set to 10, and all possible mergings were computed. We analyze two scenarios, weak ($\sigma = 0.1$) and strong couplings ($\sigma = 0.5$), and report averages over 60 random instances for each configuration. As Figure 9 (right) indicates, for $\sigma = 0.1$ BP converged in all instances, whereas for $\sigma = 0.5$ BP convergence becomes more difficult as we increase d.

Figure 8 (top) shows results for weak interactions. The TLSBP algorithm always corrects the BP error, although as d increases, the improvement is progressively reduced. We also notice that in all cases and methods the approximation of the partition function (left) is less accurate than the approximation of the marginals (right). For $d = 15$, TLSBP improvements are still about one order of magnitude for the partition function, and even better for the marginals. As in previous experiments with square lattices, the TLSBP approach is generally better than CVM in the weak

coupling regime. Here, it is also more stable, since for some dense networks the CVM error can be very large, as we can see for $d = 13$ and $d = 15$.

For strong interactions (bottom panels), we see that differences between approximations of the partition function and single-node marginals are more remarkable than in the previous case. The BP partition function is corrected by TLSBP in more than half of the instances for all degrees (see inset of Figure 8c, where we plot the fraction of instances where BP was corrected in those cases that converged), although for higher degrees, the TLSBP corrections are small using this choice of parameters. On the other hand, single-node BP marginals are corrected in almost all cases. In contrast, the CVM approach with our selection of outer clusters does not perform better than TLSBP in general. In particular, we see that CVM estimates of the partition function are very degraded

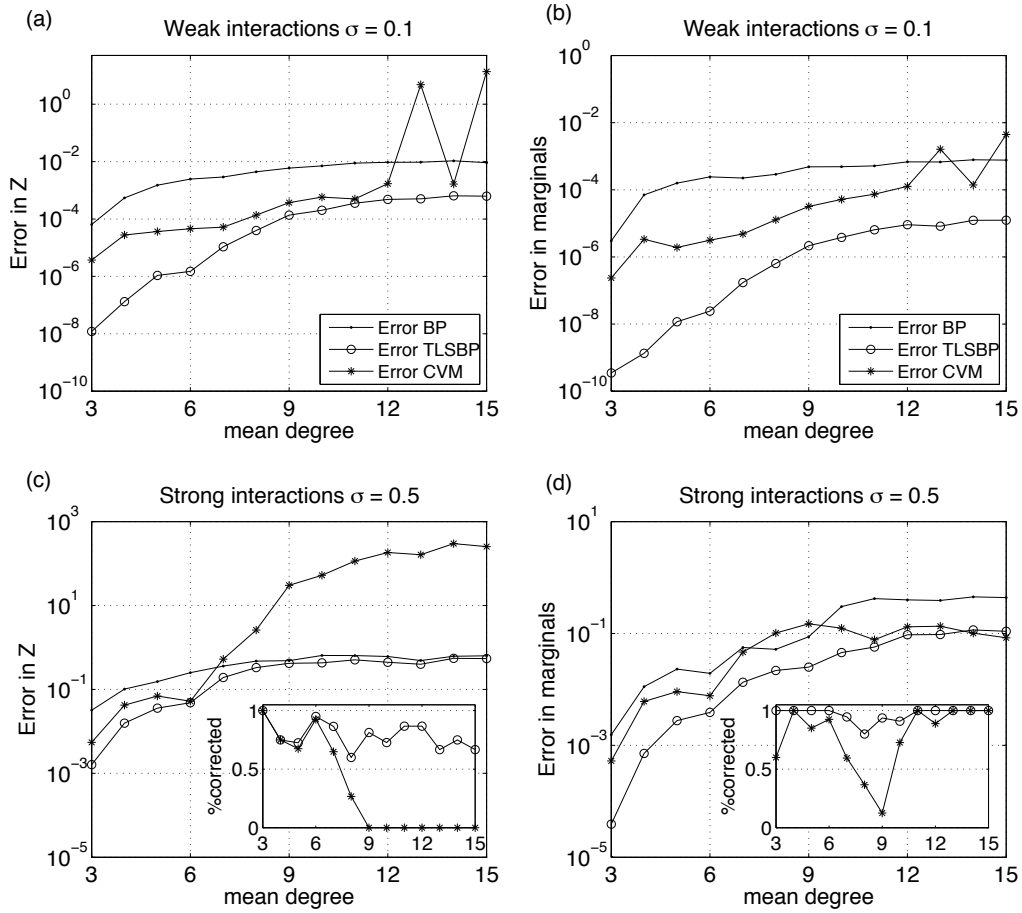


Figure 8: Results on random regular graphs. TLSBP and CVM errors as a function of the degree d. Results are averages over 60 random instances. Errors in the partition function for weak interactions **(a)**, marginals for weak interactions **(b)**, partition function for strong interactions **(c)**, and marginals for strong interactions **(d)**. Insets show percentage of instances where the BP error was corrected.

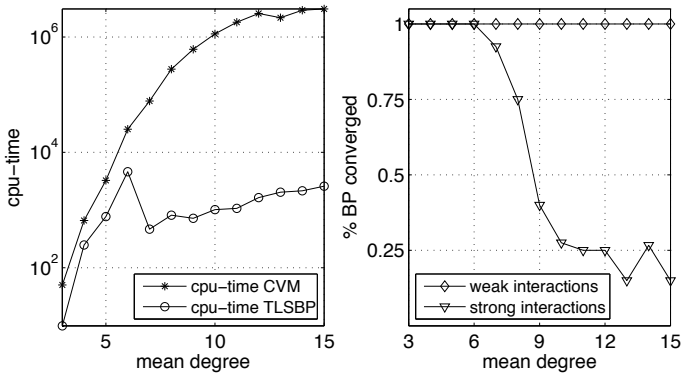


Figure 9: Results on random regular graphs. **(a)** Computation time of TLSBP and CVM. In this case, we averaged also all instances over all 60 weak and 60 strong interactions, since costs were very similar in both cases. **(b)** Fraction of the instances were BP converged. No convergence is reported when none of the four proposed schedules converged.

as networks become more dense. This unsatisfactory performance of CVM in the estimation of the partition function is not as noticeable in the marginal estimates, where BP results are often improved, although with much more variability than the TLSBP method. Interestingly, for those few instances of dense networks for which BP converged, CVM estimates of the marginals were very similar to TLSBP.

Finally, we compare computational costs in Figure 9 (left). CVM requires significantly more time to converge than the time required by TLSBP searching for loops and calculating marginals. If we analyze in detail how the TLSBP cost changes, we can notice different types of growth for $d < 7$ and for $d \geq 7$. The reason behind these two scaling tendencies can be explained by the choice of TLSBP parameters, and the bound b (the size of the largest simple loop). For $d < 7$, many simple loops of different lengths are obtained. Consequently, the cost of the merging step grows fast, since many loops with length smaller than b are produced. On the other hand, for $d \geq 7$ simple loops have similar lengths and, therefore, less combinations result in additional loops with length larger than the bound b. Without bounding the length of the loops in the merging step, we would expect the first scaling tendency ($d < 7$) also for values of $d \geq 7$.

From these experiments we can conclude that TLSBP performance is generally better than CVM in this domain. We should mention that alternative choices of regions would have lead to different CVM results, but will probably not change this conclusion.

5.3 Medical Diagnosis

We now study the performance of TLSBP on a "real-world" example, the PROMEDAS medical diagnostic network. The diagnostic model in PROMEDAS is based on a bayesian network. The global architecture of this network is similar to QMR-DT (Shwe et al., 1991). It consists of a diagnosis layer that is connected to a layer with findings. In addition, there is a layer of variables, such as age and gender, that may affect the prior probabilities of the diagnoses. Since these variables are always clamped for each patient case, they merely change the prior disease probabilities and

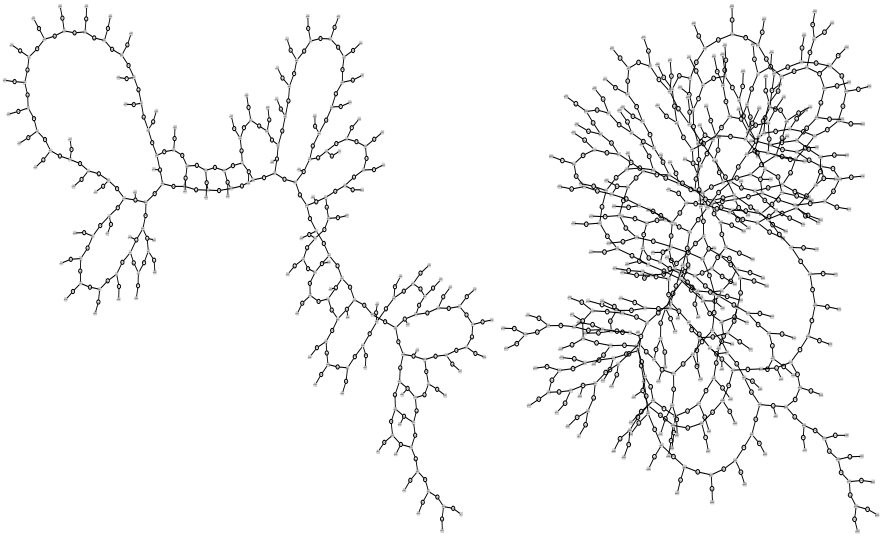


Figure 10: Examples of graph structures, corresponding to patient cases generated with one disease, after removal of unclamped findings and irrelevant disease variables and the introduction of dummy variables. Left and right graphs corresponds to an "easy" and a "difficult" case respectively.

are irrelevant for our current considerations. Diagnoses (diseases) are modeled as a priori independent binary variables causing a set of symptoms (findings) which constitute the bottom layer. The PROMEDAS network currently consists of approximately 2000 diagnoses and 1000 findings.

The interaction between diagnoses and findings is modeled with a noisy-OR structure. The conditional probability of the finding given the parents is modeled by $n+1$ numbers, n of which represent the probabilities that the finding is caused by one of the diseases and one that the finding is not caused by any of the parents.

The noisy-OR conditional probability tables with n parents can be naively stored in a table of size 2^n. This is problematic for the PROMEDAS networks since findings that are affected by more than 30 diseases are not uncommon. We use efficient implementation of noisy-OR relations as proposed by Takinawa and D'Ambrosio (1999) to reduce the size of these tables. The trick is to introduce dummy variables s and to make use of the property

$$\mathrm{OR}(x|y_1,y_2,y_3) = \sum_s \mathrm{OR}(x|y_1,s)\mathrm{OR}(s|y_2,y_3).$$

The interaction potentials on the right hand side involve at most three variables instead of the initial four (left). Repeated application of this formula reduces all tables to three interactions maximally.

When a patient case is presented to PROMEDAS, a subset of the findings will be clamped and the rest will be unclamped. If our goal is to compute the marginal probabilities of the diagnostic variables only, the unclamped findings and the diagnoses that are not related to any of the clamped findings can be summed out of the network as a preprocessing step. The clamped findings cause an effective interaction between their parents. However, the noisy-OR structure is such that when

the finding is clamped to a negative value, the effective interaction factorizes over its parents. Thus, findings can be clamped to negative values without additional computation cost (Jaakkola and Jordan, 1999).

The complexity of the problem now depends on the set of findings that is given as input. The more findings are clamped to a positive value, the larger the remaining network of disease variables and the more complex the inference task. Especially in cases where findings share more than one common possible diagnosis, and consequently loops occur, the model can become complex. We illustrate some of the graphs that result after pruning of unclamped findings and irrelevant diseases and the introduction of dummy variables for some patient cases in Figure 10.

We use the PROMEDAS model to generate virtual patient data by first clamping one disease variable to a positive value and then clamping a finding to its positive value with probability equal to the conditional distribution of the findings given this positive disease. The union of all positive findings thus obtained constitute one patient case. For each patient case, the corresponding truncated graphical model is generated. Note that the number of disease nodes in this graph can be large and hence loops can be present.

In this subsection, as well as comparing errors of single-node marginals obtained using TLSBP against CVM, we also use another loop correction approach, loop corrected belief propagation (LCBP) (Mooij and Kappen, 2007), which is based on the cavity method and also improves over BP estimates. We use the following parameters for TLSBP: $S = 100$, $M = 5$, and no bound b. Again, we apply the junction tree method to obtain exact marginals and compare the different errors. Figure 11 shows results for 146 different random instances.

We first analyze the TLSBP results compared with BP (Figure 11a). The region in light gray color indicates TLSBP improvement over BP. The observed results vary strongly because of the wide diversity of the particular instances, but we can basically differentiate two scenarios. The first set of results include those instances where the BP error is corrected almost up to machine precision. These patient cases correspond to graphs where exhaustive enumeration is tractable, and TLSBP found almost all the generalized loops. These are the dots appearing in the bottom part of Figure 11a, approximately 14% of the patient cases. Note that even for errors of the order of 10^{-2} the error was completely corrected. Apart from these results, we observe another group of instances where the BP error was not completely corrected. These cases correspond to the upper dots of Figure 11a. The results in these patient cases vary from no significant improvements to improvements of four orders of magnitude.

Figure 11b shows the performance of CVM considering all maximal factors together with all loops that consist up to four different variables as outer regions. We observe that, contrary to TLSBP, CVM in this domain performs poorly. For only one instance the CVM result is significantly better than BP. Moreover, the computation time required by CVM was much larger than TLSBP in all instances (data not shown). These results can be complemented with the study developed by Mooij and Kappen (2007), where it is shown that CVM does not perform significantly better for other choices of regions.

Figure 11c shows results of LCBP, the approach presented in Mooij and Kappen (2007) on the same set of instances. As in the case of TLSBP, LCBP significantly improves over BP. A comparison between both approaches is illustrated in Figure 11d, where those instances where TLSBP is better are marked in light gray color. For 41% of the cases TLSBP improves the LCBP results, sometimes notably. TLSBP enhancements were made at the cost of more time, as Figure 12a suggest, where in 85% of the instances TLSBP needs more time.

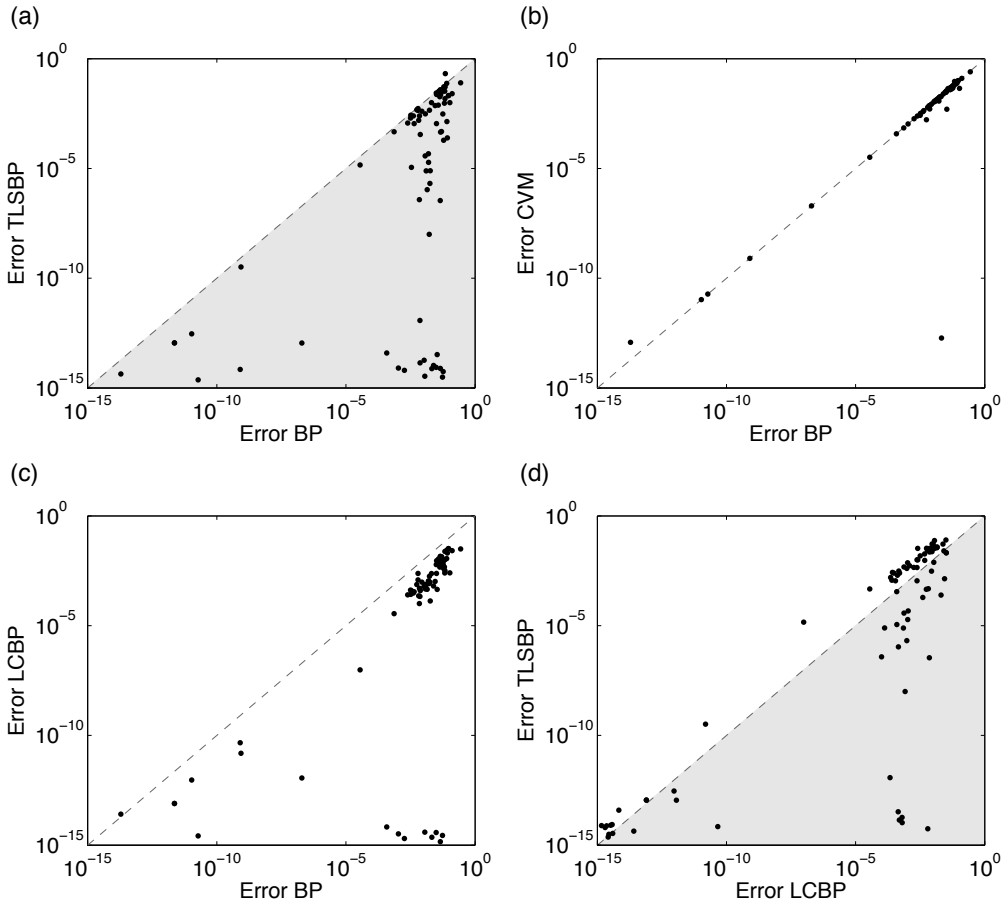


Figure 11: Results of 146 random patient cases with one disease. **(a)** TLSBP error versus BP error. **(b)** CVM error versus BP error. **(c)** LCBP error versus BP error. and **(d)** LCBP error versus TLSBP error.

To analyze the TLSBP results in more depth we plot the ratio between the error obtained by TLSBP and the BP error versus the number of generalized loops found and the CPU time. From Figure 12b we can deduce that cases where the BP error was most improved, often correspond to graphs with a small number of generalized loops found, whereas instances with highest number of loops considered have minor improvements. This is explained by the fact that some instances which contained a few loops were easy to solve and thus the BP error was significantly reduced. An example of one of those instances corresponds to the Figure 10 (left). On the contrary, there exist very loopy instances where computing some terms was not useful, even if a large number of them (more than one million) where considered. A typical instance of this type is shown in Figure 10 (right). The same argument is suggested by Figure 12c where CPU time is shown, which is often proportional to the number of loops found.

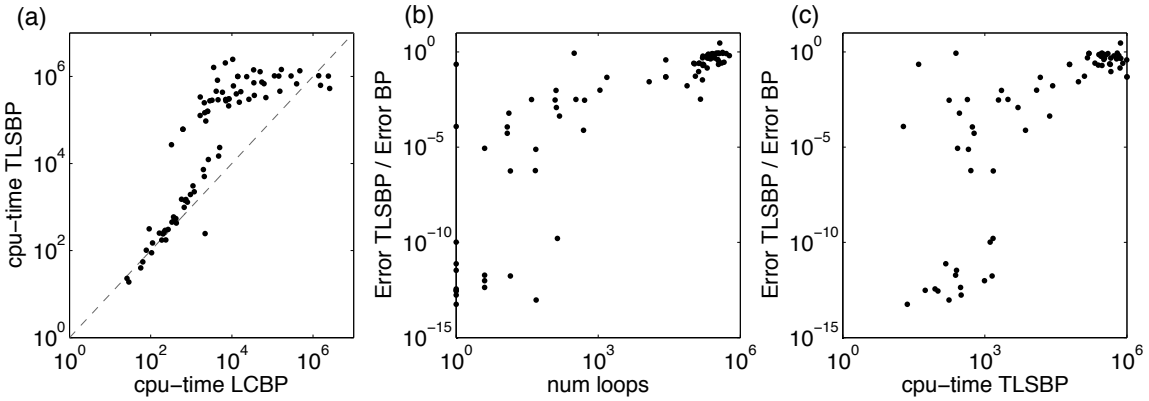


Figure 12: Results of applying TLSBP to 146 patient cases with one disease. **(a)** Relation between computational cost needed by LCBP and TLSBP. **(b)** Ratio between TLSBP and BP errors versus number of loops found. **(c)** Ratio between TLSBP and BP errors versus computation time.

In general, we can conclude that although the BP error was corrected in most of the instances, there were some cases in which TLSBP did not give significant improvements. Considering all patient cases, the BP error was corrected in more than one order of magnitude for more than 30% of the cases.

6. Discussion

We have presented TLSBP, an algorithm to compute corrections to the BP solution based on the loop series expansion proposed by Chertkov and Chernyak (2006b). [2] In general, for cases where all loops can be enumerated the method computes the exact solution efficiently. In contrast, if exhaustive enumeration is not tractable, the BP error can be reduced significantly. The performance of the algorithm does not depend directly on the size of the problem, but on how loopy the original graph is, although for larger instances it is more likely that more loops are present.

We have also shown that the performance of TLSBP strongly depends on the degree of coupling between the variables. For weak couplings, errors of BP are most prominently caused by small simple loops and truncating the series is very useful, whereas for strongly coupled variables, loop terms of different sizes tend to be of the same order of magnitude, and truncating the series can be useless. For those difficult cases, BP convergence is also difficult, and magnetizations at the fixed point tend to be close to extreme values, causing numerical difficulties in the calculation of the loop expansion formulas. In general, we can conclude that the proposed approach is useful in an intermediate regime, where BP results are not very accurate, but BP is still converging.

We have confirmed empirically that there is a correlation between the BP result and the potential improvements using TLSBP. Those cases where the BP estimate is most corrected correspond often

2. The source code of the algorithm and a subset of the data sets used in the experimental section can be downloaded from: `http://www.cns.upf.edu/vicent/`.

to cases where the BP estimate is already accurate. Whether a given BP error is acceptable or not depends on the inference task and the specific domain.

The proposed approach has been compared with CVM selecting loops of four variables and maximal factors as outer clusters. For highly regular domains with translation invariance such as square grids, CVM performs better than TLSBP in difficult instances (strong interactions). This is not surprising, since CVM exploits the symmetries on the original graph. However, for other domains such as random graphs, or medical diagnosis, TLSBP show comparable, or even better results than CVM with our choice of clusters.

The TLSBP algorithm searches the graph without considering information accessible from the BP solution, which is used to compute the loop corrections only as a final step. Therefore, it can be regarded as a blind search procedure. We have also experimented with a more "heuristic" algorithm where the search is guided in some principled way. Two modifications of the algorithm have been done in that direction:

1. One approach consisted in modifying the third step in a way that, instead of applying blind mergings, generalized loops which have larger contributions (largest $|r(C)|$) were merged preferentially. In practice, this approach tended to check all combinations of small loops which produced the same generalized loop, causing many redundant mergings. Moreover, the cost of maintaining sorted the "best" generalized loops caused a significant increase in the computational complexity. This approach did not produce more accurate results neither was a more efficient approach.

2. Also, instead of pruning the DFS search for complex loops using the parameter M, we have used the following strategy: we computed iteratively the *partial term* of the loop that is being searched, such that at each DFS step one new term using Equations (6) and (7) is multiplied with the current *partial term*. If at some point, the *partial term* was smaller than a certain threshold λ, the DFS was pruned. This new parameter λ was then used instead of M and result in an appropriate strategy for graphs with weak interactions. For cases where many terms of the same order existed, a small change of λ caused very different execution times, and often too deep searches. We concluded that using parameter M is a more suitable choice in general.

TLSBP can be easily extended in other ways. For instance, as an anytime algorithm. In this context, the partition sum or marginals can be computed incrementally as more generalized loops are being produced. This allows to stop the algorithm at any step and presumably, the more time used, the better the solution. The "improvement if allowed more time" can be a desirable property for applications in approximate reasoning (Zilberstein, 1996). Another way to extend the approach is to consider the search for loops as a *compilation* stage of the inference task, which can be done offline. Once all loops are retrieved and stored, the inference task would require much less computational cost to be performed.

During the development of this work another way of selecting generalized loops has been proposed (Chertkov and Chernyak, 2006a) in the context of Low Density Parity Check codes. Their approach tries to find only a few *critical* simple loops, related with dangerous noise configurations that lead to Linear Programming decoding failure, and use them to modify the standard BP equations. Their method shows promising results for the LDPC domain, and can be applied to any general graphical model as well, so it would be interesting to compare both approaches.

There exists another type of loop correction methods that improves BP, which is quite different from the approach discussed here (Montanari and Rizzo, 2005; Parisi and Slanina; Mooij et al., 2007; Mooij and Kappen, 2007). Their argument is based on the cavity method. BP assumes that in the absence of variable i, the distribution of its Markov blanket factorizes over the individual variables. In fact, this assumption is only approximately true, due to the loops in the graph. The first loop correction is obtained by considering the network with variable i removed and estimating the correlations in the Markov blanket. This argument can be applied recursively, yielding the higher order loop corrections. Whereas TLSBP computes exactly the corrections of a limited number of loops, the cavity based approach computes approximately the corrections due to all loops. An in-depth comparison of the efficiency and accuracy of these approaches should be made.

As a final remark, we mention the relation of the loop series expansion with a similar method originated in statistical physics, namely, the high-temperature expansion for Ising models. This expansion of the partition function is similar to the one proposed by Chertkov and Chernyak (2006b), in the sense that every term has also a direct diagrammatic representation on the graph, although not in terms of generalized loops. Note however, that the loop expansion is relative to the BP result, contrary to the high-temperature expansion. Another difference is that the high temperature expansion is an expansion in a small parameter (the inverse temperature), whereas the loop expansion has no such small parameter. Finally, another related approach is the walk-sum framework for inference in certain Gaussian Markov Models (Malioutov et al., 2006), where means and covariances between any two nodes of the graph have an interpretation in terms of an expansion of walks in the graph. They also show that Gaussian loopy BP has a walk-sum interpretation, computing all walks for the means but only a subset of walks for the variances.

Acknowledgments

We would like to thank the reviewers for their constructive suggestions that helped us to improve the paper. We also acknowledge financial support from the Càtedra Telefònica Multimèdia, the Interactive Collaborative Information Systems (ICIS) project (supported by the Dutch Ministry of Economic Affairs, grant BSIK03024), and the Dutch Technology Foundation (STW). Finally, we thank Bastian Wemmenhove for providing patient cases of the PROMEDAS medical system, and Andreas Kaltenbrunner for useful suggestions.

References

M. Chertkov and V. Y. Chernyak. Loop calculus helps to improve belief propagation and linear programming decodings of LDPC codes. In *Invited Talk at the 44th Allerton Conference*, September 2006a. URL `http://www.arxiv.org/abs/cs/0609154`.

M. Chertkov and V. Y. Chernyak. Loop series for discrete statistical models on graphs. *Journal of Statistical Mechanics: Theory and Experiment*, 2006(06):P06009, 2006b. URL `http://arxiv.org/abs/cond-mat/0601487`.

G. Elidan, I. McGraw, and D. Koller. Residual belief propagation: Informed scheduling for asynchronous message passing. In *Proceedings of the 22nd Annual Conference on Uncertainty in Artificial Intelligence (UAI-06)*, Boston, Massachussetts, July 2006. AUAI Press.

W. T. Freeman, E. C. Pasztor, and O. T. Carmichael. Learning low-level vision. *International Journal of Computer Vision*, 40(1):25–47, October 2000.

R. G. Gallagher. *Low-density Parity Check Codes*. MIT Press, 1963. ISBN 978-0262571777.

T. Heskes, K. Albers, and H. J. Kappen. Approximate inference and constrained optimization. In *Proceedings of the 19th Annual conference on Uncertainty in Artificial Intelligence (UAI-03)*, pages 313–320, San Francisco, CA, 2003. Morgan Kaufmann Publishers.

T. Jaakkola and M. I. Jordan. Variational probabilistic inference and the QMR-DT network. *Journal of Artificial Intelligence Research*, 10:291–322, 1999.

D. B. Johnson. Finding all the elementary circuits of a directed graph. *SIAM Journal on Computing*, 4(1):77–84, 1975.

M. Jordan, Z. Ghahramani, T. S. Jaakkola, and L. Saul. An introduction to variational methods for graphical models. In *Learning in Graphical Models*, pages 105–161. Cambridge, MA: MIT Press, 1999.

F. R. Kschischang, B. J. Frey, and H.-A. Loeliger. Factor graphs and the sum-product algorithm. *IEEE Transactions on Information Theory*, 47(2):498–519, February 2001.

S. L. Lauritzen and D. J. Spiegelhalter. Local computations with probabilities on graphical structures and their application to expert systems. *Journal of the Royal Statistical society. Series B-Methodological*, 50(2):154–227, 1988.

M. A. R. Leisink and H. J. Kappen. A tighter bound for graphical models. *Neural Computation*, 13 (9):2149–2171, 2001. ISSN 0899-7667.

D. Malioutov, J. Johnson, and A. Willsky. Walk-sums and belief propagation in gaussian graphical models. *Journal of Machine Learning Research*, 7(Oct):2031–2064, 2006.

R. McEliece, D. J. C. MacKay, and J. Cheng. Turbo decoding as an instance of Pearl's belief propagation algorithm. *Journal on Selected Areas of Communication*, 16(2):140–152, 1998.

M. Mézard, G. Parisi, and R. Zecchina. Analytic and algorithmic solution of random satisfiability problems. *Science*, 297(5582):812–815, August 2002. URL `http://www.sciencemag.org/cgi/content/abstract/297/5582/812`.

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. URL `http://arxiv.org/abs/cond-mat/0506769`.

J. M. Mooij and H. J. Kappen. On the properties of the Bethe approximation and loopy belief propagation on binary networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2005 (11):P11012, 2005.

J. M. Mooij and H. J. Kappen. Loop corrections for approximate inference on factor graphs. *Journal of Machine Learning Research*, 8(May):1113–1143, 2007. URL `http://arxiv.org/abs/cs/0612030`.

J. M. Mooij, B. Wemmenhove, H. J. Kappen, and T. Rizzo. Loop corrected belief propagation. In *Proceedings of the 11th International Conference on Artificial Intelligence and Statistics (AISTATS-07)*, 2007.

K. P. Murphy, Y. Weiss, and M. I. Jordan. Loopy belief propagation for approximate inference: An empirical study. In *Proceedings of the 15th Annual Conference on Uncertainty in Artificial Intelligence (UAI-99)*. Morgan Kaufmann Publishers, 1999.

J. Parisi and F. Slanina. Loop expansion around the Bethe-Peierls approximation for lattice models. *Journal of Statistical Mechanics: Theory and Experiment*, 2006(02):L02003. URL `http://stacks.iop.org/1742-5468/2006/L02003`.

J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann Publishers, San Francisco, CA, 1988. ISBN 1558604790.

A. Pelizzola. Cluster variation method in statistical physics and probabilistic graphical models. *Journal of Physics A: Mathematical and General*, 38(33):R309–R339(1), August 2005. URL `http://arxiv.org/abs/cond-mat/0508216`.

G. Potamianos and J. Goutsias. Stochastic approximation algorithms for partition function estimation of Gibbs random fields. *IEEE Transactions on Information Theory*, 43(6):1948–1965, November 1997.

M. A. Shwe, B. Middleton, D. E. Heckerman, M. Henrion, E. J. Horvitz, H. P. Lehman, and G. F. Cooper. Probabilistic diagnosis using a reformulation of the INTERNIST-1/QMR knowledge base. I. The probabilistic model and inference algorithms. *Methods of Information in Medicine*, 30(4):241–55, October 1991.

J. Sun, Y. Li, S. B. Kang, and H. Y. Shum. Symmetric stereo matching for occlusion handling. In *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR-05)*, volume 2, pages 399–406, Washington, DC, USA, 2005. IEEE Computer Society.

M. Takinawa and B. D'Ambrosio. Multiplicative factorization of noisy-MAX. In *Proceedings of the 15th Conference on Uncertainty in Artificial Intelligence (UAI-99)*, pages 622–30, San Francisco, CA, 1999. Morgan Kaufmann.

R. E. Tarjan. Enumeration of the elementary circuits of a directed graph. *SIAM Journal on Computing*, 2(3):211–216, 1973.

J. C. Tiernan. An efficient search algorithm to find the elementary circuits of a graph. *Communications of the ACM*, 13(12):722–726, 1970.

M. Wainwright, T. Jaakkola, and A. Willsky. A new class of upper bounds on the log partition function. *IEEE Transactions on Information Theory*, 51(7):2313–2335, July 2005.

M. Welling, T. Minka, and Y. W. Teh. Structured region graphs: Morphing EP into GBP. In *Proceedings of the 21th Annual Conference on Uncertainty in Artificial Intelligence (UAI-05)*, page 609, Arlington, Virginia, 2005. AUAI Press.

W. Wiegerinck, H. J. Kappen, E. W. M. T. ter Braak, W. J. P. P. Burg, M. J. Nijman, Y. L. O, and J. P. Neijt. Approximate inference for medical diagnosis. *Pattern Recognition Letters*, 20(11): 1231–1239(9), 1999.

J. S. Yedidia, W. T. Freeman, and Y. Weiss. Generalized belief propagation. In T.K. Leen, T.G. Dietterich, and V. Tresp, editors, *Advances in Neural Information Processing Systems 13 (NIPS-00)*, pages 689–695, December 2000.

J. S. Yedidia, W. T. Freeman, Y. Weiss, and A. L. Yuille. Constructing free-energy approximations and generalized belief propagation algorithms. *IEEE Transactions on Information Theory*, 51(7): 2282–2312, July 2005.

A. L. Yuille. CCCP algorithms to minimize the Bethe and Kikuchi free energies: Convergent alternatives to belief propagation. *Neural Computation*, 14(7):1691–1722, 2002. ISSN 0899-7667.

S. Zilberstein. Using anytime algorithms in intelligent systems. *AI magazine*, 17(3):73–83 (1p.1/4), 1996. ISSN 0738–4602.

Journal of Machine Learning Research 8 (2007) 2017-2045 Submitted 4/05; Revised 2/07; Published 9/07

Very Fast Online Learning of Highly Non Linear Problems

Aggelos Chariatis CHARIATISAD@DIAS.COM.GR
Alamanas 2
15125 Athens, Greece

Editor: Léon Bottou

Abstract

The experimental investigation on the efficient learning of highly non-linear problems by online training, using ordinary feed forward neural networks and stochastic gradient descent on the errors computed by back-propagation, gives evidence that the most crucial factors for efficient training are the hidden units' differentiation, the attenuation of the hidden units' interference and the selective attention on the parts of the problems where the approximation error remains high. In this report, we present global and local selective attention techniques and a new hybrid activation function that enables the hidden units to acquire individual receptive fields which may be global or local depending on the problem's local complexities. The presented techniques enable very efficient training on complex classification problems with embedded subproblems.

Keywords: neural networks, online training, selective attention, activation functions, receptive fields

1. Framework

Online supervised learning is in many cases the only practical way of learning. This includes situations where the problem size is very big, or situations where we have a non-recurring stream of input vectors that are unavailable before training begins. We examine online supervised learning using a particular class of adaptive models, the very popular feed forward neural networks, trained with stochastic gradient descent on the errors computed by back-propagation.

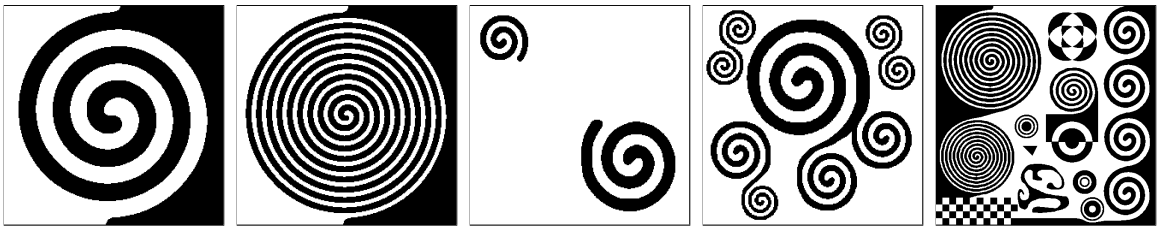
In order to easily visualize the online training dynamics of highly complex non linear problems, we are experimenting on 2:η:1 networks where the input is a point in a two dimensional image and the output is the value of the pixel at the corresponding input position. This framework allows the creation of very complex non-linear problems, just by hand-drawing the problem on a bitmap and presenting it to the network. Most problems' images in this report are 256×256 pixels in size, producing in total 65536 different samples each one.

Classification and regression problems can be modeled as black & white and gray scale images respectively. In this report we only examine training on classification problems. However, since mixed problems are possible, we are only interested on techniques that can be applied to both classification and regression.

The target of this investigation is online training where the input is not known in advance, so the input samples are treated as random and non-recurring vectors from the input space and are discarded after being used. We select and train on random samples until the average classification or RMS error is acceptable. Since both the number of training exemplars and the complexity of the underlying function are assumed unknown, we require from our training mechanism to have "initial state invariance" as a fundamental property. Thus we deliberately exclude from our arsenal any

training techniques that require a schedule to be decided ahead of training. Ideally we would like from the training mechanism to be totally invariant to the initial training parameters and network state.

This report is organized as follows: Sections 2 and 3 describe techniques for global and local selective attention. Section 4 is devoted to acceleration of training. In Section 5 we present experimental results and in Section 6 we discuss the presented techniques and give some directions for future research. Finally, Appendix A contains a description of the notations that have been used. In Figure 1 you can see some examples of problems that can be learned very efficiently using the techniques that are presented in the following sections.

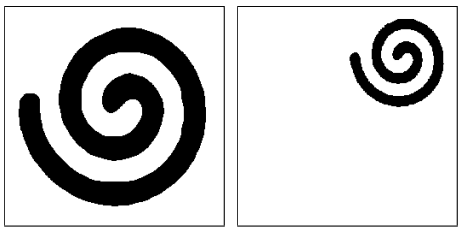


(a) (b) (c) (d) (e)

Figure 1: Examples of complex non-linear problems that can be learned efficiently.

2. Global Selective Attention - Dynamic Training Set Evolution

Consider the two problems depicted in Figure 2. Clearly, both problems are of approximately equal complexity, since they encapsulate the same image in a different scale and position within the input space. We would like to have a mechanism that will make the network capable of learning both problems at about the same speed.



(a) (b)

Figure 2: Approximately equal complexity problems.

Intuitively, the samples on the boundaries, which are the samples on positions with the highest contrast, are those that determine the complexity of each problem. During training, these samples have the property that they produce the highest errors. We thus need a method that will focus attention on samples with high error relatively to the rest. Previous work on such global selective attention has been published by many authors (Munro, 1987; Yu and Simmons, 1990; Bakker, 1992, 1993; Schapire, 1999; Zhong and Ghosh, 2000).

Of particular interest are the various boosting algorithms, such as AdaBoost (Schapire, 1999), which work by placing more emphasis on training samples that the system is currently getting wrong. Unfortunately, the most successful of these algorithms require a predefined set of samples on which training will be performed, something that is excluded from our scenario. Nevertheless, in a less constrained scenario, boosting can be applied on top of our techniques as a meta-learning algorithm, using our techniques as the base-learning algorithm.

A simple method that can provide such an adaptive selective attention mechanism, by keeping an exponential trace of the average error in the training set, is described in Algorithm 1.

$\bar{e} \leftarrow 0$
Repeat
    Pick a random sample
    Evaluate the error e for the sample
    If $e > 0.5\ \bar{e}$
        $\bar{e} \leftarrow e\ \alpha + \bar{e}\ (1 - \alpha)$
        Train
    End
Until a stopping criterion is satisfied

In this report's context, *error evaluation* and *train* are defined as:
Error Evaluation: Computation of the output values by forward propagating the activations from the input to the output layer for a single sample, plus computation of the output errors. The sample's error e is set to the quadratic mean (RMS) of the output units' errors.
Train: Back-propagation of the output errors to the hidden layer and immediate weights' adjustment.

Algorithm 1: The dynamic training set evolution algorithm.

The algorithm evaluates the errors of all samples, but trains only for samples with error greater than half the average error of the current training set. Training is initially performed for all samples, but gradually, it is concentrated on the samples at the problem's boundaries. When the error for these samples is reduced, other previously excluded samples enter the training set. Thus, samples enter and leave the training set automatically, with a tendency to train on samples with high error.

The magnitude of the constant α that determines the time scale of the exponential trace is problem specific, but in all experiments in this report it was kept fixed to 10^{-4}. The fraction of 0.5 was determined experimentally to give a good balance between sample selectivity and training set size. If it is close to 0 then we train for almost all samples. If it is close to 1 then we are at risk of making the training set starve from samples. Of course, one can choose to vary it dynamically in order to have a fixed percentage of samples in the training set, or, to not allow the training percentage to fall below a pre-specified limit.

Figure 3 shows the training set evolution for the two-spirals problem (in Figure 1a) in various training stages. The network topology was 2:64:1. You can see the training set forming gradually and tracing the problem boundaries where the error is the highest.

One could argue that such a process may be very sensitive to outliers. Experiments have shown that this does not happen. The algorithm does not try to recognize the outliers, but at least, adjusts naturally by not allowing the training set size to shrink. So, at the presence of heavy noise, the algorithm becomes ineffective, but does not introduce any additional harm. Figure 4 shows the two-

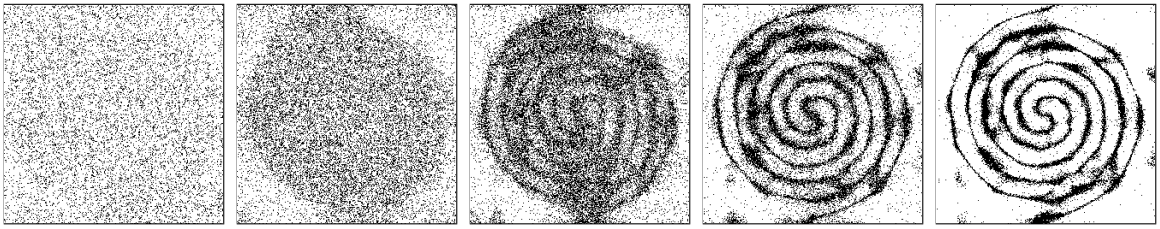


Figure 3: Training set evolution for the two-spirals problem. Under each image you can see the stage of training in trains, error-evaluations and the percentage of samples for which training is performed at the corresponding stage.

spirals problem distorted by dynamic noise and the corresponding training set after 90000 trains with 64 hidden units. You can see that the algorithm tolerates noise by not allowing the training set size to shrink. It is also interesting that at noise levels as high as 30% the algorithm can still exclude large areas of the input space from training.

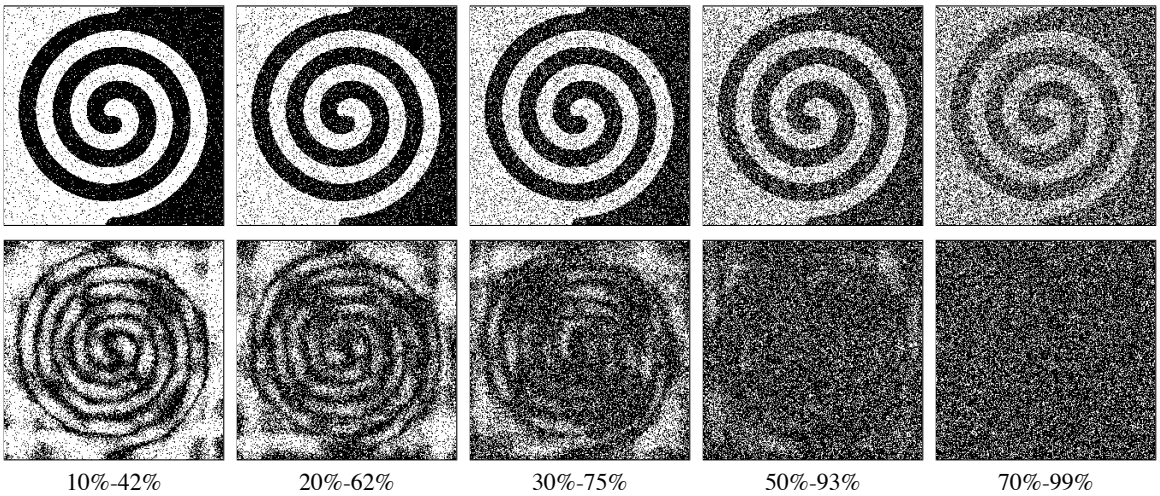


Figure 4: Top row shows the model with a visualization of the applied dynamic noise. Bottom row shows the corresponding training sets after 90000 trains. Under each pair of images you can see the percentage of noise distortion applied to the original input and the percentage of samples for which training is performed.

3. Local Selective Attention - Receptive Fields

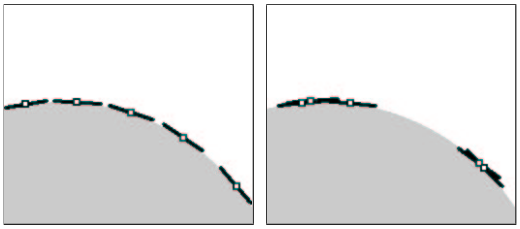
Having established a global method to focus attention on the important parts of a problem, we now come to address the main issue, which is the network training. Let first discuss the roles of the hidden and output layers in a feed forward neural network with a single hidden layer and without shortcut input-to-output connections.

The hidden layer is responsible for transforming a non linear input-to-output mapping, into a non linear input-to-hidden layer mapping, that can be mapped linearly to the output.

The output layer is responsible for learning a linear hidden-to-output mapping (which is an easy job), but most importantly, it must provide to the hidden layer error gradient information that will be used for the error credit assignment problem. In this respect, it becomes apparent that all hidden units should receive the most possibly accurate error information. That is why, we must train all hidden to output connections and back propagate the error through all these connections.

This is not the case for the hidden layer. Consider, for a classification problem, how the hidden units with sigmoidal activations partition the input space into sub areas. By adjusting the input-to-hidden weights and biases, each hidden unit develops a hyperplane that bi-partitions the input space in the most useful sense.

We would like to limit the number of hyperplanes in order to reduce the system's available degrees of freedom and obtain better generalization capabilities. At the same time, we would like to thoroughly use them in order to optimize the input output approximation. This can be done by arranging the hyperplanes to touch the problem's boundaries at regular intervals dictated by the boundary curvature, as it is shown in Figure 5a. Figure 5b, shows a suboptimal placement of the hyperplanes which causes a waste of resources. Each hidden unit must be differentiated from the others and ideally not interfere with the subproblems that the other units are trying to solve. Suppose that two hidden units are governed by the same, or nearly the same, parameters. How can we differentiate them? There are many possibilities.



(a) (b)

Figure 5: Optimal vs. suboptimal hyperplanes.

One could be, to just throw one unit away and make the output weight of the other equal to the sum of the two original output weights. That would leave the function unchanged. However, identifying these similar units during training is not easy computationally. In addition, we would have to figure out a method that would compute the best initial placement for the hyperplane of the new unit that would substitute the one that was thrown away.

Another possibility would be to add noise in the weight updates, gradually reduced with a simulated annealing schedule which should be decided before training begins. Unfortunately, the loss of initial state invariance would complicate training for unknown complex non linear problems.

To our thinking, it is much better to embed constraints into the system, so that it will not be possible for two hidden units to develop the same hyperplane. Two computationally efficient techniques to embed such constraints are described in sections 3.1 and 3.2.

Many other authors have also examined methods for local selective attention. For the related discussions see Huang and Huang (1990), Ahmad and Omohundro (1990), Baluja and Pomerleau (1995), Flake (1998), Duch et al. (1998), and Phillips and Noelle (2004).

3.1 Fixed Cascaded Inhibitory Connections

A problem with the hidden units of conventional feed forward networks is that they are all fed with the same inputs and back propagated errors and that they operate without knowing each other's existence. So, nothing prevents them from behaving identically. This lack of communication between hidden units has been addressed by researchers through hidden unit lateral connections. Agyepong and Kothari (1997) use unidirectional lateral interconnections between adjacent hidden layer units, claiming that these connections facilitate the controlled assignment of role and specialization of the hidden units. Kothari and Ensley (1998) use Gaussian lateral connections which enable the hidden decision boundaries to be global in nature but also be able to represent local structure. Numerous neural network algorithms employ bidirectional lateral inhibitory connections in order to generate competition between the hidden units. In an interesting variation described by Spratling and Johnson (2004), competition is provided by each hidden unit blocking its preferred inputs from activating other units.

We use a single hidden layer where the hidden units are considered sequenced. Each hidden unit is connected to all succeeding hidden units with a fixed connection with weight set to minus one. The hidden units get differentiated, because they receive different inputs, they produce different activations and they get back different error information. Another benefit is that they can generate higher order feature detectors, that is, the resulting hidden hyperplanes are no longer strictly linear, but they may also be curved. Considering the fixed value, -1 is used just to avoid a multiplication. Values from -0.5 to -2 give good results as well.

As it is shown in Section 5.1.1, the fixed cascaded inhibitory connections are very effective at reducing a problem's asymptotic residual error. This should be attributed to both of their abilities, to generate higher order feature detectors and to hasten the hidden units' symmetry breaking.

These connections can be implemented very efficiently with just one subtraction per hidden unit for both hidden activation and hidden error computation. In addition, the disturbance to the parallelism of the backpropagation algorithm is minimal. Most operations on the hidden units can still be done in parallel and only the final computations must be performed sequentially. We include the algorithms for the hidden activation and error computations as examples of sequential implementations. These changes can be very easily incorporated into conventional neural network code.

Hidden Activations	Hidden Error Signals
$\delta \leftarrow 0$	$\delta \leftarrow 0$
For $j \leftarrow 1 \ldots \eta$	For $j \leftarrow \eta \ldots 1$
$\quad n_j \leftarrow \delta + \vec{x} \cdot \vec{w}_j$	$\quad e_j \leftarrow \delta + \vec{r} \cdot \vec{u}_j$
$\quad h_j \leftarrow f(n_j)$	$\quad g_j \leftarrow e_j f'(n_j)$
$\quad \delta \leftarrow \delta - h_j$	$\quad \delta \leftarrow \delta - g_j$
End	End

Algorithm 2: Hidden unit activation and error computation with Fixed Cascaded -1 Connections.

3.2 Selective Training of the Hidden Units

The hidden units' differentiation can be farther magnified if each unit is not trained on all samples, but only on the samples for which it receives a high error.

We train all output units, but only the hidden units for which the error signal is higher than the RMS of the error signals of all hidden units. Typically about 10% of the hidden units are trained on each sample during early training and the percentage falls up to 2% when the network is close to the solution.

This is intuitively justified by the observation that at the beginning of training the hidden units are largely undifferentiated and receive high error signals from the whole input space. At the final stage of training, the hidden hyperplanes' linear soft decision boundaries are combined by the output layer to define arbitrarily shaped decision boundaries. For μ input dimensions, from 1 up to μ units can define an open sub-region and $\mu+1$ units are enough to define a closed convex region. With such high level constructs, each sample may be discriminated from the rest with very few hidden units. These, are the units that receive the highest error signal for the sample.

Experiments on various problems have shown that training on a fraction of the hidden units is always better (in respect to number of trains to convergence), than training all or just one hidden unit. It seems that training only one hidden unit on each sample is not sufficient for some problems (Thornton, 1992). Measurements for one of these experiments are reported in Section 5.1.1.

In addition to the convergence acceleration, the combined effect of training a fraction of the hidden units on a fraction of the samples, gives big savings in CPU usage per sample as well. This sparseness of training in respect to evaluation provides further opportunities for speedup as it is discussed in Section 4.

3.3 Centering On The Input Space

It is a well known recommendation (Schraudolph, 1998a,b; LeCun et al., 1998) that the input values should be normalized to have zero mean and unit standard deviation over each input dimension. This is achieved by subtracting from each input value the mean and dividing by the standard deviation.

For some problems, like the one in Figure 2b, the center of the input space is not equal to the center of the problem. When the input is not known in advance, the later must be computed adaptively. Moreover, since the hidden units are trained on different input samples, we should compute for each hidden unit its own mean and standard deviation over each input dimension.

For the connection between hidden unit j and input unit i we can adaptively compute the approximate mean m_{ji} and standard deviation s_{ji} over the inputs that train the hidden unit, using either exponential traces:

$$m_{ji_{(t)}} \leftarrow \beta\, x_i + (1-\beta)\, m_{ji_{(t-1)}},$$

$$q_{ji_{(t)}} \leftarrow \beta\, x_i^2 + (1-\beta)\, q_{ji_{(t-1)}},$$

$$s_{ji_{(t)}} \leftarrow (q_{ji_{(t)}} - m_{ji_{(t)}}^2)^{1/2},$$

or perturbated calculations:

$$m_{ji_{(t)}} \leftarrow m_{ji_{(t-1)}} + \beta\, (x_i - m_{ji_{(t-1)}}),$$

$$v_{ji_{(t)}} \leftarrow v_{ji_{(t-1)}} + \beta \left((x_i - m_{ji_{(t)}})\, (x_i - m_{ji_{(t-1)}}) - v_{ji_{(t-1)}} \right),$$

$$s_{ji(t)} \leftarrow v_{ji(t)}^{1/2},$$

where β is a constant that determines the time scale of exponential averaging, vector $\vec{x}$ holds the input values, matrix Q holds the means of the squared input values and matrix V holds the variances.

The means and standard deviations of a hidden unit's input connections are updated only when the hidden unit is trained. The result of this treatment is that each hidden unit is centered on a different part of the input space. This center is indirectly affected by the error that the inputs produce on the hidden unit.

The magnitude of the constant β is problem specific, but in all experiments in this report it was kept fixed to 10^{-3}. This constant must be selected large enough, so that the centers will rapidly move to their optimum locations, and small enough, so that the hidden units will see a relatively static view of the input space and the gradient descent algorithm will not be confused. As the hidden units jitter around their centers, we effectively train them on slightly shifted views of the input space, something that can assist generalization. We get something analogous to training with jitter (Reed et al., 1995), at no extra cost.

In Figure 6, the squares show where each hidden unit is centered. You can see that most are centered on the problem boundaries at regular intervals. The crosses show the standard deviations. On some directions the standard deviations are very small, which results in very high normalized input values, causing the hidden units to act as threshold units at those directions. The sloped lines show the hyperplane distance from center and the slope. These are computed for display purposes, from their theoretical formulas for a conventional network, without considering the effect of the cascaded connections.

For some units the hyperplanes shown are not exactly on the boundaries. This is because of the fixed cascaded connections that cause the hidden units to be not exactly linear discriminants. In the last picture you can see the decision surface of a hidden unit which is a bit curved and coincides with the class boundary although its calculated hyperplane is not on the boundary.

An observant reader may also notice that the hyperplane distances from the centers are very small, which implies that the corresponding biases are small as well. On the contrary, if all hidden units were centered on the center of the image, we would have the following problem. The hyperplanes of some hidden units must be positioned on the outer parts of the image. For this to happen, these units should develop large biases in respect to the weights. This would make their activations to have small variances. These small variances might need to be compensated by large output weights and biases, which would saturate the output units and in addition ill-condition the problems.

One may wonder if the hidden biases are still necessary. Since the centers are individually set, it may seem at first that they are not. However, the centers are not trained through error backpropagation, and the hyperplanes do not necessarily pass over them. The biases role is to drive the hyperplanes to the correct location and thus pull the centers in the corresponding direction.

The individual centering of the hidden units based on the samples' positions is feasible, because we train only on samples with high errors and only the hidden units with high errors. By ignoring the small errors, we effectively position the center of each hidden unit near the center of mass of the high errors that it receives. However, this centering technique can still be used even if one chooses to train on all samples and all hidden units. Then, the statistics interval should be differentiated for each hidden unit and be recomputed for each sample relatively to the normalized absolute error that each hidden unit receives. A way to do it is to set the effective statistics interval for hidden unit j

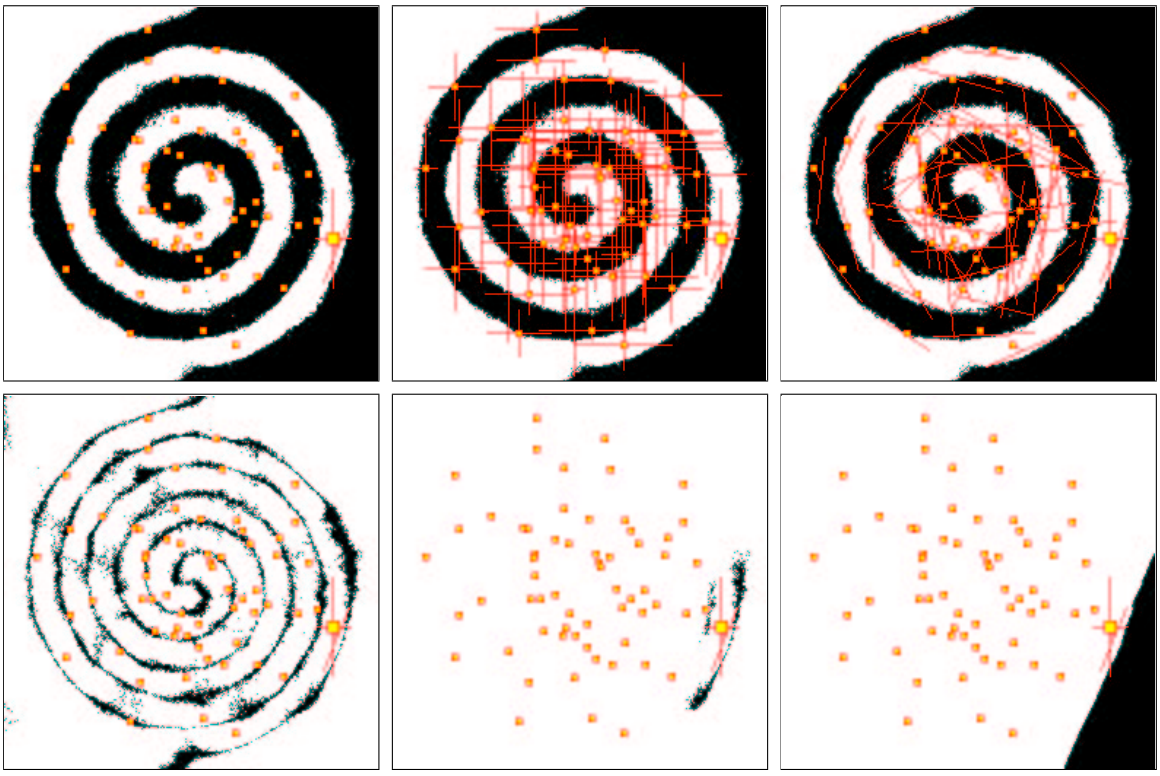


Figure 6: Hidden unit centers, standard deviations, hyperplanes, global and local training sets and a hidden unit's output. The images were captured at the final stage of training, of the problem in Figure 1a with 64 hidden units.

and sample s to:

$$\beta \frac{|e_{j,s}|}{\langle |e_j| \rangle}$$

where β is the global statistics interval, $e_{j,s}$ is the hidden unit's backpropagated error for the sample and $\langle |e_j| \rangle$ is the mean of the absolute backpropagated errors that the hidden unit receives, measured via an exponential trace. The denominator acts as a normalizer, which makes the hidden unit's mobility to be independent of the average magnitude of the errors.

Centering on other factors has been extensively investigated by Schraudolph (1998a,b). These techniques can provide further convergence acceleration, but we chose not to use them because of the additional computational overhead that they require.

3.4 A Hybrid Activation Function

As it is shown in Section 5, the aforementioned techniques enable successful training on some difficult problems like those in Figures 1a and 1b. However, if the problem contains subproblems, or put in another way, if the problem generates more than one cluster of high error density, the centering mechanism does not manage to drive the hidden unit centers to the most suitable locations. The centers are attracted by the larger subproblem or get stuck in areas between the subproblems, as shown in Figure 7.

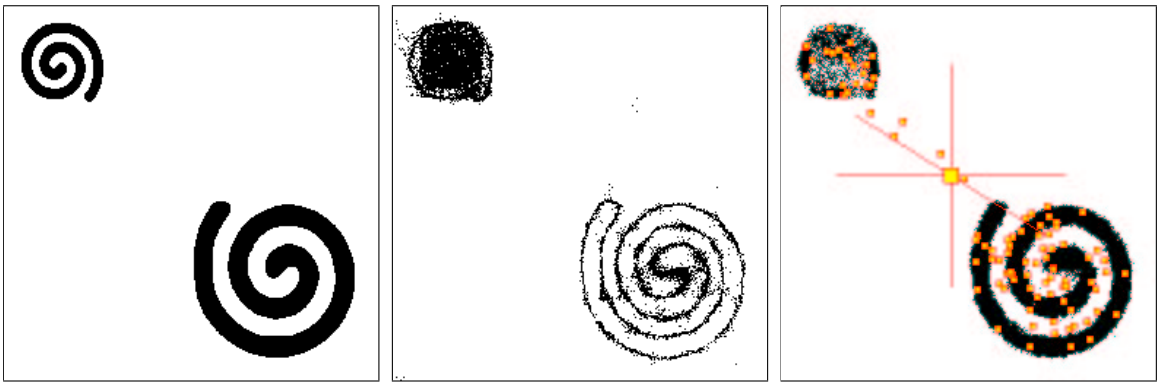


Figure 7: Model, training set, and inadequate centering

We need a mechanism that can force a hidden unit to get out of balanced but suboptimal positions. It would be nice if this mechanism could also allow the centers to migrate to various points in the input space as the need arises. It has been found that both of these requirements are fulfilled by a new hybrid activation function.

Sigmoid activations have the property that they produce hyperplanes that separate the input space globally. Our intention is to use a sigmoid like hidden activation function, because it can provide global separability, and at the same time, reduce the activation value towards zero on inputs which are not important to a hidden unit.

The Gaussian function is commonly used within radial basis function (RBF) neural networks (Broomhead and Lowe, 1988). When this function is applied to the distance of a sample to the unit's center, it produces a local response which is stronger near the center. We can then enclose the sigmoidal activation within a Gaussian envelope, by multiplying the activation with a value between 0 and 1, which is provided by applying the Gaussian function to the distance that is measured in the normalized input space.

When the number of input dimensions is large, the distance metric that must be used is not an obvious choice. Table 1 contains the distance metrics that we have considered. The most suitable distance metric seems to depend on the distribution of the samples that train the hidden units.

$\sqrt{\sum_{i=1}^{\mu} x_i^2}$	$\sqrt{\frac{1}{\mu}\sum_{i=1}^{\mu} x_i^2}$	$\sum_{i=1}^{\mu} \lvert x_i \rvert$	$\frac{1}{\mu}\sum_{i=1}^{\mu} \lvert x_i \rvert$	$\max\left(\lvert x_i \rvert\right)$
Euclidean	Euclidean Scaled	Manhattan	Manhattan Scaled	Chebyshev

Table 1: Various distance metrics that have been considered for the hybrid activation function.

In particular, if the samples follow a uniform distribution over a hypercube, then the Euclidean distance has the disturbing property that the average distance grows larger as the number of input dimensions increases and consequently the corresponding average Gaussian response decreases towards zero. As suggested by Hegland and Pestov (1999), we can make the average distance to center independent of the input dimensions, by measuring it in the normalized input space and then dividing it by the square root of the input's dimensionality. The same problem occurs for the Manhattan distance which has been claimed to be a better metric in high dimensions (Aggarwal et al., 2001). We can normalize this distance by dividing it by the input's dimensionality. A problem that

appears for both of the above rescaled distance metrics, is that for the samples that are near the axes the distances will be very much attenuated and the corresponding Gaussian responses will be close to one, something that will make the Gaussian envelopes ineffective.

A more suitable metric for this type of distributions is the Chebyshev metric whose average magnitude is independent of the dimensions. However, for reasons analogous to those mentioned above, this metric is not the most suitable if the distribution of the samples is spherical. In that case, the Euclidean distance does not need any rescaling and is the most natural distance measure. We can obtain spherical distributions by adaptively whitening them. As Plumbley (1993) and Laheld and Cardoso (1994) independently proposed, the whitening matrix Z can be adaptively computed as:

$$Z_{t+1} = Z_t - \lambda \left[\vec{z}_t \vec{z}_t^{\,T} - I\right] Z_t$$

where λ is the learning rate parameter, $\vec{z}_t = Z_t \, \vec{x}_t$ is the whitened vector and $\vec{x}_t$ is the input vector. However, we would need too many additional parameters to do it individually for each subset of samples on which each hidden unit is trained.

For the above reasons (and because of lack of a justified alternative), in the implementation of these techniques we typically use the Euclidean metric when the number of input dimensions is up to three and the Chebyshev metric in all other cases. We have also replaced the usual tanh (sigmoidal) and Gaussian (bell-like) functions, by similar functions which do not involve exponentials (Elliott, 1993).

For each hidden unit j we first compute the net-input n_j to the hidden unit (that is, the weighted distance of the sample to the hyperplane), as the inner product of normalized inputs and weights plus the bias:

$$\begin{aligned} z_{ji} &= \frac{x_i - m_{ji}}{s_{ji}}, \\ n_j &= \vec{z}_j \cdot \vec{w}_j. \end{aligned}$$

We then compute the sample's distance d_j to the center of the unit which is measured in the normalized input space:

$$d_j = \left\|\vec{z}_j\right\|.$$

Finally, we compute the activation h_j as:

$$\begin{aligned} a_j &= Elliott(n_j) = \frac{n_j}{(1+\left|n_j\right|)}, \\ b_j &= bell(d_j) = \frac{1}{(1+d_j^2)}, \\ h_j &= a_j \, b_j. \end{aligned}$$

Since d_j is not a function of $\vec{w}_j$, we treat b_j as a constant for the calculation of the activation derivative with respect to n_j, which becomes:

$$\frac{\partial h_j}{\partial n_j} = b_j \, (1 - \left|a_j\right|)^2.$$

The hybrid activation function, which by definition may only be used for hidden units connected to the input layer, enables these units to acquire selective attention capabilities on the input space. Each hidden unit may have a global or local receptive field on each input dimension. The size of this dimensional receptive field depends on the standard deviation which is computed for the corresponding dimension.

This activation makes balanced positions between subproblems to be unstable. As soon as the center is changed by a small amount, it will be attracted by the nearest subproblem. This is because the unit's activation and the corresponding error will be increased for samples towards the nearest subproblem and decreased at the other direction. Hidden units can still be centered between subproblems but only if their movement at either direction causes a large error for samples at the opposite direction, that is, if they are absolutely necessary at their current position.

Additionally, if a unit is centered near a subproblem that produces low errors and the unit is not necessary in that area, then it may migrate to other areas that still have high errors. This unit center migration has been observed in all experiments on complex problems. This may be due to the non-linear response of the bell function, and its long tails which keep the activation above zero for all input samples.

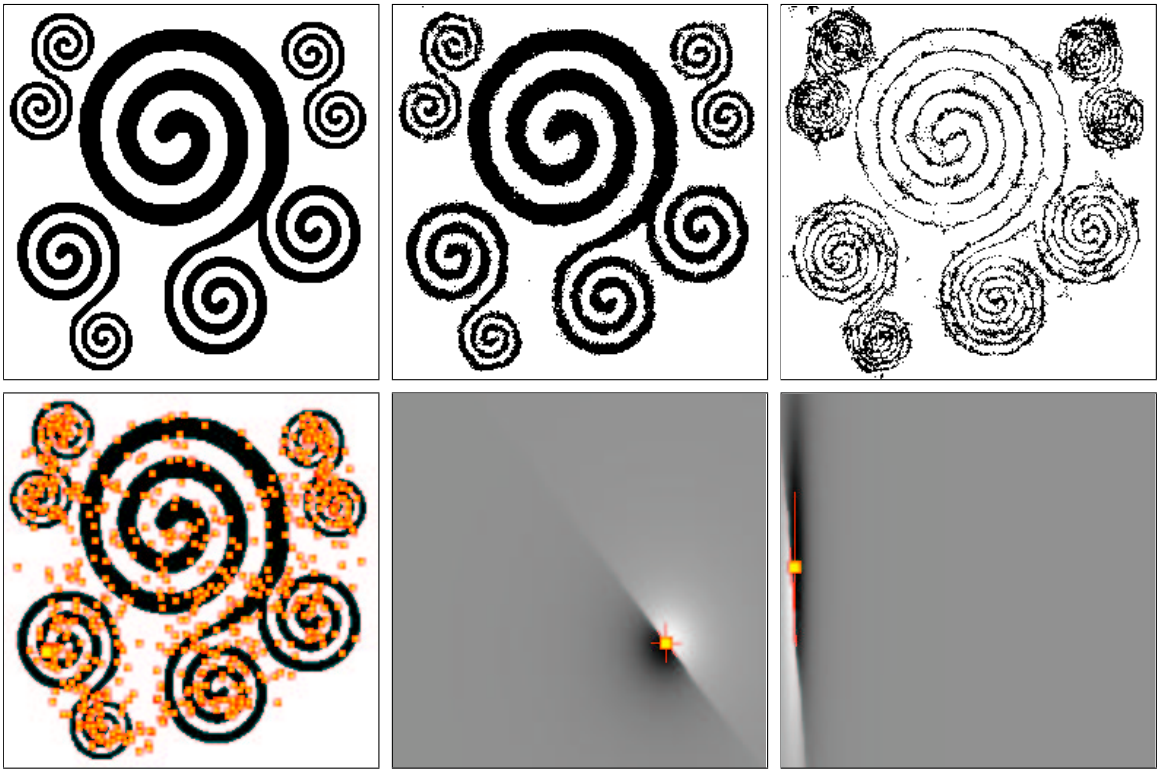


Figure 8: Model, evaluation, training set, hidden unit centers and two hidden unit outputs showing the effect of the hybrid activation function. The images were captured at the final stage of training, of the problem in Figure 1d with 700 hidden units.

In Figure 8 you can see a complex problem with 9 clusters of high errors. The hidden units place their centers on all clusters and are able to solve the problem. In the last two images, you can see the

effect of the hybrid activation function which attenuates the activation at points far from the center in respect to the standard deviation on each dimension. One unit develops a near circular local receptive field and one other develops an elongated ellipsoidal receptive field. The later provides almost global separation in the vertical direction and becomes a useful discriminant for two of the subproblems.

One may find similarities between this hybrid activation function and the Square-MLP architecture described by Flake (1998). The later, partially implements higher order neurons by duplicating the number of input units and setting the new input values equal to the squares of the original inputs. This architecture enables the hidden units to form local features of various shapes, but not the locally constrained sigmoid formed by our proposal. In contrast, the hybrid activation function does not need any additional parameters beyond those that are already used for centering and it has the additional benefit, which is realized by the local receptive fields in conjunction with the small biases and the symmetric sigmoid, that the hidden activations will have a mean close to zero. As discussed by Schraudolph (1998a,b) and LeCun et al. (1998), this is very beneficial for the output layer training.

However, there is still room for improvement. As it was also observed by Flake (1998), the orientations of the receptive field ellipses are always at the direction of one of the input axes. This limitation is expected to hinder performance when training hidden units which have sloped hyperplanes. Figure 9 shows a complex problem at the middle of training. Units with sloped hyperplanes are trained on samples whose input values are highly correlated. This can slowdown learning by itself, but in addition, the standard deviations cannot get sufficiently small and as a result the receptive field cannot be sufficiently shrunk at the direction perpendicular to the hyperplane. As a result the hidden unit's activation unnecessarily interferes with the activations of nearby units.

Although it may be possible to address the correlation problem with a more sophisticated training method that uses second order gradient information, like Stochastic Meta Descent (Schraudolph, 1999, 2002), the orientations of the receptive fields will still be limited. In Section 6.2 we discuss possible directions for further research that may circumvent this limitation.

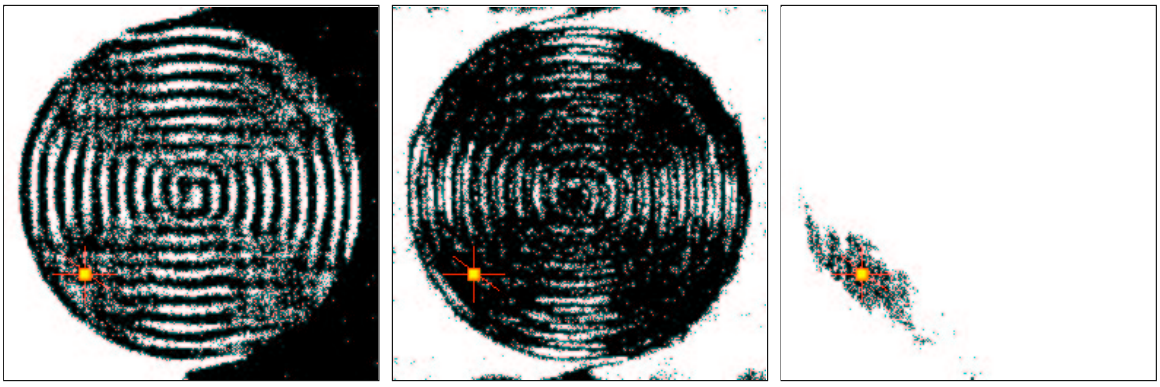


Figure 9: Evaluation and global and local training sets during middle training for the problem in Figure 1b. It can be seen that a hidden unit with a sloped hyperplane is trained on samples with highly correlated input values. Samples that are separated by horizontal or vertical hyperplanes are easier to be learned.

4. Further Speedups

In this section we first describe an implementation technique that reduces the computational requirements of the error evaluation phase and then we give references to methods that have been proposed by other authors for the acceleration of the training phase.

4.1 Evaluation Speedup

Two of the discussed techniques, training only for samples with high errors, and then, training only the hidden units with high error, make the error-evaluation phase to be the most processing demanding phase for the solution of a given problem. In addition, some other techniques, like board game learning through temporal difference methods, require many evaluations to be performed before each train. We can speedup evaluation by the following observation:

For many problems, only part of the input is changed on successive samples. For example, for a backgammon program with 200 input units (with raw board data and not any additional features encoded), very few inputs will change on successive positions. Even on two dimensional problems such as images, we can arrange to train on samples selected by random changes on the X and Y dimensions alternatively. This process of only resampling one coordinate at a time is also known as "Gibbs sampling" and it nicely generalises to more than two coordinates (Geman and Geman, 1984).

Thus, we can keep in memory all intermediate results from the evaluation, and recalculate only for the inputs that have changed. This implementation technique requires more storage, especially for high dimensional inputs. Fortunately, storage is not an issue on modern hardware.

4.2 Training Speedup

Many authors have proposed methods for speeding-up online training by using second order gradient information in order to dynamically vary either the learning rate or the momentum (see LeCun et al., 1993; Leen and Orr, 1993; Murata et al., 1996; Harmon and Baird, 1996; Orr and Leen, 1996; Almeida et al., 1997; Amari, 1998; Schraudolph, 1998c, 1999, 2002; Graepel and Schraudolph, 2002).

As it is shown in the next section, our techniques enable standard stochastic gradient descent with momentum to efficiently solve all the highly non-linear problems that have been investigated. However, the additional speed up that an accelerating algorithm can give is a nice thing to have. Moreover, these accelerating algorithms automatically reduce the learning rate when we are close to a solution (by sensing the oscillations in the error gradient) something that we should do through annealing if we wanted the best possible solution.

We use the Incremental Delta-Delta (IDD) accelerating algorithm (Harmon and Baird, 1996), an incremental nonlinear extension to Jacobs' (1988) Delta-Delta algorithm, because of its simplicity and relatively small processing requirements. IDD computes an individual learning rate λ for each weight w as:

$$\lambda(t) = e^{\xi(t)},$$

$$\xi(t+1) = \xi(t) + \theta \frac{\Delta w(t+1)}{\lambda(t)} \Delta w(t),$$

where θ is the meta-learning rate which we typically set to 0.1.

5. Experimental Results

In order to measure the effectiveness of the described techniques on various classes of problems, we performed several experiments. Each experiment was replicated 10 times with different random initial weights using matched random seeds and the means and standard deviations of the results were plotted in the corresponding figures.

For the experiments we used a single hidden layer, the cross entropy error function, the logistic or softmax activation function for the output units and the Elliott or hybrid activation function for the hidden units. Output to hidden layer weights and biases were initialized to zero. Hidden to input layer weights were initialized to random numbers from a normal distribution and then rescaled so that the incoming weights to each hidden unit had norm unity. Hidden unit biases were initialized to a uniform random number between zero and one.

The curves in the figures are labelled with a combination of the following letters which indicate the techniques that were applied:

B – Adjust weights using stochastic gradient descent with momentum 0.9 and fixed learning rate $0.1/\sqrt{c}$ where c is the number of incoming connections to the unit.

A – Adjust weights using IDD with meta-learning rate 0.1 and initial learning rate $1/\sqrt{c}$ where c is as above.

L – Use fixed cascaded inhibitory connections as described in Section 3.1.

S – Skip weights adjustment for samples with low error as described in Section 2.

U – Skip weights adjustment for hidden units with low error as described in Section 3.2.

C – Use individual means and stdevs for each hidden to input connection as described in Section 3.3.

H – Use the hybrid activation function as described in Section 3.4.

For the 'B' training method we deliberately avoided an annealing schedule for the learning rate, since this would destroy the initial state invariance of our techniques. Instead, we used a fixed small learning rate which we compensated with a large momentum. For the 'A' method, we used a small meta-learning rate, to avoid instabilities due to the high non-linearities of the examined problems. It is important to note that for both training methods the learning parameters were fixed to the above values and not optimized to each individual problem.

For the 'C' technique, the centers of the hidden units where initially set to the center of the input space and the standard deviations were set to one third of the distance between the extreme values of each dimension. When the technique was not used, a global preprocessing was applied which normalized the input samples to have zero mean and unit standard deviation.

5.1 Two Input Dimensions

In this section we give experimental results for the class of problems that we have mainly examined, that is, problems in two input and one output dimensions, for which we have dense and noiseless training samples from the whole input space. In the figures, we measure the average classification error in respect to the stage of training. The classification error was averaged via an exponential trace with time scale 10^{-4}.

5.1.1 COMPARISON OF TECHNIQUE COMBINATIONS

For these experiments we used the two-spirals problem shown in Figures 1a, 3, 4 and 6. We chose this problem as a non trivial representative of the class of problems that during early training generate a single cluster of high error density. The goal of this experiment is to measure the effectiveness of various technique combinations and then to measure how well the best technique combination scales with the size of the hidden layer.

Figures 10 and 11 show the average classification error in respect to the number of evaluated samples and processing cycles respectively for 13 technique combinations. For these experiments we used 64 hidden units. The standard deviations were not plotted in order to keep the figures uncluttered. Figure 10 has also been split to Figures 12 and 13 in order to show the related error bars.

Comparing the curves B vs. BL and BS vs. BLS on Figures 10 and 11, we can see that the fixed cascaded inhibitory connections reduce the asymptotic residual error by more than half. This also applies, but to a lesser degree, when we skip weight updates for hidden units with low errors (B vs. BU, BS vs. BSU). When used in combination, we can see a speed-up of convergence but the asymptotic error is only marginally further improved (BLU and BLSU).

In Figure 11, it can be seen that skipping samples with low errors can speed-up convergence and reduce the asymptotic error as well (BLU vs. BLSU). This is a very intriguing result, in the sense that it implies that the system can learn faster and better by throwing away information.

Both Figures 10 and 11 show the BLUCH curve to diverge. Considering the success of the BLSUCH curve, we can imply that skipping samples is necessary for the hybrid activation. However, the real problem, which was found out by viewing the dynamics of training, is that the centering mechanism does not work correctly when we train on all samples. A possible remedy may be to modify the statistics interval which is used for centering, as it is described at the end of Section 3.3.

BLSUC vs. BLSU shows that centering further reduces the remaining asymptotic error to half and converges much faster as well.

Comparing curve BLSUCH vs. BLSUC, we see that the hybrid activation function does better, but only marginally. This was expected since this problem has a single region of interest, so the ability of H to focus on multiple regions simultaneously is not exercised. This is the reason for the additional experiments in Section 5.1.2.

BLSUCH and ALSUCH were the most successful technique combinations, with the later being a little faster. Nevertheless, it is very impressive that standard stochastic gradient descent with momentum can approach the best asymptotic error in less than a second, when using a modern 3.2 GHz processor.

Figure 14 shows the average classification error in respect to the number of evaluated samples, for the ALSUCH technique combination and various hidden layer sizes. It can be seen that the asymptotic error is almost inversely proportional to the number of hidden units. This is a good indication that our techniques use the available resources efficiently. It is also interesting, that the convergence rates to the corresponding asymptotic errors are quite fast and about the same for all hidden layer sizes.

5.1.2 HYBRID VS. CONVENTIONAL ACTIVATION

For these experiments we used the two dimensional problem depicted in Figures 1c and 7. We chose this problem as a representative of the class of problems that during early training generate

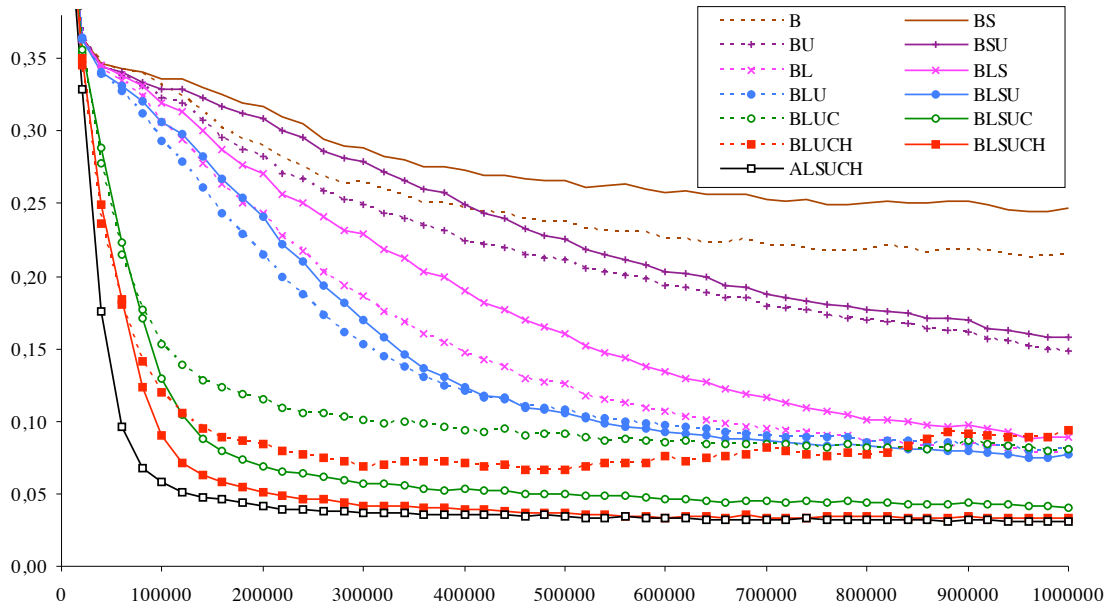


Figure 10: Average classification error vs. number of evaluated samples for various technique combinations, while training the problem in Figure 1a with 64 hidden units. The standard deviations have been omitted for clarity.

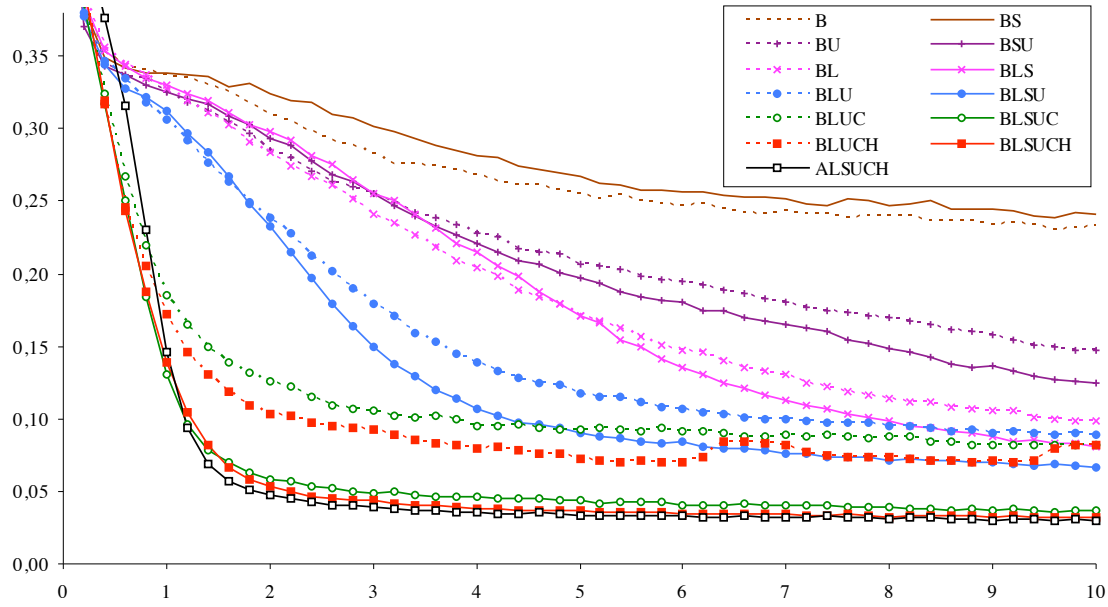


Figure 11: Average classification error vs. Intel IA32 CPU cycles in billions, for various technique combinations, while training the problem in Figure 1a with 64 hidden units. The horizontal scale also corresponds to seconds when run on a 1 GHz processor. The standard deviations have been omitted for clarity.

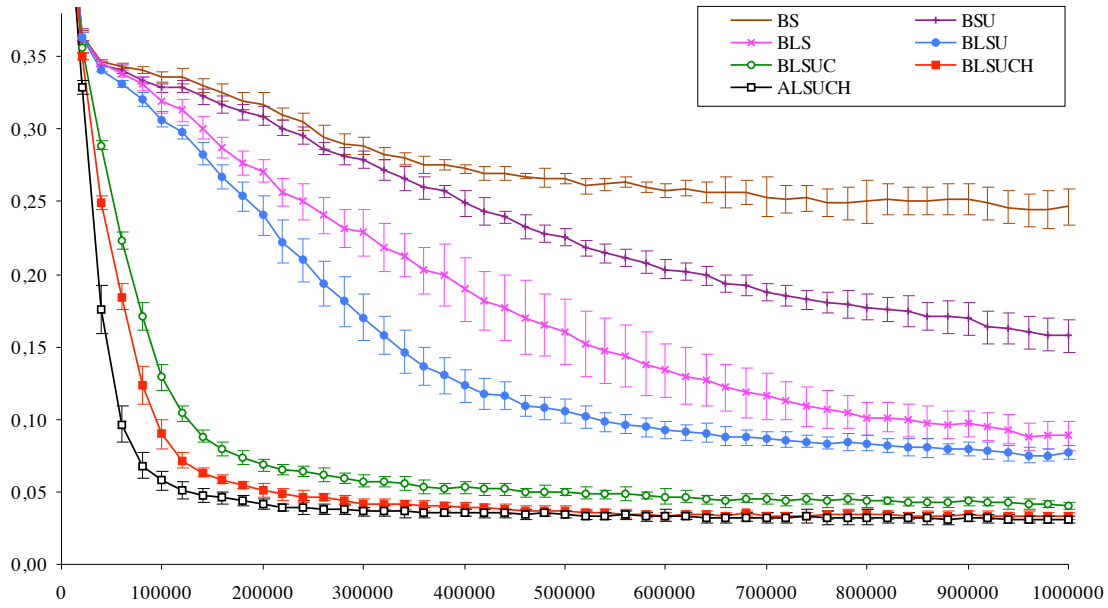


Figure 12: Part of Figure 10 showing error bars for technique combinations which employ S.

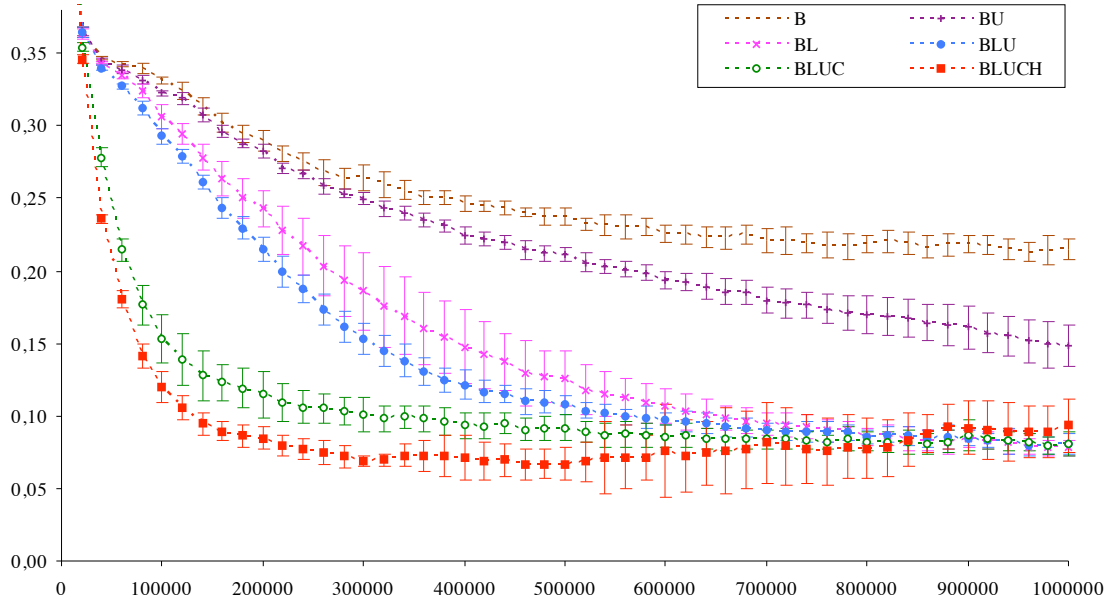


Figure 13: Part of Figure 10 showing error bars for technique combinations which do not employ S.

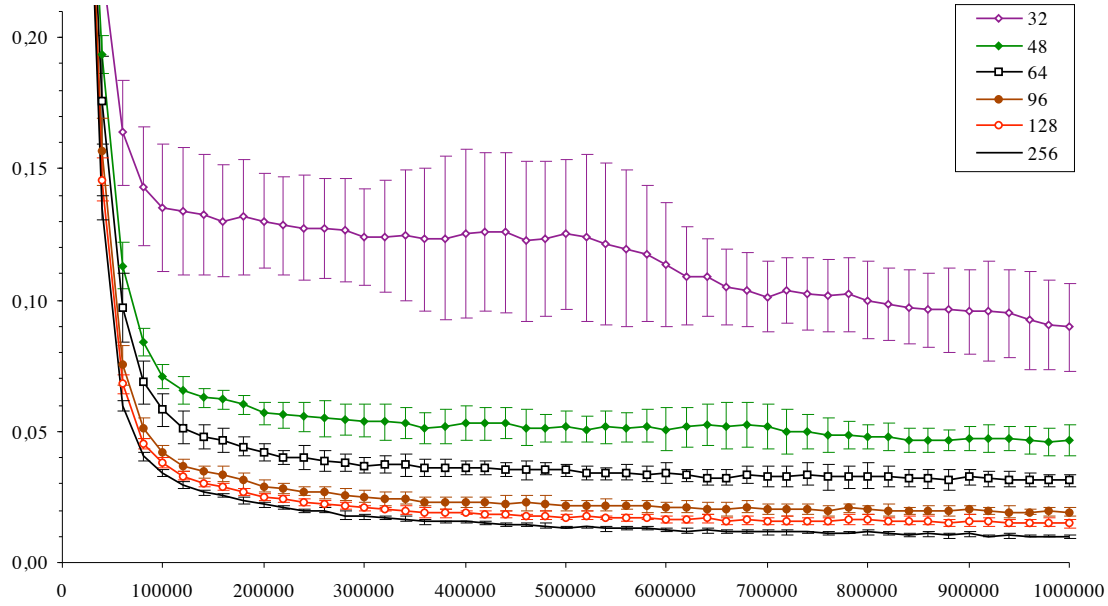


Figure 14: Average classification error vs. number of evaluated samples for various hidden layer sizes, while training the problem in Figure 1a with the ALSUCH technique combination.

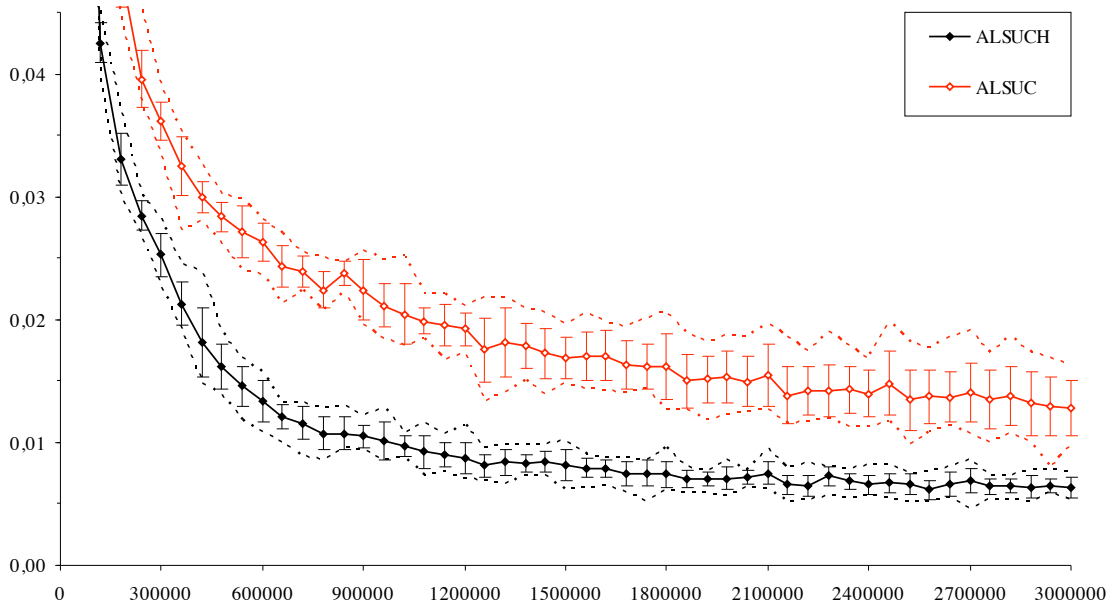
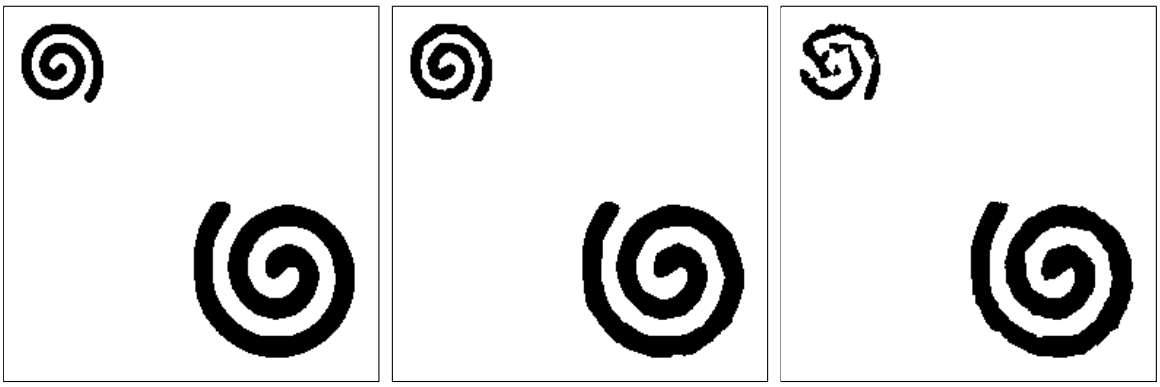


Figure 15: Average classification error vs. number of evaluated samples for the ALSUCH and ALSUC technique combinations, while training the problem in Figure 1c with 100 hidden units. The dashed lines show the minimum and maximum observed values.

small clusters of high error density of various sizes. For this kind of problems we typically obtain very small residuals for the classification error, although the problem may not have been learned. This is because we measure the error on the whole input space and for these problems most of the input space is trivial to be learned. The problem's complexities are confined in very small areas. The dynamic training set evolution algorithm is able to locate these areas, but we need much more sample presentations, since most of the samples are not used for training.

The goal of this experiment is to measure the effectiveness of the hybrid activation function at coping with the varying sizes of the subproblems. For these experiments we used 100 hidden units.

Figure 15 shows that the ALSUCH technique, which employs the hybrid activation function, reduced the asymptotic error to half in respect to the ALSUC technique. As all of the visual inspections revealed, one of which is reproduced in Figure 16, the difference in the residual errors of the two curves is due to the insufficient approximation of the smaller subproblem by the ALSUC technique.



Model ALSUCH ALSUC

Figure 16: ALSUCH vs. ALSUC approximations for a problem with two sub-problems.

5.2 Higher Input and Output Dimensions

In order to evaluate our techniques on a problem with higher input and output dimensions, we selected a standard benchmark, the Letter recognition database from the UCI Machine Learning Repository (Newman et al., 1998).

This database consists of 20000 samples that use 16 integer attributes to classify the 26 letters of the English alphabet. This problem is characterized by a medium input dimensionality and a large output dimensionality. The later, makes it a very challenging problem for any classifier.

This problem differs from those on which we have experimented so far, in that we do not have the whole input space at our disposal for training. We must train on a limited number of samples and then test the system's generalization abilities on a separate test set. Although we have not taken any special measures to assist generalization, the experimental results indicate that our techniques have the inherent ability to generalize well, when given noiseless exemplars.

An observation that applies to this problem is that the IDD accelerated training method could not do better than standard stochastic gradient descent with momentum. Thus, we report results using

the BLSUCH technique combination which is computationally more efficient than the ALSUCH technique.

For this experiment, which involves more than two output classes, we used the softmax activation function at the output layer.

Table 2 contains previously published results showing the classification accuracy of various classifiers. The most successful of them were the AdaBoosted versions of the C4.5 decision-tree algorithm and of a feed forward neural network with two hidden layers. Both classifier ensembles required quite a lot of machines in order to achieve that high accuracy.

Classifier	**Test Error %**	**Reference**
Naive Bayesian classifier	25,3	Ting and Zheng (1999)
AdaBoost on Naive Bayesian classifier	24,1	Ting and Zheng (1999)
Holland-style adaptive classifier	17,3	Frey and Slate (1991)
C4.5	13,8	Freund and Schapire (1996)
AdaBoost on C4.5 (100 machines)	3,3	Freund and Schapire (1996)
AdaBoost on C4.5 (1000 machines)	3,1	Schapire et al. (1997)
CART	12,4	Breiman (1996)
AdaBoost on CART (50 machines)	3,4	Breiman (1996)
16-70-50-26 MLP (500 online epochs)	6,2	Schwenk and Bengio (1998)
AdaBoost on 16-70-50-26 MLP (20 machines)	2,0	Schwenk and Bengio (1998)
AdaBoost on 16-70-50-26 MLP (100 machines)	1,5	Schwenk and Bengio (2000)
Nearest Neighbor	4,3	Fogarty (1992)

Table 2: A compilation of previously reported best error rates on the test set for the UCI Letters Recognition Database.

Figure 17 shows the average error reduction in respect to the number of online epochs, for the BLSUCH technique combination and various hidden layer sizes. As suggested in the database's documentation, we used the first 16000 samples for training and for measuring the training accuracy and the rest 4000 samples to measure the predictive accuracy. The solid and dashed curves show the test and training set errors respectively. Similarly to ensemble methods, we can observe two interesting phenomena which both seem to contradict the Occam's razor principle.

The first observation is that the test error stabilizes or continues to slightly decrease even after the training error has been zeroed. What is really happening is that the RMS error for the training set (which is related to the confidence of classification) continues to decrease even after the classification error has been zeroed, something that is also beneficiary for the test set's classification error.

The second observation is that increasing the network's capacity does not lead to over fitting. Although the training set error can be zeroed with just 125 hidden units, increasing the number of hidden units reduces the residual test error as well. We attribute this phenomenon to the conjecture that the hidden units' differentiation results in a smoother approximation (as suggested by Figure 5 and the related discussion).

Comparing our results with those in Table 2, we can also observe the following: The 16-125-26 MLP (5401 weights) reached a 4.6% misclassification error on average, which is 26% better than the 6.2% of the 16-70-50-26 MLP (6066 weights), despite the fact that it had fewer weights, a simpler

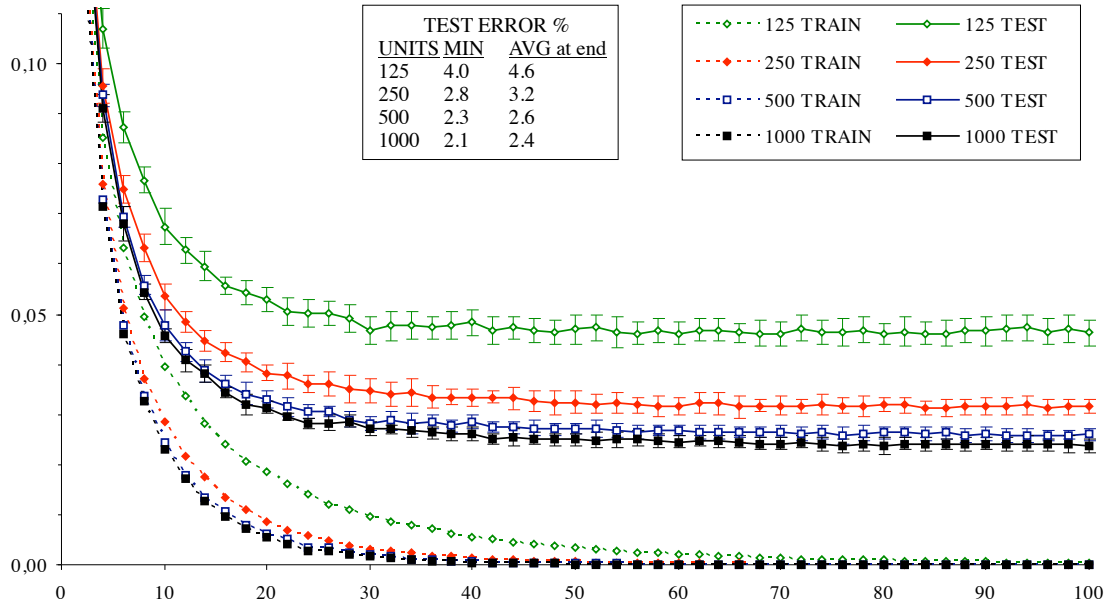


Figure 17: Average error reduction vs. number of online epochs for various hidden layer sizes, while training on the UCI Letters Recognition Database with the BLSUCH technique combination. The solid and dashed curves show the test and training set errors respectively. The standard deviations for the training set errors have been omitted for clarity. The embedded table contains the minimum observed errors across all trials and epochs, and the average errors across all trials at epoch 100.

architecture with one hidden layer only and it was trained for a far less number of online epochs. It is indicative that the asymptotic residual classification error on the test set was reached in about 30 online epochs.

The 16-1000-26 MLP (43026 weights) reached a 2.4% misclassification error on average, which is the third best published result following the AdaBoosted 16-70-50-26 MLPs with 20 and 100 machines (121320 and 606600 weights respectively). The lowest observed classification error was 2.1% and was reached in one of the 10 runs at the 80th epoch. It must be stressed that the above results were obtained without any optimization of the learning rate, without a learning rate annealing schedule and within a by far shorter training time.

All MLPs with 250 hidden units and above, gave results which put them at the top of the list of non-ensemble techniques and they even outperformed Adaboost on C4.5 with 100 machines.

Similarly to Figure 14, we also see that the convergence rates to the corresponding asymptotic errors on the test set are quite fast and about the same for all hidden layer sizes.

6. Discussion and Future Research

We have presented global and local selective attention techniques that can help neural network training to concentrate on the difficult parts of complex non-linear problems. A new hybrid activation function has also been presented that enables the hidden units to acquire individual receptive fields

in the input space. These individual receptive fields may be global or local depending on the problem's local complexities.

The success of the new activation function is due to the fact that it depends on two distances. The first is the weighted distance of a sample to the hidden unit's hyperplane. The second is the distance to the hidden unit's center. We need both distances and neither of them is sufficient. The first helps us discriminate and the second helps us localize.

The dynamic training set evolution algorithm locates the sub-areas of the input space where the problem resides. The fixed cascaded inhibitory connections and the selective training of a subset of the hidden units on each sample, force the hidden units to get differentiated and attack different subproblems. The individual centering of the hidden units at different points in the input space, adaptively conditions the network to the problem's local structures and enables each hidden unit to solve a well-conditioned subproblem. In coordination with the above, the hidden units' limited receptive fields allow training to follow a divide and conquer paradigm where each hidden unit only solves a local subproblem. The solutions to the subproblems are then combined by the output layer to give a solution to the original problem.

In the reported experiments we initialized the hidden weights and biases so that the hidden hyperplanes would cover the whole input space at random positions and orientations. The initial norm of the weights was also adjusted so that the net-input to each hidden unit would fall in the transition between the linear and non-linear range of the activation function. These specific initializations were necessary for standard backpropagation. On the contrary, we have found that the combined techniques are insensitive to the initial weights and biases, as long as their values are small. We have repeated the experiments with hidden biases set to zero and hidden weight norms set to 10^{-3} and the results where equivalent to those reported in Section 5. However, the choice of the best initial learning rate is still problem specific.

An additional and important characteristic of these techniques is that training of the hidden layer does not depend solely on gradient information. Gradient based techniques can only perform local optimization by locating a local minimum of the error function when the system is already at the basin of attraction of that minimum. Stochastic training has a potential of escaping from a shallow basin, but only when the basin is not very wide. Once there, the system cannot escape towards a different basin with a lower minimum. On the contrary, in our model some of the hidden layer's free parameters (the weights) are trained through gradient descent on the error, whereas some other (the means and standard deviations) are "trained" from the statistical properties of the back-propagated errors. Each hidden unit places its center near the center of mass of the error that it receives and limits its visibility only to the area of the input space where it produces a significant error. This model makes the hidden error landscape to constantly change. We conjecture that during training, paths connecting the various error basins are continuously emerging and vanishing. As a result the system can explore much more of the solution space. It is indicative that in all the reported experiments, all trials converged to a solution with more or less the same residual error irrespectively of the initial network state.

The combination of the presented techniques enables very fast training on complex classification problems with embedded subproblems. By focusing on the problem's details and efficiently utilizing the available resources, they make feasible the solution of very difficult problems (like the one in Figure 1e), provided that the adequate number of hidden units has been used. Although other machine learning techniques can do the same, to our knowledge this is the first report that this can be done using ordinary feed forward neural networks and backpropagation, in an online, adaptive

and memory-less scenario, where the input exemplars are unknown before training and discarded after being used.

In the following we discuss some areas that deserve further investigation.

6.1 Generalization and Regression

For the classes of problems that were investigated, we had noiseless exemplars and the whole input space at our disposal for training, so there was no danger of overfitting. Thus, we did not use any mechanism to assist generalization. This does not mean of course that the network just stored the input output mapping, as a lookup table would do. By putting constraints on the positions and orientations of the hidden unit hyperplanes and by limiting their receptive fields, we reduced the system's available degrees of freedom, and the network arranged its resources in a way to achieve the best possible input-output mapping approximation.

The experiments on the Letter Recognition Database showed remarkable generalization capabilities. However, when we train on noisy samples or when the number of training samples is small in respect to the size and complexity of the input space, we have the danger of overfitting. It remains to be examined how the described techniques are affected by methods that avoid overfitting, such as, training with jitter, error regularization, target smoothing and sigmoid gain attenuation (Reed et al., 1995). This consideration also applies to regression problems which usually require smoother approximations. Although early experiments give evidence that the presented techniques can be applied to regression problems as well, we feel that some smoothing technique must be included in the training framework.

6.2 Receptive Fields Limited Orientations

As it was noted in Section 3.4, the orientations of the receptive field ellipses are limited to have the direction of one of the input axes. This hinders training performance by not allowing the receptive fields to be adequately shrunk at the direction perpendicular to the hyperplane. In addition, hidden units with sloped hyperplanes are trained on highly correlated input values. These problems are expected to be exaggerated in high dimensional input spaces.

We would cure both of these problems simultaneously, if we could individually transform the input for each hidden unit through adaptive whitening, or, if we could present to each hidden unit a rotated view of the input space, such that, one of the axes to be perpendicular to the hyperplane and the rest to be parallel to the hyperplane. Unfortunately, both of the above transformations would require too many additional parameters. An approximation (for 2 dimensional problems) that we are currently investigating upon is the following:

For each input vector we compute K vectors rotated around the center of the input space with successive angle increments equal to $\pi/(2K)$. Our purpose is to obtain uniform rotations between 0 and $\pi/4$. Every a few hundred training steps, we reassign to each hidden unit the most appropriate input representation and adjust the affected parameters (weights, means and stdevs). The results are promising.

6.3 Dynamic Cascaded Inhibitory Connections

Regarding the fixed cascaded inhibitory connections, it must be examined whether it is better to make the strength of the connections, dynamic. Minus one is OK when the weights are small. How-

ever as the weights get larger, the inhibitory connections get less and less effective to differentiate the hidden units. We can try to make them relative to each hidden unit's average absolute net-input or alternatively to make them trainable. It has been observed that increasing the strength of these connections enables the hidden units to generate more curved discriminant functions, which is very beneficiary for some problems.

6.4 Miscellaneous

More experiments need to be done, in order to evaluate the effectiveness of the hybrid activation function on highly non-linear problems in many dimensions. High dimensional input spaces have a multitude of disturbing properties in regard to distance and density metrics, which may affect the hybrid activation in yet unknown ways.

Last, we must devise a training mechanism, that will be invariant to the initial learning rate and that will vary automatically the number of hidden units as each problem requires.

Acknowledgments

I would like to thank all participants in my threads in usenet comp.ai.neural-nets, for their fruitful comments on early presentations of the subjects in this report. Special thanks to Aleks Jakulin for his support and ideas on further research that can make these results even better and to Greg Heath for bringing to my attention the perturbated forms for the calculation of sliding window statistics. I also thank the area editor Léon Bottou and the anonymous reviewers for their valuable comments and for helping me to bring this report in shape for publication.

Appendix A. Notational Conventions

The following list contains the meanings of the symbols that have been used in this report. Symbols with subscripts are used either as scalars or as vectors and matrices when the subscripts are omitted. For example, w_{ji} is a single weight, $\vec{w}_j$ is a weight vector and W is a weight matrix.

α – A constant that determines the time scale of the exponential trace of the average training-set error within the dynamic training set evolution algorithm.

β – A constant that determines the time scale of the exponential trace of the input means and standard deviations.

δ – An accumulator for the efficient implementation of the fixed cascaded inhibitory connections.

η – The number of hidden units.

μ – The number of input units.

f – The hidden units' squashing function.

i – Index enumerating the input units.

j – Index enumerating the hidden units.

k – Index enumerating the output units.

a_j – The hidden unit's activation computed from the sample's weighted distance to the hidden unit's hyperplane.

b_j – The hidden unit's activation attenuation computed from the sample's distance to the hidden unit's center.

d_j – The sample's distance to the hidden unit's center.

e_j – The hidden unit's accumulated back propagated errors.

g_j – The hidden unit's error signal $\left(f'(n_j)\, e_j\right)$.

h_j – The hidden unit's activation.

m_{ji} – The mean of the values received by hidden unit j from input unit i.

n_j – The net-input to the hidden unit.

q_{ji} – The mean of the squared values received by hidden unit j from input unit i.

r_k – The error of output unit k.

s_{ji} – The standard deviation of the values received by hidden unit j from input unit i.

u_{jk} – The weight of the connection from hidden unit j to output unit k.

v_{ji} – The variance of the values received by hidden unit j from input unit i.

w_{ji} – The weight of the connection from hidden unit j to input unit i.

x_i – The value of input unit i.

z_{ji} – The normalized input value received by hidden unit j from input unit i. It is currently computed as the z-score of the input value. A better alternative would be to compute the vector $\vec{z}_j$ by multiplying the input vector $\vec{x}$ with a whitening matrix Z_j.

References

C. C. Aggarwal, A. Hinneburg, and D. A. Keim. On the surprising behavior of distance metrics in high dimensional spaces. In J. Van den Bussche and V. Vianu, editors, *Proceedings of the 8th International Conference on Database Theory (ICDT)*, volume 1973 of *Lecture Notes in Computer Science*, pages 420–434. Springer, 2001.

K. Agyepong and R. Kothari. Controlling hidden layer capacity through lateral connections. *Neural Computation*, 9(6):1381–1402, 1997.

S. Ahmad and S. Omohundro. A network for extracting the locations of point clusters using selective attention. In *Proceedings of the 12th Annual Conference of the Cognitive Science Society, MIT*, 1990.

L. B. Almeida, T. Langlois, and J. D. Amaral. On-line step size adaptation. Technical Report INESC RT07/97, INESC/IST, Rua Alves Redol 1000 Lisbon, Portugal, 1997.

S. Amari. Natural gradient works efficiently in learning. *Neural Computation*, 10(2):251–276, 1998.

P. Bakker. Don't care margins help backpropagation learn exceptions. In A. Adams and L. Sterling, editors, *Proceedings of the 5th Australian Joint Conference on Artificial Intelligence*, pages 139–144, 1992.

P. Bakker. Exception learning by backpropagation: A new error function. In P. Leong and M. Jabri, editors, *Proceedings of the 4th Australian Conference on Neural Networks*, pages 118–121, 1993.

S. Baluja and D. Pomerleau. Using the representation in a neural network's hidden layer for task-specific focus of attention. In *IJCAI*, pages 133–141, 1995.

L. Breiman. Bias, variance, and arcing classifiers. Technical Report 460, Statistics Department, University of California, 1996.

D. S. Broomhead and D. Lowe. Multivariate functional interpolation and adaptive networks. *Complex Systems*, 2(3):321–355, 1988.

W. Duch, K. Grudzinski, and G. H. F. Diercksen. Minimal distance neural methods. In *World Congress of Computational Intelligence*, pages 1299–1304, 1998.

D. L. Elliott. A better activation function for artificial neural networks. Technical Report TR 93-8, The Institute for Systems Research, University of Maryland, College Park, MD, 1993.

G. W. Flake. Square unit augmented, radially extended, multilayer perceptrons. In G. B. Orr and K. R. Müller, editors, *Neural Networks: Tricks of the Trade*, volume 1524 of *Lecture Notes in Computer Science*, pages 145–163. Springer, 1998.

T. C. Fogarty. Technical note: First nearest neighbor classification on frey and slate's letter recognition problem. *Machine Learning*, 9(4):387–388, 1992.

Y. Freund and R. E. Schapire. Experiments with a new boosting algorithm. In *ICML*, pages 148–156, 1996.

P. W. Frey and D. J. Slate. Letter recognition using holland-style adaptive classifiers. *Machine Learning*, 6:161–182, 1991.

S. Geman and D. Geman. Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6(6):721–741, 1984.

T. Graepel and N. N. Schraudolph. Stable adaptive momentum for rapid online learning in nonlinear systems. In J. R. Dorronsoro, editor, *Proceedings of the International Conference on Artificial Neural Networks (ICANN)*, volume 2415 of *Lecture Notes in Computer Science*, pages 450–455. Springer, 2002.

M. Harmon and L. Baird. Multi-player residual advantage learning with general function approximation. Technical Report WL-TR-1065, Wright Laboratory, Wright-Patterson Air Force Base, OH 45433-6543, 1996.

M. Hegland and V. Pestov. Additive models in high dimensions. *Computing Research Repository (CoRR)*, cs/9912020, 1999.

S. C. Huang and Y. F. Huang. Learning algorithms for perceptrons using back propagation with selective updates. *IEEE Control Systems Magazine*, pages 56–61, April 1990.

R.A. Jacobs. Increased rates of convergence through learning rate adaptation. *Neural Networks*, 1: 295–307, 1988.

R. Kothari and D. Ensley. Decision boundary and generalization performance of feed-forward networks with gaussian lateral connections. In S. K. Rogers, D. B. Fogel, J. C. Bezdek, and B. Bosacchi, editors, *Applications and Science of Computational Intelligence, SPIE Proceedings*, volume 3390, pages 314–321, 1998.

B. Laheld and J. F. Cardoso. Adaptive source separation with uniform performance. In *Proc. EUSIPCO*, pages 183–186, September 1994.

Y. LeCun, P. Simard, and B. Pearlmutter. Automatic learning rate maximization by on-line estimation of the hessian's eigenvectors. In S. Hanson, J. Cowan, and L. Giles, editors, *Advances in Neural Information Processing Systems*, volume 5, pages 156–163. Morgan Kaufmann Publishers, San Mateo, CA, 1993.

Y. LeCun, L. Bottou, G. B. Orr, and K.-R. Mueller. Efficient backprop. In G. B. Orr and K.-R. Müller, editors, *Neural Networks: Tricks of the Trade*, volume 1524 of *Lecture Notes in Computer Science*, pages 9–50. Springer, 1998.

T. K. Leen and G. B. Orr. Optimal stochastic search and adaptive momentum. In J. D. Cowan, G. Tesauro, and J. Alspector, editors, *Proceedings of the 7th NIPS Conference (NIPS), Advances in Neural Information Processing Systems 6*, pages 477–484. Morgan Kaufmann, 1993.

P. W. Munro. A dual back-propagation scheme for scalar reinforcement learning. In *Proceedings of the 9th Annual Conference of the Cognitive Science Society, Seattle, WA*, pages 165–176, 1987.

N. Murata, K. Müller, A. Ziehe, and S. Amari. Adaptive on-line learning in changing environments. In M. Mozer, M. I. Jordan, and T. Petsche, editors, *Advances in Neural Information Processing Systems 9 (NIPS)*, pages 599–605. MIT Press, 1996.

D. J. Newman, S. Hettich, C.L. Blake, and C.J. Merz. UCI repository of machine learning databases, 1998.

G. B. Orr and T. K. Leen. Using curvature information for fast stochastic search. In M. Mozer, M. I. Jordan, and T. Petsche, editors, *Advances in Neural Information Processing Systems 9 (NIPS)*, pages 606–612. MIT Press, 1996.

J. L. Phillips and D. C. Noelle. Reinforcement learning of dimensional attention for categorization. In *Proceedings of the 26th Annual Meeting of the Cognitive Science Society*, 2004.

M. Plumbley. A hebbian/anti-hebbian network which optimizes information capacity by orthonormalizing the principal subspace. In *Proc. IEE Conf. on Artificial Neural Networks, Brighton, UK*, pages 86–90, 1993.

R. Reed, R.J. Marks, and S. Oh. Similarities of error regularization, sigmoid gain scaling, target smoothing, and training with jitter. *IEEE Transactions on Neural Networks*, 6(3):529–538, 1995.

R. E. Schapire. A brief introduction to boosting. In T. Dean, editor, *Proceedings of the 16th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1401–1406. Morgan Kaufmann, 1999.

R. E. Schapire, Y. Freund, P. Barlett, and W. S. Lee. Boosting the margin: A new explanation for the effectiveness of voting methods. In D. H. Fisher, editor, *Proceedings of the 14th International Conference on Machine Learning (ICML)*, pages 322–330. Morgan Kaufmann, 1997.

N. N. Schraudolph. Fast curvature matrix-vector products for second-order gradient descent. *Neural Computation*, 14(7):1723–1738, 2002.

N. N. Schraudolph. Centering neural network gradient factors. In G. B. Orr and K. R. Müller, editors, *Neural Networks: Tricks of the Trade*, volume 1524 of *Lecture Notes in Computer Science*, pages 207–226. Springer, 1998a.

N. N. Schraudolph. Accelerated gradient descent by factor-centering decomposition. Technical Report IDSIA-33-98, Istituto Dalle Molle di Studi sull'Intelligenza Artificiale, 1998b.

N. N. Schraudolph. Online local gain adaptation for multi-layer perceptrons. Technical Report IDSIA-09-98, Istituto Dalle Molle di Studi sull'Intelligenza Artificiale, Galleria 2, CH-6928 Manno, Switzerland, 1998c.

N. N. Schraudolph. Local gain adaptation in stochastic gradient descent. In *ICANN*, pages 569–574. IEE, London, 1999.

H. Schwenk and Y. Bengio. Boosting neural networks. *Neural Computation*, 12(8):1869–1887, 2000.

H. Schwenk and Y. Bengio. Training methods for adaptive boosting of neural networks for character recognition. In M. Jordan, M. Kearns, and S. Solla, editors, *Advances in Neural Information Processing Systems 10*. MIT Press, Cambridge, MA, 1998.

M. W. Spratling and M. H. Johnson. Neural coding strategies and mechanisms of competition. *Cognitive Systems Research*, 5(2):93–117, 2004.

C. Thornton. The howl effect in dynamic-network learning. In *Proceedings of the International Conference on Artificial Neural Networks*, pages 211–214, 1992.

K. M. Ting and Z. Zheng. Improving the performance of boosting for naive bayesian classification. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 296–305, 1999.

Y. H. Yu and R. F. Simmons. Descending epsilon in back-propagation: A technique for better generalization. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN)*, volume 3, pages 167–172, 1990.

S. Zhong and J. Ghosh. Decision boundary focused neural network classifier. In *Intelligent Engineering Systems Through Artificial Neural Networks (ANNIE)*. ASME Press, 2000.

Journal of Machine Learning Research 8 (2007) 2047-2081 Submitted 9/05; Revised 11/06; Published 9/07

Unlabeled Compression Schemes for Maximum Classes*

Dima Kuzmin DIMA@CSE.UCSC.EDU
Manfred K. Warmuth MANFRED@CSE.UCSC.EDU
Computer Science Department
University of California, Santa Cruz
1156 High Street
Santa Cruz, CA, 95064

Editor: John Shawe-Taylor

Abstract

Maximum concept classes of VC dimension d over n domain points have size $\binom{n}{\leq d}$, and this is an upper bound on the size of any concept class of VC dimension d over n points. We give a compression scheme for any maximum class that represents each concept by a subset of up to d unlabeled domain points and has the property that for any sample of a concept in the class, the representative of exactly one of the concepts consistent with the sample is a subset of the domain of the sample. This allows us to compress any sample of a concept in the class to a subset of up to d unlabeled sample points such that this subset represents a concept consistent with the entire original sample. Unlike the previously known compression scheme for maximum classes (Floyd and Warmuth, 1995) which compresses to labeled subsets of the sample of size equal d, our new scheme is tight in the sense that the number of possible unlabeled compression sets of size at most d equals the number of concepts in the class.

Keywords: compression schemes, VC dimension, maximum classes, one-inclusion graph

1. Introduction

Consider the following type of protocol between a learner and a teacher. Both agree on a domain and a class of concepts (subsets of the domain). For instance, the domain could be the plane and a concept the subset defined by an axis-parallel rectangle (see Figure 1). The teacher gives a set of training examples (labeled domain points) to the learner. The labels of this set are consistent with a concept (rectangle) that is hidden from the learner. The learner's task is to predict the label of the hidden concept on a new test point.

Intuitively, if the training and test points are drawn from some fixed distribution, then the labels of the test point can be predicted accurately provided the number of training examples is large enough. The sample size should grow with the inverse of the desired accuracy and with the complexity or "dimension" of the concept class. The most basic notion of dimension in this context is the Vapnik-Chervonenkis dimension. This dimension is the size d of the maximum cardinality subset of domain points such that all 2^d labeling patterns can be realized by a concept in the class. The VC dimension of axis-parallel rectangles is 4, since it is possible to label any set of 4 points in all possible ways as long as no point lies inside the *orthogonal hull* of the other 3 points (where

*. Supported by NSF grant CCR CCR 9821087. Some work on this paper was done while the authors were visiting National ICT Australia.

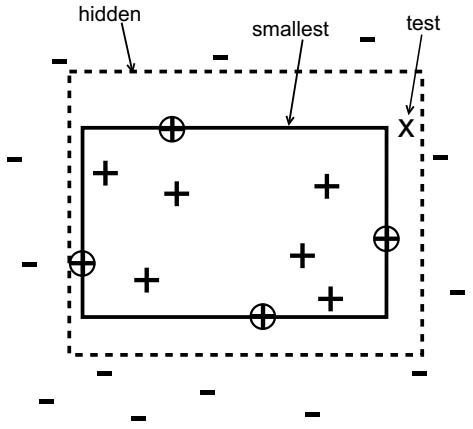


Figure 1: An example set consistent with some axis-parallel rectangle. Also shown is the smallest axis-parallel rectangle containing the subsample of circled points. This rectangle is consistent with all examples and the set of circled points represents a consistent concept. The hidden rectangle generating the data is dashed. "x" is the next test point.

the orthogonal hull is defined as the smallest rectangle containing the points); also for any 5 points, at least one of the points lies inside the orthogonal hull containing the remaining 4 points and this disallows at least one of the 2^5 patterns.

This paper deals with an alternate notion of dimension used in machine learning that is related to compression (Littlestone and Warmuth, 1986). It stems from the observation that you can often select a subset of the training examples to represent a hypothesis consistent with all training examples. For instance, in the case of rectangles, it is sufficient to keep only the uppermost, lowermost, leftmost and rightmost positive point. There are up to 4 points in the subsample (since some of the 4 extreme points might coincide.) The orthogonal hull of the subsample will always be consistent with the entire sample.

More generally, a compression scheme is defined by two mappings (Littlestone and Warmuth, 1986): one mapping samples of concepts in the class to subsamples, and the other one mapping the resulting subsamples to hypotheses on the domain of the class. Compression schemes must have the property that the subsample always represents a hypothesis consistent with the entire original sample. However note that the reconstructed hypothesis doesn't have to lie in the original concept class. It only needs to be consistent with the original sample. The subsamples represent hypotheses and are called *representatives* in this paper. A compression scheme can be viewed as a set of representatives of hypotheses with the property that every sample of the class contains a representative of a consistent hypothesis. The size of the compression scheme is the size of the largest representative, and the minimum size of a compression scheme for a class serves as an alternate measure of complexity.

Note that in the case of rectangles we need to keep at most 4 points and 4 also is the Vapnik-Chervonenkis dimension of that class. One of the most tantalizing conjectures in learning theory is the following (Floyd and Warmuth, 1995; Warmuth, 2003): *For any concept class of VC dimension d, there is a compression scheme of size at most d.*

The size of the compression scheme also replaces the VC dimension in the PAC sample size bounds (Littlestone and Warmuth, 1986; Floyd and Warmuth, 1995; Langford, 2005). However, in the case of compression schemes, the proofs of these bounds are much simpler. There are many practical algorithms based on compression schemes (e.g., Marchand and Shawe-Taylor, 2002, 2003). Also, any algorithm with a mistake bound M leads to a compression scheme of size M (Floyd and Warmuth, 1995).

Let's consider some more illustrative examples of compression schemes. Unions of up to k intervals on the real line form a concept class of VC dimension $2k$. We can compress a sample from this class to the following set of points: the leftmost "$+$" point in the sample, the leftmost "$-$" point to the right of the last selected point, the leftmost "$+$" further to the right of the last selected point, and so forth; stop when there are no more points whose label is opposite to the last selected point. It is easy to see that at most $2k$ points are kept when the original sample is consistent with a union of k intervals. Also the labels of the entire original sample can be reconstructed from this subsample. Note that in this case the labels of the subsample are always alternating starting with a "$+$". Thus these labels are redundant and the above scheme can be interpreted as compressing to unlabeled subsamples of size at most the VC dimension $2k$.

Support Vector Machines also lead to a simple labeled compression scheme for halfspaces (sets of the form $\{\boldsymbol{x} \in \mathbb{R}^n : \boldsymbol{w} \cdot \boldsymbol{x} \geq b\}$), because only the set of support vectors is needed to reconstruct the hyperplane consistent with the original sample. Of course, the number of support vectors can be quite big. However, it suffices to keep any set of *essential* support vectors and these sets have size $n+1$, where n is the dimension of the feature space (von Luxburg et al., 2004). Not surprisingly, $n+1$ is also the VC dimension of arbitrary halfspaces of dimension n. However, the labels of a set of essential support vectors are not redundant and this provides an example of a labeled compression scheme for halfspaces. There also exists a compression scheme for the same class that compresses to at most $n+1$ *unlabeled points* (Ben-David and Litman, 1998). However this scheme is not constructive.

The compression scheme conjecture is easily proven for intersection-closed concept classes (Helmbold et al., 1992), which include the class of axis-parallel rectangles as a special case. More importantly, the conjecture was shown to be true for *maximum classes*. A finite class of VC dimension d over n domain points is maximum if its size is equal to $\binom{n}{\leq d}$, which is the upper bound on the size of any concept class of VC dimension d over n points. An infinite class of VC dimension d is maximum if all restrictions to a finite subset of domain of size at least d are maximum classes of dimension d.

Of the example concept classes discussed so far, the class of up to k intervals on the real line is maximum. The class of halfspaces in $\mathbb{R}^n$ is not maximum, but it is in fact a union of two classes of VC dimension n which are "almost maximum": positive halfspaces and negative halfspaces (Floyd, 1989). *Positive* halfspaces are those that contain the "point" $(\infty, 0, \ldots, 0)$ and negative halfspaces are those that contain $(-\infty, 0, \ldots, 0)$. Both classes of halfspaces are almost maximum in the sense that the restriction to any set of points in general position always produces a maximum class. Finally, the class of axis-parallel rectangles is not maximum since for any five points at least two labelings are not realizable.

In Floyd and Warmuth (1995) it was shown that for all maximum classes there always exist compression schemes that compress to *exactly d labeled examples*. In this paper, we give an alternate compression scheme for finite maximum classes. Even though we do not resolve the conjecture for arbitrary classes, we have uncovered a great deal of new combinatorics. Our new scheme com-

	x_1	x_2	$\mathbf{x_3}$	$\mathbf{x_4}$	$r(c)$
c_1	0	0	0	0	$\emptyset$
$\mathbf{c_2}$	**0**	**0**	**1**	**0**	$\{\mathbf{x_3}\}$
c_3	0	0	1	1	$\{x_4\}$
c_4	0	1	0	0	$\{x_2\}$
c_5	0	1	0	1	$\{x_3,x_4\}$
c_6	0	1	1	0	$\{x_2,x_3\}$
c_7	0	1	1	1	$\{x_2,x_4\}$
c_8	1	0	0	0	$\{x_1\}$
c_9	1	0	1	0	$\{x_1,x_3\}$
c_{10}	1	0	1	1	$\{x_1,x_4\}$
c_{11}	1	1	0	0	$\{x_1,x_2\}$

Figure 2: Illustration of the unlabeled compression scheme for some maximum concept class. The representatives for each concept are indicated in the right column and also as the underlined positions in each row. Suppose the sample is $\mathbf{x_3}=\mathbf{1},\mathbf{x_4}=\mathbf{0}$. The set of concepts consistent with that sample is $\{c_2,c_6,c_9\}$. The representative of exactly one of these concepts is entirely contained in the sample domain $\{\mathbf{x_3},\mathbf{x_4}\}$. For our sample this representative is $\{\mathbf{x_3}\}$ which represents $\mathbf{c_2}$. So the compressed sample becomes $\{\mathbf{x_3}\}$.

presses any sample consistent with a concept to *at most d unlabeled points* from the sample. If m is the size of the sample, then there are $\binom{m}{\leq d}$ sets of points of size up to d. For maximum classes, the number of different labelings induced on any set of size m is also $\binom{m}{\leq d}$. Thus our new scheme is "tight". In the previous labeled scheme, the number of possible representatives was much bigger than the number of concepts.

The new unlabeled scheme also has many interesting combinatorial properties. Let us represent finite classes as a binary table (see Figure 2) where the rows are concepts and the columns are all the points in the domain. Our compression scheme represents concepts by subsets of size at most d and for any $k \leq d$, the concepts represented by subsets of size up to k will form a maximum class of VC dimension k. Our scheme compresses as follows: After receiving a set of examples, we first restrict ourselves to concepts that are consistent with the sample. We then compress to a representative of a consistent concept that is completely contained in the sample domain (see Figure 2). As our main result we will prove that for our choice of representatives, for any sample there always will be exactly one of the consistent concepts whose representative is completely contained in the sample domain.

Our new unlabeled compression scheme is connected to a certain undirected graph, called the *one-inclusion graph*, that characterizes the concept class on a set of example points (Haussler et al., 1994): the vertices are the possible labelings of the example points and there is an edge between two concepts if they disagree on a single point. The edges are naturally labeled by the differing points (see Figure 4).

Any prediction algorithm can be used to *orient* the edges of the one-inclusion graphs as follows. Assume we are given a labeling of some m points $x_1,\ldots,x_m$ and an unlabeled test point x. If there is still an ambiguity as to how x should be labeled, then this corresponds to an x-labeled edge in

	x_1	x_2	x_3	x_4
c_1	0	0	1	0
c_2	0	1	0	0
c_3	0	1	1	0
c_4	1	0	1	0
c_5	1	1	0	0
c_6	1	1	1	0
c_7	0	0	1	1
c_8	0	1	0	1
c_9	1	0	0	0
c_{10}	1	0	0	1

,

Figure 3: A maximal class of VCdim 2 with 10 concepts. Maximum concept classes of VCdim 2 have $\binom{4}{\leq 2} = 11$ concepts (see Figure 2).

one-inclusion graph for the set $\{x_1,\ldots,x_m,x\}$. This edge connects the two possible extensions of the labeling of $x_1,\ldots,x_m$ to the test point x. If the algorithm predicts b, then orient the edge toward the concept that labels x with bit b.

The vertices in the one-inclusion graph represent the possible labelings of $\{x_1,\ldots,x_m,x\}$ produced by the target concepts and if the prediction is averaged over all permutations of the $m+1$ points, then the probability of predicting wrong is $\frac{d}{m+1}$, where d is the out-degree of the target. Therefore the canonical optimal algorithm predicts with an orientation of the one-inclusion graphs that minimizes the maximum out-degree (Haussler et al., 1994; Li et al., 2002) and in Haussler et al. (1994) it was shown that this outdegree is at most the VC dimension.

How is this all related to our new compression scheme for maximum classes? We show that for any edge labeled with x, exactly one of the two representatives of the incident concepts contains the point x. Thus by orienting the edges toward concept that does not have x, we immediately obtain an orientation of the one-inclusion graphs in which all vertices have maximum outdegree at most d (which is the best possible). Again such a d-orientation immediately leads to prediction algorithms with a worst case expected mistake bound of $\frac{d}{m+1}$, where m is the sample size (Haussler et al., 1994), and this bound is optimal[1] (Li et al., 2002).

The conjecture whether there always exists a compression scheme of size at most the VC dimension remains open. For finite domains it clearly suffices to resolve the conjecture for maximal classes (i.e., classes where adding any concept would increase the VC dimension). We do not know of any natural example of a maximal concept class that is not maximum or closely related. However, it is easy to find small artificial maximal classes (see Figure 3). We believe that much of the new methodology developed in this paper for maximum classes will be useful in deciding the general conjecture in the positive and think that in this paper we made considerable progress toward this goal. In particular, we developed a refined recursive structure of finite concept classes and made the connection to orientations of the one-inclusion graphs. Also, our scheme constructs a certain unique matching that is interesting in its own right.

1. Predicting with a d-orientation of the one-inclusion graphs is also conjectured to lead to optimal algorithms in the PAC model of learning (Warmuth, 2004).

Even though the unlabeled compression schemes for maximum classes are tight in some sense, they are not unique. There is a strikingly simple "peeling algorithm" that always seems to produce a valid unlabeled compression scheme for maximum classes: construct the one-inclusion graph for the domain; iteratively peel off a lowest degree vertex and represent a concept c by the set of data points incident to vertex c in the remaining graph when c was removed from the graph (see Figure 7 for an example run). However, we have no proof of correctness of this algorithm and the resulting schemes do not have as much recursive structure as the ones produced by our recursive algorithm for which we have correctness proof. For the small example given in Figure 2, both algorithms can produce the same scheme.

1.1 Outline of the Paper

Some basic definitions are provided in Section 2. We then define unlabeled compression schemes in Section 3 and characterize the properties of the representation mappings of such schemes and their relation to the one-inclusion graph. In this section we also discuss the simple Min-Peeling Algorithm in more detail. This algorithm always seems to provide an unlabeled compression scheme even though we currently do not have a correctness proof for it. Section 4 discusses linear arrangements, which are special maximum concept classes, and discuss how to interpret unlabeled compression schemes for these classes. Next, in Section 5, we briefly summarize the old scheme for maximum classes from Floyd and Warmuth (1995) which compresses to labeled subsamples, whereas ours uses unlabeled ones. The core of the paper is in Section 6, where we give a recursive algorithm for constructing an unlabeled compression scheme with a detailed proof of correctness. Section 7 contains additional combinatorial lemmas about the structure of maximum classes and unlabeled compression schemes. In Section 8, we discuss how to possibly extend various compression schemes for maximum classes to the more general case of maximal classes. We conclude in Section 9 with a large number of combinatorial open problems that we have encountered in this research.

2. Definitions

Let X be a domain, where we allow $X = \emptyset$. A concept c is a mapping from X to $\{0,1\}$. We can also view a concept c as a characteristic function of a subset of $\mathrm{dom}(c)$, that is, for any domain point $x \in \mathrm{dom}(c)$, $c(x) = 1$ iff $x \in c$. A concept class C is a set of concepts with the same domain (denoted as $\mathrm{dom}(C)$). Such a class is represented by a binary table (see Figure 2), where the rows correspond to concepts and the columns to points in $\mathrm{dom}(C)$.

Alternatively, C can be represented as a subgraph of the Boolean hypercube of dimension $|\mathrm{dom}(C)|$. Each dimension corresponds to a particular domain point, the vertices are the concepts in C and two concepts are connected with an edge if they disagree on the label of a single point. This graph is called the *one-inclusion graph* of C (Haussler et al., 1994). Note that each edge is naturally labeled by the single dimension/point on which the incident concepts disagree (see Figure 4). The *set of incident dimensions* of a vertex c in a one-inclusion graph G is the set of dimensions labeling the edges incident to c. We denote this set as $I_G(c)$. Its size equals the degree of c in G.

We denote the *restriction* of a concept c onto $A \subseteq \mathrm{dom}(c)$ as $c|A$. This concept has domain A and labels that domain consistently with c. The restriction of an entire class is denoted as $C|A$. This restriction is produced by simply removing all columns not in A from the table for C and collapsing

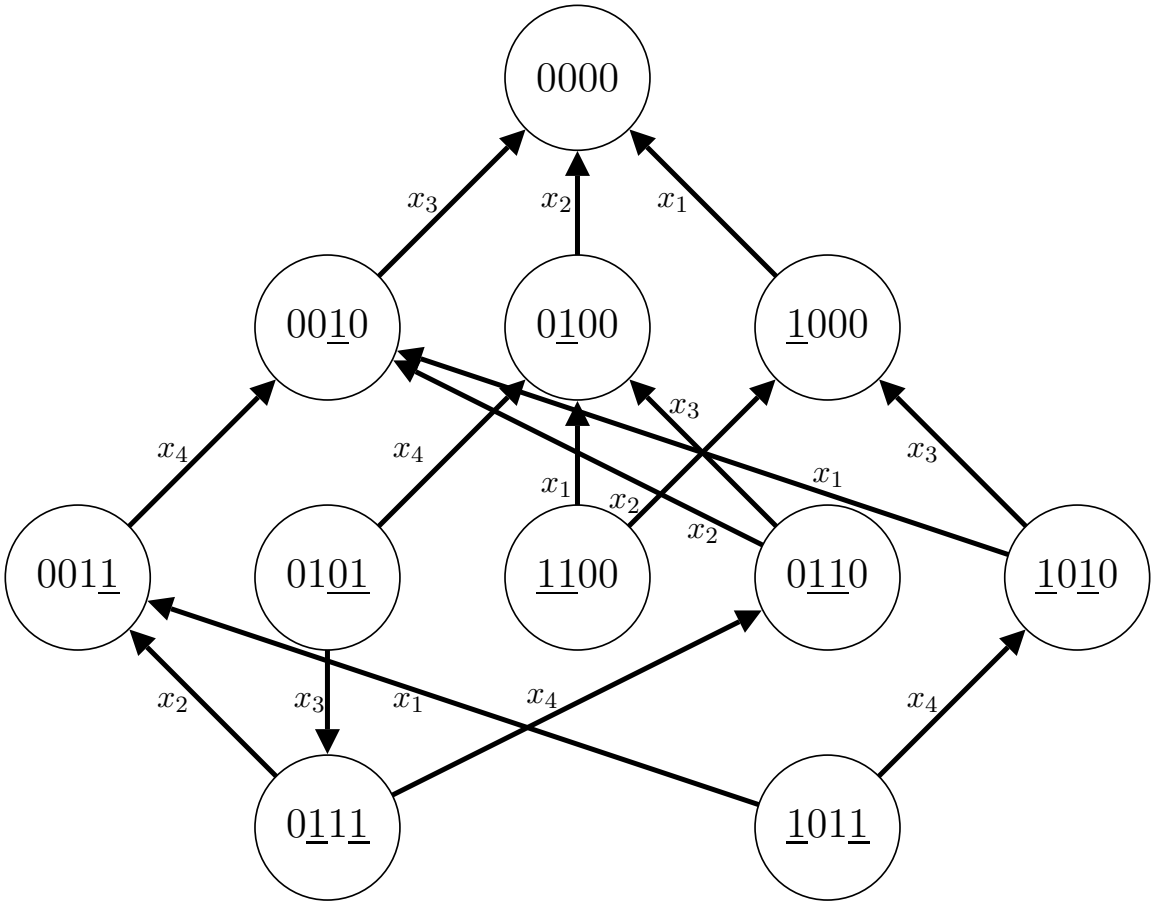


Figure 4: One-inclusion graph for the concept class from Figure 2. The concepts are the vertices of the graph and edges are labeled with the single differing dimension. Each concept c is given as a bit pattern and the set of underlined dimensions indicates its representative $r(c)$. Arrows show the d-orientation derived from the compression scheme.

x_2	x_3	x_4
0	0	0
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1

$C - x_1$

x_2	x_3	x_4
0	0	0
0	1	0
0	1	1
1	0	0

C^{x_1}

x_1	x_2	x_3	x_4
0	1	0	1
0	1	1	0
0	1	1	1

$\mathrm{tail}_{x_1}(C)$

Figure 5: The reduction, restriction and the tail of the concept class from Figure 2 wrt x_1.

identical rows.[2] Also the one-inclusion graph for the restriction $C|A$ is now a subgraph of the Boolean hypercube of dimension $|A|$ instead of the full dimension $|C|$. We use $c - x$ as shorthand for $c|(\mathrm{dom}(C) \setminus \{x\})$ and let $C - x$ denote $\{c - x | c \in C\}$ (produced by removing column x from the table, see Figure 5). A *sample* of a concept c is any restriction $c|A$ for some $A \subseteq \mathrm{dom}(c)$.

The *reduction* C^x of a concept class C wrt a dimension $x \in \mathrm{dom}(C)$ consists of all those concepts in $C - x$ that have two possible extensions onto concepts in C. All such concepts correspond to an edge labeled with x in the one-inclusion graph (see Figure 5). In summary, the class C^x is a subset of $C - x$ that has the same reduced domain $X - \{x\}$.

The *tail* of concept class C on dimension x consists of all concepts that do not have an edge labeled with x. Thus it corresponds to the subset of $C - x$ that has a unique extension onto the full domain. We denote the tail of C on dimension x as $\mathrm{tail}_x(C)$. The class C can therefore be partitioned as $0C^x \dot\cup 1C^x \dot\cup \mathrm{tail}_x(C)$, where $\dot\cup$ denotes the disjoint union and bC^x consists of all concepts in C^x extended with bit b in dimension x. Note that tails have the same domain as the original class, whereas the reduction and restriction are classes that have a reduced domain.

A finite set of dimensions $A \subseteq \mathrm{dom}(C)$ is *shattered* by a concept class C if for any possible labeling of A, the class C contains a concept consistent with that labeling (i.e., $\mathrm{size}(C|A) = 2^{|A|}$).[3] The *Vapnik-Chervonenkis dimension* of a concept class C is the size of a maximum subset that is shattered by that class (Vapnik, 1982). We denote this dimension as $\mathrm{VCdim}(C)$. Note that if $|C| = 1$, then $\mathrm{VCdim}(C) = 0$.[4]

In this paper we use the binomial coefficients $\binom{n}{d}$, for integers $n \geq 0$ and d, where $\binom{n}{d} = 0$ for $d > n$ or $d < 0$ and $\binom{0}{0} = 1$. We make use of the following identity which holds for $n > 0$: $\binom{n}{d} = \binom{n-1}{d} + \binom{n-1}{d-1}$. Let $\binom{n}{\leq d}$ be a shorthand for the binomial sums $\sum_{i=0}^{d} \binom{n}{i}$. Then we have a similar identity for the binomial sums when $n > 0$: $\binom{n}{\leq d} = \binom{n-1}{\leq d} + \binom{n-1}{\leq d-1}$.

2. We define $c|\emptyset = \emptyset$. Note that $C|\emptyset = \{\emptyset\}$ if $C \neq \emptyset$ and $\emptyset$ otherwise.

3. The notations $\mathrm{size}(A)$ and $|A|$ both denote the number of elements in set A.

4. $\mathrm{VCdim}(\{\emptyset\}) = 0$ and $\mathrm{VCdim}(\emptyset)$ is defined to be -1.

From Vapnik and Chervonenkis (1971) and Sauer (1972) we know that for all concept classes with VC dimension d: $|C| \leq \binom{|\mathrm{dom}(C)|}{\leq d}$ (generally known as Sauer's lemma). A concept class C with $\mathrm{VCdim}(C) = d$ is called *maximum* (Welzl, 1987) if for all finite subsets Y of the domain $\mathrm{dom}(C)$, $\mathrm{size}(C|Y) = \binom{|Y|}{\leq d}$. If C is a maximum class with $d = \mathrm{VCdim}(C)$, then $\forall x \in \mathrm{dom}(C)$, the classes $C - x$ and C^x are also maximum classes and have VC dimensions d and $d-1$, respectively (Welzl, 1987). From this it follows that for finite domains, a concept class C is maximum iff $\mathrm{size}(C) = \binom{|\mathrm{dom}(C)|}{\leq d}$.

A concept class C is called *maximal* if adding any other concept to C will increase its VC dimension. Any maximum class on a finite domain is also maximal (Welzl, 1987). However, there exist finite maximal classes, which are not maximum (see Figure 3 for an example).

From now on we only consider finite classes. As our main result we construct an unlabeled compression scheme for any finite maximum class. The existence of an unlabeled scheme for infinite maximum classes then follows from a compactness theorem given in Ben-David and Litman (1998). The proof of that theorem is, however, non-constructive.

3. Unlabeled Compression Scheme

Our unlabeled compression scheme for maximum classes represents the concepts as *unlabeled* subsets of $\mathrm{dom}(C)$ of size at most d. For any $c \in C$ we call $r(c)$ its *representative*. Intuitively we want concepts to disagree on their representatives. We say that two different concepts *clash* wrt r if $c|(r(c) \cup r(c')) = c'|(r(c) \cup r(c'))$.

Main definition: A *representation mapping* r of a maximum concept class C must have the following two properties:

1. r is a bijection between C and (unlabeled) subsets of $\mathrm{dom}(C)$ of size at most $\mathrm{VCdim}(C)$ and
2. no two concepts in C clash wrt r.

The following lemma shows how the non-clashing requirement can be used to find a unique representative for each sample.

Lemma 1 *Let r be any bijection between a finite maximum concept class C of VC dimension d and subsets of $\mathrm{dom}(C)$ of size at most d. Then the following two statements are equivalent:*

1. *No two concepts clash wrt r.*
2. *For all samples s of a concept from C, there is exactly one concept $c \in C$ that is consistent with s and $r(c) \subseteq \mathrm{dom}(s)$.*

Based on this lemma it is easy to see that a representation mapping r for a maximum concept class C defines a compression scheme as follows. For any sample s of C we *compress* s to the unique representative $r(c)$ such that c is consistent with s and $r(c) \subseteq \mathrm{dom}(s)$. Reconstruction is even simpler, since r is bijective: if s is compressed to the set $r(c)$, then we reconstruct to the concept c. See Figure 2 for an example of how compression and reconstruction work.

Proof of Lemma 1

$2 \Rightarrow 1$: Proof by contrapositive. Assume $\neg 1$, that is, there $\exists c, c' \in C, c \neq c'$ s.t. $c|r(c) \cup r(c') = c'|r(c) \cup r(c')$. Then let $s = c|r(c) \cup r(c')$. Clearly both c and c' are consistent with s and $r(c), r(c') \subseteq \mathrm{dom}(s)$. This negates 2.

$1 \Rightarrow 2$: At a high level, for any sample domain $\mathrm{dom}(s)$ there are as many representatives $r(c) \subseteq \mathrm{dom}(s)$ as there are different samples having that domain. The no clashing condition implies that all concepts with representatives in $\mathrm{dom}(s)$ are different from each other on $\mathrm{dom}(s)$, thus every sample has to get at least one representative.

For a more detailed proof assume $\neg 2$, that is, there is a sample s for which there are either zero or (at least) two consistent concepts c for which $r(c) \subseteq \mathrm{dom}(s)$. If two concepts $c, c' \in C$ are consistent with s and $r(c), r(c') \subseteq \mathrm{dom}(s)$, then $c|r(c) \cup r(c') = c'|r(c) \cup r(c')$ (which is $\neg 1$). If there is no concept c consistent with s for which $r(c) \subseteq \mathrm{dom}(s)$, then since

$$\mathrm{size}(C|\mathrm{dom}(s)) = \binom{|\mathrm{dom}(s)|}{\leq d} = |\{c : r(c) \subseteq \mathrm{dom}(s)\}| \ .$$

there must be another sample s' with $\mathrm{dom}(s') = \mathrm{dom}(s)$ for which there are two such concepts. So again $\neg 1$ is implied. □

Once we have a valid representation mapping for some maximum concept class C, we can easily derive a valid mapping for any restriction of the class $C|A$ by compressing every restricted concept. This is discussed in the following corollary.

Corollary 2 *For any maximum class C and $A \subseteq \mathrm{dom}(C)$, if r is a representation mapping for C then a representation mapping for $C|A$ can be constructed as follows. For any $c \in C|A$, let $r_A(c)$ be the representative of the unique concept $c' \in C$, such that $c'|A = c$ and $r(c') \subseteq A$.*

Proof The construction of the mapping for $C|A$ essentially tells us to treat the concept c as a sample from C and to compress it. Thus we can apply Lemma 1 to see that $r_A(c) \subseteq A$ is always uniquely defined. Now we need to show that r_A satisfies the conditions of the Main Definition. Since the representatives $r_A(c)$ are subsets of A, the non-clashing property for the representation mapping r_A for $C|A$ follows from the non-clashing condition for r for C. The bijection property follows from a counting argument like the one used in the proof of Lemma 1, since $\mathrm{size}(C|A) = \mathrm{size}(\{r(c) \text{ s.t. } r(c) \subseteq A\})$.■

The following lemmas and corollaries will be stated only for the concept class C itself. However, in light of Corollary 2 they will also hold for all restrictions $C|A$.

We first show that a representation mapping r for a maximum classes can be used to derive a d-orientation for the one-inclusion graph of the class (i.e., an orientation of the edges such that the outdegree of every vertex is at most d). As discussed in the introduction such orientations lead to a prediction algorithm with a worst-case expected mistake bound of $\frac{d}{t}$ at trial t.

Lemma 3 *For any representation mapping r of a maximum concept class C and the one-inclusion graph of C, any edge $c \stackrel{x}{-} c'$ in the graph has the property that its associated dimension x lies in exactly one of the representatives $r(c)$ or $r(c')$.*

Proof Since c and c' differ only on dimension x and $c|r(c) \cup r(c') \neq c'|r(c) \cup r(c')$, x lies in at least one of $r(c), r(c')$. Next we will show that x lies in exactly one.

We say an edge *charges* its incident concept if the dimension of the edge lies in the representative of this concept. Every edge charges at least one of its incident concepts and each concept c can receive at most $|r(c)|$ charges. So the number of charges is lower bounded by the number of edges

and upper bounded by the total size of all representations. We complete the proof of the lemma by showing that the number of edges equals the total size of all representatives. This means that no edge can charge both of its incident concepts and each point labeling an edge must lie in exactly one of the representations of its incident concepts.

There are $|C^x|$ edges labeled with dimension x in the one-inclusion graph for C. Since there are n dimensions and C^x is always maximum and of dimension $d-1$, the total number of edges in the graph is $n\binom{n-1}{\leq d-1}$, where $n = |\text{dom}(C)|$, $d = \text{VCdim}(C)$. (This formula is also a special case of Lemma 15.) The total size of all representatives is the same number because:

$$\sum_{c \in C} |r(c)| = \sum_{i=0}^{d} i\binom{n}{i} = n\sum_{i=1}^{d}\binom{n-1}{i-1} = n\binom{n-1}{\leq d-1} \ .$$

■

The above lemma lets us orient the one-inclusion graphs for the class.

Corollary 4 *For any representation mapping of maximum class C and the one-inclusion graph of C, directing each edge away from the concept whose representative contains the dimension of the edge creates a d-orientation of the one-inclusion graph for the class.*

Proof The outdegree of every concept is equal to size of its representative, which is $\leq d$. ■

The lemma also implies that the representatives of concepts are always subsets of the set of incident dimensions in the one-inclusion graphs.

Corollary 5 *Any representation mapping r of a maximum class C has the property that for any concept $c \in C$, its representative $r(c)$ is a subset of the dimensions incident to c in the one-inclusion graph for C.*

Proof From the counting argument in the proof of Lemma 3 we see that for every $x \in r(c)$ there must exist an edge leaving c in the graph labeled with x. ■

3.1 The Min-Peeling Algorithm

As discussed at the end of the introduction, there is a simple algorithm that always seems to construct a correct representation mapping for any maximum class. The algorithm iteratively removes any lowest degree vertex from the one-inclusion graph for the remaining class and sets the representative $r(c)$ to the set of dimensions of the edges incident to c when c was removed from the graph. The algorithm is formally stated in Figure 6. An illustration of several iterations of the algorithm is given in Figure 7.

Unfortunately, we do not have a proof that this algorithm always produces a correct unlabeled compression scheme for maximum classes. As one of the steps in the correctness proof we would need the following: By iteratively removing the lowest degree vertex from a maximum class, we never arrive at a subgraph whose lowest degree vertex has a degree larger than the VC dimension of the remaining class. A natural conjecture is that any class of VC dimension d has a vertex of degree at most d in its one-inclusion graph. However, an elegant counterexample to this conjecture was constructed in Rubinstein et al. (2007a). Note that their counterexample is not a maximum class and

Min-Peeling Algorithm
Input: a finite maximum concept class C.
Output: a representation mapping r for C
Let G be the one-inclusion graph for C

While $C \neq \emptyset$

1. Choose a minimum-degree vertex c among those in C
2. $r(c) :=$ set of dimension incident to c in the graph
3. Remove c from G and C

Figure 6: The **Min-Peeling** Algorithm for constructing an unlabeled compression scheme for maximum classes.

thus it does not contradict the Min-Peeling Algorithm. Maximum classes and classes obtained by peeling them appear to have a special structure that always ensures existence of a degree $\leq d$ vertex, where d is the VC dimension of the remaining class. However a complete proof of this statement needs to be found.

The representation mappings produced by the Min-Peeling Algorithm have less structure than the representation mappings produced by the Tail Matching Algorithm discussed in Section 6. In particular, they do not necessarily satisfy the condition of Lemma 12, which states the subset of concepts corresponding to representations of size up to k forms a maximum class of VC dimension k. For the maximum class of Figure 2, both algorithms can produce the same representation mapping.

Also note how the Min-Peeling Algorithm immediately leads to a d-orientation of the one-inclusion graph: as we peel away a vertex, its edges are naturally oriented away from the vertex before the edges are removed. Since each vertex has degree at most d when it is peeled away, the outdegree of each vertex will be at most d. Moreover, the resulting orientation of the one-inclusion graph is acyclic because all edges are oriented from a vertex toward a vertex that is removed later. In other words, the list of vertices produced by the Min-Peeling Algorithm is a topological ordering of the oriented one-inclusion graph (see Figure 8). As we shall see later, the Tail Matching Algorithm also produces a topological order of the graph. By Corollary 4, every representation mapping induces a d-orientation of the one-inclusion graph. However we found examples where a valid representation mapping for a maximum class induces a cyclic orientation (not shown).

4. Linear Arrangements

In this section we visualize many of our basic notations and unlabeled compression schemes for simple linear arrangements, which are special maximum classes. An unlabeled compression scheme for linear arrangements is also described in Ben-David and Litman (1998).

A *linear arrangement* is a collection of oriented hyperplanes in $\mathbb{R}^d$. The cells of the arrangement are the concepts and the planes the dimensions of the concept class. The orientations of the planes

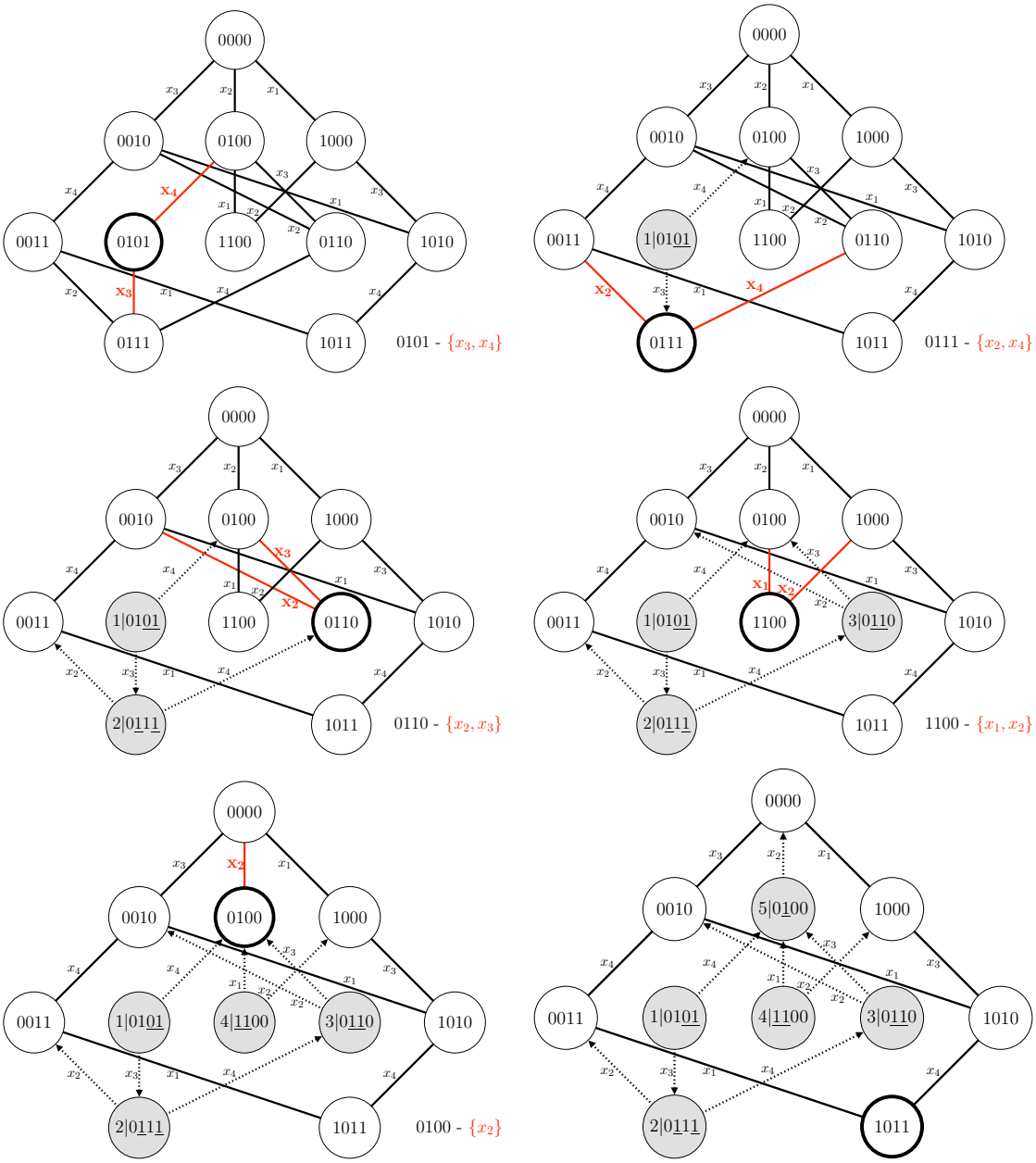


Figure 7: Illustration of a run of the **Min-Peeling** Algorithm. Each figure corresponds to an iteration of the algorithm: the bold circle indicates the chosen concept to be removed and the grey circles are the concepts that were removed previously. By completing the run, one can produce the compression scheme given in Figure 2.

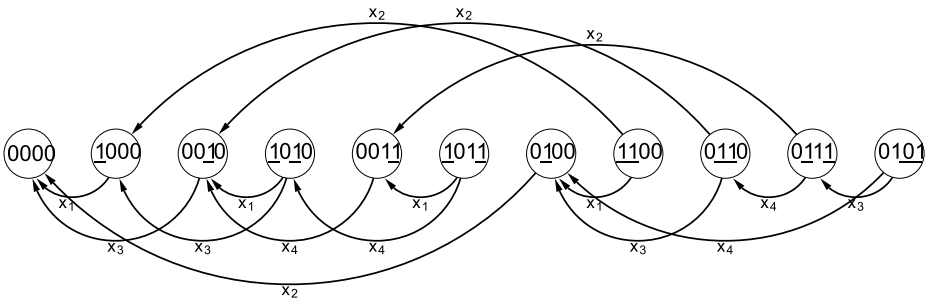


Figure 8: Topological order and d-orientation produced by a run of Min-Peeling Algorithm for some maximum class.

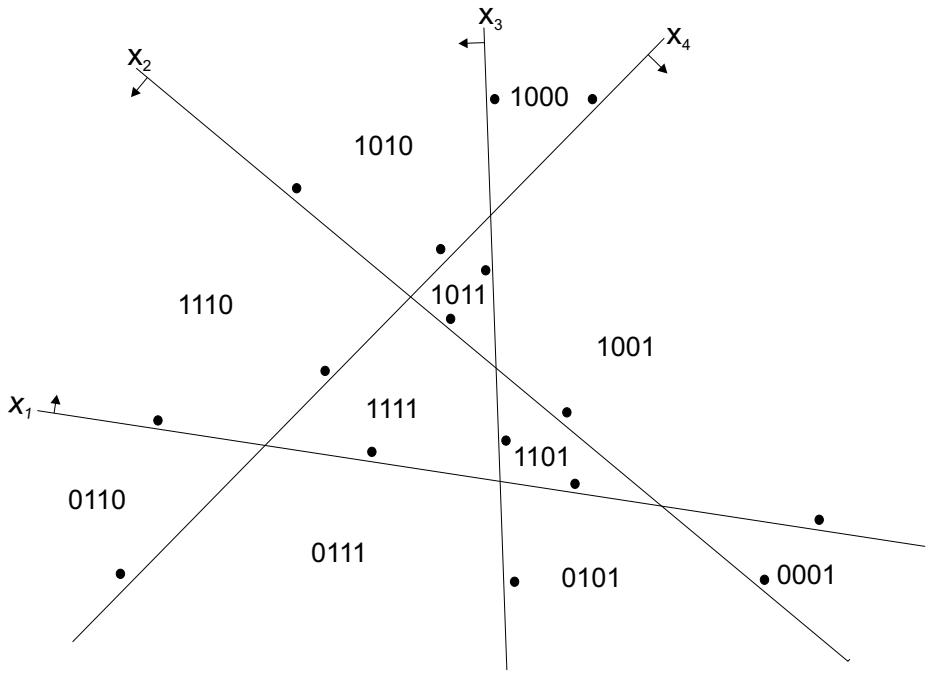


Figure 9: An example linear arrangement. The cells of the arrangement represent the concepts and the planes the dimensions of the class. All cells on the upper side of a hyperplane (indicated by an arrow) are labeled one in the corresponding dimension and the cells on the lower side are labeled zero. Up to d hyperplanes bordering a cell are marked and the set of dimensions of these marked planes forms the representative of the cell in the unlabeled compression scheme.

are indicated by arrows (See Figure 9). All cells lying above the plane corresponding to dimension x label x with one, and the cells below label x with zero. If the n planes are in *general position* (any d hyperplanes have a unique point in common and any $d+1$ hyperplanes have no point in

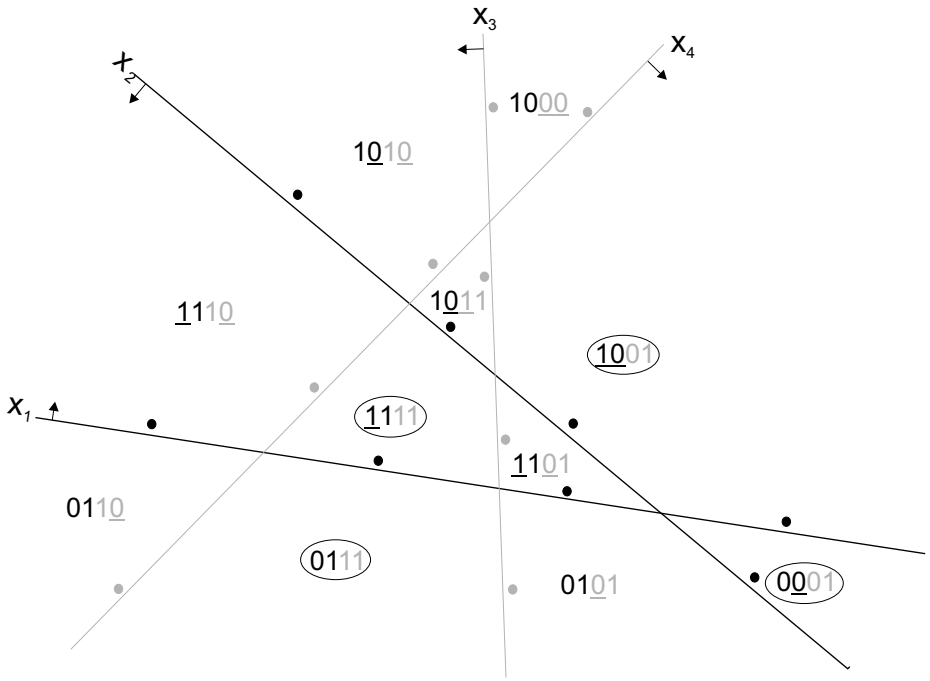


Figure 10: Restriction $C-\{x_3,x_4\}$ of the linear arrangements concept class of Figure 9. The planes x_3 and x_4 are grayed. Several cells of the full arrangement are combined into bigger cells of the restricted arrangement. One of the subcells forming each big cell (circled) has the property that all dimensions in its representative are still available (none of the corresponding hyperplanes were grayed). The representative of this subcell represents the big cell. For example the subcells $1110, 1111, 1101$ form the larger cell for sample $(x_1,1),(x_2,1)$ and only 1111 (circled) is represented by a subset of non-grayed hyperplanes: $r(1111)=\{x_1\}$, and therefore the sample is compressed to $\{x_1\}$.

common) then the arrangement is called *simple*. Such arrangements are maximum classes because their VC dimension is $\min\{n,d\}$ and they have $\binom{n}{\leq d}$ cells (Edelsbrunner, 1987). However not all finite maximum classes are representable as linear arrangements (Floyd, 1989).

The vertices of the one-inclusion graph are the cells of the arrangement and edges connect neighboring cells. A restriction $C-x$ corresponds to removing the x plane from the arrangement. Pairs of cells that border this plane are now combined to larger cells (an example of a double restriction is given in Figure 10). Samples are combined cells produced by removing some hyperplanes. The reduction C^x is the arrangement in the space of dimension $d-1$ induced by the projection of the remaining $n-1$ planes onto the x plane. The subclasses $1C^x$ and $0C^x$ correspond to the cells directly above or below the x plane, respectively. All cells not bordering the x plane form the subclass $\mathrm{tail}_x(C)$.

By Corollary 5, the representative of a concept in an unlabeled compression scheme is always a subset of the incident dimensions (here the bordering hyperplanes) in the one-inclusion graph. So we can indicate the representatives of each cell by marking the inside borders with the corresponding

neighbouring cells (See Figure 9). Each cell marks at most d bordering hyperplanes and no cell marks the same set of hyperplanes. The no-clashing condition of our Main Definition means that any two cells are on opposite sides of at least one hyperplane marked by one of the cells. Also by Lemma 3, any boundary shared between any two cells is marked on exactly one side.

We now visualize how we compress a sample, that is, we restate the process described in Figure 2 for the special case of linear arrangements. Recall that a sample s corresponds to a combined cell produced by removing the hyperplanes in $\mathrm{dom}(c) \setminus \mathrm{dom}(s)$ where each of the original cells corresponds to a concept consistent with the sample. One (and only one) of the original cells in the combined cell that corresponds to the sample is marking only hyperplanes from the surviving set $\mathrm{dom}(s)$ (circled in Figure 10). We compress the sample to that set of marked dimensions and reconstruct based on the represented original cell. Note that if the selected original cell marks any plane, then it must always be at the boundary of the combined cell, since cells in the middle do not border any of the remaining hyperplanes.

It is interesting to observe how our algorithms construct representation mappings for linear arrangements.

Conjecture 1. *Sweeping the arrangement with a hyperplane that is not parallel to any plane in the arrangement produces a compression scheme as follows: as soon as a cell is completely swept, it marks the planes of all bordering currently live cells. The resulting sequential assignments of representatives to concepts corresponds to a run of the Min-Peeling Algorithm.*

In particular we conjecture that sweeping as prescribed iteratively completes minimum degree cells.

The recursive Tail Matching Algorithm of Section 6 chooses some plane x and first finds a compression scheme for the projection C^x of the linear arrangement onto the this plane. Each projected cell from C^x corresponds to two cells, one from $0C^x$ and one from $1C^x$. The algorithm uses the scheme for C^x for all concepts in $0C^x$, that is, all cells bordering the x plane from below. The sibling cells in $1C^x$ right above the plane receive the same marks but also an additional mark from the x plane. The recursive algorithm uses exactly d marks for all vertices in $\mathrm{tail}_x(C)$ (the cells not bordering the x plane). However, this assignment cannot be easily visualized.

Note that one of the planes has the property that the markings produced by the recursive algorithm all lie on one side of the plane. We initially conjectured that there always exist representation schemes that place the marks on the same side for all planes. However we found small counterexamples to this conjecture (not shown).

Simple linear arrangements are known to have the following property: the shortest path between any two cells is always equal to the Hamming distance between the cells (Edelsbrunner, 1987). Surprisingly, we were able to show in Lemma 14 that all maximum classes have this property.

5. Comparison with Old Scheme

In the unlabeled compression schemes introduced in this paper, each subset of up to d domain points represents a unique concept in the class, and every sample of a concept contains exactly one subset that represents a concept consistent with the sample. Before we show that there always exist unlabeled compression schemes, we present the old compression scheme for maximum classes from Floyd and Warmuth (1995) in a concise way that brings out the difference between both schemes.

In the old scheme every set of *exactly d labeled* points represents a concept. Let u denote such a set of d labeled points. By the properties of maximum classes, the reduction $C^{\mathrm{dom}(u)}$ is a maximum class of VC dimension 0, that is, just a single concept on the domain $\mathrm{dom}(C) \setminus \mathrm{dom}(u)$.[5] Augmenting this concept with any of the 2^d labelings of $\mathrm{dom}(u)$, leads to a concept in C on the full domain. Let c_u denote the concept in C represented by the labeled set u in this way.

Note that there are $2^d\binom{n}{d}$ labeled subsets of size d when the domain size is n, and the number of concepts in the maximum class C is $\binom{n}{\leq d}$. This means that some concepts have multiple representatives in the old scheme. In Figure 11 we give both compression schemes for the maximum class used in the previous figures.

We first reason that every concept in C is represented by some labeled subset u of the domain of size d. Since the one-inclusion graph for C is connected (see Gurvits, 1997, or Lemma 14 of this paper), any concept c has an edge along some dimension x. Therefore, $c-x$ lies in C^x. Inductively we can find a labeled set v of size $d-1$ that represents $c-x$ in C^x. Now let $u = v \cup \{(x, c(x))\}$. Clearly, $c_u = c$ (since $C^{\mathrm{dom}(u)} = (C^x)^{\mathrm{dom}(v)}$).

We still need to show that for every sample s of C with at least d points, there is at least one labeled subset u of size d that represents a concept consistent with the entire sample. Since the restriction $C|\mathrm{dom}(s)$ is a maximum class of VC dimension d, it follows from the previous paragraph that there is a labeled subset u representing the concept s of $C|\mathrm{dom}(s)$. However, u also represents a concept c in C. It suffices to show that u represents the same concept on $\mathrm{dom}(s)$ wrt both classes C and $C|\mathrm{dom}(s)$.

Assume the concept for C labels some point x in $\mathrm{dom}(s) \setminus \mathrm{dom}(u)$ with 0 and the concept for $C|\mathrm{dom}(s)$ labels this point with 1. Then from the construction of the representations for C it follows that there are 2^d concepts in C that label x with 0 and $\mathrm{dom}(u)$ in all possible ways. Similarly there are 2^d concepts in $C|\mathrm{dom}(s)$ labeling x with 1. The latter concepts extend to concepts in C and therefore the $d+1$ points $\mathrm{dom}(u) \cup \{x\}$ are shattered by class C, which is a contradiction.

6. Tail Matching Algorithm for Constructing an Unlabeled Compression Scheme

The unlabeled compression scheme for any maximum class can be found by the recursive algorithm given in Figure 13. For any $x \in \mathrm{dom}(C)$, there are two "copies" of C^x in the original class, one in which the concepts in C^x are extended in the x dimension with label 0 and one with extension $(x,1)$. This algorithm first finds a representation mapping r for C^x to subsets of size up to $d-1$ of $\mathrm{dom}(C) \setminus x$. It then uses this mapping for the $(x,0)$ extension and adds x to all the representatives in the other extension. Finally, the algorithm completes r by finding the representatives for $\mathrm{tail}_x(C)$ via the subroutine given in Figure 14.

For correctness, it suffices to show that the constructed mapping satisfies both conditions of our Main Definition. We begin with some additional definitions and a sequence of lemmas.

For $a \in \{0,1\}$ and $c \in C - x$, ac denotes a concept formed from c by extending it with (x,a). It is usually clear from the context what the missing x dimension is. Similarly, aC^x denotes the concept class formed by extending all the concepts in C^x with (x,a). Each dimension $x \in \mathrm{dom}(C)$ can be used to split class C into three disjoint sets: $C = 0C^x \dot{\cup} 1C^x \dot{\cup} \mathrm{tail}_x(C)$.

5. Here $C^{\mathrm{dom}(u)}$ is just the consecutive reduction on all d dimensions in $\mathrm{dom}(u)$. The result of this operation does not depend on the order of the reductions (Welzl, 1987).

x_1	x_2	x_3	x_4	Unlab.	Labeled Representatives
0	0	0	0	$\emptyset$	$\{(x_1,0),(x_2,0)\},\{(x_1,0),(x_3,0)\},\{(x_2,0),(x_3,0)\}$
0	0	1	0	$\{x_3\}$	$\{(x_1,0),(x_3,1)\},\{(x_1,0),(x_4,0)\},\{(x_2,0),(x_4,0)\}$ $\{(x_2,0),(x_3,1)\}$
0	0	1	1	$\{x_4\}$	$\{(x_1,0),(x_4,1)\},\{(x_2,0),(x_4,1)\}$
0	1	0	0	$\{x_2\}$	$\{(x_1,0),(x_2,1)\},\{(x_2,1),(x_3,0)\},\{(x_3,0),(x_4,0)\}$
1	0	0	0	$\{x_1\}$	$\{(x_1,1),(x_2,0)\},\{(x_1,1),(x_3,0)\}$
1	0	1	0	$\{x_1,x_3\}$	$\{(x_1,0),(x_3,1)\},\{(x_1,1),(x_4,0)\}$
1	0	1	1	$\{x_1,x_4\}$	$\{(x_1,1),(x_4,1)\}$
1	1	0	0	$\{x_1,x_2\}$	$\{(x_1,1),(x_2,1)\}$
0	1	0	1	$\{x_3,x_4\}$	$\{(x_3,0),(x_4,1)\}$
0	1	1	0	$\{x_2,x_3\}$	$\{(x_2,1),(x_3,1)\},\{(x_2,1),(x_4,0)\},\{(x_3,1),(x_4,0)\}$
0	1	1	1	$\{x_2,x_4\}$	$\{(x_2,1),(x_4,1)\},\{(x_3,1),(x_4,1)\}$

Figure 11: The new unlabeled compression scheme and the old labeled compression scheme for a maximum class.

	Unlabeled	Labeled
Compression	Consider all concepts consistent with the sample and choose the concept whose representative lies completely in the domain of the sample. Compress to that representative	Compress to any set of d labeled points u in the sample such that the single concept $C^{\mathrm{dom}(u)}$ is consistent with the sample
Reconstruction	Predict with the represented concept	Predict with the single concept in $C^{\mathrm{dom}(u)}$ that is extended with the examples from u
# of representatives:	$\binom{n}{\leq d}$	$2^d\binom{n}{d}$

Figure 12: Comparison of the two compression schemes.

A *forbidden labeling* (Floyd and Warmuth, 1995) of a class C of VC dimension d is a labeled set f of $d+1$ points in $\mathrm{dom}(C)$ that is not consistent with any concept in C. We first note that for a maximum class of VC dimension d there is exactly one forbidden labeling f for each set of $d+1$ dimensions in $\mathrm{dom}(C)$. This is because the restriction $C|\mathrm{dom}(f)$ is maximum with dimension d and its size is thus $2^{d+1}-1$. Also if $C=\emptyset$, then $\mathrm{VCdim}(C)=-1$ and the empty set is the only forbidden labeling.

Our Tail Matching Algorithm assigns all concepts in $\mathrm{tail}_x(C)$ a forbidden labeling of the class C^x of size $(d-1)+1$. Since $c|r(c)$ is now a forbidden labeling for C^x, clashes between the $\mathrm{tail}_x(C)$ and C^x are avoided. If n is the domain size of C, then the number of such forbidden labelings is $\binom{n-1}{d}$. The class $\mathrm{tail}_x(C)$ contains the same number of concepts, since $C-x=C^x\dot{\cup}(\mathrm{tail}_x(C)-x)$

and C^x and $C-x$ are maximum classes:

$$|\text{tail}_x(C)| = |C-x| - |C^x| = \binom{n-1}{\leq d} - \binom{n-1}{\leq d-1} = \binom{n-1}{d}. \tag{1}$$

We next show that every tail concept contains some forbidden labeling of C^x and each such forbidden labeling occurs in at least one tail concept. Since any finite maximum class is maximal, adding any concept increases the VC dimension. Adding any concept in $\text{tail}_x(C)-x$ to C^x increases the dimension of C^x to d. Therefore all concepts in $\text{tail}_x(C)$ contain at least one forbidden labeling of C^x. Furthermore, since $C-x$ shatters all sets of size d and $C-x = C^x \dot{\cup} (\text{tail}_x(C)-x)$, all forbidden labels of C^x appear in the tail.

We will now show that the Tail Subroutine actually constructs a *matching* between the forbidden labelings of size d for C^x and the tail concepts that contain them. This matching is unique (Theorem 10 below) and using these matched forbidden labelings as representatives avoids clashes between tail concepts.

We begin by establishing a recursive structure for the tail (see Figure 15 for an example).

Lemma 6 *Let C be a maximum class and $x \neq y$ be two dimensions in* $\text{dom}(C)$. *If we denote* $\text{tail}_x(C^y)$ *as* $\{c_i : i \in I\}$ *and* $\text{tail}_x(C-y)$ *as* $\{c_j : j \in J\}$ *(where $I \cap J = \emptyset$),*[6] *then there exist bit values* $\{a_i : i \in I\}, \{a_j : j \in J\}$ *for the y dimension such that* $\text{tail}_x(C) = \{a_i c_i : i \in I\} \dot{\cup} \{a_j c_j : j \in J\}$.

Proof First note that the sizes add up as they should (see Equation 1 for the tail size calculation):

$$|\text{tail}_x(C)| = \binom{n-1}{d} = \binom{n-2}{d-1} + \binom{n-2}{d} = |\text{tail}_x(C^y)| + |\text{tail}_x(C-y)|.$$

Next we will show that any concept in $\text{tail}_x(C^y)$ and $\text{tail}_x(C-y)$ can be mapped to a concept in $\text{tail}_x(C)$ by extending it with a suitable y bit. We also have to account for the possibility that there can be some concepts $c \in \text{tail}_x(C^y) \cap \text{tail}_x(C-y)$. Concepts in the intersection will need to be mapped back to two different concepts of $\text{tail}_x(C)$.

Consider some concept $c \in \text{tail}_x(C^y)$. Since $c \in C^y$, both extensions $0c$ and $1c$ exist in C. (Note that the first bit is the y position.) If at least one of the extensions lies in $\text{tail}_x(C)$, then we can choose one of the extensions and map c to it. Assume that neither $0c$ and $1c$ lie in $\text{tail}_x(C)$. This means that these concepts both have x edges to some concepts $0c', 1c'$, respectively. But then $c' \in C^y$ and therefore (c, c') forms an x edge in C^y. Thus $c \notin \text{tail}_x(C^y)$, which is a contradiction.

Now consider a concept $c \in \text{tail}_x(C-y)$. It might have one or two y extensions in C. Assume $0c$ was an extension outside of the $\text{tail}_x(C)$. Then this extension has an x edge to some $0c'$ and therefore (c, c') forms an x edge in $C-y$. It follows that all extensions of c will be in the tail.

Finally, we need to avoid mapping back to the same concept in $\text{tail}_x(C)$. This can only happen for concepts in $c \in \text{tail}_x(C^y) \cap \text{tail}_x(C-y)$. In this case $0c, 1c \in C$, and by the previous paragraph, both lie in $\text{tail}_x(C)$. So we can arbitrarily map $c \in \text{tail}_x(C^y)$ to $0c$ and $c \in \text{tail}_x(C-y)$ to $1c$. ■

The next lemma shows that the order of the restriction and reduction operations is interchangeable (see Figure 16 for an illustration).

6. Note that while $C^y \subseteq C-y$, this does not imply that $\text{tail}_x(C^y) \subseteq \text{tail}_x(C-y)$, as the deletion of the concepts $(C-y) \setminus C^y$ from $C-y$ can remove x edges as well, and thus introduce new tail concepts. See Figure 15 for an example.

Tail Matching Algorithm
Input: a maximum concept class C
Output: a representation mapping r for C

1. If $\mathrm{VCdim}(C) = 0$ (i.e., C contains only one concept c), then $r(c) := \emptyset$.
Otherwise, pick any $x \in \mathrm{dom}(C)$ and recursively find a representation mapping $\tilde{r}$ for C^x.
2. Expand $\tilde{r}$ to $0C^x \cup 1C^x$:

$$\forall c \in C^x : r(c \cup \{x = 0\}) := \tilde{r}(c) \text{ and } r(c \cup \{x = 1\}) := \tilde{r}(c) \cup x$$

3. Extend r to $\mathrm{tail}_x(C)$ via the subroutine of Figure 14.

Figure 13: The recursive algorithm for constructing an unlabeled compression scheme for maximum classes.

Tail Subroutine
Input: a maximum concept class C, $x \in \mathrm{dom}(C)$
Output: an assignment of representatives to $\mathrm{tail}_x(C)$

1. If $\mathrm{VCdim}(C) = 0$ (i.e., $C = \{c\} = \mathrm{tail}_x(C)$), then $r(c) := \emptyset$.
If $\mathrm{VCdim}(C) = |\mathrm{dom}(C)|$, then $\mathrm{tail}_x(C) = \emptyset$ and $r := \emptyset$.
Otherwise, pick some $y \in \mathrm{dom}(C)$, $y \neq x$ and recursively find representatives for $\mathrm{tail}_x(C^y)$ and $\mathrm{tail}_x(C - y)$.
2. $\forall c \in \mathrm{tail}_x(C^y) \setminus \mathrm{tail}_x(C - y)$, find $c' \in \mathrm{tail}_x(C)$, s.t. $c' - y = c$, $r(c') := r(c) \cup \{y\}$.
3. $\forall c \in \mathrm{tail}_x(C - y) \setminus \mathrm{tail}_x(C^y)$, find $c' \in \mathrm{tail}_x(C)$, s.t. $c' - y = c$, $r(c') := r(c)$.
4. $\forall c \in \mathrm{tail}_x(C^y) \cap \mathrm{tail}_x(C - y)$, consider the concepts $0c, 1c \in \mathrm{tail}_x(C)$. Let r_1 be the representative for c from $\mathrm{tail}_x(C^y)$ and r_2 be the one from $\mathrm{tail}_x(C - y)$. Suppose, wlog, that $0c|r_1 \cup \{y\}$ is a sample not consistent with any concept in C^x. Then $r(0c) := r_1 \cup \{y\}$, $r(1c) := r_2$.

Figure 14: The **Tail Subroutine** for finding tail representatives

Lemma 7 *For any maximum class C and two dimensions $x \neq y$ in* $\mathrm{dom}(C)$, $C^x - y = (C - y)^x$.

Proof We first show that $C^x - y \subseteq (C - y)^x$. Take any $c \in C^x - y$. By the definition of restriction, there exists a bit a_y such that $a_y c \in C^x$. Since concepts in C^x always have two extensions in C, it

x_1	x_3	x_4
0	0	0
0	1	0
0	1	1
1	0	0
1	1	0
1	1	1
0	0	1

$C - x_2$

x_1	x_3	x_4
0	0	0
1	0	0
0	1	0
0	1	1

C^{x_2}

x_1	x_2	x_3	x_4	
0	1	0	1	$\text{tail}_{x_1}(C - x_2)$
0	1	1	0	$\text{tail}_{x_1}(C^{x_2})$
0	1	1	1	$\text{tail}_{x_1}(C^{x_2})$

$\text{tail}_{x_1}(C)$

Figure 15: Illustration of Lemma 6 which shows that $\text{tail}_{x_1}(C)$ can be composed from $\text{tail}_{x_1}(C^{x_2})$ and $\text{tail}_{x_1}(C - x_2)$; class C is from Figure 2, tails in classes are separated by horizontal lines and the last column for $\text{tail}_{x_1}(C)$ indicates whether the concept comes from $\text{tail}_{x_1}(C^{x_2})$ or $\text{tail}_{x_1}(C - x_2)$.

0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	1	0
1	1	1

$C - x_2$

0	0	0
0	1	0
0	1	1
1	0	0

C^{x_1}

0	0
1	0
1	1

$C^{x_1} - x_2 = (C - x_2)^{x_1}$

Figure 16: Illustration of Lemma 7 which shows that $C^{x_1} - x_2 = (C - x_2)^{x_1}$: class C is given in Figure 2.

follows that $0a_yc, 1a_yc \in C$. By first restricting these two concepts in y and then reducing in x we have $0c, 1c \in C - y$ and $c \in (C - y)^x$, respectively.

Both $(C - y)^x$ and $C^x - y$ are maximum classes with the same domain size $|\text{dom}(C)| - 2$ and the same VC dimension. Therefore both have the same size, and since $C^x - y \subseteq (C - y)^x$, they are in fact equal. ■

Corollary 8 *Any forbidden labeling of $(C - y)^x$ is also a forbidden labeling of C^x.*

Proof By the previous lemma, the forbidden labelings of $(C - y)^x$ and $C^x - y$ are the same. The corollary now follows from the fact that the forbidden labelings of $C^x - y$ are exactly all forbidden labeling of C^x that do not contain y. ■

Lemma 9 *If we have a forbidden labeling for C^{xy} of size $d-1$, then there exists a bit value for the y dimension such that extending this forbidden labeling with this bit results in a forbidden labeling of size d for C^x.*

Proof We will establish a bijection between forbidden labelings of C^x of size d that contain y and forbidden labelings of size $d-1$ for C^{xy}. Since C^x is a maximum class of VC dimension $d-1$, it has $\binom{n-1}{d}$ forbidden labelings of size d, one for every set of d dimensions from $\mathrm{dom}(C) \setminus x$. Exactly $\binom{n-2}{d-1}$ of these forbidden labelings contain y and this is also the total number of forbidden labelings of size $d-1$ for C^{xy}.

We map the forbidden labelings of size d for C^x that contain y to labelings of size $d-1$ by discarding the y dimension. Assume that a labeling constructed this way is not forbidden in C^{xy}. Then by extending the concept that contains this labeling with both $(y,0)$ and $(y,1)$ back to C^x, we will hit the original forbidden set, thus forming a contradiction.

It follows that every forbidden set is mapped to a different forbidden labeling and by the counting argument above we see that all forbidden sets are covered. Thus the mapping is a bijection and the inverse of this mapping proves the lemma. ■

Theorem 10 *Let C be any maximum class C of VC dimension d and domain size n. For any $x \in \mathrm{dom}(C)$ we can construct a bipartite graph between the $\binom{n-1}{d}$ concepts in $\mathrm{tail}_x(C)$ and the $\binom{n-1}{d}$ forbidden labelings of size d for C^x with an edge between a concept and a forbidden labeling if this labeling is contained in the concept. All such graphs have a unique matching.*

Proof The proof is an induction on $n = |\mathrm{dom}(C)|$ and d. More precisely, we induct on the pairs (n,d) (where $n \geq d$) in lexicographic order. The minimal element of this order is $(0,0)$.

Note that in the Tail Matching Algorithm 13 we actually stop when $n = d$, in which case we have a complete hypercube with no tail and the matching is empty. Also for $d = 0$, there is a single concept which is always in the tail and gets matched to the empty set.

Inductive hypothesis: For any maximum class $\widetilde{C}$, such that $(|\mathrm{dom}(\widetilde{C}|, \mathrm{VCdim}(\widetilde{C})) < (n,d)$ the statement of the theorem holds.

Inductive step. Let $x, y \in \mathrm{dom}(C)$ and $x \neq y$. By Lemma 6, we can compose $\mathrm{tail}_x(C)$ from $\mathrm{tail}_x(C^y)$ and $\mathrm{tail}_x(C-y)$. Since $\mathrm{VCdim}(C^x) = d-1$ and $|\mathrm{dom}(C-x)| = n-1$,[7] then $(n-1,d), (n-1,d-1) < (n,d)$ and we can use the inductive hypothesis for these classes and assume that the desired matchings already exist for $\mathrm{tail}_x(C^y)$ and $\mathrm{tail}_x(C-y)$.

Now we need to combine these matchings to form a matching for $\mathrm{tail}_x(C)$. See Figure 14 for a description of this process. Concepts in $\mathrm{tail}_x(C-y)$ are matched to forbidden labelings of $(C-y)^x$ of size d. By Lemma 8, any forbidden labeling of $(C-y)^x$ is also a forbidden labeling of C^x. Thus this part of the matching transfers to the appropriate part of $\mathrm{tail}_x(C)$ without alterations. On the other hand, $\mathrm{tail}_x(C^y)$ is matched to labelings of size $d-1$. We can make them labelings of size d by adding some value for the y coordinate. Some care must be taken here. Lemma 9 tells us that one of the two extensions will in fact have a forbidden labeling of size d (that includes the y coordinate). In the case where just one of two possible extensions of a concept in $\mathrm{tail}_x(C^y)$ is in the $\mathrm{tail}_x(C)$, there are no problems: the single concept will be the concept of Lemma 9, since the other concept lies in C^x and thus does not contain any forbidden labelings. There is also the possibility

7. $\mathrm{VCdim}(C-x) = d$, unless $n = d$, in which case it would obviously drop by one as well.

that both extensions are in $\text{tail}_x(C)$. From the proof of Lemma 6 we see that this only happens to the concepts that are in $\text{tail}_x(C^y) \cap \text{tail}_x(C-y)$. Then, by Lemma 9, we can figure out which extension corresponds to the forbidden labeling involving y and use that for the $\text{tail}_x(C^y)$ matching. The other extension will correspond to the $\text{tail}_x(C-y)$ matching. Essentially, where before Lemma 6 told us to map the intersection $\text{tail}_x(C^y) \cap \text{tail}_x(C-y)$ back to $\text{tail}_x(C)$ by assigning a bit arbitrarily, we now choose a bit in a specific way.

So far we have shown that the matching exists. We still need to verify its uniqueness. From any matching for $\text{tail}_x(C)$ we will show how to construct matchings for $\text{tail}_x(C-y)$ and $\text{tail}_x(C^y)$ with the property that two different matchings for $\text{tail}_x(C)$ will disagree with the constructed matchings for either $\text{tail}_x(C-y)$ or $\text{tail}_x(C^y)$. Now, uniqueness follows by induction.

Consider any concept c in $\text{tail}_x(C)$, such that $c-y \in \text{tail}_x(C^y) \setminus \text{tail}_x(C-y)$. Then $c-y$ lies in $C-y$, but not in $\text{tail}_x(C-y)$. Therefore $c-y$ must belong to either $0(C-y)^x$ or $1(C-y)^x$, which means that this concept cannot contain a forbidden set for $(C-y)^x$. We claim that any forbidden set of c for C^x must contain y. Otherwise such a set would be forbidden for $C^x - y$, which by Lemma 7 equals $(C-y)^x$. By a similar argument, concepts $c \in \text{tail}_x(C)$, such that $c-y \in \text{tail}_x(C-y) \setminus \text{tail}_x(C^y)$ have to be matched to forbidden sets that do not contain y (since a forbidden set of size d for C^x that contains y, becomes a forbidden set of size $d-1$ for C^{xy} just by removing y, and condition $c-y \notin \text{tail}_x(C^y)$ implies that c does not contain any such forbidden sets).

From these two facts it follows that if a concept in $\text{tail}_x(C)$ is matched to a forbidden set containing y, then $c-y \in tail_x(C^y)$, and if it is matched to a set not containing y, then $c-y \in tail_x(C-y)$. We conclude that a matching for $\text{tail}_x(C)$ splits into a matching for $\text{tail}_x(C-y)$ and a matching for $\text{tail}_x(C^y)$. This implies that if there are two matchings for all of $\text{tail}_x(C)$, then there are either two matchings for $\text{tail}_x(C-y)$ or two matchings for $\text{tail}_x(C^y)$. ■

Theorem 11 *The Tail Matching Algorithm of Figure 13 returns a representation mapping that satisfies both conditions of the Main Definition.*

Proof Proof by induction on $d = \text{VCdim}(C)$. The base case is $d = 0$: this class has only one concept which is represented by the empty set.

The algorithm recurses on C^x and $\text{VCdim}(C^x) = d-1$. Thus we can assume that it has a correct representation mapping for C^x that uses sets of size at most $d-1$ for the representatives.

Bijection condition: The representation mapping for C is composed of a bijection between $1C^x$ and all sets of size $\leq d$ containing x, a bijection between $0C^x$ and all sets of size $< d$ that do not contain x, and finally a bijection between $\text{tail}_x(C)$ sets of size equal d that do not contain x.

No clashes condition: By the inductive assumption there cannot be any clashes internally within each of the subclasses $0C^x$ and $1C^x$, respectively. Clashes between $0C^x$ and $1C^x$ cannot occur because such concepts are always differentiated on the x bit and x belongs to all representatives of $1C^x$. By Theorem 10, we know that concepts in the tail are assigned to representatives that define a forbidden labeling for C^x. Therefore, clashes between $\text{tail}_x(C)$ and $0C^x$, $1C^x$ are avoided. Finally, we need to argue that there cannot be any clashes internally within the tail. By Theorem 10, the matching between concepts in $\text{tail}_x(C)$ and forbidden labeling of C^x is unique. So if this matching resulted in a clash, that is, $c_1|r_1 \cup r_2 = c_2|r_1 \cup r_2$, then both c_1 and c_2 would contain the forbidden labelings specified by representative r_1 and r_2. By swapping the assignment of forbidden labels between c_1 and c_2 (i.e., c_1 is assigned to $c_1|r_2$ and c_2 to $c_2|r_1$) we would create a new valid matching,

thus contradicting the uniqueness of the matching. ■

Note that by Corollary 4, the unlabeled compression scheme produced by our recursive algorithm induces a d-orientation of the one-inclusion graph: orient each edge away from the concept that contains the dimension of the edge in its representative. As was the case for the orientation produced by the Min-Peeling Algorithm, the resulting orientation is acyclic. As a matter of fact a topological order can be constructed by ordering the concepts of C as follows: $0C^x, 1C^x, \text{tail}_x(C)$. The concept within $\text{tail}_x(C)$ can be ordered arbitrarily and the concepts within $0C^x$ and $1C^x$ are ordered recursively based on the topological order for C^x.

7. Miscellaneous Lemmas

We conclude with some miscellaneous lemmas that highlight the combinatorics underlying the unlabeled compression schemes for maximum classes. The first one shows that the representatives constructed by our Tail Matching Algorithm induce a nesting of maximum classes. This is a special property, because there are cases where the simpler Min-Peeling Algorithm produces a representation mapping that does not have this property (not shown).

Lemma 12 *Let C be a maximum concept class with VC dimension d and let r be a representation mapping for C produced by the Tail Matching Algorithm. For $0 \le k \le d$, let $C_k = \{c \in C \text{ s. t. } |r(c)| \le k\}$. Then C_k is a maximum concept class of VC dimension k.*

Proof Proof by induction on d. Base case $d = 0$: the class has only one concept and the lemma clearly holds.

The lemma trivially holds for $k = 0$ or $k = d$. Otherwise let $x \in \text{dom}(C)$ be the first dimension used in the recursion of the Tail Matching Algorithm and assume by induction that the lemma holds for C^x. Consider which concepts in C belong to C_k. Clearly none of the concepts in $\text{tail}_x(C)$ lie in C_k because their representatives are of size $d > k$. From the recursion of the algorithm it follows that $C_k = 0C^x_k \cup 1C^x_{k-1}$, that is, it consists of all concepts in $0C^x$ with representatives of size $\le k$ in the mapping for C^x, plus all the concepts in $1C^x$ with representatives of size $\le k-1$ in the mapping for C^x. By the inductive assumption, C^x_k and C^x_{k-1} are maximum classes with VC dimension k and $k-1$, respectively. Furthermore, the definition of C_k implies that $C^x_{k-1} \subset C^x_k$.

Since $|C_k| = |0C^x_k| + |1C^x_{k-1}| = \binom{n-1}{\le k} + \binom{n-1}{\le k-1} = \binom{n}{\le k}$, the class C_k has the right size and $\text{VCdim}(C_k) \ge k$. We still need to show that C_k does not shatter any set of size $k+1$. Consider any such set that does not contain x. This set would have to be shattered by $C_k - x = C^x_k \cup C^x_{k-1} = C^x_k$, which is impossible. Now consider any set A of size $k+1$ that does contain x. All the 1 values for the x coordinate happen in the $1C^x_{k-1}$ part of C_k. Thus $A \setminus x$ must be shattered by C^x_{k-1} whose VC dimension is again one too low. ■

We actually proved that the C_k produced by the representation mapping of the Tail Matching Algorithm always satisfy the recurrence $C_k = 0C^x_k \cup 1C^x_{k-1}$. On the other hand there are nestings of maximum concept classes $C_0 \subset C_1 \subset \ldots \subset C_d = C$, where C_k has VC dimension k, for which the above recurrence does not hold (not shown).

Open Problem 1. *We do not know whether for any nesting $C_0 \subset C_1 \subset \ldots \subset C_d = C$ of maximum classes, where C_k has VC dimension k, there always exists a representation mapping that induces this nesting.*

We now consider the connectivity of the one-inclusion graphs of maximum classes. It was known previously that they are connected (Gurvits, 1997). We show in Lemma 14 that the length of the shortest path between any two concepts in these graphs is always the Hamming distance between the concepts. This property was previously known for the one-inclusion graphs of linear arrangements, which are special maximum classes. The following technical lemma is necessary to prove the property for arbitrary maximum classes.

We use $I_C(c)$ to denote the set of dimensions incident to c in the one-inclusion graph for C and let $E(C)$ denote the set of all edges of the graph.

Lemma 13 *For any maximum class C and $x \in \mathrm{dom}(C)$, restricting wrt x does not change the sets of incident dimensions of concepts in $\mathrm{tail}_x(C)$, that is, $\forall c \in \mathrm{tail}_x(C)$, $I_C(c) = I_{C-x}(c-x)$.*

Proof Let (c, c') be any edge leaving a concept $c \in \mathrm{tail}_x(C)$. By the definition of $\mathrm{tail}_x(C)$, this edge cannot be an x edge, and therefore c and c' agree on x and $(c-x, c'-x)$ is an edge in $C-x$. It follows that $I_C(c) \subseteq I_{C-x}(c-x)$ when $c \in \mathrm{tail}_x(C)$.

If $I_C(c)$ is a strict subset of $I_{C-x}(c-x)$ for some $c \in \mathrm{tail}_x(C)$, then the number of edges incident to $\mathrm{tail}_x(C) - x = (C-x) \setminus C^x$ in $C-x$ is larger than the number of edges incident to $\mathrm{tail}_x(C)$ in C. The first number is a difference between the sizes of edge sets of the two maximum classes $C-x$ and C^x. Recall that if C is maximum on domain of size n and has VC dimension d, then its edge set $E(C)$ has size $n\binom{n-1}{\leq d-1}$ (see proof of Lemma 3 or Lemma 15). Thus the first number is

$$\begin{aligned} |E(C-x)| - |E(C^x)| &= (n-1)\binom{n-2}{\leq d-1} - (n-1)\binom{n-2}{\leq d-2} \\ &= (n-1)\binom{n-2}{d-1} = d\binom{n-1}{d}. \end{aligned}$$

Furthermore, the second number is the number of edges in C minus the number of intra edges in $0C^x$ and $1C^x$, respectively, minus the number of cross edges between $0C^x$ and $1C^x$:

$$\begin{aligned} |E(C)| - 2|E(C^x)| - |C^x| &= n\binom{n-1}{\leq d-1} - 2(n-1)\binom{n-2}{\leq d-2} - \binom{n-1}{\leq d-1} \\ &= (n-1)\left(\binom{n-1}{\leq d-1} - 2\binom{n-2}{\leq d-2}\right) \\ &= (n-1)\left(\binom{n-2}{\leq d-1} - \binom{n-2}{\leq d-2}\right) \\ &= (n-1)\binom{n-2}{d-1} = d\binom{n-1}{d}. \end{aligned}$$

Thus the two numbers are the same and we have a contradiction. ■

Lemma 14 *In the one-inclusion graph for a maximum concept class C, the length of the shortest path between any two concepts is equal to their Hamming distance.*

Proof The proof will proceed by induction on $|\mathrm{dom}(C)|$. The lemma trivially holds when $|\mathrm{dom}(C)| = 0$ (i.e., $C = \emptyset$). Let c_1, c_2 be any two concepts in a maximum class C of domain size $n > 0$ and let

$x \in \text{dom}(C)$. Since $C-x$ is a maximum concept class with a reduced domain size, there is a shortest path P between $c_1 - x$ and $c_2 - x$ in $C-x$ of length equal their Hamming distance. The class $C-x$ is partitioned into C^x and $\text{tail}_x(C)-x$. If $\hat{c}_1$ is the first concept of P in C^x and $\hat{c}_2$ the last, then by induction on the maximum class C^x (also of reduced domain size), there is a shortest path between $\hat{c}_1$ and $\hat{c}_2$ that only uses concepts of C^x. Thus we can assume that P begins and ends with a segment in $\text{tail}_x(C)-x$ and has a segment of C^x concepts in the middle, where some of the three segments may be empty.

We partition $\text{tail}_x(C)$ into $\text{tail}_{x=0}(C)$ and $\text{tail}_{x=1}(C)$. There are no edges between these two sets because they would have to be x edges. There are also no edges between the restrictions $\text{tail}_{x=0}(C)-x$ and $\text{tail}_{x=1}(C)-x$ of the two sets, because by Lemma 13 these edges would also exist between the original sets $\text{tail}_{x=0}(C)$ and $\text{tail}_{x=1}(C)$. It follows that any segment of P from $\text{tail}_x(C)-x$ must be from the same part of the tail. Also if the initial segment and final segment of P are both non-empty and from different parts of the tail, then the middle C^x segment cannot be empty.

We can now construct a shortest path P' between c_1 and c_2 from the path P. If $c_1(x) = c_2(x)$ then we extend the concepts in P with $x = c_1(x)$ to obtain a path P' between c_1 and c_2 in C of the same length. Note that from the above discussion, all concepts in the beginning and ending tail segments of P come from the part of the $\text{tail}_x(C)$ that label x with $c_1(x) = c_2(x)$. Also for the middle segment of P we have the freedom to use label $c_1(x)$.

If $c_1(x) \neq c_2(x)$, then P must contain a concept $\tilde{c}_1$ in C^x, because if all concepts in P lied in $\text{tail}_x(C)-x$ then this would imply an edge between a concept in $\text{tail}_{x=0}(C)-x$ and a concept in $\text{tail}_{x=1}(C)-x$. We now construct a new path P' in C of length $|P|+1$ which is one more than the Hamming distance $|P|$ between c_1-x and c_2-x: extend the concepts up to $\tilde{c}_1$ in P with label $c_1(x)$ on x; then cross to the sibling concept $\tilde{c}_2$ which disagrees with $\tilde{c}_1$ only on its x dimension; finally extend the concepts in path P from $\tilde{c}_2$ onward with label $c_2(x)$ on x. ■

We already know that the number of vertices and edges in the one-inclusion graph of a maximum class of domain size n and VC dimension d is $\binom{n}{\leq d}$ and $n\binom{n-1}{\leq d-1}$, respectively. Since vertices and edges are hypercubes of dimension 0 and 1, respectively, these bounds are special cases of the below lemma and corollary, where we bound the number of hypercubes of dimension r, for $0 \leq r \leq d$.

Lemma 15 *Let C be any class of domain size n and VC dimension d. Then the number of hypercubes of dimension $0 \leq r \leq d$ which are subgraphs of the one-inclusion graph for C is at most $\binom{n}{r}\binom{n-r}{\leq d-r}$.*

Proof Pick any subset $A \subseteq \text{dom}(C)$ of size r. Recall that C^A consists of all concepts in $C|(\text{dom}(C)-A)$ with the property that all $2^{|A|}$ extensions to the original domain $\text{dom}(C)$ are in C. Thus any concept in the reduced class C^A defines a hypercube of dimension $|A|$ which is a subgraph of the original one-inclusion graph for C. Also from the definition of C^A it follows that all hypercubes that are subgraphs using the dimension set A correspond to a concept in C^A. Note that two hypercubes from the same C^A have no common concepts (vertices), but hypercubes from different restriction sets of the same size may overlap on their vertex set but they are never identical.

From the above discussion it follows that the total number of hypercubes of dimension r is the total size of all C^A, where A has size r. Since the reductions C^A are classes of domain size $n-r$ and VC dimensions at most $d-r$, the inequalities of the lemma follow from Sauer's lemma. ■

Corollary 16 *For maximum classes of domain size n and VC dimension d, all $d+1$ inequalities of the previous lemma are tight. Also for any class C of domain size n and VC dimension d, if one of the inequalities is tight, then C is maximum and they are all tight.*

Proof For maximum classes we have that for any set A of size $0 \leq r \leq d$, the reduction C^A of C is also a maximum class on domain size $n-r$ and VC dimension $d-r$ (Welzl, 1987; Floyd and Warmuth, 1995). Therefore for maximum classes all inequalities are tight.

Observe that since $|C| = |C^x| + |C - x|$, it follows that if C^x and $C - x$ are maximum, then $|C| = \binom{n-1}{\leq d-1} + \binom{n-1}{\leq d} = \binom{n}{\leq d}$ and C is maximum as well.

If the inequality of the previous lemma is tight for some size r, then for all sets of this size, C^A is a maximum class of VC dimension $n-r$. We will show by the usual double induction on n and d, that in this case C is maximum. Essentially, for C^x the inequality for size $r-1$ is tight, since for all A containing x, $C^A = (C^x)^{(A \setminus x)}$. Furthermore, for $C - x$ the inequality for size r is tight, since for all A not containing x, $(C-x)^A \supseteq C^A - x$ and $C^A - x$ is maximum because C^A is maximum. ■

If the following lemma could be proven for any concept class produced by peeling minimum degree vertices off a maximum class, then this would be sufficient to prove the non-clashing condition for the representation map produced by the Min-Peeling Algorithm. However the current proof only holds for maximum classes, which is the base case.

Lemma 17 *In a maximum class C the labeling of the set of incident dimensions of any concept c uniquely identifies the concept, that is:*

$$\forall c' \in C : c' \neq c \;\Leftrightarrow\; c|I_C(c) \neq c'|I_C(c). \tag{2}$$

Proof We employ an induction on $|\mathrm{dom}(C)|$. The base case is $|\mathrm{dom}(C)| = \mathrm{VCdim}(C)$. In this case, C is a complete hypercube. Note that if $I_C(c) = \mathrm{dom}(C)$, then $c|I_C(c) = c|\,\mathrm{dom}(C) = c$ and Equation (2) follows from the uniqueness of each concept. In the hypercube all concepts have this property.

For the general case, if $I_C(c) \neq \mathrm{dom}(C)$ pick $x \notin I_C(c)$ for which $c \in \mathrm{tail}_x(C)$. We have to show that $\forall c' \neq c$, $c|I_C(c) \neq c'|I_C(c)$. Consider the maximum concept class $C - x$ and its concept $c - x$. Because of the reduced domain, we know by induction that

$$\forall c'' \in C - x : c'' \neq c - x \;\Leftrightarrow\; c - x|I_{C-x}(c-x) \neq c''|I_{C-x}(c-x).$$

Since $c'' = c' - x$, for some $c' \in C$, we can let quantification run over $c' \in C$ and the above is equivalent to

$$\forall c' \in C : c' - x \neq c - x \;\Leftrightarrow\; c - x|I_{C-x}(c-x) \neq c' - x|I_{C-x}(c-x).$$

Also since $c \in \mathrm{tail}_x(C)$ does not have an x edge, $\forall c' \in C : c' - x \neq c - x$ is equivalent to $\forall c' \in C : c' \neq c$ and by Lemma 13, $I_{C-x}(c-x) = I_C(c)$. This gives us the equivalent statement: $\forall c' \in C : c' \neq c \;\Leftrightarrow\; c - x|I_C(c) \neq c' - x|I_C(c)$. Finally, since $x \notin I_C(c)$, $c - x|I_C(c) = c|I_C(c)$ and $c' - x|I_C(c) = c'|I_C(c)$, giving us Equation (2). ■

x_1	x_2	x_3	x_4	r
0	0	0	0	$\emptyset$
0	0	1	0	$\{x_3\}$
0	1	0	0	$\{x_2\}$
1	0	0	0	$\{x_1\}$
0	1	1	0	$\{x_2,x_3\}$
1	0	1	0	$\{x_1,x_3\}$
1	1	0	0	$\{x_1,x_2\}$
0	1	1	1	$\{x_1,x_4\}$
1	0	1	1	$\{x_2,x_4\}$
1	1	0	1	$\{x_3,x_4\}$
1	1	1	1	$\{x_4\}$

Table 1: A maximal class that does not have a compression scheme with a representation mapping from sets of domain points of size at most $\mathrm{VCdim}(C) = 2$ to concepts. However if we are allow mappings to concepts in and outside of the class, then the no-clashing condiction can still be satisfied and a valid scheme exists. In the given solution, $\{x_4\}$ represents 1111, which is not a concept in the class.

Conjecture 2. *The above lemma holds for all classes produced by iteratively peeling minimum degree vertices off a maximum class.*

8. Discussion of Possible Compression Schemes for Maximal Classes

This section discusses the possibility of proving the compression scheme conjecture in the general case. Any finite or infinite concept class is called *maximal* if no concept can be added without increasing the VC dimension. Any concept class can be embedded into a maximal class by adding as many concepts to the class as possible until no new concept can be added without increasing the VC dimension. Figure 3 presents an example of a maximal class. All finite maximum classes are also maximal, but there are infinite maximum classes which are not maximal (Floyd and Warmuth, 1995). For the rest of this section maximal means: finite, maximal and not maximum.

A natural idea for constructing a compression scheme for any class is to embed that class into some other class, for which a compression scheme is known. However adding any concepts to a maximal class increases its VC dimension, so we would want an embedding that does not increase the VC dimension too much. The question whether and how this can be done is an intriguing open problem of its own.

The old labeled compression scheme for maximum classes cannot be extended to maximal classes due to the nature of its mapping between representatives and represented concepts. The old scheme compresses to a labeled set u of size $\mathrm{VCdim}(C) = d$ and u represents the single concept in the d-fold reduction $C^{\mathrm{dom}(u)}$ extended with the d examples of u (See Section 5). In the case of

x_1	x_2	x_3	x_4	r
0	0	1	1	$\{x_1,x_2\},\{x_3,x_4\}$
0	1	0	0	$\{x_3\}$
0	1	0	1	$\{x_2\}$
0	1	1	0	$\{x_1\}$
1	0	0	0	$\{x_2,x_3\}$
1	0	0	1	$\{x_1,x_3\}$
1	0	1	0	$\{x_1,x_2\}$
1	1	0	0	$\{x_1,x_4\}$
1	1	0	1	$\{x_2,x_4\}$
1	1	1	0	$\{x_3,x_4\}$

Table 2: A maximal class that does have an unlabeled scheme where concepts in the class have multiple representatives. The no-clashing holds for any representatives of different concepts, and for all samples there is exactly one consistent concept with a representative in the sample domain.

maximal classes, many d-fold reductions will be empty. Thus it is unclear, which concepts the corresponding set of d labeled examples should represent. Essentially, the old scheme relied on the fact that maximum classes are unions of hypercubes of dimension $\mathrm{VCdim}(C)$. Maximal classes do not have this property. Their one-inclusion graphs can be disconnected. In particular, there are maximal classes whose one-inclusion graph has several isolated vertices, that is, vertices with no incident edges (not shown). Nevertheless, it may be possible to somehow cover maximal classes with hypercubes of dimension d or slightly larger.

Now we consider the possibility of generalizing our new unlabeled compression scheme from finite maximum classes to finite maximal ones. Recall that our scheme has the property that for any sample from the class, there is *exactly one* representative contained within the domain of the sample whose concept is consistent with the sample. Of particular importance in achieving this property was the no-clashing condition for the representatives of concepts. The following lemma describes the effect of having non-clashing representatives for arbitrary concept classes.

Lemma 18 *Let r be any injection between a finite concept class C of VC dimension d and subsets of* $\mathrm{dom}(C)$ *of size at most d. Then the following two statements are equivalent:*

1. *No two concepts clash wrt r.*
2. *For all samples s from C, there is at most one concept $c \in C$ that is consistent with s and $r(c) \subseteq \mathrm{dom}(s)$.*

Proof If there are two concepts c and c' that are consistent with s and $r(c), r(c') \subseteq \mathrm{dom}(s)$, then the concepts clash because $c|r(c) \cup r(c') = c'|r(c) \cup r(c')$. Conversely, if two concepts c and c' clash, then the sample $c|r(c) \cup r(c')$ is consistent with at least two concepts c and c' that satisfy $r(c), r(c') \subseteq \mathrm{dom}(s)$. ■

Intuitively, the no-clashing condition causes the representatives to be spread out as much as possible in an attempt to cover all the samples of concepts in the class. Lemma 18 says that there is *at most one* representative whose concept is consistent with the sample. However we also need the *at least one* condition, which assures that every sample can be compressed. For maximum classes, the latter condition was assured by a counting argument: the number of concepts in $C|\operatorname{dom}(s)$ and the number of subsets of size up to d in $\operatorname{dom}(s)$ is the same; also all such subsets must represent some concept and the no-clashing condition assured that these concepts disagreed on $\operatorname{dom}(s)$. It follows that for each sample, there is always at least one representative in its domain that represents a consistent concept.

There are maximal concept classes of VC dimension d that shatter any set of size d (see Table 2). There are $|C|$ representatives in total and this is less than the total number of subsets of size up to d (since C is maximal but not maximum). Therefore, for some domain of size d, there are 2^d concepts but less than that many representatives over the domain.

Of course it makes sense to use all subsets of size up to d by assigning some concepts more than one representative. Note that two representatives of the same concept always clash. However, we still must avoid clashes between representatives of different concepts. A compression scheme is valid if for any sample domain, the number of concepts on the domain equals the number of concepts that have at least one representative inside the domain. There exists such a scheme with multiple representatives for the maximal class of Table 2.

Unfortunately, Table 1 presents a maximal concept class that does not have an unlabeled compression scheme of size equal the VC dimension with multiple representatives of concepts in the class. We checked that for any assignment of multiple representatives to concepts in this class, there is always some sample that cannot be compressed, that is, there is no representative of a consistent concept in the sample domain. On the other hand, it is very easy to produce an unlabeled compression scheme for this class if we let some subsets of size at most d represent hypotheses outside of the class. Note that in the example of Table 1, the no-clashing condition is satisfied not only for all concept pairs, but also for the additional hypothesis and any concept in the class.

For the class of Table 1, there also is a compression scheme that maps labeled sets of size equal d to just concepts in the class. We do not know whether such schemes exist for arbitrary finite concept classes. However, there is an infinite maximum (but not maximal) concept class of VC dimension one with the following property: there is no scheme when labeled points must represent concepts in the class, but there is a scheme when labeled points can represent hypotheses outside of the class (Eaton, 2005). Also, there is no unlabeled compression scheme for positive halfspaces in $\mathbb{R}^2$ in which the compression sets (of size at most two) represent positive halfspaces (Neylon, 2006b).

As has become apparent, there are many variations of compression schemes. We now give a unified notation for all these variations. A compression scheme for a concept class C is essentially defined by a mapping f from representatives to hypotheses which are arbitrary subsets of $\operatorname{dom}(C)$. Note that the direction of this mapping is opposite to the representation mapping r used for maximum classes.

To define the mapping f, we first choose a set R of representatives which are sets of labeled and/or unlabeled sample points in $\operatorname{dom}(C)$. The mapping f maps R to hypotheses, which are arbitrary subsets of $\operatorname{dom}(C)$. Note that f is not required to be injective, allowing for multiple representatives of the same hypothesis. The size of the scheme is the maximum size of any set in R. The mapping f produces a compression scheme as follows:

- **Compression.** For any sample s of a concept in C, consider the set of all representatives $r \in R$ that lie in s and compress to any such r for which the hypothesis $f(r)$ is consistent with the sample s.
- **Reconstruction.** Use hypothesis $f(r)$ to reconstruct the labels of the original sample s.
- **Validity.** Mapping f gives a valid compression scheme, if every sample of a concept in C can be compressed as above.

The no-clashing condition seems to be useful to construct compression schemes for maximal classes as well. However, as we shall see, many techniques for assuring this condition for maximum classes are not applicable in the more general case. Recall that our Tail Matching Algorithm used the forbidden sets of C^x to represent the concepts in $tail_x(C)$. This immediately prevented clashes between tail concepts and the rest of the class. However, in maximal classes the number of tail concepts can be larger than the number of forbidden sets of C^x. For example, in the maximal class of Figure 3 all tails have size 4, but there are only $3 = \binom{4-1}{2}$ forbidden sets for C^{x_i}.

Of course, we can consider other splits of C into two parts and use forbidden sets of one part as representatives for the other. A natural idea is to split C into a concept class C' of VC dimension one lower than C and a *shell* $C \setminus C'$ of size at most $\binom{n}{d}$, which is the number of forbidden sets when the domain size is n. For maximum classes, such splits always exist (see Lemma 12), but we do not know this for maximal classes. However, we found a particular maximal class (not shown) with a split of the above form for which there are two concepts in the shell that contain only one forbidden set and this set is the same. Thus, we have a single forbidden set available for representing two concepts, and therefore using forbidden sets as representatives does not seem to work for maximal classes.

The Min-Peeling Algorithm provided a simple way of constructing a scheme for maximum classes. For maximal classes, this algorithm fails on simple examples. Part of the problem seems to be that maximal classes have too few edges. Thus a potential idea is to add "virtual" edges to maximal classes, so that a richer set of representatives is obtained. We hope to add edges so that the representatives produced by the Min-Peeling Algorithm satisfy the no-clashing condition. Our tests along these lines were inconclusive.

We conclude this section by discussing which lemmas of the previous sections still hold for maximal classes. We have already discussed in this section how the no-clashing condition partially carries over to maximal classes. For a maximum class C, both $C - x$ and C^x are again maximum. This recursion lies at the core of many of the techniques. Maximal classes can still be split as $C = 0C^x \dot\cup 1C^x \dot\cup \mathrm{tail}_x(C)$, but now C^x and $C - x$ are not necessarily maximal and they do not have specific sizes. Our Tail Matching Algorithm relied on a further decomposition of the class $\mathrm{tail}_x(C)$ given in Lemma 6: for any concept in $\mathrm{tail}_x(C - y)$ or $\mathrm{tail}_x(C^y)$, we can extend these concepts with a y bit to concepts in $\mathrm{tail}_x(C)$. These extensions also exist for maximal classes, but we cannot get all concepts in $\mathrm{tail}_x(C)$ this way. Finally, for maximal classes, the equality of Lemma 13 becomes a subset relationship, that is, $I_C(c) \subseteq I_{C-x}(c - x)$.

9. Conclusions and Combinatorial Open Problems

The main open problem of whether there always exist ompression schemes of size at most the VC dimension still remains open. (For a general discussion of the allowable schemes see Section

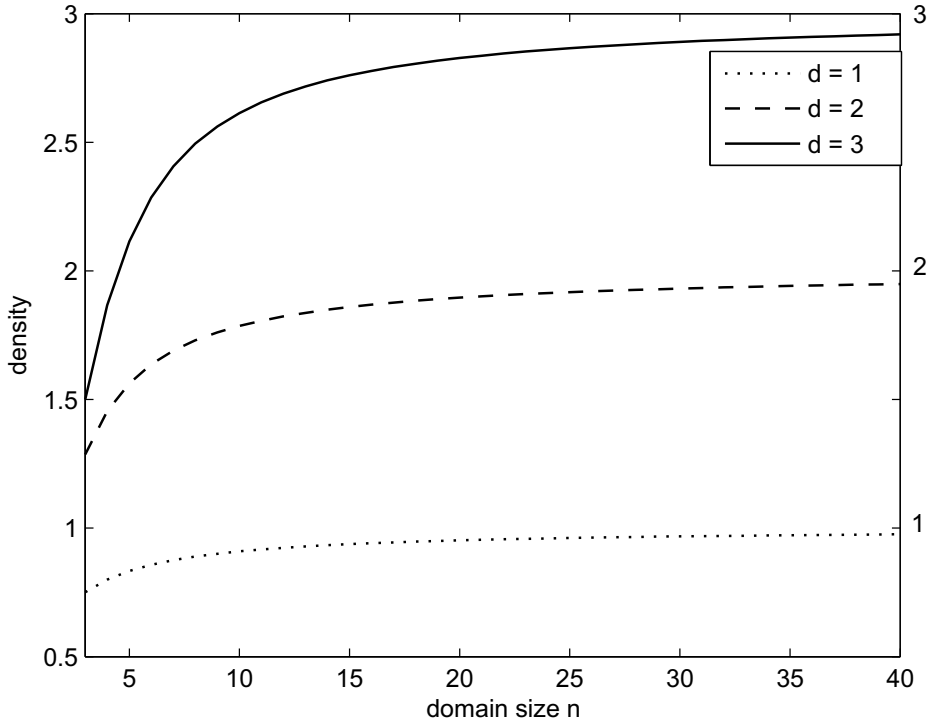


Figure 17: Density curves D_d^n (as a function of n) of the one-inclusion graphs of maximum classes with VC dimension $d = 1, 2, 3$.

8.) In this paper we gave two algorithms for constructing an unlabeled compression scheme for maximum classes. These schemes have many interesting combinatorial properties. We gave a correctness proof for the recursive Tail Matching Algorithm, however the correctness of the simpler Min-Peeling Algorithm still remains to be shown. We already gave a number of conjectures in connection with the latter algorithm: Does sweeping a linear arrangement always correspond to a run of the Min-Peeling Algorithm (Conjecture 4)? Does Lemma 17 hold for partially peeled maximum classes (Conjecture 7)? Does any one-inclusion graph of VC dimension d that results from peeling a maximum class always have a vertex of degree at most d? It is already known, that general classes can have minimum degree larger than d (Rubinstein et al., 2007a).

In our empirical tests (not shown) we actually always found at least $d+1$ vertices of degree at most d in maximum and peeled classes (instead of just one vertex). We were able to prove (not shown) that maximum classes of VC dimension d have at least one vertex of degree d. However we have not been able to prove that maximum classes have at least $d+1$ such vertices.

Notice that in connection with the last conjecture, the obvious counting arguments are off by a factor of two: since the sum of vertex degrees is equal to twice the number of edges, the density of any graph of minimum degree d has to be bigger than at least $\frac{d}{2}$ and not d. Thus there must be something particular about maximum classes and their peelings that forces them to have a low degree vertex. This is likely related to the fact that maximum classes are unions of hypercubes of dimension d.

The *density* of a graph is the ratio of the number of edges to the number of vertices. It is already known that one-inclusion graphs of VC dimension d can have density at most d (Haussler et al., 1994). For a maximum class of domain size n and VC dimension d this density can be expressed as:

$$D_d^n = \frac{n\binom{n-1}{\leq d-1}}{\binom{n}{\leq d}} = \frac{n\sum_{i=0}^{d-1}\frac{i+1}{n}\binom{n}{i+1}}{\binom{n}{\leq d}} = \frac{\sum_{i=1}^{d} i\binom{n}{i}}{\binom{n}{\leq d}} \leq \frac{d\binom{n}{\leq d}}{\binom{n}{\leq d}} = d.$$

Figure 17 plots the density curves D_d^n as a function of the domain size n for various values of the VC dimension d. These curves always start at $d/2$, which is the density of the complete hypercube and $\lim_{n\to\infty} D_d^n = d$.

We previously conjectured that maximum classes are the densest. This was recently proven in Rubinstein et al. (2007b). Specifically, they show that any one-inclusion graph of domain size n and VC dimension d has density at most D_d^n. This is an improvement on the previously known density bound.

Our constructions for unlabeled compression schemes only apply to *finite* maximum classes whereas the original labeled compression scheme for maximum classes is applicable for infinite maximum classes as well. The existence of unlabeled compression schemes for infinite maximum classes of size equal to the VC dimension does follow from the compactness property of compression schemes as shown in Ben-David and Litman (1998). That theorem is, however, a non-constructive existence result and is therefore not completely satisfactory.

One of the most important natural infinite classes is the class of positive halfspaces (halfspaces containing $(\infty, 0, \ldots, 0)$). There are labeled compression schemes for this class that reconstruct only with halfspaces (e.g., compressing to a set of essential support vectors, von Luxburg et al., 2004). An unlabeled compression scheme is also known to exist (Ben-David and Litman, 1998) (via a nonconstructive proof) but it would be interesting to find a simple constructive unlabeled compression scheme for this class. Recall that the VC dimension of positive halfspaces in $\mathbb{R}^n$ is n. For the case of $n = 1, 2$, it is easy to find unlabeled compression schemes (not shown). However for $n = 2$, it is necessary that the sets of size at most two represent hypotheses which are not halfspaces (Neylon, 2006a).[8]

Open Problem 2. *Find a constructive unlabeled compression scheme of size n for the class of positive halfspaces in* $\mathbb{R}^n$.

One of the simplest ways to obtain compression schemes for arbitrary classes would be to embed them into maximum classes and then use one of the existing algorithms.

Open Problem 3. *For any concept class C, does there always exist a maximum class of VC dimension at most a constant times larger than* $\mathrm{VCdim}(C)$ *that contains C as a subset?*

8. The construction in Neylon (2006a) requires a set of points not in general position, but this requirement can be removed (Neylon, 2006b). Moreover, it is possible to restrict the class of positive halfspaces to an everywhere dense subset of $\mathbb{R}^n$ with the property that all finite subsets of this set are in general position (Neylon, 2006b). This restriction is a natural infinite maximum class with no unlabeled compression scheme that reconstructs with halfspaces.

Acknowledgments

We thank Sally Floyd for her personal encouragement and brilliant insights, Sanjoy Dasgupta for the discussions leading to Lemma 14, and Tyler Neylon for the discussion of unlabeled compression schemes for positive halfspaces.

References

S. Ben-David and A. Litman. Combinatorial variability of Vapnik-Chervonenkis classes with applications to sample compression schemes. *Discrete Applied Mathematics*, 86:3 – 25, 1998.

F. Eaton. Private communication, 2005.

H. Edelsbrunner. *Algorithms in Combinatorial Geometry*, volume 10 of *EATCS Monographs on Theoretical Computer Science*. Springer-Verlag, Berlin, New York, 1987. ISBN 038713722X.

S. Floyd. *Space-bounded learning and the Vapnik-Chervonenkis Dimension (Ph.D)*. PhD thesis, U.C. Berkeley, December 1989. ICSI Tech Report TR-89-061.

S. Floyd and M. K. Warmuth. Sample compression, learnability, and the Vapnik-Chervonenkis dimension. *Machine Learning*, 21(3):269–304, 1995.

L. Gurvits. Linear algebraic proofs of VC-dimension based inequalities. In Shai Ben-David, editor, *EuroCOLT '97, Jerusalem, Israel, March 1997*, pages 238–250. Springer Verlag, March 1997.

D. Haussler, N. Littlestone, and M. K. Warmuth. Predicting $\{0,1\}$ functions on randomly drawn points. *Inform. Comput.*, 115(2):248–292, 1994.

D. Helmbold, R. Sloan, and M. K. Warmuth. Learning integer lattices. *SIAM J. Comput.*, 21(2): 240–266, 1992.

J. Langford. Tutorial on practical prediction theory for classification. *Journal of Machine Learning Research*, 6:273–306, 2005.

Y. Li, P. M. Long, and A. Srinivasan. The one-inclusion graph algorithm is near optimal for the prediction model of learning. *Transaction on Information Theory*, 47(3):1257–1261, 2002.

N. Littlestone and M. K. Warmuth. Relating data compression and learnability. Unpublished manuscript, obtainable at http://www.cse.ucsc.edu/~manfred/pubs/T1.pdf, June 10 1986.

M. Marchand and J. Shawe-Taylor. The Decision List Machine. In *Advances in Neural Information Processing Systems 15*, pages 921–928. MIT-Press, Cambridge, MA, USA, 2003.

M. Marchand and J. Shawe-Taylor. The Set Covering Machine. *Journal of Machine Learning Research*, 3:723–746, 2002.

T. Neylon. *Sparse Solutions to Linear Prediction Problems*. PhD thesis, New York University, Courant Institute of Mathematical Sciences, May 2006a.

T. Neylon. Private communication, 2006b.

B.I.P. Rubinstein, P. Bartlett, and J. H. Rubinstein. Shifting: One-inclusion mistake bounds and sample compression. Technical Report UCB/EECS-2007-86, EECS Department, University of California, Berkeley, Jun 2007a. URL http://www.eecs.berkeley.edu/Pubs/TechRpts/2007/EECS-2007-86.html.

B.I.P. Rubinstein, P. L. Bartlett, and J. H. Rubinstein. Shifting, one-inclusion mistake bounds and tight multiclass expected risk bounds. In B. Schölkopf, J. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems 19*, pages 1193–1200. MIT Press, Cambridge, MA, 2007b.

N. Sauer. On the density of families of sets. *Journal of Combinatorial Theory (A)*, 13:145–147, 1972.

V. N. Vapnik. *Estimation of Dependences Based on Empirical Data*. Springer-Verlag, New York, 1982.

V. N. Vapnik and A. Y. Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probab. and its Applications*, 16(2):264–280, 1971.

U. von Luxburg, O. Bousquet, and B. Schölkopf. A compression approach to support vector model selection. *Journal of Machine Learning Research*, 5:293–323, April 2004.

M. K. Warmuth. Compressing to VC dimension many points. In *Proceedings of the 16th Annual Conference on Learning Theory (COLT 03)*, Washington D.C., USA, August 2003. Springer. Open problem.

M. K. Warmuth. The optimal PAC algorithm. In *Proceedings of the 17th Annual Conference on Learning Theory (COLT 04)*, Banff, Canada, July 2004. Springer. Open problem.

E. Welzl. Complete range spaces. Unpublished notes, 1987.

Journal of Machine Learning Research 8 (2007) 2083-2120 Submitted 5/07; Revised 8/07; Published 9/07

Refinable Kernels*

Yuesheng Xu YXU06@SYR.EDU
Haizhang Zhang HZHANG12@SYR.EDU
Department of Mathematics
Syracuse University
Syracuse, NY 13244, USA

Editor: Bernhard Schölkopf

Abstract

Motivated by mathematical learning from training data, we introduce the notion of *refinable kernels*. Various characterizations of refinable kernels are presented. The concept of refinable kernels leads to the introduction of *wavelet-like reproducing kernels*. We also investigate a refinable kernel that forms a Riesz basis. In particular, we characterize refinable translation invariant kernels, and refinable kernels defined by refinable functions. This study leads to multiresolution analysis of reproducing kernel Hilbert spaces.

Keywords: refinable kernels, refinable feature maps, wavelet-like reproducing kernels, dual kernels, learning with kernels, reproducing kernel Hilbert spaces, Riesz bases

1. Introduction

The main purpose of this paper is to introduce the notion of refinable kernels, wavelet-like reproducing kernels and multiresolution analysis of a reproducing kernel Hilbert space. Before proceeding to the motivation, it is worthwhile to know that there has been a large body of literature on similar notions such as refinable functions (Cavaretta et al., 1991; Daubechies, 1992), multiresolution analysis of $L^2(\mathbb{R})$ (Mallat, 1989; Meyer, 1992) and kernels constructed by wavelet functions (Amato et al., 2006; Rakotomamonjy and Canu, 2005; Rakotomamonjy et al., 2005). The connection of these well-known notions with those to be presented will become clear as we proceed this study.

We first motivate the concept of refinable kernels by learning via a kernel. Let X be a prescribed set which is called in the theory of learning an input space and is associated with an output space $Y \subseteq \mathbb{C}$. A typical learning task aims at inferring from a finite set of training data $\mathbf{z} := \{(x_j, y_j) : j \in \mathbb{N}_m\}$, where $\mathbb{N}_m := \{1, 2, \ldots, m\}$, a function f from X to Y so that $f(x)$ gives a satisfactory output of an input $x \in X$. A popular choice of f is a minimizer of a certain error functional. Specifically, we let $\mathcal{H}$ be a given class of functions on X, $Q : \mathbb{C} \times \mathbb{C} \to \mathbb{R}_+$ be a *loss function* (Schölkopf and Smola, 2002) measuring how well g fits the training data $\mathbf{z}$, $\mathcal{N} : \mathcal{H} \to \mathbb{R}_+$ be a controller of the set of functions in $\mathcal{H}$ from which we choose f, and μ be a positive regularization parameter. The function f may be chosen as

$$\arg\min_{g \in \mathcal{H}} \sum_{j \in \mathbb{N}_m} Q(g(x_j), y_j) + \mu \mathcal{N}(g). \tag{1}$$

*. Dedicated to Dr. Charles Micchelli's 65th birthday for friendship and esteem.

The selection of the function class $\mathcal{H}$ is critical for the behavior of f and thus it deserves special attention. In practice, $\mathcal{H}$ may be chosen through a *kernel* K on X, a function from $X \times X$ to $\mathbb{C}$ such that for all finite sets $\mathbf{t} := \{t_j : j \in \mathbb{N}_n\} \subseteq X$ the matrix

$$K[\mathbf{t}] := [K(t_j, t_k) : j,k \in \mathbb{N}_n] \tag{2}$$

is hermitian and positive semi-definite (see, for example, Cucker and Smale, 2002; Schölkopf and Smola, 2002; Shawe-Taylor and Cristianini, 2004; Vapnik, 1998). The importance of kernels in learning is that the function evaluation $K(x,y)$ is able to measure the similarity of $x,y \in X$. A kernel K on X corresponds to a Hilbert space

$$\mathcal{H}_K := \overline{\operatorname{span}}\{K(\cdot,y) : y \in X\} \tag{3}$$

of functions on X with an inner product determined by

$$(K(\cdot,y), K(\cdot,x))_{\mathcal{H}_K} = K(x,y), \quad x,y \in X. \tag{4}$$

The space $\mathcal{H}_K$ is a *reproducing kernel Hilbert space* (RKHS), that is, point evaluations are continuous linear functionals on $\mathcal{H}_K$ (Aronszajn, 1950). Moreover, $\mathcal{H}_K$ is the only Hilbert space of functions on X such that for all $x \in X$, we have that $K(\cdot,x) \in \mathcal{H}_K$ and

$$f(x) = (f, K(\cdot,x))_{\mathcal{H}_K}, \quad f \in \mathcal{H}_K. \tag{5}$$

Due to Equation (5), K is often interpreted as the *reproducing kernel* of $\mathcal{H}_K$. A RKHS has exactly one reproducing kernel (Aronszajn, 1950). To construct a learning function $f : X \to Y$ from the training data $\mathbf{z}$, we start with a kernel K on X. Choose in (1) $\mathcal{H} := \mathcal{H}_K$ and $\mathcal{N} := \|\cdot\|^2_{\mathcal{H}_K}$, the square of the norm on $\mathcal{H}_K$. In this case, the minimization problem in (1) reduces to a regularization in the RKHS, which has received much attention in the literature (see, for example, Bousquet and Elisseeff, 2002; Cucker and Smale, 2002; Micchelli and Pontil, 2005a,b; Mukherjee et al., 2006; Schölkopf and Smola, 2002; Smale and Zhou, 2003; Steinwart and Scovel, 2005; Vapnik, 1998; Wahba, 1999; Walder et al., 2006; Ying and Zhou, 2007; Zhang, 2004, and the references cited therein). In this setting, the *representer theorem* in learning (see, for example, Kimeldorf and Wahba, 1971; Micchelli and Pontil, 2004; Schölkopf et al., 2001; Schölkopf and Smola, 2002; Shawe-Taylor and Cristianini, 2004; Walder et al., 2006) asserts that there exists $\mathbf{c} := [c_j : j \in \mathbb{N}_m]^T \subseteq \mathbb{C}^m$, depending on the training data $\mathbf{z}$ and the kernel K, such that the minimizer (1) is

$$f = \sum_{j \in \mathbb{N}_m} c_j K(\cdot, x_j). \tag{6}$$

The choice of kernels K is certainly one of the most important issues in the above learning scheme via regularization. It is often based on the training data $\mathbf{z}$ currently available to us. However, the old training data may be updated to $\mathbf{z}' := \{(x'_j, y'_j) : j \in \mathbb{N}_{m'}\}$ by adding to $\mathbf{z}$ more new samples from $X \times Y$. The kernels that we use should offer us a convenient way to update the kernel. In other words, we are looking for kernels with the ability of learning dynamically expanding training data. Specifically, we demand kernels K having the feature that there is a cheap way of updating K to a new kernel K' such that $\mathcal{H}_K \preceq \mathcal{H}_{K'}$. Here and throughout the paper, we make the convention that whenever we write $\mathcal{W}_1 \preceq \mathcal{W}_2$ for Hilbert spaces $\mathcal{W}_1, \mathcal{W}_2$, the inclusion is in the sense that $\mathcal{W}_1 \subseteq \mathcal{W}_2$

and for all $u,v \in \mathcal{W}_1$, $(u,v)_{\mathcal{W}_1} = (u,v)_{\mathcal{W}_2}$. The inclusion $\mathcal{H}_K \preceq \mathcal{H}_{K'}$ has a natural interpretation. When we have a larger training data set $\mathbf{z}'$, we may expect that the minimization

$$\min_{g\in\mathcal{H}_{K'}} \sum_{j\in\mathbb{N}_{m'}} Q(g(x'_j),y'_j) + \mu' \|g\|^2_{\mathcal{H}_{K'}}, \tag{7}$$

yields a better predictor f' than f. This becomes possible only if the new space $\mathcal{H}_{K'}$ includes $\mathcal{H}_K$ as a subspace. Moreover, we require the updating from K to K' to have the feature that computing minimizer f' from (7) should be able to make use of the previously computed minimizer f from (1). Moreover, when the representation (6) for f is available, we want to process it efficiently.

This motivates us to introduce the concept of refinable kernels. We shall study characterizations of a refinable kernel, fundamental properties of refinable kernels and wavelet-like reproducing kernels, and multiscale structures of a RKHS induced by refinable kernels. It is important to note that the concept of wavelet-like reproducing kernels differs from that of "wavelet kernels" in Amato et al. (2006), Rakotomamonjy and Canu (2005), and Rakotomamonjy et al. (2005). The earlier means the kernels defined by the difference of kernels at two consecutive scales while the latter means the kernels defined by a linear combination of dilations and translations of wavelet functions. This paper is organized in seven sections. We present in Section 2 two characterizations of refinable kernels. Section 3 is devoted to wavelet-like reproducing kernels and a multiscale decomposition of the RKHS of a refinable kernel. In Section 4, we investigate refinable kernels of a Riesz type and we also introduce the notion of a multiresolution analysis for a RKHS. As concrete examples of refinable kernels, we formulate in Sections 5 and 6, respectively, conditions for translation invariant kernels and kernels defined by a refinable function to be refinable. In Section 7, we have a brief discussion of potential applications of refinable kernels and make a conclusion.

2. Characterizations of γ-Refinable Kernels

We define in this section γ-refinable kernels and present their characterizations. Let $\gamma : X \to X$ be a given bijective mapping and let ι denote the identity mapping from X to itself. We introduce a sequence of mappings from X to itself by recursions

$$\gamma_{-1} := \gamma^{-1}, \ \ \gamma_{-n-1} := \gamma^{-1} \circ \gamma_{-n}, \ \text{ and } \ \gamma_0 := \iota, \ \ \gamma_n := \gamma \circ \gamma_{n-1}, \ \ n \in \mathbb{N}.$$

A kernel K on X is called γ-*refinable* if there exists a positive constant λ depending only on K and γ such that

$$\mathcal{H}_K \preceq \mathcal{H}_{K_1},$$

where

$$K_1(x,y) := \lambda K(\gamma(x),\gamma(y)), \ \ x,y \in X.$$

The constant λ is a normalization factor which ensures the inner product on $\mathcal{H}_K$ identical to that on $\mathcal{H}_{K_1}$. Examples of the mappings γ include the dilation mapping $x \to 2x$ in $\mathbb{R}^d$ and in general, the dilation mapping $x \to Ax$ where A is an expanding matrix. We simply call the γ-refinable kernels in this case *refinable kernels*.

Let K be a kernel on the input space X, and λ a positive constant. For any bijective mapping γ and any $n \in \mathbb{Z}$, we let

$$K_n(x,y) := \lambda^n K(\gamma_n(x),\gamma_n(y)), \ \ x,y \in X. \tag{8}$$

We next identify the RKHS defined by K_n.

Theorem 1 *If K is a kernel on X, then for each $n \in \mathbb{Z}$, the RKHS of kernel K_n is*

$$\mathcal{H}_{K_n} = \{f \circ \gamma_n : f \in \mathcal{H}_K\} \tag{9}$$

with inner product

$$(f,g)_{\mathcal{H}_{K_n}} := \lambda^{-n}(f \circ \gamma_{-n}, g \circ \gamma_{-n})_{\mathcal{H}_K}, \;\; f,g \in \mathcal{H}_{K_n}. \tag{10}$$

Proof It can be shown that K_n is a kernel on X. We set $\mathbb{H}_n := \{f \circ \gamma_n : f \in \mathcal{H}_K\}$ and introduce an inner product on $\mathbb{H}_n$ by

$$(f,g)_{\mathbb{H}_n} := \lambda^{-n}(f \circ \gamma_{-n}, g \circ \gamma_{-n})_{\mathcal{H}_K}, \;\; f,g \in \mathbb{H}_n.$$

It is clear that $\mathbb{H}_n$ is a Hilbert space with this inner product. Note that for any $x \in X$, $K_n(\cdot,x) \in \mathbb{H}_n$ and for any $f \in \mathbb{H}_n$, $f \circ \gamma_{-n} \in \mathcal{H}_K$. Hence, for $f \in \mathbb{H}_n$, we obtain by (5) and the definition of the inner product $(\cdot,\cdot)_{\mathcal{H}_{K_n}}$ that for $x \in X$

$$f(x) = (f \circ \gamma_{-n})(\gamma_n(x)) = (f \circ \gamma_{-n}, K(\cdot, \gamma_n(x)))_{\mathcal{H}_K} = \lambda^n (f, K(\gamma_n(\cdot), \gamma_n(x))_{\mathbb{H}_n}.$$

Combining this equation with the definition of the kernel K_n leads to $f(x) = (f, K_n(\cdot,x))_{\mathbb{H}_n}$, $x \in X$. This implies that K_n is the reproducing kernel for $\mathbb{H}_n$. By the unique correspondence between a RKHS and its reproducing kernel, we conclude that $\mathcal{H}_{K_n} = \mathbb{H}_n$. ■

A direct consequence of Theorem 1 is that if K is a γ-refinable kernel then for each $n \in \mathbb{Z}$, the kernel K_n is γ-refinable. This result justifies the usage of the same mapping γ to update K_n to K_{n+1} in (8) for each $n \in \mathbb{Z}$.

Proposition 2 *If the kernel K is γ-refinable, then for each $n \in \mathbb{Z}$, K_n is γ-refinable. Conversely, if for some $n \in \mathbb{Z}$, K_n is γ-refinable, then K is γ-refinable.*

Proof Suppose that K is γ-refinable. Then, we have that $\mathcal{H}_K \preceq \mathcal{H}_{K_1}$. Let $f \in \mathcal{H}_{K_n}$. By Theorem 1, we deduce that $f \circ \gamma_{-n} \in \mathcal{H}_K \preceq \mathcal{H}_{K_1}$, which implies that $f \in \mathcal{H}_{K_{n+1}}$. In addition, Theorem 1, Equation (10) and the γ-refinability of K ensure that for $f,g \in \mathcal{H}_{K_n}$,

$$\begin{aligned} (f,g)_{\mathcal{H}_{K_n}} &= \lambda^{-n}(f \circ \gamma_{-n}, g \circ \gamma_{-n})_{\mathcal{H}_K} = \lambda^{-n}(f \circ \gamma_{-n}, g \circ \gamma_{-n})_{\mathcal{H}_{K_1}} \\ &= \lambda^{-n-1}(f \circ \gamma_{-n-1}, g \circ \gamma_{-n-1})_{\mathcal{H}_K} = (f,g)_{\mathcal{H}_{K_{n+1}}}. \end{aligned}$$

This confirms that K_n is γ-refinable.

Conversely, we suppose that K_n is γ-refinable for some $n \in \mathbb{Z}$. Since

$$K(x,y) = \lambda^{-n} K_n(\gamma_{-n}(x), \gamma_{-n}(y)), \;\; x,y \in X,$$

the arguments in the proof of the first statement of this proposition show that K is γ-refinable. ■

The next result follows immediately from the last proposition.

Corollary 3 *If K is γ-refinable, then for all $f,g \in \mathcal{H}_K$ and $n \in \mathbb{N}$, $f,g \in \mathcal{H}_{K_n}$ and $(f,g)_{\mathcal{H}_{K_n}} = (f,g)_{\mathcal{H}_K}$.*

We now present our first characterization of γ-refinable kernels.

Theorem 4 *A kernel K is γ-refinable if and only if*

$$K(\gamma_{-1}(\cdot),x) \in \mathcal{H}_K, \ \text{for all } x \in X \tag{11}$$

and

$$(K(\gamma_{-1}(\cdot),y),K(\gamma_{-1}(\cdot),x))_{\mathcal{H}_K} = \lambda K(x,y), \ \text{for all } x,y \in X, \tag{12}$$

where λ is the same constant in the definition of a γ-refinable kernel.

Proof By Proposition 2, K is γ-refinable if and only if

$$\mathcal{H}_{K_{-1}} \preceq \mathcal{H}_K. \tag{13}$$

Suppose that K is γ-refinable. The definition of kernel K_{-1} leads to

$$K(\gamma_{-1}(\cdot),x) = \lambda K_{-1}(\cdot,\gamma(x)) \in \mathcal{H}_{K_{-1}}, \ \text{for } \ x \in X,$$

for some constant λ. This combined with relation (13) ensures the validity of (11). By (13), (10) and (4), we obtain for all $x,y \in X$ that

$$\begin{aligned}(K(\gamma_{-1}(\cdot),y),K(\gamma_{-1}(\cdot),x))_{\mathcal{H}_K} &= (K(\gamma_{-1}(\cdot),y),K(\gamma_{-1}(\cdot),x))_{\mathcal{H}_{K_{-1}}} \\ &= \lambda(K(\cdot,y),K(\cdot,x))_{\mathcal{H}_K} = \lambda K(x,y),\end{aligned}$$

which is Equation (12).

Conversely, we suppose that (11) holds and (12) is satisfied with some constant λ, and we prove that inclusion relation (13) is valid. Let $f \in \mathcal{H}_K$. By (3), there exists a sequence

$$f_n \in \operatorname{span}\{K(\cdot,y) : y \in X\}, \ \ n \in \mathbb{N}$$

that converges to f in $\mathcal{H}_K$. Equation (11) implies that $f_n \circ \gamma_{-1} \in \mathcal{H}_K$, $n \in \mathbb{N}$, and Equation (12) implies for all $m,n \in \mathbb{N}$ that

$$\|f_m \circ \gamma_{-1} - f_n \circ \gamma_{-1}\|_{\mathcal{H}_K} = \lambda^{1/2}\|f_m - f_n\|_{\mathcal{H}_K}.$$

Therefore, $f_n \circ \gamma_{-1}$ is a Cauchy sequence in $\mathcal{H}_K$, whose limit is denoted by f_{-1}. Recall that point evaluations are continuous linear functionals on $\mathcal{H}_K$. An application of this fact yields that

$$f_{-1}(x) = \lim_{n\to\infty}(f_n \circ \gamma_{-1})(x), \ \ x \in X. \tag{14}$$

In the same manner, since f_n converges to f in $\mathcal{H}_K$, we have for each $x \in X$ that

$$(f \circ \gamma_{-1})(x) = \lim_{n\to\infty}(f_n \circ \gamma_{-1})(x). \tag{15}$$

We observe from (14) and (15) that $f \circ \gamma_{-1} = f_{-1}$. It is hence proved that $f \circ \gamma_{-1} \in \mathcal{H}_K$ for each $f \in \mathcal{H}_K$. This combined with (9) shows that the elements of $\mathcal{H}_{K_{-1}}$ are contained in $\mathcal{H}_K$. Finally, we verify by (12) for each $f,g \in \mathcal{H}_K$ that

$$(f \circ \gamma_{-1}, g \circ \gamma_{-1})_{\mathcal{H}_K} = (f_{-1},g_{-1})_{\mathcal{H}_K} = \lim_{n\to\infty}(f_n \circ \gamma_{-1}, g_n \circ \gamma_{-1})_{\mathcal{H}_K} = \lambda \lim_{n\to\infty}(f_n,g_n)_{\mathcal{H}_K} = \lambda(f,g)_{\mathcal{H}_K}.$$

By the above equation and (10), the inner product on $\mathcal{H}_{K_{-1}}$ coincides with the one on $\mathcal{H}_K$. We conclude that (13) holds true and complete the proof. ■

Another characterization of γ-refinable kernels K is in terms of a *feature map* for K. A function Φ from X to a Hilbert space $\mathcal{W}$ is called a feature map for the kernel K if

$$K(x,y) = (\Phi(x), \Phi(y))_{\mathcal{W}}, \;\; x,y \in X. \tag{16}$$

We call the Hilbert space $\mathcal{W}$ the feature space of K. It is known (Aronszajn, 1950) that K is a kernel on X if and only if there exists a map $\Phi : X \to \mathcal{W}$ satisfying (16). In the next result, we identify the RKHS $\mathcal{H}_K$ in terms of a feature map Φ for K. To state the result, we denote by $\Phi(X)$ the image of X under Φ, $\overline{\text{span}}\,\Phi(X)$ the closure of $\text{span}\,\Phi(X)$ in $\mathcal{W}$, and P_Φ the orthogonal projection from $\mathcal{W}$ onto $\overline{\text{span}}\,\Phi(X)$.

Lemma 5 *Let K be a kernel having a representation (16) in terms of a feature map Φ from X to $\mathcal{W}$. Then $\mathcal{H}_K = \{(\Phi(\cdot), u)_{\mathcal{W}} : u \in \mathcal{W}\}$ with inner product*

$$((\Phi(\cdot), u)_{\mathcal{W}}, (\Phi(\cdot), v)_{\mathcal{W}})_{\mathcal{H}_K} = (P_\Phi v, P_\Phi u)_{\mathcal{W}}, \;\; u, v \in \mathcal{W}. \tag{17}$$

A proof of this result in the special case that K is a *Hilbert-Schmidt kernel* was provided in Opfer (2006) (see Lemma 3.4, Theorem 3.5 therein). The proof works for the general case described in Lemma 5. The result can also be found in Micchelli and Pontil (2005a). In the application of Lemma 5, it is always convenient to assume that there holds

$$\overline{\text{span}}\,\Phi(X) = \mathcal{W} \tag{18}$$

since otherwise $\mathcal{W}$ can be replaced by $\overline{\text{span}}\,\Phi(X)$. If (18) holds then for each $f \in \mathcal{H}_K$ there exists a unique $u_f \in \mathcal{W}$ such that $f = (\Phi(\cdot), u_f)_{\mathcal{W}}$. Moreover, one can see by Lemma 5 that the linear transformation Γ from $\mathcal{H}_K$ to $\mathcal{W}$ defined by

$$\Gamma f := u_f \tag{19}$$

is an *isomorphism*, that is, it is one-to-one, onto and satisfies $\|\Gamma f\|_{\mathcal{W}} = \|f\|_{\mathcal{H}_K}$, for $f \in \mathcal{H}_K$.

We call a feature map Φ from X to $\mathcal{W}$ γ-*refinable* provided that there is a bounded linear operator T on $\mathcal{W}$ such that

$$\lambda^{-1/2} \Phi \circ \gamma_{-1} = T\Phi, \tag{20}$$

where $\lambda^{-1/2}$ plays the role of a normalization parameter. Throughout this paper, we mean that T is a function from a Hilbert space $\mathcal{W}$ to itself whenever we say that T is an operator on $\mathcal{W}$. Recall that a linear operator A on $\mathcal{W}$ is *isometric* if for all $u \in \mathcal{W}$, $\|Au\|_{\mathcal{W}} = \|u\|_{\mathcal{W}}$. One can see that A is isometric if and only if A^*A is equal to the identity operator on $\mathcal{W}$, where A^* denotes the *adjoint operator* of A.

We characterize a refinable kernel in terms of its feature map.

Theorem 6 *Suppose that K is a kernel on X with a feature map $\Phi : X \to \mathcal{W}$ satisfying (18). Then K is γ-refinable if and only if Φ is γ-refinable and the adjoint operator T^* of T in (20) is isometric.*

Proof Suppose that Φ is γ-refinable, that is, it satisfies (20) for some bounded operator T on $\mathcal{W}$, and suppose that T^* is isometric. We first observe by (16) and (20) for each $x \in X$ that

$$K(\gamma_{-1}(\cdot),x) = (\Phi \circ \gamma_{-1}(\cdot), \Phi(x))_{\mathcal{W}} = \lambda^{1/2}(T\Phi(\cdot), \Phi(x))_{\mathcal{W}} = \lambda^{1/2}(\Phi(\cdot), T^*\Phi(x))_{\mathcal{W}}. \tag{21}$$

Lemma 5 with the equation above yields that for each $x \in X$, $K(\gamma_{-1}(\cdot),x) \in \mathcal{H}_K$. Moreover, (18) implies that P_Φ is the identity operator on $\mathcal{W}$. This fact, together with Equations (21) and (17) ensures for all $x, y \in X$ that

$$(K(\gamma_{-1}(\cdot),y), K(\gamma_{-1}(\cdot),x))_{\mathcal{H}_K} = \lambda((\Phi(\cdot), T^*\Phi(y))_{\mathcal{W}}, (\Phi(\cdot), T^*\Phi(x))_{\mathcal{W}})_{\mathcal{H}_K} = \lambda(T^*\Phi(x), T^*\Phi(y))_{\mathcal{W}}.$$

By hypothesis, TT^* is the identity. Hence, the right hand side of the above equation becomes $\lambda(\Phi(x), \Phi(y))_{\mathcal{W}}$, which is equal to $\lambda K(x,y)$, since Φ is a feature map for K. That is, (12) holds. We conclude by Theorem 4 that K is γ-refinable.

Conversely, suppose that K is γ-refinable, that is, $\mathcal{H}_{K_{-1}} \preceq \mathcal{H}_K$. We shall choose a bounded linear operator T on $\mathcal{W}$ such that Φ satisfies (20) and T^* is isometric. By Theorem 1 and Lemma 5, functions in $\mathcal{H}_{K_{-1}}$ have the form $(\Phi \circ \gamma_{-1}(\cdot), u)_{\mathcal{W}}$, $u \in \mathcal{W}$. The inclusion $\mathcal{H}_{K_{-1}} \preceq \mathcal{H}_K$ implies that for each $u \in \mathcal{W}$ there exists $v_u \in \mathcal{W}$ such that

$$\lambda^{-1/2}(\Phi \circ \gamma_{-1}(\cdot), u)_{\mathcal{W}} = (\Phi(\cdot), v_u)_{\mathcal{W}}. \tag{22}$$

Equation (18) ensures that for each $u \in \mathcal{W}$ there is a unique $v_u \in \mathcal{W}$ satisfying (22). Let A denote the map $u \to v_u$ and observe that A is a linear operator on $\mathcal{W}$. We shall prove that it is isometric. Since by (16) for all $x, y \in X$

$$K_{-1}(x,y) = \lambda^{-1}K(\gamma_{-1}(x), \gamma_{-1}(y)) = \lambda^{-1}(\Phi(\gamma_{-1}(x)), \Phi(\gamma_{-1}(y)))_{\mathcal{W}},$$

the map $\Phi_{-1} := \lambda^{-1/2}\Phi \circ \gamma_{-1} : X \to \mathcal{W}$ is a feature map for K_{-1}. Since γ is a bijective map from X to itself, $P_{\Phi_{-1}}$ is also equal to the identity operator on $\mathcal{W}$. Therefore, by Lemma 5, we have for all $u \in \mathcal{W}$ that

$$\left\| \lambda^{-1/2}(\Phi \circ \gamma_{-1}(\cdot), u)_{\mathcal{W}} \right\|_{\mathcal{H}_{K_{-1}}} = \|u\|_{\mathcal{W}}. \tag{23}$$

Likewise, condition (18) and Lemma 5 imply that

$$\|(\Phi(\cdot), v_u)_{\mathcal{W}}\|_{\mathcal{H}_K} = \|v_u\|_{\mathcal{W}}. \tag{24}$$

In addition, by the relation $\mathcal{H}_{K_{-1}} \preceq \mathcal{H}_K$ and (22), there holds

$$\left\| \lambda^{-1/2}(\Phi \circ \gamma_{-1}(\cdot), u)_{\mathcal{W}} \right\|_{\mathcal{H}_{K_{-1}}} = \|(\Phi(\cdot), v_u)_{\mathcal{W}}\|_{\mathcal{H}_K}. \tag{25}$$

Combining Equations (23), (24) and (25) shows that A is isometric. By (22) we conclude for all $u \in \mathcal{W}$ that

$$\lambda^{-1/2}(\Phi \circ \gamma_{-1}(\cdot), u)_{\mathcal{W}} = (\Phi(\cdot), Au)_{\mathcal{W}} = (A^*\Phi(\cdot), u)_{\mathcal{W}}.$$

We choose $T := A^*$ and observe from the above equation that (20) holds. Thus, Φ is γ-refinable and T^* is isometric. ■

3. Wavelet-like Reproducing Kernels

This section is devoted to developing a multiscale decomposition of the RKHS $\mathcal{H}_K$ of a γ-refinable kernel K. Specifically, we construct the nontrivial orthogonal complement of $\mathcal{H}_{K_n}$ in $\mathcal{H}_{K_{n+1}}$. In this regard, an issue important to us is when $\mathcal{H}_{K_n}$ is a proper subspace of $\mathcal{H}_{K_{n+1}}$. Our first result concerns this proper inclusion question. Let $\mathcal{R}(A)$ and $\mathcal{N}(A)$ denote the range and null space of an operator A on $\mathcal{W}$, respectively. We also denote for every $\mathcal{V} \subseteq \mathcal{W}$ by $\mathcal{V}^\perp$ the set of all elements in $\mathcal{W}$ that are orthogonal to $\mathcal{V}$.

Theorem 7 *Suppose that K defined by (16) is γ-refinable and the feature map Φ satisfies (18). Then $\mathcal{H}_{K_{-1}}$ is a proper subspace of $\mathcal{H}_K$ if and only if the operator T in (20) is not injective. Moreover, if $\mathcal{H}_{K_{-1}}$ is a proper subspace of $\mathcal{H}_K$, then for all $n \in \mathbb{Z}$, $\mathcal{H}_{K_n}$ is a proper subspace of $\mathcal{H}_{K_{n+1}}$.*

Proof Since K is γ-refinable, by Theorem 6, the feature map Φ is γ-refinable and T^* is isometric. Hence, by (20), functions in $\mathcal{H}_{K_{-1}}$ are of the form

$$\lambda^{-1/2}(\Phi \circ \gamma_{-1}(\cdot), u)_{\mathcal{W}} = (T\Phi(\cdot), u)_{\mathcal{W}} = (\Phi(\cdot), T^* u)_{\mathcal{W}}, \ \ u \in \mathcal{W}. \tag{26}$$

On the other hand, Lemma 5 ensures that functions in $\mathcal{H}_K$ have the form

$$(\Phi(\cdot), u)_{\mathcal{W}}, \ \ u \in \mathcal{W}. \tag{27}$$

The isometry of T^* guarantees that $\mathcal{R}(T^*)$ is a closed subspace of $\mathcal{W}$. This fact, together with Equations (26), (27) and the isomorphism Γ introduced in (19), implies that $\mathcal{H}_{K_{-1}}$ is a proper subspace of $\mathcal{H}_K$ if and only if $\mathcal{R}(T^*)$ is a proper subspace of $\mathcal{W}$. The relation $\mathcal{R}(T^*)^\perp = \mathcal{N}(T)$ (see, Conway, 1990, page 35) proves that $\mathcal{H}_{K_{-1}}$ is a proper subspace of $\mathcal{H}_K$ if and only if $\mathcal{N}(T) \neq \{0\}$. Hence, the first claim of this theorem is valid. The proof of the second statement is straightforward. ■

Theorem 7 allows us to construct the nontrivial orthogonal complement of $\mathcal{H}_{K_n}$ in $\mathcal{H}_{K_{n+1}}$. For this purpose, we define

$$G := K_1 - K, \ \text{ and } \ G_n(x,y) := \lambda^n G(\gamma_n(x), \gamma_n(y)), \ \ x, y \in X, \ \ n \in \mathbb{Z}.$$

Theorem 8 *Suppose that K is a γ-refinable kernel on X. Then the following statements hold:*

(1) For each $n \in \mathbb{Z}$, G_n is a kernel on X.

(2) There holds $\mathcal{H}_{G_n} \preceq \mathcal{H}_{K_{n+1}}$, and $\mathcal{H}_{G_n}$ is the orthogonal complement of $\mathcal{H}_{K_n}$ in $\mathcal{H}_{K_{n+1}}$.

(3) For each $n \in \mathbb{Z}$ that

$$\mathcal{H}_{G_n} = \{f \circ \gamma_n : f \in \mathcal{H}_G\} \tag{28}$$

and the inner product on $\mathcal{H}_{G_n}$ satisfies

$$(f,g)_{\mathcal{H}_{G_n}} = \lambda^{-n}(f \circ \gamma_{-n}, g \circ \gamma_{-n})_{\mathcal{H}_G}, \ \ f, g \in \mathcal{H}_{G_n}. \tag{29}$$

Proof Since K is γ-refinable, we have by Proposition 2 that K_n is γ-refinable, namely, $\mathcal{H}_{K_n} \preceq \mathcal{H}_{K_{n+1}}$. Let $\mathcal{W}_n$ be the orthogonal complement of $\mathcal{H}_{K_n}$ in $\mathcal{H}_{K_{n+1}}$. It is clear that $\mathcal{W}_n$ is a RKHS with the inner product of $\mathcal{H}_{K_{n+1}}$. By a property of reproducing kernels (see, Aronszajn, 1950, page 345), the sum of K_n and the kernel of $\mathcal{W}_n$ is equal to K_{n+1}. Therefore, G is the kernel of $\mathcal{W}_0$, and it is observed

by (8) that the kernel of $\mathcal{W}_n$ is G_n. This proves (1) and (2). The result (3) follows directly from Theorem 1. ■

A direct consequence of Theorem 8 (2) is that for all $n \in \mathbb{Z}$ and for all $f, g \in \mathcal{H}_{G_n}$,

$$(f,g)_{\mathcal{H}_{G_n}} = (f,g)_{\mathcal{H}_{K_{n+1}}}.$$

Theorem 8 leads to the decomposition

$$\mathcal{H}_{K_{n+1}} = \mathcal{H}_{K_n} \oplus \mathcal{H}_{G_n},$$

where the notation $A \oplus B$ denotes the orthogonal direct sum of A and B. We call G_n the *wavelet-like reproducing kernels* and in particular, G the initial wavelet-like kernel. It is clear that the initial wavelet-like kernel G is nontrivial if and only if $\mathcal{H}_{K_{-1}}$ is a proper subspace of $\mathcal{H}_K$. Repeatedly using the above decomposition with $n = -1$, we have the decomposition for the RKHS

$$\mathcal{H}_K = \mathcal{H}_{G_{-1}} \oplus \cdots \oplus \mathcal{H}_{G_{-m}} \oplus \mathcal{H}_{K_{-m}}, \ \ m \geq 1. \tag{30}$$

One should notice the difference between wavelet-like reproducing kernels that we introduce here and the "wavelet kernels" studied in Amato et al. (2006), Rakotomamonjy and Canu (2005), and Rakotomamonjy et al. (2005). The latter are a class of Hilbert-Schmidt kernels defined as a superposition of dilations and translations of a wavelet function.

We now consider the decomposition (30) when $m \to \infty$. To this end, we define the space

$$\mathcal{H}_{-\infty} := \bigcap_{n \in \mathbb{Z}} \mathcal{H}_{K_n}$$

and we describe the space in the next theorem.

Theorem 9 *Suppose that K defined by (16) is γ-refinable and the feature map Φ satisfies (18). Then the closed subspace $\mathcal{H}_{-\infty}$ of $\mathcal{H}_K$ has the form*

$$\mathcal{H}_{-\infty} = \left\{ (\Phi(\cdot), u)_{\mathcal{W}} : u \in \bigcap_{n \in \mathbb{N}} \mathcal{R}((T^*)^n) \right\}. \tag{31}$$

Moreover, $\mathcal{H}_{-\infty} = \{0\}$ if and only if

$$\lim_{n \to \infty} \|T^n u\|_{\mathcal{W}} = 0, \quad \text{for all } u \in \mathcal{W}. \tag{32}$$

Proof Since $\mathcal{H}_{K_n}$, $n \leq 0$, are closed subspaces of $\mathcal{H}_K$, $\mathcal{H}_{-\infty}$ is a closed subspace of $\mathcal{H}_K$. By (20), we use induction to conclude for each $n \in \mathbb{N}$ that

$$\lambda^{-n/2} (\Phi \circ \gamma_{-n}(\cdot), u)_{\mathcal{W}} = (T^n \Phi(\cdot), u)_{\mathcal{W}} = (\Phi(\cdot), (T^*)^n u)_{\mathcal{W}}, \ \ u \in \mathcal{W}. \tag{33}$$

By Theorem 1 and Equation (27), functions in $\mathcal{H}_{K_{-n}}$ are of the form $\lambda^{-n/2} (\Phi \circ \gamma_{-n}(\cdot), u)_{\mathcal{W}}$. This combined with (33) proves formula (31).

It remains to prove the second statement. Since for each $n \in \mathbb{N}$, $(T^*)^n$ is isometric, $\mathcal{R}((T^*)^n)$ is a closed subspace of $\mathcal{W}$. Therefore,

$$\mathcal{W}_{-\infty} := \bigcap_{n\in\mathbb{N}} \mathcal{R}((T^*)^n)$$

is a closed subspace of $\mathcal{W}$. It suffices to show that $\mathcal{W}_{-\infty} = \{0\}$ if and only if (32) holds. Suppose that (32) is satisfied. Let $v \in \mathcal{W}_{-\infty}$ and by the definition of $\mathcal{W}_{-\infty}$, there exists for each $n \in \mathbb{N}$ a $v_n \in \mathcal{W}$ such that $(T^*)^n v_n = v$. Since T^* is isometric, $\|v_n\|_{\mathcal{W}} = \|v\|_{\mathcal{W}}$, which ensures that for each $u \in \mathcal{W}$ and $n \in \mathbb{N}$,

$$|(u,v)_{\mathcal{W}}| = |(u,(T^*)^n v_n)_{\mathcal{W}}| = |(T^n u, v_n)_{\mathcal{W}}| \le \|T^n u\|_{\mathcal{W}} \|v_n\|_{\mathcal{W}} = \|T^n u\|_{\mathcal{W}} \|v\|_{\mathcal{W}}.$$

Let $n \to \infty$ in the above inequality and by condition (32) we conclude that each $u \in \mathcal{W}$ is orthogonal to $\mathcal{W}_{-\infty}$. Consequently, $\mathcal{W}_{-\infty}$ contains only the zero element.

Conversely, we suppose that $\mathcal{W}_{-\infty} = \{0\}$. By the relation that

$$\bigcap_{n\in\mathbb{N}} \mathcal{R}((T^*)^n) = \left(\bigcup_{n\in\mathbb{N}} \mathcal{N}(T^n) \right)^{\perp}, \tag{34}$$

the union of $\mathcal{N}(T^n)$, $n \in \mathbb{N}$, is dense in $\mathcal{W}$. Let $u \in \mathcal{W}$. For each $\varepsilon > 0$ there exists an $m \in \mathbb{N}$ and $v \in \mathcal{N}(T^m)$ such that $\|u - v\|_{\mathcal{W}} \le \varepsilon$. For each operator A on $\mathcal{W}$, its norm $\|A\|$ is defined as

$$\|A\| := \sup\{\|Aw\|_{\mathcal{W}} : w \in \mathcal{W},\ \|w\|_{\mathcal{W}} = 1\}.$$

An operator on $\mathcal{W}$ has the same norm as its adjoint (Conway, 1990). Since T^* is isometric, we have $\|T\| = \|T^*\| = 1$. By the definition of the norm of an operator on $\mathcal{W}$, there holds

$$\|Tw\|_{\mathcal{W}} \le \|w\|_{\mathcal{W}}, \quad w \in \mathcal{W},$$

We get from the above equation for all $n \ge m$ that $T^n v = 0$ and

$$\|T^n u\|_{\mathcal{W}} = \|T^n u - T^n v\|_{\mathcal{W}} \le \|u - v\|_{\mathcal{W}} \le \varepsilon.$$

This verifies (32) and completes the proof. ■

The decomposition (30) can now be extended to the decomposition

$$\mathcal{H}_K = (\mathcal{H}_{-\infty}) \bigoplus_{n\in\mathbb{N}} \mathcal{H}_{G_{-n}}. \tag{35}$$

This decomposition gives a multiresolution analysis (Mallat, 1989; Meyer, 1992) of the RKHS $\mathcal{H}_K$, in terms of a sequence of orthogonal subspaces, each of which is a RKHS corresponding to the wavelet-like kernels.

In passing, we make an additional remark on condition (32). It has a close relation with the translation invariant subspaces in Hardy spaces (Beurling, 1949), which in turn has an important application to the Bedrosian identity (Yu and Zhang, 2006). Under the assumption that the linear span of the eigenelements of T is dense in $\mathcal{W}$, the condition is equivalent to that all the eigenvalues of T have the absolute value less than one (see Beurling, 1949, and the references therein).

To close this section, we prove a corollary of Theorem 7, which concerns the finite dimensional feature space and presents an example of *trivial* refinable kernels, in the sense that its wavelet-like kernel is the zero kernel.

Corollary 10 *If K defined by (16) is γ-refinable, the feature map Φ satisfies (18), and the feature space $\mathcal{W}$ is finite dimensional, then $\mathcal{H}_{K_{-1}} = \mathcal{H}_K$.*

Proof Since K is γ-refinable, by Theorem 6, T^* is isometric, or equivalently, TT^* is equal to the identity operator on $\mathcal{W}$. It follows that for every $w \in \mathcal{W}$, there holds $w = T(T^*w)$. Therefore, T is a surjective operator on $\mathcal{W}$. Since $\mathcal{W}$ is finite dimensional, T must be injective as well. By Theorem 7, there holds $\mathcal{H}_{K_{-1}} = \mathcal{H}_K$. ■

As an application of Corollary 10, we investigate the finite *dot-product kernel* (FitzGerald et al., 1995). Set $\mathbb{Z}_+ := \mathbb{N} \cup \{0\}$, and $\mathbb{A}_n := \{\alpha := (\alpha_j : j \in \mathbb{N}_d) \in \mathbb{Z}_+^d : \sum_{j\in\mathbb{N}_d} \alpha_j = n\}$, $n \in \mathbb{Z}_+$. It can be seen that the kernel

$$K(x,y) := \sum_{\alpha\in\mathbb{A}_n} c_\alpha x^\alpha y^\alpha, \quad x, y \in \mathbb{R}^d,$$

is refinable on $\mathbb{R}^d$, where c_α, $\alpha \in \mathbb{A}_n$, are positive constants. A feature map Φ for this kernel is

$$\Phi(x) := [\sqrt{c_\alpha} x^\alpha : \alpha \in \mathbb{A}_n] \in \ell^2(\mathbb{A}_n), \quad x \in \mathbb{R}^d$$

and the feature space $\ell^2(\mathbb{A}_n)$ is of finite dimension. By Corollary 10, K is a trivial refinable kernel on $\mathbb{R}^d$ in the sense that $\mathcal{H}_{K_{-1}} = \mathcal{H}_K$. Examples of nontrivial refinable kernels on $\mathbb{R}^d$ will be given in Sections 5 and 6.

4. γ-Refinable Kernels of a Riesz Type

The study in this section is motivated by the representation (6), which indicates a need of a countable subset $\mathcal{X}$ of X such that the linear span of $K_\mathcal{X} := \{K(\cdot, x) : x \in \mathcal{X}\}$ is dense in $\mathcal{H}_K$. In practical computations, it is also desirable to have a convenient and stable way of finding an approximation $\tilde{f} \in \operatorname{span} K_\mathcal{X}$ of a function $f \in \mathcal{H}_K$. This leads to consideration of requiring $K_\mathcal{X}$ to be a frame or a Riesz basis for $\mathcal{H}_K$.

We recall the basic concept of frames and Riesz bases (cf., Daubechies, 1992; Duffin and Schaeffer, 1952; Mallat, 1998; Young, 1980). Let J be an index set. A family of elements $\{\varphi_j : j \in J\}$ in a Hilbert space $\mathcal{W}$ forms a *frame* if there exist $0 < \alpha \le \beta < \infty$ such that for all $f \in \mathcal{W}$

$$\alpha \|f\|_\mathcal{W}^2 \le \sum_{j\in J} |(f, \varphi_j)_\mathcal{W}|^2 \le \beta \|f\|_\mathcal{W}^2. \tag{36}$$

The constants α, β are called the *frame bounds* for $\{\varphi_j : j \in J\}$. We call a frame $\{\varphi_j : j \in J\}$ a *tight frame* when its two frame bounds are equal, that is, $\alpha = \beta$. If, in addition to (36), $\{\varphi_j : j \in J\}$ is a linearly independent set, we call it a *Riesz basis* for $\mathcal{W}$. A Riesz basis $\{\varphi_j : j \in J\}$ is *equivalent* to an orthonormal basis $\{\psi_j : j \in J\}$ for $\mathcal{W}$, namely, there exists a bounded linear operator L on $\mathcal{W}$ having a bounded inverse such that $L(\psi_j) = \varphi_j$, $j \in J$.

An arbitrary element in $\mathcal{W}$ can be expressed as a linear combination of a frame $\{\varphi_j : j \in J\}$ for $\mathcal{W}$. We denote by $\ell^2(J)$ the Hilbert space of the square summable sequences on J with the inner product $(c,d)_{\ell^2(J)} := \sum_{j\in J} c_j \overline{d_j}$. Define the *frame operator* F from $\mathcal{W}$ to $\ell^2(J)$ by setting for all $f \in \mathcal{W}$, $(Ff)_j := (f, \varphi_j)_\mathcal{W}$, for all $j \in J$. Then F^*F is a bounded positive self-adjoint operator on $\mathcal{W}$ with a bounded inverse and

$$\{\tilde{\varphi}_j := (F^*F)^{-1}\varphi_j : j \in J\}$$

is a frame for $\mathcal{W}$. We shall refer to it as the *dual frame* of $\{\varphi_j : j \in J\}$ since we have for all $f \in \mathcal{W}$

$$f = \sum_{j\in J} (f, \varphi_j)_{\mathcal{W}} \tilde{\varphi}_j = \sum_{j\in J} (f, \tilde{\varphi}_j)_{\mathcal{W}} \varphi_j. \tag{37}$$

When $\{\varphi_j : j \in J\}$ is a Riesz basis, one can see from (37) that $(\varphi_j, \tilde{\varphi}_k)_{\mathcal{W}} = \delta_{j,k}$, $j,k \in J$. We hence say in this case that $\{\varphi_j : j \in J\}$ and $\{\tilde{\varphi}_j : j \in J\}$ constitute a pair of *biorthogonal bases* for $\mathcal{W}$.

Let $\mathcal{Z} := \{z_j : j \in J\} \subseteq X$ be a countable set. For a kernel K on X, we are interested in the conditions for which the set $K_{\mathcal{Z}} := \{K(\cdot, z_j) : j \in J\}$ is a Riesz basis for $\mathcal{H}_K$. Let us begin with a simple necessary and sufficient condition which follows directly from the reproducing property (5) and the definition of a Riesz basis.

Proposition 11 *The family $K_{\mathcal{Z}}$ is a Riesz basis for $\mathcal{H}_K$ with frame bounds $0 < \alpha \le \beta < \infty$ if and only if for every finite subset $\mathcal{X} \subseteq \mathcal{Z}$, $K[\mathcal{X}]$ is nonsingular, and for all $f \in \mathcal{H}_K$*

$$\alpha \|f\|^2_{\mathcal{H}_K} \le \sum_{j\in J} |f(z_j)|^2 \le \beta \|f\|^2_{\mathcal{H}_K}.$$

We remark that Proposition 11 is closely related to the concept of universal kernels. If X is a topological space, we call a kernel K on X a *universal kernel* if for all compact $\mathcal{X} \subseteq X$, the linear span of $\{K(\cdot, y) : y \in \mathcal{X}\}$ is dense in $C(\mathcal{X})$, the Banach space of continuous functions on $\mathcal{X}$. Various characterizations of universal kernels are studied in Micchelli et al. (2006). By a result of Zhou (2003), universal kernels have the property that for all finite subsets $\mathbf{x}$ of X, $K[\mathbf{x}]$ is nonsingular.

We next present a characterization for $K_{\mathcal{Z}}$ to be a Riesz basis for $\mathcal{H}_K$ in terms of a uniqueness set $\mathcal{Z}$. We call $\mathcal{X} \subseteq X$ a *uniqueness set* for $\mathcal{H}_K$ if there is not a nontrivial $f \in \mathcal{H}_K$ that vanishes on $\mathcal{X}$ (Micchelli et al., 2003, 2006). We shall also need the matrix

$$\Lambda := [K(z_j, z_k) : j,k \in J]. \tag{38}$$

Note that each bounded operator $\mathcal{A} : \ell^2(J) \to \ell^2(J)$ can be represented via a unique matrix $[\mathcal{A}_{j,k} : j,k \in J]$ as

$$(\mathcal{A}c)_j := \sum_{k\in J} \mathcal{A}_{j,k} c_k, \quad c \in \ell^2(J), \quad j \in J.$$

For simplicity, we shall not distinguish a linear operator on $\ell^2(J)$ from its corresponding representation matrix. The matrix associated with the adjoint operator $\mathcal{A}^*$ of $\mathcal{A}$ is

$$(\mathcal{A}^*)_{j,k} := \overline{\mathcal{A}_{k,j}}, \quad j,k \in J.$$

We also denote by $\mathcal{A}^T$ and $\bar{\mathcal{A}}$ the transpose and conjugate of a matrix $\mathcal{A}$, respectively, namely,

$$(\mathcal{A}^T)_{j,k} = \mathcal{A}_{k,j}, \quad (\bar{\mathcal{A}})_{j,k} = \overline{\mathcal{A}_{j,k}}, \quad j,k \in J.$$

Proposition 12 *The family $K_{\mathcal{Z}}$ forms a Riesz basis for $\mathcal{H}_K$ if and only if $\mathcal{Z}$ is a uniqueness set for $\mathcal{H}_K$ and there exist $0 < \alpha \le \beta < \infty$ such that*

$$\alpha \|c\|^2_{\ell^2(J)} \le (\Lambda c, c)_{\ell^2(J)} \le \beta \|c\|^2_{\ell^2(J)}. \tag{39}$$

Proof The proof follows directly from the fact (see, for example, Young, 1980, page 32) that for $\{\varphi_j : j \in J\}$ to be a Riesz basis for a Hilbert space $\mathcal{W}$ with frame bounds $0 < \alpha \le \beta < \infty$, it is necessary and sufficient that its linear span is dense in $\mathcal{W}$ and for all $c \in \ell^2(J)$

$$\alpha \|c\|^2_{\ell^2(J)} \le \left\| \sum_{j \in J} c_j \varphi_j \right\|^2_{\mathcal{W}} \le \beta \|c\|^2_{\ell^2(J)}.$$

■

The third characterization is in terms of a feature map for the kernel K.

Theorem 13 *Let K be given by (16). Then $K_{\mathcal{Z}}$ is a Riesz basis for $\mathcal{H}_K$ if and only if $\Phi(\mathcal{Z})$ is a Riesz basis for $\overline{\operatorname{span}}\Phi(X)$.*

Proof We only discuss the case when (18) holds true, for the other can be handled in a similar way. Since the operator Γ defined by (19) is an isomorphism from $\mathcal{H}_K$ onto $\mathcal{W}$, $K_{\mathcal{Z}}$ is a Riesz basis if and only if $\Gamma(K_{\mathcal{Z}})$ is a Riesz basis for $\Gamma(\mathcal{H}_K) = \mathcal{W}$. By (16) and the definition of Γ, $\Gamma(K_{\mathcal{Z}}) = \Phi(\mathcal{Z})$. This completes the proof. ■

The following corollary is a direct consequence of Theorem 13.

Corollary 14 *Let K be given by (16). If (18) holds then $K_{\mathcal{Z}}$ is a Riesz basis for $\mathcal{H}_K$ if and only if the features $\Phi(\mathcal{Z})$ of $\mathcal{Z}$ is a Riesz basis for the feature space $\mathcal{W}$.*

For the simplicity of notations, we set

$$\phi_{n,j} := \lambda^{n/2} K(\gamma_n(\cdot), z_j), \quad (n, j) \in \mathbb{Z} \times J.$$

Note that $\phi_{0,j} = K(\cdot, z_j)$, $j \in J$. In the following presentation, we shall adopt the convention that we use m, n to denote integers and j, k, l to denote indices in J. The next result regards the sequence $\phi_{n,j}$ being a Riesz basis for $\mathcal{H}_{K_n}$. Since the proof is standard (cf., Daubechies, 1992), we omit the details.

Proposition 15 *Suppose that $K_{\mathcal{Z}}$ is a Riesz basis for $\mathcal{H}_K$ with frame bounds $0 < \alpha \le \beta < \infty$. Then for each $n \in \mathbb{Z}$, $\{\phi_{n,j} : j \in J\}$ is a Riesz basis for $\mathcal{H}_{K_n}$ with the same frame bounds α, β.*

For the rest of this section we assume that $K_{\mathcal{Z}}$ is a Riesz basis for $\mathcal{H}_K$. This assumption implies that the linear operator S on $\mathcal{H}_K$ defined by

$$Sf := \sum_{j \in J} f(z_j) K(\cdot, z_j), \quad f \in \mathcal{H}_K \tag{40}$$

is bounded positive self-adjoint and so is its inverse operator S^{-1} on $\mathcal{H}_K$ (see, for example, Daubechies, 1992, pages 58–59). Note that S is a special case of the operator F^*F introduced at the beginning of this section. A particular example of the operator S was studied in Smale and Zhou (2007) to approximate integral operators.

Proposition 16 *Suppose that $K_{\mathcal{Z}}$ is a Riesz basis for $\mathcal{H}_K$ and let S be the operator on $\mathcal{H}_K$ defined by (40). Then the function*

$$\tilde{K}(x,y) := (S^{-1}K(\cdot,y))(x), \quad x,y \in X \tag{41}$$

is a kernel on X and the corresponding RKHS is

$$\mathcal{H}_{\tilde{K}} = \{f : f \in \mathcal{H}_K\} \tag{42}$$

with inner product $(f,g)_{\mathcal{H}_{\tilde{K}}} := (Sf,g)_{\mathcal{H}_K}$.

Proof Since for all $x,y \in X$

$$(S^{-1}K(\cdot,y))(x) = (S^{-1}(K(\cdot,y)),K(\cdot,x))_{\mathcal{H}_K}$$

and S^{-1} is positive self-adjoint, the function $\tilde{K}$ defined by (41) is a kernel on X. Clearly, $\mathcal{W} := \{f : f \in \mathcal{H}_K\}$ with inner product $(f,g)_{\mathcal{W}} := (Sf,g)_{\mathcal{H}_K}$ is a RKHS. We also observe by (5) and the fact that $S = S^*$ for each $f \in \mathcal{H}_K$ and $x \in X$ that

$$f(x) = (f,K(\cdot,x))_{\mathcal{H}_K} = (Sf,S^{-1}(K(\cdot,x)))_{\mathcal{H}_K} = (f,S^{-1}(K(\cdot,x)))_{\mathcal{W}} = (f,\tilde{K}(\cdot,x))_{\mathcal{W}},$$

which implies that $\tilde{K}$ is the kernel of $\mathcal{W}$. Consequently, we have (42). ■

We call $\tilde{K}$ defined by (41) the *dual kernel* of K. By the general theory of frames introduced at the beginning of this section, the dual Riesz basis of $K_{\mathcal{Z}}$ is

$$\tilde{K}_{\mathcal{Z}} := \{\tilde{K}(\cdot,z_j) : j \in J\}. \tag{43}$$

To obtain an explicit expression of this dual basis, we need to understand the operator S^{-1}. We shall use notation $\tilde{\Lambda}$ for the inverse Λ^{-1} of matrix Λ defined by (38).

Theorem 17 *If $K_{\mathcal{Z}}$ is a Riesz basis for $\mathcal{H}_K$, then for each $n \in \mathbb{Z}$, the dual Riesz basis $\{\tilde{\phi}_{n,j} : j \in J\}$ of $\{\phi_{n,j} : j \in J\}$ for $\mathcal{H}_{K_n}$ has the form*

$$\tilde{\phi}_{n,j} = \sum_{k\in J} \tilde{\Lambda}_{k,j}\phi_{n,k}, \quad j \in J \tag{44}$$

and

$$S^{-1}f = \sum_{j\in J}\left(\sum_{k\in J}\tilde{\Lambda}_{j,k}f(z_k)\right)\left(\sum_{l\in J}\tilde{\Lambda}_{l,j}K(\cdot,z_l)\right), \quad f \in \mathcal{H}_K. \tag{45}$$

Proof Since $K_{\mathcal{Z}}$ is a Riesz basis for $\mathcal{H}_K$, by Proposition 12, (39) holds for some positive constants α,β. We hence have for all $c \in \ell^2(J)$ that (see, for example, Daubechies, 1992, page 58)

$$\beta^{-1}\|c\|^2_{\ell^2(J)} \le (\tilde{\Lambda}c,c)_{\ell^2(J)} \le \alpha^{-1}\|c\|^2_{\ell^2(J)}. \tag{46}$$

Moreover, by Proposition 15, $\{\phi_{n,j} : j \in J\}$ is a Riesz basis for $\mathcal{H}_{K_n}$. This combined with (46) shows that $\tilde{\phi}_{n,j}$ defined by (44) belong to $\mathcal{H}_{K_n}$. By (10) and (5), we have for all $j,k \in J$ that

$$(\tilde{\phi}_{n,j},\phi_{n,k})_{\mathcal{H}_{K_n}} = (\tilde{\phi}_{0,j},K(\cdot,z_k))_{\mathcal{H}_K} = \tilde{\phi}_{0,j}(z_k) = \sum_{l\in J}\tilde{\Lambda}_{l,j}K(z_k,z_l) = \sum_{l\in J}\Lambda_{k,l}\tilde{\Lambda}_{l,j} = \delta_{j,k},$$

which shows that $\{\tilde{\phi}_{n,j} : j \in J\}$ is the dual Riesz basis of $\{\phi_{n,j} : j \in J\}$ in $\mathcal{H}_{K_n}$.

We next establish the representation (45) of S^{-1}. It follows that for each $f \in \mathcal{H}_K$ the function (denoted by g) in the right hand side of (45) is in $\mathcal{H}_K$. Recalling the definition of matrix $\tilde{\Lambda}$, the function g satisfies for $j \in J$ that

$$g(z_j) = \sum_{l \in J} \tilde{\Lambda}_{j,l} f(z_l).$$

As a consequence, we have by the definition (40) of S for all $k \in J$ that

$$(Sg)(z_k) = \sum_{j \in J} \left(\sum_{l \in J} \tilde{\Lambda}_{j,l} f(z_l) \right) K(z_k, z_j) = \sum_{l \in J} f(z_l) \sum_{j \in J} \tilde{\Lambda}_{j,l} \Lambda_{k,j} = \sum_{l \in J} f(z_l) \delta_{k,l} = f(z_k).$$

Since, by Proposition 12, $\mathcal{Z}$ is a uniqueness set for $\mathcal{H}_K$, we have $Sg = f$. ■

We remark that the functions defined by Equation (44) satisfy

$$\tilde{\phi}_{n,j} = \lambda^{n/2} \tilde{\phi}_{0,j} \circ \gamma_n, \quad j \in J, \ n \in \mathbb{Z}.$$

Another implication of Theorem 17 is that $\tilde{\phi}_{0,j}$, $j \in J$, are the *interpolating functions* on $\mathcal{Z}$, that is,

$$\tilde{\phi}_{0,j}(z_k) = \delta_{j,k}, \quad j,k \in J.$$

For each $n \in \mathbb{Z}$ we introduce the sampling operator $I_{n,J}$ by

$$(I_{n,J} f)_j := \lambda^{-n/2} f(\gamma_{-n}(z_j)), \quad j \in J, \quad f \in \mathcal{H}_{K_n}.$$

It is pointed out that a special case of $I_{0,J}$ has been introduced in Smale and Zhou (2007). A function $f \in \mathcal{H}_{K_n}$ can be completely recovered from its sample $I_{n,J} f$, that is,

$$f = \sum_{j \in J} (\tilde{\Lambda} I_{n,J} f)_j \phi_{n,j} = \sum_{j \in J} (I_{n,J} f)_j \tilde{\phi}_{n,j}. \tag{47}$$

In particular, we have the representation for our original kernel K

$$K(x,y) = \sum_{j,k \in J} K(x, z_j) \tilde{\Lambda}_{j,k} K(z_k, y), \quad x, y \in X. \tag{48}$$

The Riesz basis provides a characterization of γ-refinable kernels in terms of the sampling operator, which we present next.

Theorem 18 *Suppose that $K_{\mathcal{Z}}$ is a Riesz basis for $\mathcal{H}_K$. Then K is γ-refinable if and only if*

$$\{I_{-1,J} \phi_{0,j} : j \in J\} \subseteq \ell^2(J), \tag{49}$$

$$(\tilde{\Lambda} I_{-1,J} \phi_{0,j}, I_{-1,J} \phi_{0,k})_{\ell^2(J)} = K(z_k, z_j), \quad j,k \in J, \tag{50}$$

and

$$\phi_{-1,j} = \sum_{k \in J} (\tilde{\Lambda} I_{-1,J} \phi_{0,j})_k K(\cdot, z_k), \quad j \in J. \tag{51}$$

Proof Suppose that conditions (49), (50) and (51) hold true. For $j,k \in J$, we set $\mathcal{C}_{j,k} := (\tilde{\Lambda} I_{-1,J} \phi_{0,j})_k$. By (49) and (46), $[\mathcal{C}_{j,k} : k \in J] \in \ell^2(J)$. Since $K_{\mathcal{Z}}$ is a Riesz basis for $\mathcal{H}_K$, it follows from (51) that $\phi_{-1,j} \in \mathcal{H}_K$, $j \in J$. Equations (50), (47) and (10) imply for all $j,k \in J$ that

$$(\phi_{-1,j}, \phi_{-1,k})_{\mathcal{H}_K} = K(z_k, z_j) = (\phi_{-1,j}, \phi_{-1,k})_{\mathcal{H}_{K_{-1}}}. \tag{52}$$

For each $f \in \mathcal{H}_{K_{-1}}$, we have by Proposition 15 a sequence $f_n \in \operatorname{span}\{\phi_{-1,j} : j \in J\}$, $n \in \mathbb{N}$ that converges to f in $\mathcal{H}_{K_{-1}}$. By (52), there holds for each $n \in \mathbb{N}$ that $f_n \in \mathcal{H}_K$ and

$$(f_n, f_n)_{\mathcal{H}_K} = (f_n, f_n)_{\mathcal{H}_{K_{-1}}}. \tag{53}$$

This means that f_n is a Cauchy sequence in $\mathcal{H}_K$. There hence exists $g \in \mathcal{H}_K$ that is the limit of f_n in $\mathcal{H}_K$. We get by (5) for each $x \in X$ that

$$g(x) = (g, K(\cdot,x))_{\mathcal{H}_K} = \lim_{n\to\infty} (f_n, K(\cdot,x))_{\mathcal{H}_K} = \lim_{n\to\infty} f_n(x) = \lim_{n\to\infty} (f_n, K_{-1}(\cdot,x))_{\mathcal{H}_{K_{-1}}} = f(x).$$

Therefore, $f \in \mathcal{H}_K$, and by (53)

$$(f,f)_{\mathcal{H}_{K_{-1}}} = \lim_{n\to\infty} (f_n, f_n)_{\mathcal{H}_{K_{-1}}} = \lim_{n\to\infty} (f_n, f_n)_{\mathcal{H}_K} = (g,g)_{\mathcal{H}_K} = (f,f)_{\mathcal{H}_K}.$$

We conclude that $\mathcal{H}_{K_{-1}} \preceq \mathcal{H}_K$, that is, K is γ-refinable.

Conversely, suppose that $\mathcal{H}_{K_{-1}} \preceq \mathcal{H}_K$. Then since $K_{\mathcal{Z}}$ is a Riesz basis for $\mathcal{H}_K$ with the dual basis $\tilde{K}_{\mathcal{Z}}$ defined by (43), we let $n = 0$ and $f = \phi_{-1,j}$ in (47) to get that (49), (51) hold true. The inclusion $\mathcal{H}_{K_{-1}} \preceq \mathcal{H}_K$ also implies (52). Through a calculation, we notice that (50) is a consequence of (52). The proof is complete. ■

In the rest of this section, we construct a frame for the RKHS $\mathcal{H}_G$ of the wavelet-like kernel $G := K_1 - K$.

Lemma 19 *Suppose that K is γ-refinable and $K_{\mathcal{Z}}$ is a Riesz basis for $\mathcal{H}_K$ with frame bounds $0 < \alpha \le \beta < \infty$. Then*

$$\psi_{0,j} := \lambda^{-1/2} G(\cdot, \gamma_{-1}(z_j)), \quad j \in J$$

form a frame for $\mathcal{H}_G$ with the same frame bounds α, β.

Proof By the definition $G = K_1 - K$, we have that

$$\psi_{0,j} = \phi_{1,j} - \lambda^{-1/2} K(\cdot, \gamma_{-1}(z_j)), \quad j \in J.$$

Let $f \in \mathcal{H}_G$. Since f is orthogonal to $\mathcal{H}_K$ in $\mathcal{H}_{K_1}$, we have for each $j \in J$ that

$$(f, \psi_{0,j})_{\mathcal{H}_G} = (f, \psi_{0,j})_{\mathcal{H}_{K_1}} = (f, \phi_{1,j})_{\mathcal{H}_{K_1}},$$

where the first equality holds because of Theorem 8 (2). Applying Proposition 15 and $\|f\|_{\mathcal{H}_G} = \|f\|_{\mathcal{H}_{K_1}}$ yields that $\{\psi_{0,j} : j \in J\}$ is a frame for $\mathcal{H}_G$ with frame bounds α, β. ■

The frame for the RKHS $\mathcal{H}_G$ is now translated to a frame for the RKHSs $\mathcal{H}_{G_n}$.

Proposition 20 *Suppose that K is γ-refinable and $K_{\mathcal{Z}}$ is a Riesz basis for $\mathcal{H}_K$ with frame bounds $0<\alpha\le\beta<\infty$. Then for each $n\in\mathbb{Z}$, $\psi_{n,j}:=\lambda^{n/2}\psi_{0,j}\circ\gamma_n$, $j\in J$ form a frame for $\mathcal{H}_{G_n}$ with the same frame bounds α,β. Furthermore, there holds*

$$\psi_{n,j}=\sum_{k\in J}\mathcal{D}_{j,k}\phi_{n+1,k}, \tag{54}$$

where

$$\mathcal{D}_{j,k}:=\delta_{j,k}-\lambda^{-1}\sum_{l\in J}\tilde{\Lambda}_{k,l}K(\gamma_{-1}(z_l),\gamma_{-1}(z_j)). \tag{55}$$

Proof Arguments similar to those in the proof of Proposition 15 with Lemma 19, equations (28) and (29) prove the first claim of this proposition. For each $n\in\mathbb{Z}$, by Theorem 17, $\{\tilde{\phi}_{n+1,k}:k\in J\}$ is the dual Riesz basis of $\{\phi_{n+1,k}:k\in J\}$ for $\mathcal{H}_{K_{n+1}}$. Since $\mathcal{H}_{G_n}\preceq\mathcal{H}_{K_{n+1}}$, we obtain for each $j\in J$

$$\psi_{n,j}=\sum_{k\in J}(\psi_{n,j},\tilde{\phi}_{n+1,k})_{\mathcal{H}_{K_{n+1}}}\phi_{n+1,k}.$$

By (44) we confirm that $(\psi_{n,j},\tilde{\phi}_{n+1,k})_{\mathcal{H}_{K_{n+1}}}$ is equal to $\mathcal{D}_{j,k}$ defined by (55), proving the result. ■

We next present the reconstruction of a function $f\in\mathcal{H}_{G_n}$ from its samples $(f,\psi_{n,j})_{\mathcal{H}_{G_n}}$, $j\in J$. To describe the reconstruction, we remark that for each $(n,j)\in\mathbb{Z}\times J$ there holds

$$\phi_{n,j}=\sum_{k\in J}\mathcal{C}_{j,k}\phi_{n+1,k} \tag{56}$$

and

$$\tilde{\phi}_{n,j}=\sum_{k\in J}\tilde{\mathcal{C}}_{j,k}\tilde{\phi}_{n+1,k}, \tag{57}$$

where

$$\tilde{\mathcal{C}}_{j,k}:=\lambda^{-1/2}\sum_{l\in J}\tilde{\Lambda}_{l,j}K(\gamma_{-1}(z_k),z_l).$$

For $j,k\in J$, we let

$$\tilde{\mathcal{D}}_{j,k}:=\delta_{j,k}-\sum_{l\in J}\overline{\mathcal{C}_{l,j}}\tilde{\mathcal{C}}_{l,k}.$$

Theorem 21 *Suppose that K is γ-refinable and $K_{\mathcal{Z}}$ is a Riesz basis for $\mathcal{H}_K$. For each $n\in\mathbb{Z}$, the functions*

$$\tilde{\psi}_{n,j}:=\tilde{\phi}_{n+1,j}-\sum_{k\in J}\overline{\mathcal{C}_{k,j}}\tilde{\phi}_{n,k},\quad j\in J \tag{58}$$

constitute a frame for $\mathcal{H}_{G_n}$ and have the representation

$$\tilde{\psi}_{n,j}=\sum_{k\in J}\tilde{\mathcal{D}}_{j,k}\tilde{\phi}_{n+1,k}. \tag{59}$$

Moreover, there holds for all $f\in\mathcal{H}_{G_n}$ that

$$f=\sum_{j\in J}(f,\tilde{\psi}_{n,j})_{\mathcal{H}_{G_n}}\psi_{n,j}=\sum_{j\in J}(f,\psi_{n,j})_{\mathcal{H}_{G_n}}\tilde{\psi}_{n,j}. \tag{60}$$

Proof Since K is γ-refinable, by Proposition 2, $\mathcal{H}_{K_n} \preceq \mathcal{H}_{K_{n+1}}$. This together with Theorem 17 follows that the functions $\tilde{\psi}_{n,j}$ defined by (58) are in $\mathcal{H}_{K_{n+1}}$. Moreover, it can be verified by (56) that they are orthogonal to $\mathcal{H}_{K_n}$. Therefore, $\{\tilde{\psi}_{n,j} : j \in J\} \subseteq \mathcal{H}_{G_n}$. Arguments similar to those in the proof of Proposition 20 yield that $\tilde{\psi}_{n,j}$, $j \in J$ form a frame for $\mathcal{H}_{G_n}$ with the same frame bounds as those of $\{\tilde{\phi}_{0,j} : j \in J\}$ for $\mathcal{H}_K$. Equation (59) is obtained by substituting (57) into (58).

We next prove the first equality of (60). To this end, for any fixed $f \in \mathcal{H}_{G_n}$ we define

$$g := \sum_{j\in J} (f, \tilde{\psi}_{n,j})_{\mathcal{H}_{G_n}} \psi_{n,j}$$

and observe that $g \in \mathcal{H}_{G_n}$. It can be verified that

$$\begin{aligned} g &= \sum_{j\in J} (f, \tilde{\phi}_{n+1,j})_{\mathcal{H}_{K_{n+1}}} \psi_{n,j} \\ &= \sum_{j\in J} (f, \tilde{\phi}_{n+1,j})_{\mathcal{H}_{K_{n+1}}} (\phi_{n+1,j} - \lambda^{-(n+1)/2} K_n(\cdot, \gamma_{-n-1}(z_j))) \\ &= f - \lambda^{-(n+1)/2} \sum_{j\in J} (f, \tilde{\phi}_{n+1,j})_{\mathcal{H}_{K_{n+1}}} K_n(\cdot, \gamma_{-n-1}(z_j)). \end{aligned}$$

The above equation ensures that $g - f \in \mathcal{H}_{G_n} \cap \mathcal{H}_{K_n}$, which implies that $g = f$. Likewise, we may prove the second equality of (60). ■

Suppose that K is a kernel on X and K_n, $n \in \mathbb{Z}$, are defined by (8). The RKHS $\mathcal{H}_K$ is said to have a multiresolution analysis if

$$\cdots \preceq \mathcal{H}_{K_{-2}} \preceq \mathcal{H}_{K_{-1}} \preceq \mathcal{H}_K, \ \bigcap_{n\in\mathbb{N}} \mathcal{H}_{K_{-n}} = \{0\},$$

and $K_{\mathcal{Z}}$ with a countable set $\mathcal{Z} := \{z_j : j \in J\} \subseteq X$ is a Riesz basis for $\mathcal{H}_K$. The following theorem characterizes a RKHS that has a multiresolution analysis.

Theorem 22 *Let K be a kernel on an input space X with a feature map Φ from X to a Hilbert space $\mathcal{W}$ that satisfies (18). The RKHS $\mathcal{H}_K$ has a multiresolution analysis if and only if Φ is refinable, that is, it satisfies (20) for some bounded linear operator T on $\mathcal{W}$ whose adjoint T^* is isometric, T has the property (32) and there exists a countable subset $\mathcal{Z}$ of X such that $\Phi(\mathcal{Z})$ is a Riesz basis for $\mathcal{W}$.*

Proof The result of this theorem follows directly from Theorems 6, 9 and 13. ■

Suppose that $\mathcal{H}_K$ has a multiresolution analysis. By (35), Lemma 19 and Proposition 20, we have that

$$\mathcal{H}_K = \bigoplus_{n\in\mathbb{N}} \mathcal{H}_{G_{-n}}$$

and $\lambda^{(n-1)/2} G(\gamma_n(\cdot), \gamma_{-1}(z_j))$, $j \in J$ form a frame for $\mathcal{H}_{G_n}$, $n \le -1$. The multiresolution analysis on $\mathcal{H}_K$ is hence generated by the wavelet-like kernel G.

To close this section, we present the decomposition and reconstruction algorithms. These algorithms are analogues to the Mallat algorithms (cf., Mallat, 1989) in wavelet analysis. They are

important for fast computation. With Equations (56), (57), (54) and (59), we now establish a recursive scheme for the decomposition (30). For

$$f \in \bigcup_{m\geq 0} \mathcal{H}_{K_m},$$

we denote by $P_n f$ the orthogonal projection of f onto $\mathcal{H}_{K_n}$, for $n \leq 0$. We have that

$$P_{n+1}f = P_n f + \sum_{j\in J} (f, \tilde{\psi}_{n,j})_{\mathcal{H}_{G_n}} \psi_{n,j} = P_n f + \sum_{j\in J} (f, \psi_{n,j})_{\mathcal{H}_{G_n}} \tilde{\psi}_{n,j}, \;\; n \leq -1.$$

We define four vectors

$$\alpha_n := [(f, \phi_{n,j})_{\mathcal{H}_{K_n}} : j \in J], \;\; \tilde{\alpha}_n := [(f, \tilde{\phi}_{n,j})_{\mathcal{H}_{K_n}} : j \in J],$$

$$\beta_n := [(f, \psi_{n,j})_{\mathcal{H}_{G_n}} : j \in J] \;\text{ and }\; \tilde{\beta}_n := [(f, \tilde{\psi}_{n,j})_{\mathcal{H}_{G_n}} : j \in J].$$

By (47), the projection $P_n f$ is completely described by α_n or $\tilde{\alpha}_n$. Likewise, by (60), the *difference* $P_{n+1}f - P_n f$ between two levels of consecutive projections is completely determined by β_n or $\tilde{\beta}_n$. We introduce the matrix notations:

$$\mathcal{C} := [\mathcal{C}_{j,k} : j,k \in J], \; \tilde{\mathcal{C}} := [\tilde{\mathcal{C}}_{j,k} : j,k \in J], \; \mathcal{D} := [\mathcal{D}_{j,k} : j,k \in J], \; \tilde{\mathcal{D}} := [\tilde{\mathcal{D}}_{j,k} : j,k \in J].$$

Suppose that α_0 is given and for each $n \leq -1$ we then use $\mathcal{C}$ and $\mathcal{D}$ to decompose α_n, $n \leq 0$ recursively to obtain α_n, β_n

$$\alpha_n = \overline{\mathcal{C}}\, \alpha_{n+1}, \;\; \beta_n = \overline{\mathcal{D}}\, \alpha_{n+1}. \tag{61}$$

Conversely, α_{n+1} can be reconstructed from α_n and β_n by using $\tilde{\mathcal{C}}$ and $\tilde{\mathcal{D}}$

$$\alpha_{n+1} = \tilde{\mathcal{C}}^T \alpha_n + \tilde{\mathcal{D}}^T \beta_n, \;\; n \leq -1. \tag{62}$$

Alternatively, the decomposition and reconstruction process can start from $\tilde{\alpha}_0$

$$\tilde{\alpha}_n = \overline{\tilde{\mathcal{C}}}\, \tilde{\alpha}_{n+1}, \;\; \tilde{\beta}_n = \overline{\tilde{\mathcal{D}}}\, \tilde{\alpha}_{n+1}, \;\; \tilde{\alpha}_{n+1} = \mathcal{C}^T \tilde{\alpha}_n + \mathcal{D}^T \tilde{\beta}_n, \;\; n \leq -1. \tag{63}$$

Since $\{\phi_{0,j} : j \in J\}$ and $\{\tilde{\phi}_{0,j} : j \in J\}$ are Riesz bases for $\mathcal{H}_K$ and Riesz bases are equivalent to orthonormal bases,

$$\{[(f, \phi_{0,j})_{\mathcal{H}_K} : j \in J] : f \in \mathcal{H}_K\} = \{[(f, \tilde{\phi}_{0,j})_{\mathcal{H}_K} : j \in J] : f \in \mathcal{H}_K\} = \ell^2(J),$$

we have by (61), (62) and (63) that

$$\tilde{\mathcal{C}}^T \overline{\mathcal{C}} + \tilde{\mathcal{D}}^T \overline{\mathcal{D}} = \mathcal{C}^T \overline{\tilde{\mathcal{C}}} + \mathcal{D}^T \overline{\tilde{\mathcal{D}}} = I, \tag{64}$$

where I is the identity matrix. We say that $\mathcal{C}, \tilde{\mathcal{C}}, \mathcal{D}, \tilde{\mathcal{D}}$ form a *perfect reconstruction system* if they satisfy (64).

5. Refinable Translation Invariant Kernels

In this section, we consider refinable kernels with specializing our input space to $\mathbb{R}^d$, $d \in \mathbb{N}$, the mapping γ to the dilation mapping $x \to 2x$ in $\mathbb{R}^d$ and kernels to *translation invariant* kernels K on $\mathbb{R}^d$, that is, for all $x, y, a \in \mathbb{R}^d$

$$K(x-a, y-a) = K(x, y).$$

In other words, the main purpose of this section is to characterize refinable translation invariant kernels on $\mathbb{R}^d$.

We need the notation of *Fourier transform* defined for $f \in L^1(\mathbb{R}^d)$ as

$$\hat{f}(t) := \frac{1}{(2\pi)^d} \int_{\mathbb{R}^d} f(x) e^{-i(x,t)} dx, \ \ t \in \mathbb{R}^d,$$

where (x,t) denotes the inner product of x, t in $\mathbb{R}^d$. The Fourier transform $\hat{f}$ of $f \in L^p(\mathbb{R}^d)$, $1 < p \leq \infty$, is defined in the weak sense (Grafakos, 2004). If both $f, \hat{f}$ belong to $L^1(\mathbb{R}^d)$ then there holds

$$f(x) = \int_{\mathbb{R}^d} \hat{f}(t) e^{i(x,t)} dt, \ \ x \in \mathbb{R}^d.$$

Clearly, K is translation invariant if and only if there exists a function $k : \mathbb{R}^d \to \mathbb{C}$ such that

$$K(x, y) = k(x-y), \ \ x, y \in \mathbb{R}^d. \tag{65}$$

Note that in this section k will always denote a function. It was established by Bochner (1959) that if k is continuous on $\mathbb{R}^d$ then (65) defines a kernel if and only if there exists a finite positive Borel measure μ on $\mathbb{R}^d$ such that

$$k(x) = \int_{\mathbb{R}^d} e^{i(x,t)} d\mu(t), \ \ x \in \mathbb{R}^d. \tag{66}$$

We shall consider only measures μ that are *absolutely continuous* with respect to the Lebesgue measure. This means that $\mu(A) = 0$ whenever the Lebesgue measure $|A|$ of a Borel subset $A \subseteq \mathbb{R}^d$ is zero. By the Radon-Nikodym theorem (Rudin, 1987), μ in (66) is absolutely continuous with respect to the Lebesgue measure if and only if $\hat{k}$ is a nonnegative Lebesgue integrable function on $\mathbb{R}^d$.

Specifically, we shall characterize nonnegative $\hat{k} \in L^1(\mathbb{R}^d)$ for which the kernel K given by

$$K(x, y) = k(x-y) = \int_{\mathbb{R}^d} e^{i(x-y,t)} \hat{k}(t) dt, \ \ x, y \in \mathbb{R}^d \tag{67}$$

is refinable, that is, there holds

$$\mathcal{H}_{K_{-1}} \preceq \mathcal{H}_K. \tag{68}$$

Note that in this section K_j, $j \in \mathbb{Z}$ are defined through a positive constant λ as

$$K_j(x, y) = \lambda^j K(2^j x, 2^j y), \ \ x, y \in \mathbb{R}^d. \tag{69}$$

We shall use j to denote integers while m, n, l to denote elements in $\mathbb{Z}^d$. We shall also discuss conditions for $K_{\mathbb{Z}^d} := \{K(\cdot, n) : n \in \mathbb{Z}^d\}$ to be a Riesz basis for $\mathcal{H}_K$.

We next identify a feature map for the kernel K. Let $L^2(\mathbb{R}^d, \hat{k}dt)$ be the space of Borel measurable functions $f : \mathbb{R}^d \to \mathbb{C}$ such that $\int_{\mathbb{R}^d} |f(t)|^2 \hat{k}(t)dt < \infty$. It is a Hilbert space with the inner product

$$(f,g)_{L^2(\mathbb{R}^d,\hat{k}dt)} := \int_{\mathbb{R}^d} f(t)\overline{g(t)}\hat{k}(t)dt.$$

We observe from (67) that

$$K(x,y) = (\Phi(x), \Phi(y))_{L^2(\mathbb{R}^d,\hat{k}dt)}, \;\; x,y \in \mathbb{R}^d,$$

where the feature map $\Phi : \mathbb{R}^d \to L^2(\mathbb{R}^d, \hat{k}dt)$ is defined by $\Phi(x)(t) := e^{i(x,t)}, \;\; t \in \mathbb{R}^d$. It is clear that $\operatorname{span}\Phi(\mathbb{R}^d)$ is dense in $L^2(\mathbb{R}^d, \hat{k}dt)$. By Lemma 5, the functions in $\mathcal{H}_K$ are of the form

$$f_\Phi := (\Phi(\cdot), f)_{L^2(\mathbb{R}^d,\hat{k}dt)}, \;\; f \in L^2(\mathbb{R}^d, \hat{k}dt) \tag{70}$$

and the inner product on $\mathcal{H}_K$ is given by

$$(f_\Phi, g_\Phi)_{\mathcal{H}_K} = (g,f)_{L^2(\mathbb{R}^d,\hat{k}dt)}, \;\; f,g \in L^2(\mathbb{R}^d, \hat{k}dt). \tag{71}$$

We now present a special result on characterization of refinable translation invariant kernels. For this purpose, we set

$$\Omega := \{t \in \mathbb{R}^d : \hat{k}(t) > 0\} \tag{72}$$

and denote by χ_Ω the characteristic function of Ω. We remark that Ω is a Lebesgue measurable subset of $\mathbb{R}^d$.

Theorem 23 *Let K be the translation invariant kernel given in (67) through a nonnegative $\hat{k} \in L^1(\mathbb{R}^d)$ and let Ω be defined by (72). Then K is a refinable kernel on $\mathbb{R}^d$ if and only if*

$$\Omega \subseteq 2\Omega \tag{73}$$

and

$$\hat{k} = \frac{\lambda}{2^d}\chi_\Omega \hat{k}\left(\frac{\cdot}{2}\right). \tag{74}$$

Proof This proof is based on Theorem 4. Through a change of variables, we have for each $y \in \mathbb{R}^d$ that

$$K\left(\frac{x}{2}, y\right) = 2^d \int_{\mathbb{R}^d} e^{i(x,t)} e^{-i2(y,t)} \hat{k}(2t)dt, \;\; x \in \mathbb{R}^d.$$

It follows by (70) that $K(\frac{\cdot}{2}, y) \in \mathcal{H}_K$ for each $y \in \mathbb{R}^d$ if and only if there exists a nonnegative $g \in L^2(\mathbb{R}^d, \hat{k}dt)$ such that

$$\hat{k}(2\cdot) = g\hat{k}. \tag{75}$$

Suppose that (75) is valid. Then it can be verified by the uniqueness of Fourier transforms, and (71) that

$$\left(K\left(\frac{\cdot}{2}, y\right), K\left(\frac{\cdot}{2}, x\right)\right)_{\mathcal{H}_K} = \lambda K(x,y), \quad \text{for all } x,y \in \mathbb{R}^d$$

if and only if

$$2^d g^2 \hat{k} = \lambda \hat{k}(2\cdot). \tag{76}$$

Suppose that (73) and (74) are true. We then obtain by hypothesis (74) that equations (75) and (76) hold true for $g = \frac{\lambda}{2^d}\chi_{\frac{\Omega}{2}}$, which, by (73), is contained in $L^2(\mathbb{R}^d, \hat{k}dt)$. Conversely, if (75) and (76) are valid then (73) follows from (75), and (74) is a consequence of (73), (75) and (76). ■

In the next corollary, we prove special properties of a refinable translation invariant kernel.

Corollary 24 *Let K be a refinable translation invariant kernel defined by (67). If $\hat{k}$ is nontrivial, then $\lambda \geq 1$ and if $\Omega = \frac{\Omega}{2} \neq \emptyset$, then $\hat{k}(t)$ is not continuous at $t = 0$.*

Proof Since K is refinable, by Theorem 23, Equations (73) and (74) hold. We then observe by (74) that

$$\int_{\Omega} \hat{k}(t)dt = \lambda \int_{\frac{\Omega}{2}} \hat{k}(t)dt. \tag{77}$$

The inclusion (73) and Equation (77) imply that there must hold $\lambda \geq 1$ since $\hat{k}$ is nontrivial.

We next prove that $\hat{k}(t)$ is not continuous at $t = 0$. Assume to the contrary that $\hat{k}$ is continuous at $t = 0$ and $\Omega = \frac{\Omega}{2} \neq \emptyset$. In this case, we first see from (77) that $\lambda = 1$ and then by (74) for each $t \in \Omega$ that

$$\hat{k}(t) = \lim_{j\to\infty} \frac{\hat{k}(2^{-j}t)}{2^{dj}} = 0$$

because by hypothesis $\hat{k}(t)$ is continuous at $t = 0$. This contradicts the assumption that $\Omega \neq \emptyset$. ■

As a consequence of this corollary, we have the following interesting observation. For an inclusion relation of the RKHSs of Gaussian kernels, see Walder et al. (2007).

Corollary 25 *The Gaussian kernels*

$$G_\sigma(x,y) := \exp\left(-\sigma\|x-y\|^2\right), \;\; x,y \in \mathbb{R}^d, \;\; \sigma > 0,$$

where $\|x\| := (x,x)^{1/2}$, are not refinable.

Proof Since the Gaussian kernels can be represented as

$$G_\sigma(x,y) = \frac{1}{(2\pi)^d} \int_{\mathbb{R}^d} \left(\frac{\pi}{\sigma}\right)^{d/2} e^{i(x-y,t)} e^{-\frac{\|t\|^2}{4\sigma}} dt,$$

and $e^{-\frac{\|\cdot\|^2}{4\sigma}}$, supported on the whole $\mathbb{R}^d$, is clearly continuous at $t = 0$, by Corollary 24, they are not refinable. ■

We now present a nontrivial refinable translation invariant kernel.

Corollary 26 *For $a,b,\sigma \geq 0$, let $\hat{k} := \|\cdot\|^\sigma \chi_{[-a,b]^d}$. Then the kernel K defined by (67) with $\hat{k}$ is refinable with $\Omega = [-a,b]^d \setminus \{0\}$ and $\lambda = 2^{\sigma+d}$.*

Proof It can be verified directly that $\hat{k}$ satisfies condition (73) and (74) with $\Omega = [-a,b]^d \setminus \{0\}$ and $\lambda = 2^{\sigma+d}$. Hence, by Theorem 23, K is refinable. ■

We next characterize when $\mathcal{H}_{K_{-1}}$ is a proper subspace of $\mathcal{H}_K$ if K is a refinable translation invariant kernel. For this purpose, we identify the feature map for such a K with

$$\lambda^{-1/2}\Phi\left(\frac{\cdot}{2}\right) = T\Phi,$$

where

$$Tf := \lambda^{-1/2} f\left(\frac{\cdot}{2}\right)\chi_\Omega, \quad f \in L^2(\mathbb{R}^d, \hat{k}dt). \tag{78}$$

Theorem 27 *Suppose that a kernel K defined by (67) is refinable on $\mathbb{R}^d$ and Ω is defined by (72). Then $\mathcal{H}_{K_{-1}}$ is a proper subspace of $\mathcal{H}_K$ if and only if*

$$|2\Omega - \Omega| > 0. \tag{79}$$

If (79) holds true then $\lambda > 1$ and

$$\bigcap_{j\in\mathbb{Z}} \mathcal{H}_{K_j} = \{0\}. \tag{80}$$

Proof Set $f \in L^2(\mathbb{R}^d, \hat{k}dt)$. We observe by (78) that $Tf = 0$ if and only if $f(x) = 0$, a.e. $x \in \frac{\Omega}{2}$. Therefore, $\mathcal{N}(T)$ is nontrivial if and only if (79) is true. The first statement of this theorem hence follows from Theorem 7.

To prove the second statement, we suppose that (79) holds. We prove (80) by verifying condition (32) in Theorem 9. To this end, we note by (78) for each $j \in \mathbb{N}$ that

$$T^j f = \lambda^{-\frac{j}{2}} f\left(\frac{\cdot}{2^j}\right)\chi_\Omega$$

and

$$\hat{k}(2^j\cdot) = \left(\frac{\lambda}{2^d}\right)^j \hat{k}\chi_{\frac{\Omega}{2^j}}. \tag{81}$$

The above two equations imply that

$$\|T^j f\|^2_{L^2(\mathbb{R}^d,\hat{k}dt)} = \int_{\frac{\Omega}{2^j}} |f(t)|^2 \hat{k}(t)dt. \tag{82}$$

We integrate both sides of (81) over $\mathbb{R}^d$ to get that

$$\frac{1}{\lambda^j}\int_\Omega \hat{k}(t)dt = \int_{\frac{\Omega}{2^j}} \hat{k}(t)dt.$$

This with (79) implies that $\lambda > 1$. Hence,

$$\lim_{j\to\infty}\int_{\frac{\Omega}{2^j}} \hat{k}(t)dt = 0,$$

which with (82) ensures that

$$\lim_{j\to\infty} \|T^j f\|_{L^2(\mathbb{R}^d,\hat{k}dt)} = 0,$$

proving the result. ■

Let us turn to establishing conditions for $K_{\mathbb{Z}^d}$ to be a Riesz basis for $\mathcal{H}_K$. We begin with a technical lemma.

Lemma 28 *The family $K_{\mathbb{Z}^d}$ is a Riesz basis for $\mathcal{H}_K$ if and only if $\mathcal{E}_{\mathbb{Z}^d} := \{e^{i(n,t)} : n \in \mathbb{Z}^d\}$ is a Riesz basis for $L^2(\mathbb{R}^d, \hat{k}dt)$.*

Proof This lemma follows from Theorem 13 and the density of $\operatorname{span}\Phi(\mathbb{R}^d)$ in $L^2(\mathbb{R}^d, \hat{k}dt)$. ■

Our next approach is based on the characterization of Riesz bases mentioned at the beginning of the proof for Proposition 12. We shall use the property of Ω that for all Lebesgue measurable sets $A \subseteq \Omega$ with $|A| > 0$ there exists a $\sigma > 0$ such that $\left|\{t \in A : \hat{k}(t) \geq \sigma\}\right| > 0$.

Lemma 29 *The linear span of $\mathcal{E}_{\mathbb{Z}^d}$ is dense in $L^2(\mathbb{R}^d, \hat{k}dt)$ if and only if*

$$|\Omega \cap (\Omega + 2n\pi)| = 0, \;\; n \in \mathbb{Z}^d \setminus \{0\}. \tag{83}$$

Proof Suppose that there exists a nonzero $n \in \mathbb{Z}^d$ such that $|\Omega \cap (\Omega + 2n\pi)| > 0$. For $\sigma_1 > 0, a \in \mathbb{R}$, $0 < \delta < \pi$ we set

$$A_1 := \{t : t \in \Omega \cap (\Omega + 2n\pi), \hat{k}(t) \geq \sigma_1\} \cap [a, a+\delta]^d.$$

We can choose some $\sigma_1 > 0$, $a \in \mathbb{R}$ such that A_1 has nonzero Lebesgue measure. Since the set $A_1 - 2n\pi$ is contained in Ω with $|A_1 - 2n\pi| = |A_1| > 0$, we can find a $\sigma_2 > 0$ such that $|A_2| > 0$, where $A_2 := \{t : t \in A_1 - 2n\pi, \hat{k}(t) \geq \sigma_2\}$. Set $A := A_2 + 2n\pi$, $\sigma := \min\{\sigma_1, \sigma_2\}$. The set A so constructed has the properties that $|A| > 0$, $A \cap (A - 2n\pi) = \emptyset$ and $\hat{k}(t) \geq \sigma$, for $t \in A \cup (A - 2n\pi)$. Using this set, we define a function $f \in L^2(\mathbb{R}^d, \hat{k}dt)$ as

$$f(t) := \begin{cases} -1, & t \in A, \\ 1, & t \in A - 2n\pi, \\ 0, & \text{otherwise.} \end{cases}$$

For an arbitrary $g \in \mathcal{E} := \operatorname{span}\mathcal{E}_{\mathbb{Z}^d}$, we have that

$$\begin{aligned} \int_{\mathbb{R}^d} |g(t) - f(t)|^2 \hat{k}(t)dt &\geq \int_A |g(t) - f(t)|^2 \hat{k}(t)dt + \int_{A-2n\pi} |g(t) - f(t)|^2 \hat{k}(t)dt \\ &\geq \sigma \int_A (|g(t) + 1|^2 + |g(t) - 1|^2)dt \geq \sigma|A|. \end{aligned}$$

This shows that $\mathcal{E}$ would not be dense in $L^2(\mathbb{R}^d, \hat{k}dt)$ if (83) were invalid.

On the other hand, suppose that (83) is true. For $n \in \mathbb{Z}^d$, we define $\tilde{\Omega}_n := (\Omega - 2n\pi) \cap [0, 2\pi]^d$ and observe that these sets satisfy the condition

$$\bigcup \{\tilde{\Omega}_n : n \in \mathbb{Z}^d\} \subseteq [0, 2\pi]^d, \;\; |\tilde{\Omega}_n \cap \tilde{\Omega}_m| = 0, \;\; n \neq m.$$

Let

$$\rho(t) := \begin{cases} \hat{k}(t + 2n\pi), & t \in \tilde{\Omega}_n, \;\; n \in \mathbb{Z}^d, \\ 0, & \text{otherwise} \end{cases}$$

and for $f \in L^2(\mathbb{R}^d, \hat{k}dt)$ we introduce a new function $g \in L^2([0,2\pi]^d, \rho dt)$ by setting

$$g(t) := \begin{cases} f(t+2n\pi), & t \in \tilde{\Omega}_n, \ \ n \in \mathbb{Z}^d, \\ 0, & \text{otherwise.} \end{cases}$$

Since $\mathcal{E}$ is dense in $L^2([0,2\pi]^d, \rho dt)$, for each $\varepsilon > 0$ there exists $\tilde{f} \in \mathcal{E}$ such that

$$\|g - \tilde{f}\|_{L^2([0,2\pi]^d, \rho dt)} < \varepsilon.$$

Note that

$$\|f - \tilde{f}\|_{L^2(\mathbb{R}^d, \hat{k}dt)} = \|g - \tilde{f}\|_{L^2([0,2\pi]^d, \rho dt)}.$$

Combining the two relations above proves the lemma. ■

We next present a necessary and sufficient condition for $K_{\mathbb{Z}^d}$ to be a Riesz basis for $\mathcal{H}_K$ if K is a translation invariant kernel defined by (67) through a nonnegative $\hat{k} \in L^1(\mathbb{R}^d)$.

Theorem 30 *Let K be a translation invariant kernel defined by (67) through a nonnegative $\hat{k} \in L^1(\mathbb{R}^d)$ and Ω be defined by (72). Then $K_{\mathbb{Z}^d}$ is a Riesz basis for $\mathcal{H}_K$ if and only if*

$$\sum_{n \in \mathbb{Z}^d} \chi_\Omega(\cdot + 2n\pi) = 1, \ \ a.e. \tag{84}$$

and there exist $0 < \alpha \le \beta < \infty$ such that

$$\alpha \le \hat{k}(t) \le \beta, \quad a.e.\ t \in \Omega. \tag{85}$$

Proof By Lemma 28, $K_{\mathbb{Z}^d}$ is a Riesz basis for $\mathcal{H}_K$ if and only if the linear span of $\mathcal{E}_{\mathbb{Z}^d}$ is dense in $L^2(\mathbb{R}^d, \hat{k}dt)$, which is equivalent to (83) by Lemma 29,

$$\sum_{n \in \mathbb{Z}^d} c_n e^{i(n,t)} \in L^2(\mathbb{R}^d, \hat{k}dt), \ \ c \in \ell^2(\mathbb{Z}^d) \tag{86}$$

and for some constants $0 < \alpha \le \beta < \infty$

$$\alpha(2\pi)^d \|c\|^2_{\ell^2(\mathbb{Z}^d)} \le \left\| \sum_{n \in \mathbb{Z}^d} c_n e^{i(n,t)} \right\|^2_{L^2(\mathbb{R}^d, \hat{k}dt)} \le \beta(2\pi)^d \|c\|^2_{\ell^2(\mathbb{Z}^d)}, \ \ c \in \ell^2(\mathbb{Z}^d). \tag{87}$$

One can use arguments similar to those on pages 139–140 of Daubechies (1992) to show that relation (86) and inequality (87) hold true if and only if there exist $0 < \alpha \le \beta < \infty$ such that

$$\alpha \le \sum_{n \in \mathbb{Z}^d} \hat{k}(\cdot + 2n\pi) \le \beta, \quad \text{a.e.} \tag{88}$$

The proof is completed by noting that (83) and (88) hold true if and only if (84) and (85) are satisfied. ■

Inequality (88) was established for a different purpose in Smale and Zhou (2004), where it was proved that $\Lambda := [K(m,n) : m,n \in \mathbb{Z}^d]$ satisfies (39) for $J := \mathbb{Z}^d$ if and only if (88) holds true.

In the next theorem, we construct $\hat{k}$ that satisfy conditions (73), (74), (84) and (85) to obtain a refinable kernel K on $\mathbb{R}^d$ with $K_{\mathbb{Z}^d}$ being a Riesz basis for $\mathcal{H}_K$.

Theorem 31 *Let K be defined by (67) where nonnegative $\hat{k} \in L^1(\mathbb{R}^d)$ is continuous at 0. Then K is refinable and $K_{\mathbb{Z}^d}$ is a Riesz basis for $\mathcal{H}_K$ if and only if $\lambda = 2^d$ and*

$$\hat{k} = \eta \chi_\Omega, \ \ a.e. \tag{89}$$

where η is a positive constant, and Ω satisfies (73) and (84). Moreover, if (84) and (89) hold true then functions

$$\frac{1}{\sqrt{\eta}(2\pi)^{d/2}} K(\cdot, n), \ \ n \in \mathbb{Z}^d \tag{90}$$

form an orthonormal basis for $\mathcal{H}_K$.

Proof Suppose that $\lambda = 2^d$, $\hat{k}$ is given by (89) with a positive constant η, and Ω satisfies (73) and (84). By Theorem 23, K defined by (67) is refinable, and by Theorem 30, $K_{\mathbb{Z}^d}$ is a Riesz basis for $\mathcal{H}_K$.

Conversely, we suppose that $\hat{k} \in L^1(\mathbb{R}^d)$ is continuous at 0, K is refinable and $K_{\mathbb{Z}^d}$ is a Riesz basis for $\mathcal{H}_K$. These hypotheses imply that there hold (73), (74), (84) and (85). Choose $\eta = \hat{k}(0)$. Repeatedly using Equation (74) with iterations, we have for all $t \in \Omega$ that

$$\hat{k}(t) = \left(\frac{\lambda}{2^d}\right)^j \hat{k}\left(\frac{t}{2^j}\right), \ \ j \in \mathbb{N}.$$

This formula with condition (85) ensures that $\lambda = 2^d$. Equation (89) follows from the formula above, (85) and the continuity of $\hat{k}$ at 0. This completes the proof of the necessary and sufficient condition.

It remains to show that the functions defined by (90) form an orthonormal basis for $\mathcal{H}_K$. Since (84) and (89) are true, we obtain by direct computation for all $m, n \in \mathbb{Z}^d$ that

$$\begin{aligned}
\frac{1}{\eta(2\pi)^d}(K(\cdot,n), K(\cdot,m))_{\mathcal{H}_K} &= \frac{1}{(2\pi)^d} \int_\Omega e^{i(m-n,t)} dt \\
&= \frac{1}{(2\pi)^d} \int_{\mathbb{R}^d} e^{i(m-n,t)} \chi_\Omega(t) dt \\
&= \frac{1}{(2\pi)^d} \int_{[0,2\pi]^d} e^{i(m-n,t)} \sum_{l \in \mathbb{Z}^d} \chi_\Omega(\cdot + 2l\pi) dt \\
&= \frac{1}{(2\pi)^d} \int_{[0,2\pi]^d} e^{i(m-n,t)} dt = \delta_{m,n}.
\end{aligned}$$

This proves that functions defined by (90) constitute an orthonormal basis for $\mathcal{H}_K$ and completes the proof. ■

By noting that (84) can be interpreted as that $\Omega + 2n\pi$, $n \in \mathbb{Z}^d$, form a *tiling* of $\mathbb{R}^d$, we construct examples of refinable kernels K such that $K_{\mathbb{Z}^d}$ are Riesz bases for $\mathcal{H}_K$. The readers are referred to Grünbaum and Shephard (1989) for the subject of tiling. We describe our examples in the following two corollaries.

Corollary 32 *For $a \in [0, 2\pi]$, let $\Omega := [-a, 2\pi - a]^d$. Then the kernel K defined by (67) with $\hat{k}$ having the form (89) is a refinable kernel such that $K_{\mathbb{Z}^d}$ is a Riesz basis for $\mathcal{H}_K$.*

We remark that when $a = \pi$, k is the well-known sinc function. The result in the last corollary can be extended.

Corollary 33 *Suppose that* α *and* β *are constants satisfying either* $\frac{\pi}{2} \le \alpha \le \beta \le \frac{2\pi}{3}$ *or* $\frac{2\pi}{3} \le \alpha \le \beta \le \pi$. *Let*

$$\Omega := ([-2\pi+\alpha, -2\pi+\beta] \cup [-\pi, -\beta] \cup [-\alpha, \alpha] \cup [\beta, \pi] \cup [2\pi-\beta, 2\pi-\alpha])^d. \tag{91}$$

Then the kernel K defined by (67) with $\hat{k}$ *having the form (89) with* $\eta = 1$ *is a refinable kernel such that* $K_{\mathbb{Z}^d}$ *is a Riesz basis for* $\mathcal{H}_K$. *Moreover,*

$$k(x) = \prod_{j\in\mathbb{N}_d} \frac{2\sin\alpha x_j}{x_j} + \prod_{j\in\mathbb{N}_d} \frac{4\sin\frac{\pi-\beta}{2}x_j}{x_j}\cos\frac{\pi+\beta}{2}x_j + \prod_{j\in\mathbb{N}_d} \frac{4\sin\frac{\beta-\alpha}{2}x_j}{x_j}\cos\frac{\alpha+\beta}{2}x_j, \ \ x \in \mathbb{R}^d,$$

where x_j *denotes the* j*-th component of* x.

When both α and β in (91) are chosen as π, k is also reduced to the sinc function.

6. Refinable Kernels Defined by Refinable Functions

We present in this section a construction of refinable kernels via refinable functions. For a complete reference of refinable functions, the readers are referred to Cavaretta et al. (1991) and Daubechies (1992). As in the last section, we assume that the mapping γ has the form $x \to 2x$ throughout this section.

Let φ be a compactly supported continuous function on $\mathbb{R}^d$ that is *refinable*, namely, there exists $h := [h_n : n \in \mathbb{Z}^d]$ such that

$$\varphi\left(\frac{\cdot}{2}\right) = \sum_{n\in\mathbb{Z}^d} h_n \varphi(\cdot - n). \tag{92}$$

We always assume that φ is nontrivial, and the cardinality of $\{n : h_n \neq 0, \ n \in \mathbb{Z}^d\}$ is finite by the compact support of φ. Suppose further that we have an infinite matrix A satisfying for some positive constants α, β that

$$\alpha\|c\|^2_{\ell^2(\mathbb{Z}^d)} \le (Ac, c)_{\ell^2(\mathbb{Z}^d)} \le \beta\|c\|^2_{\ell^2(\mathbb{Z}^d)}, \ \ c \in \ell^2(\mathbb{Z}^d). \tag{93}$$

The above inequality implies that A is a bounded positive self-adjoint operator on $\ell^2(\mathbb{Z}^d)$ and its inverse A^{-1} is also bounded positive self-adjoint (see, for example, Daubechies, 1992, page 58).

Motivated by (48), associated with the matrix A we define our kernel K by

$$K(x, y) := (A\Psi(x), \Psi(y))_{\ell^2(\mathbb{Z}^d)}, \ \ x, y \in \mathbb{R}^d, \tag{94}$$

where Ψ is a mapping from $\mathbb{R}^d$ to $\ell^2(\mathbb{Z}^d)$ given by $\Psi(x) := [\varphi(x-n) : n \in \mathbb{Z}^d], \ \ x \in \mathbb{R}^d$. Assuming that $A := [A_{m,n} : m, n \in \mathbb{Z}^d]$, we have that

$$K(x, y) = \sum_{m\in\mathbb{Z}^d} \sum_{n\in\mathbb{Z}^d} A_{m,n}\varphi(x-n)\overline{\varphi(y-m)}, \ \ x, y \in \mathbb{R}^d. \tag{95}$$

Kernels in the form

$$\sum_{n\in\mathbb{Z}^d} \psi(x-n)\overline{\psi(y-n)} \tag{96}$$

constructed by a refinable function ψ were considered in Opfer (2006), and kernels defined as a superposition of frame elements in RKHS were discussed in Gao et al. (2001), Opfer (2006), and

Rakotomamonjy and Canu (2005). When A is the identity infinite matrix I, we see that K defined by (95) has the form (96) with $\psi = \varphi$. We are interested in the necessary and sufficient condition for K to degenerate to the form (96) for some function ψ on $\mathbb{R}^d$ such that the series converges for all $x, y \in \mathbb{R}^d$. We need the following technical lemma, whose proof is standard and thus is omitted.

Lemma 34 *The linear span of $\Psi(\mathbb{R}^d)$ is dense in $\ell^2(\mathbb{Z}^d)$, that is, $\Psi(\mathbb{R}^d)^\perp = \{0\}$.*

The next proposition shows that kernels in the form (95) are more general than those in the degenerate form (96) and in general cannot be written in the degenerate form.

Proposition 35 *Let A be an infinite matrix satisfying (93) and K be defined by (95) through a compactly supported continuous function φ. Then K can be represented as (96) if and only if*

$$A_{m,n} = A_{m-n,0}, \quad m, n \in \mathbb{Z}^d. \tag{97}$$

Proof Suppose that the kernel K defined by (95) has the form (96). It follows that for all $x, y \in \mathbb{R}^d$ and for $l \in \mathbb{Z}^d$, $K(x-l, y-l) = K(x, y)$. Using (95), we rewrite the above equation as

$$\sum_{m\in\mathbb{Z}^d} \sum_{n\in\mathbb{Z}^d} A_{m-l,n-l} \varphi(x-n)\overline{\varphi(y-n)} = \sum_{m\in\mathbb{Z}^d} \sum_{n\in\mathbb{Z}^d} A_{m,n} \varphi(x-n)\overline{\varphi(y-n)}.$$

By Lemma 34, we have for all $m, n, l \in \mathbb{Z}^d$ that $A_{m-l,n-l} = A_{m,n}$. In this equation, letting $l = n$ yields (97).

Conversely, we suppose that (97) is satisfied. For $n \in \mathbb{Z}^d$, we set $a_n := A_{n,0}$ and observe that for all $c \in \ell^2(\mathbb{Z}^d)$

$$(Ac, c)_{\ell^2(\mathbb{Z}^d)} = \frac{1}{(2\pi)^d} \int_{[0,2\pi]^d} \left(\sum_{n\in\mathbb{Z}^d} a_n e^{i(n,t)} \right) \left| \sum_{n\in\mathbb{Z}^d} c_n e^{i(n,t)} \right|^2 dt.$$

This with (93) implies that

$$\alpha \le \sum_{n\in\mathbb{Z}^d} a_n e^{i(n,t)} \le \beta, \quad \text{a.e. } t \in [0, 2\pi]^d,$$

where the constants α and β are the lower and upper bound in (93). Therefore, there exists $b \in \ell^2(\mathbb{Z}^d)$ such that

$$\sum_{n\in\mathbb{Z}^d} b_n e^{i(n,t)} = \left(\sum_{n\in\mathbb{Z}^d} a_n e^{i(n,t)} \right)^{1/2}, \quad \text{a.e. } t \in [0, 2\pi]^d.$$

We then define the matrix B by setting $B_{m,n} := b_{m-n,0}$, $m, n \in \mathbb{Z}^d$. Clearly, we have $B = B^*$ and $A = B^2$, which ensures that

$$K(x, y) = (B\Psi(x), B\Psi(y))_{\ell^2(\mathbb{Z}^d)}, \quad x, y \in \mathbb{R}^d.$$

One can see that K can be rewritten as (96) with $\psi := \sum_{m\in\mathbb{Z}^d} B_{0,m} \varphi(\cdot - m)$. ■

The main purpose of this section is to formulate conditions on h and A so that the kernel K in the form (95) is refinable. Our discussions will be based on the following result concerning the RKHS $\mathcal{H}_K$.

Proposition 36 *The RKHS of the kernel K defined by (94) is*

$$\mathcal{H}_K := \{c_\Psi := (\Psi(\cdot), c)_{\ell^2(\mathbb{Z}^d)} : c \in \ell^2(\mathbb{Z}^d)\}$$

with inner product

$$(c_\Psi, d_\Psi)_{\mathcal{H}_K} = (A^{-1}d, c)_{\ell^2(\mathbb{Z}^d)}, \quad c, d \in \ell^2(\mathbb{Z}^d).$$

Proof Since the operator A satisfies (93), there exists a bounded positive self-adjoint operator $A^{1/2}$ on $\ell^2(\mathbb{Z}^d)$ such that $A^{1/2}A^{1/2} = A$ (see, Conway, 1990, ,page 240). It is observed that K has the following feature map representation $K(x,y) = (\Phi(x), \Phi(y))_{\ell^2(\mathbb{Z}^d)}, x, y \in \mathbb{R}^d$, where

$$\Phi := A^{1/2}\Psi. \tag{98}$$

The proposition now follows immediately from Lemmas 5 and 34. ■

Let λ be a fixed positive number and K_j, $j \in \mathbb{Z}$ be defined as in (69). We shall give a characterization for K to be refinable, that is, (68) holds. To this end, we introduce the infinite matrix H associated with h as

$$H_{m,n} := h_{n-2m}, \quad m, n \in \mathbb{Z}^d. \tag{99}$$

It can be seen by the generalized Minkowski inequality that the matrix H induces a bounded operator on $\ell^2(\mathbb{Z}^d)$. In fact, we have for each $c \in \ell^2(\mathbb{Z}^d)$ that

$$\|Hc\|_{\ell^2(\mathbb{Z}^d)} \le \left(\sum_{n\in\mathbb{Z}^d} |h_n|\right) \|c\|_{\ell^2(\mathbb{Z}^d)}.$$

We next characterize refinable kernels in terms of matrices A and H.

Theorem 37 *Suppose that φ is a nontrivial compactly supported refinable function satisfying (92). Then the kernel K defined by (94) is refinable if and only if*

$$HA^{-1}H^*A = \lambda I. \tag{100}$$

Proof The function $\Phi : \mathbb{R}^d \to \ell^2(\mathbb{Z}^d)$ defined by (98) is a feature map for K. We observe by (92) that it satisfies a refinement equation

$$\lambda^{-1/2}\Phi\left(\frac{\cdot}{2}\right) = \lambda^{-1/2}A^{1/2}\Psi\left(\frac{\cdot}{2}\right) = \lambda^{-1/2}A^{1/2}H\Psi = \lambda^{-1/2}A^{1/2}HA^{-1/2}\Phi,$$

where $A^{-1/2}$ denotes the inverse of $A^{1/2}$. Setting

$$T := \lambda^{-1/2}A^{1/2}HA^{-1/2}, \tag{101}$$

by Theorem 6, K is refinable if and only if T^* is isometric, or equivalently, $TT^* = I$. The proof is complete by noting that equation $TT^* = I$ has the form (100). ■

We need the following lemma to study the proper inclusion of $\mathcal{H}_{K_{-1}}$ in $\mathcal{H}_K$.

Lemma 38 *Let $[a_n : n \in \mathbb{Z}^d]$ be a nontrivial vector in $\ell^2(\mathbb{Z}^d)$ with a finite number of nonzero components. Then the linear span of $\{\tilde{a}_m := [a_{m-n} : n \in \mathbb{Z}^d] : m \in \mathbb{Z}^d\}$ is dense in $\ell^2(\mathbb{Z}^d)$, and $\tilde{a}_m$, $m \in \mathbb{Z}^d$ are linearly independent.*

Proposition 39 *Let φ be a nontrivial compactly supported continuous refinable function on $\mathbb{R}^d$, A, h satisfy (93) and (100), and K be defined by (94). Then $\mathcal{H}_{K_{-1}}$ is a proper subspace of $\mathcal{H}_K$.*

Proof By Theorem 7, $\mathcal{H}_{K_{-1}}$ is a proper subspace of $\mathcal{H}_K$ if and only if the null space $\mathcal{N}(T)$ of operator T defined by (101) contains nonzero elements in $\ell^2(\mathbb{Z}^d)$, which is equivalent to that

$$\mathcal{N}(H) \neq \{0\}. \tag{102}$$

Set $\tilde{b}_m := [h_{n-m} : n \in \mathbb{Z}^d], m \in \mathbb{Z}^d$. Assume that $\mathcal{N}(H) = \{0\}$. This implies that span $\{\tilde{b}_{2m} : m \in \mathbb{Z}^d\}$ is dense in $\ell^2(\mathbb{Z}^d)$. Choose $l \in \mathbb{Z}^d \setminus 2\mathbb{Z}^d$. Since $[h_n : n \in \mathbb{Z}^d]$ has a finite number of nonzero components, $\tilde{b}_l$ can be represented as a finite linear combination of $\tilde{b}_{2m}$, $m \in \mathbb{Z}^d$. However, Lemma 38 ensures that $\tilde{b}_m$, $m \in \mathbb{Z}^d$ are linearly independent. This contradiction implies the validity of (102). ■

When A and H commute, Equation (100) reduces to

$$HH^* = \lambda I. \tag{103}$$

Through a scaling of the matrix H, one may consider $HH^* = I$. This equation arose also in the construction of orthonormal wavelets and it has been well understood in the one-dimensional case (cf., Daubechies, 1992). For a special class of solutions h of (103) in the multidimensional case, see Chen et al. (2003) and Chen et al. (2007). These h can be used to construct A and H satisfying (93) and (100).

Proposition 40 *Let h be a solution of (103). Then for each real number $a \in \mathbb{R} \setminus \{\pm\lambda^{-1/2}\}$ the matrix A defined by*

$$A := ((I + aH)(I + aH^*))^{-1} \tag{104}$$

satisfies (93) with $\alpha := (1 + |a|\sqrt{\lambda})^{-2}$ and $\beta := (1 - |a|\sqrt{\lambda})^{-2}$, and (100).

Proof For $a \in \mathbb{R} \setminus \{\pm\lambda^{-1/2}\}$ we set $B := (I + aH)(I + aH^*)$. By direct computation, we obtain for each $c \in \ell^2(\mathbb{Z}^d)$ that

$$(Bc, c)_{\ell^2(\mathbb{Z}^d)} = (1 + a^2\lambda)\|c\|^2_{\ell^2(\mathbb{Z}^d)} + a((H + H^*)c, c)_{\ell^2(\mathbb{Z}^d)}. \tag{105}$$

Equation (103) leads to the estimates

$$\|Hc\|_{\ell^2(\mathbb{Z}^d)} \leq \sqrt{\lambda}\|c\|_{\ell^2(\mathbb{Z}^d)}, \text{ and } \|H^*c\|_{\ell^2(\mathbb{Z}^d)} \leq \sqrt{\lambda}\|c\|_{\ell^2(\mathbb{Z}^d)}.$$

Equation (105) with these two estimates implies that

$$(1 - |a|\sqrt{\lambda})^2\|c\|^2_{\ell^2(\mathbb{Z}^d)} \leq (Bc, c)_{\ell^2(\mathbb{Z}^d)} \leq (1 + |a|\sqrt{\lambda})^2\|c\|^2_{\ell^2(\mathbb{Z}^d)}.$$

The inverse operator A of B hence satisfies (93) with $\alpha := (1 + |a|\sqrt{\lambda})^{-2}$ and $\beta := (1 - |a|\sqrt{\lambda})^{-2}$.

It remains to show that A satisfies Equation (100). It can be verified by (103) that the following equation holds

$$H(I+aH)(I+aH^*)H^* = \lambda(I+aH)(I+aH^*).$$

By the definition (104) of A, this is equivalent to the equation $HA^{-1}H^* = \lambda A^{-1}$, which confirms that A satisfies (100). ■

The last proposition provides a class of refinable kernels given by (95) that never degenerate to the form (96).

Proposition 41 *Let h satisfy (103), where matrix H is defined by (99), and matrix A be of the form (104) for some $a \in \mathbb{R} \setminus \{\pm\lambda^{-1/2}\}$. If there exists at least one $n \in \mathbb{Z}^d$ such that the real part $\mathrm{Re}\,(h_n)$ of h_n is not zero then A satisfies (97) if and only if $a = 0$.*

Proof If $a = 0$ then $A = I$ satisfies (97). Let $a \in \mathbb{R} \setminus \{0, \pm\lambda^{-1/2}\}$. One can see that A satisfies (97) if and only if A^{-1} does. Since $A^{-1} = (1+a^2\lambda)I + a(H+H^*)$, it satisfies (97) only if $H+H^*$ does. Choose $n \in \mathbb{Z}^d$ such that $\mathrm{Re}\,(h_n) \neq 0$. There exists an $m \in \mathbb{Z}^d$ such that $\mathrm{Re}\,(h_m) \neq \mathrm{Re}\,(h_n)$ since $h \in \ell^2(\mathbb{Z}^d)$. As a consequence, we have

$$(H+H^*)_{-m,-m} = 2\,\mathrm{Re}\,(h_m) \neq 2\,\mathrm{Re}\,(h_n) = (H+H^*)_{-n,-n}.$$

This shows that $H+H^*$ does not satisfy (97). The proof is complete. ■

By Propositions 35 and 41, if h is a real vector in $\ell^2(\mathbb{Z}^d)$ satisfying (103) then for all $a \in \mathbb{R} \setminus \{0, \pm\lambda^{-1/2}\}$, the refinable kernel K defined by (95) through A given in (104) can not be rewritten as (96).

We now turn to an investigation of the intersection of $\mathcal{H}_{K_j}$, for $j \in \mathbb{Z}$, where K_j are kernels defined by (69).

Theorem 42 *Suppose that φ is a nontrivial compactly supported continuous refinable function on $\mathbb{R}^d$, h satisfies (103), A satisfies (93) and (100), and K is defined by (94). Then (80) holds true if and only if*

$$h_n \neq 0, \ \textit{for at least one } n \in \mathbb{Z}^d \setminus 2\mathbb{Z}^d. \tag{106}$$

If (106) does not hold then

$$\bigcap_{j\in\mathbb{Z}} \mathcal{H}_{K_j} = \{(\Psi(\cdot), c)_{\ell^2(\mathbb{Z}^d)} : c \in \mathcal{N}(H)^\perp\}. \tag{107}$$

Proof Let $\tilde{\mathcal{F}}$ be the function from $\ell^2(\mathbb{Z}^d)$ to $L^2([0,2\pi]^d)$ defined for $c \in \ell^2(\mathbb{Z}^d)$ by $(\tilde{\mathcal{F}}c)(t) := \sum_{n\in\mathbb{Z}^d} c_n e^{i(n,t)}$, $t \in [0,2\pi]^d$. Set $B := \lambda^{-1/2}H$, $m_0 := \tilde{\mathcal{F}}(\lambda^{-1/2}h)$, and let $\{\nu_j : j \in \mathbb{N}_{2^d}\} \subseteq \mathbb{Z}^d$ denote the set of extreme points of cube $[0,1]^d$. By condition (32) in Theorem 9, (80) holds true if and only if

$$\lim_{j\to\infty} \|T^j c\|_{\ell^2(\mathbb{Z}^d)} = 0, \quad \text{for all } c \in \ell^2(\mathbb{Z}^d), \tag{108}$$

where T is the operator defined by (101). Noting that $T^j = A^{1/2}B^jA^{-1/2}$, Equation (108) is equivalent to the condition

$$\lim_{j\to\infty} (2\pi)^{-d/2} \|\tilde{\mathcal{F}}(B^j c)\|_{L^2([0,2\pi]^d)} = \lim_{j\to\infty} \|B^j c\|_{\ell^2(\mathbb{Z}^d)} = 0, \quad \text{for all } c \in \ell^2(\mathbb{Z}^d). \tag{109}$$

For each $c \in \ell^2(\mathbb{Z}^d)$, we define $m_1 := \tilde{\mathcal{F}} c$. It can be verified by direct calculation that

$$\tilde{\mathcal{F}}(B^j c)(t) = \left(\frac{1}{2^d} \sum_{j \in \mathbb{N}_{2^d}} m_0(-t - \pi \nu_j) m_1(t + \pi \nu_j) \right) \left(\frac{1}{2^d} \sum_{j \in \mathbb{N}_{2^d}} m_0(-t - \pi \nu_j) \right)^{j-1}, \ t \in [0, 2\pi]^d.$$

Equation (103) can be rewritten in terms of m_0 as

$$\sum_{j \in \mathbb{N}_{2^d}} |m_0(\cdot + \pi \nu_j)|^2 = 2^d. \tag{110}$$

The Cauchy-Schwartz inequality with (110) ensures for all $t \in [0, 2\pi]^d$ that

$$\left| \sum_{j \in \mathbb{N}_{2^d}} m_0(t + \pi \nu_j) \right| \le 2^d, \tag{111}$$

where the equality holds at a point $t_0 \in [0, 2\pi]^d$ if and only if

$$m_0(t_0 + \pi \nu_j) = a, \ \ j \in \mathbb{N}_{2^d} \ \text{ for some } \ a \in \mathbb{C} \ \text{ with } \ |a| = 1. \tag{112}$$

If $h_n = 0$ for each $n \in \mathbb{Z}^d \setminus 2\mathbb{Z}^d$ then $m_0(\cdot + \pi \nu_j) = m_0$ for all $j \in \mathbb{N}_{2^d}$. This together with (110) implies that (112) holds for all $t_0 \in [0, 2\pi]^d$. Thus, the equality in (111) holds for all $t \in [0, 2\pi]^d$. We now choose c such that $m_1 = \overline{m_0(-\cdot)}$ and clearly for such a c (109) does not hold. Consequently, (80) does not hold and this is equivalent to saying that (80) implies (106).

Conversely, suppose that (106) holds. By the fact that the zeros of a nontrivial real-analytic function on $\mathbb{R}^d$ form a set of Lebesgue measure zero, the set of points $t \in [0, 2\pi]^d$ for which the equality in (111) holds has zero Lebesgue measure. Therefore, for a fixed $c \in \ell^2(\mathbb{Z}^d)$, $\tilde{\mathcal{F}}(B^j c)$ goes to zero almost everywhere on $[0, 2\pi]^d$. Since

$$|\tilde{\mathcal{F}}(B^j c)(t)| \le \left(\frac{1}{2^d} \sum_{j \in \mathbb{N}_{2^d}} |m_1(t + \pi \nu_j)|^2 \right)^{1/2}, \ t \in [0, 2\pi]^d,$$

Equation (109) holds true by the Lebesgue dominated convergence theorem. Thus, we conclude that (80) holds.

Now, suppose that (106) does not hold. Note that $c \in \ell^2(\mathbb{Z}^d)$ is in the union of $\mathcal{N}(T^j)$, $j \in \mathbb{N}$ if and only if $A^{-1/2} c \in \mathcal{N}(H)$. We use Theorem 7 and (34) to get that

$$\bigcap_{j \in \mathbb{Z}} \mathcal{H}_{K_j} = \{ (A^{1/2} \Psi(\cdot), c)_{\ell^2(\mathbb{Z}^d)} : c \in (A^{1/2} \mathcal{N}(H))^{\perp} \}.$$

The above equation can be rewritten as (107). ■

We next present a characterization for $K_{\mathbb{Z}^d}$ to be a Riesz basis for $\mathcal{H}_K$.

Theorem 43 *Let K be defined by (94) through a matrix A satisfying (93) and a compactly supported continuous function φ on $\mathbb{R}^d$. Then $K_{\mathbb{Z}^d}$ is a Riesz basis for $\mathcal{H}_K$ if and only if the polynomial*

$$q(t) := \sum_{n \in \mathbb{Z}^d} \varphi(n) e^{i(n,t)}, \ \ t \in \mathbb{R}^d \tag{113}$$

has no zeros.

Proof By Theorem 13 and condition (93), $K_{\mathbb{Z}^d}$ is a Riesz basis for $\mathcal{H}_K$ if and only if $\tilde{\varphi}_m := [\varphi(m-n) : n \in \mathbb{Z}^d]$, for $m \in \mathbb{Z}^d$ form a Riesz basis for $\ell^2(\mathbb{Z}^d)$. Lemma 38 states that $\tilde{\varphi}_m$, $m \in \mathbb{Z}^d$ are linearly independent if there exists $t \in \mathbb{Z}^d$ such that $\varphi(t) \neq 0$. Therefore, $\{\tilde{\varphi}_m : m \in \mathbb{Z}^d\}$ is Riesz basis for $\ell^2(\mathbb{Z}^d)$ if and only if there exist $0 < \alpha \leq \beta < \infty$ such that for every $c \in \ell^2(\mathbb{Z}^d)$ there holds

$$\alpha \|c\|^2_{\ell^2(\mathbb{Z}^d)} \leq \sum_{m \in \mathbb{Z}^d} |(\tilde{\varphi}_m, c)_{\ell^2(\mathbb{Z}^d)}|^2 \leq \beta \|c\|^2_{\ell^2(\mathbb{Z}^d)}. \tag{114}$$

Since

$$\sum_{m \in \mathbb{Z}^d} |(\tilde{\varphi}_m, c)_{\ell^2(\mathbb{Z}^d)}|^2 = \frac{1}{(2\pi)^d} \int_{[0,2\pi]^d} \left| \sum_{n \in \mathbb{Z}^d} \varphi(n) e^{i(n,t)} \right|^2 \left| \sum_{n \in \mathbb{Z}^d} \overline{c_n} e^{i(n,t)} \right|^2 dt,$$

Equation (114) holds for all $c \in \ell^2(\mathbb{Z}^d)$ if and only if $\alpha \leq |q(t)|^2 \leq \beta$, $t \in [0,2\pi]^d$. The theorem hence follows from the continuity of q on $\mathbb{R}^d$. ■

We conclude this section by a result regarding a multiresolution analysis for $\mathcal{H}_K$.

Theorem 44 *Let φ be a nontrivial compactly supported continuous refinable function on $\mathbb{R}^d$, h satisfy (103), A satisfy (93), and K be defined by (94). Then $\mathcal{H}_K$ has a multiresolution analysis with $K_{\mathbb{Z}^d}$ being a Riesz basis for $\mathcal{H}_K$ if and only if there holds (100), (106), and the polynomial (113) has no zeros.*

Proof This result is a direct consequence of Theorems 37, 42 and 43. ■

7. A Discussion of Applications and Conclusions

For the completeness of the paper, in this section we discuss how refinable kernels can be used to efficiently update kernels for learning from increasing training data. Here we only use a simple learning example to illustrate the main points. The general case requires further substantial research and it will be reported on a different occasion.

In this special example, we assume that the input space X is $\mathbb{R}$ and that the *initial* training data set is given by $\mathbf{z} := \{(j, y_j) : j \in \mathbb{B}_m\} \subseteq X \times Y$, where we have set for each $n \in \mathbb{N}$, $\mathbb{B}_n := \{-n, \ldots, -1, 0, 1, \ldots, n\}$. Let K be a kernel on X and consider the loss function $Q(p,q) := |p-q|^2$, for $p, q \in \mathbb{C}$. This loss function is important in practice (for example, in regularization networks Evgeniou et al., 2000; Schölkopf and Smola, 2002; Vapnik, 1998). The predictor f from the training data $\mathbf{z}$ is hence the minimizer of

$$\min_{g \in \mathcal{H}_K} \sum_{j \in \mathbb{B}_m} |g(j) - y_j|^2 + \mu \|g\|^2_{\mathcal{H}_K},$$

where μ is a positive regularization parameter. The representer theorem (Kimeldorf and Wahba, 1971; Schölkopf et al., 2001; Schölkopf and Smola, 2002) ensures in this case that

$$f = \sum_{j \in \mathbb{B}_m} c_j K(\cdot, j). \tag{115}$$

Here, the vector $\mathbf{c} := [c_j : j \in \mathbb{B}_m]^T$ satisfies the linear system

$$(\mu I_{2m+1} + K[\mathbb{B}_m])\mathbf{c} = \mathbf{y}, \tag{116}$$

where I_n denotes the $n \times n$ identity matrix and $\mathbf{y} := [y_j : j \in \mathbb{B}_m]^T$.

Suppose that the initial training data set is updated to a new data set $\mathbf{z}' := \{(j/2, y'_j) : j \in \mathbb{B}_{2m}\}$ where $y'_{2j} = y_j$ for $j \in \mathbb{B}_m$. We divide $\mathbf{x}' := \mathbb{B}_{2m}/2$ into two disjoint subsets $\mathbf{x}'_1 := \mathbb{B}_m/2$ and $\mathbf{x}'_2 = \mathbb{B}_{2m}/2 \setminus \mathbb{B}_m/2$, and $\mathbf{y}' := \{y'_j : j \in \mathbb{B}_{2m}\}$ into $\mathbf{y}'_1$ and $\mathbf{y}'_2$, accordingly. For convenience, we set $\mathbf{x}'_2 := \{x'_{2,j} : j \in \mathbb{N}_{2m}\}$. If K is refinable on $X = \mathbb{R}$ then we update the kernel K to a new kernel $K_1 := \lambda K(2\cdot, 2\cdot)$. A new predictor f' is then obtained as the minimizer of

$$\min_{g \in \mathcal{H}_{K_1}} \sum_{j \in \mathbb{B}_{2m}} |g(j/2) - y'_j|^2 + \mu' \|g\|^2_{\mathcal{H}_{K_1}},$$

where μ' is an updated regularization parameter. By the representer theorem, we have that

$$f' = \sum_{j \in \mathbb{B}_m} c'_{1,j} K(\cdot, j/2) + \sum_{j \in \mathbb{N}_{2m}} c'_{2,j} K(\cdot, x'_{2,j}). \tag{117}$$

The above vectors $\mathbf{c}'_1 := [c'_{1,j} : j \in \mathbb{B}_m]^T$ and $\mathbf{c}'_2 := [c'_{2,j} : j \in \mathbb{N}_{2m}]^T$ satisfy the linear system

$$\begin{bmatrix} \mu' I_{2m+1} + K_1[\mathbf{x}'_1] & K_1[\mathbf{x}'_1, \mathbf{x}'_2] \\ K_1[\mathbf{x}'_2, \mathbf{x}'_1] & \mu' I_{2m} + K_1[\mathbf{x}'_2] \end{bmatrix} \begin{bmatrix} \mathbf{c}'_1 \\ \mathbf{c}'_2 \end{bmatrix} = \begin{bmatrix} \mathbf{y}'_1 \\ \mathbf{y}'_2 \end{bmatrix}, \tag{118}$$

where $K_1[\mathbf{x}'_1, \mathbf{x}'_2] := [K_1(p,q) : p \in \mathbf{x}'_1, q \in \mathbf{x}'_2]$.

It can be easily seen that the computational advantages offered by the refinability of the kernel include:

- **Efficient updating the kernel**. Kernels K_n in all scales can be efficiently updated from a refinable kernel K.
- **Improvement of the predictor**. Since $\mathcal{H}_K \preceq \mathcal{H}_{K_1}$, the class of candidate functions for the predictor is enlarged and the initial predictor f is in $\mathcal{H}_{K_1}$. Consequently, we can expect an improvement in approximation quality from the initial predictor to the new predictor f'.
- **Efficiency in setting up the coefficient matrix**. By refinability, we observe that the block matrix $K_1[\mathbf{x}'_1]$ in (118) satisfies the relation

  $$K_1[\mathbf{x}'_1] = K_1[\mathbb{B}_m/2] = \lambda K[\mathbb{B}_m].$$

  Therefore, the coefficient matrix of system (118) is an augmentation of that of system (116). As a result, we do not need to recompute the entries in the block $K_1[\mathbf{x}'_1]$.
- **Fast solving the linear system for the updated data set**. Because of the special structure in its coefficient matrix that results from the refinability of the kernel, the linear system can be solved efficiently by a fast algorithm analogous to the multi-level (wavelet) method (cf., Chen et al., 2005, 2006a,b).
- **Fast algorithms for processing the predictor**. Since the predictor (115) or (117) is expressed as a linear combination of the kernel, when the kernel is refinable we can use the (Mallat-type) decomposition and reconstruction algorithms developed in Section 4 to process the predictor.

Finally, we close this paper with the conclusion: Motivated by efficient mathematical learning, we introduce the notion of refinable kernels and characterize various types of refinable kernels. Examples of refinable kernels are presented. A special learning example illustrates that refinable kernels should provide computational advantages for solving various learning problems. Important examples of refinable kernels and their applications deserve further investigation.

Acknowledgments

Yuesheng Xu is supported in part by the US National Science Foundation under grants CCR-0407476 and DMS-0712827, by National Aeronautics and Space Administration under Cooperative Agreement NNX07AC37A, by the Natural Science Foundation of China under grants 10371122 and 10631080, and by the Education Ministry of the People's Republic of China under the Changjiang Scholar Chair Professorship Program through Sun Yat-sen University. He is also with the Academy of Mathematics and Systems Science, Chinese Academy of Sciences, Beijing 100080, P. R. China. Haizhang Zhang is supported by Syracuse University under a Syracuse University Graduate Fellowship.

References

U. Amato, A. Antoniadis and M. Pensky. Wavelet kernel penalized estimation for non-equispaced design regression. *Stat. Comput.*, 16: 37–55, 2006.

N. Aronszajn. Theory of reproducing kernels. *Trans. Amer. Math. Soc.*, 68: 337–404, 1950.

A. Beurling. On two problems concerning linear transformations in Hilbert space. *Acta Math.*, 81: 239–255, 1949.

S. Bochner. *Lectures on Fourier Integrals with an Author's Supplement on Monotonic Functions, Stieltjes Integrals, and Harmonic Analysis*. Annals of Mathematics Studies 42, Princeton University Press, New Jersey, 1959.

O. Bousquet and A. Elisseeff. Stability and generalization. *Journal of Machine Learning Research*, 2: 499–526, 2002.

A. S. Cavaretta, W. Dahmen and C. A. Micchelli. Stationary subdivision. *Mem. Amer. Math. Soc.*, 93, no. 453, 1991.

Q. Chen, C. A. Micchelli, S. Peng and Y. Xu. Multivariate filters banks having a matrix factorization. *SIAM J. Matrix Anal. Appl.*, 25: 517-531, 2003.

Q. Chen, C. A. Micchelli and Y. Xu. On the matrix completion problem for multivariate filter bank construction. *Adv. Comput. Math.*, 26: 173–204, 2007.

Z. Chen, B. Wu and Y. Xu. Multilevel augmentation methods for solving operator equations. *Numer. Math. J. Chinese Univ.*, 14: 31–55, 2005.

Z. Chen, B. Wu and Y. Xu. Multilevel augmentation methods for differential equations. *Adv. Comput. Math.*, 24: 213–238, 2006.

Z. Chen, Y. Xu and H. Yang. A multilevel augmentation method for solving ill-posed operator equations. *Inverse Problems*, 22: 155–174, 2006.

J. B. Conway. *A Course in Functional Analysis*. 2nd Edition, Springer-Verlag, New York, 1990.

F. Cucker and S. Smale. On the mathematical foundations of learning. *Bull. Amer. Math. Soc.*, 39: 1–49, 2002.

I. Daubechies. *Ten Lectures on Wavelets*. CBMS-NSF Regional Conference Series in Applied Mathematics 61, Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 1992.

R. J. Duffin and A. C. Schaeffer. A class of nonharmonic Fourier series. *Trans. Amer. Math. Soc.*, 72: 341–366, 1952.

T. Evgeniou, M. Pontil and T. Poggio. Regularization networks and support vector machines. *Adv. Comput. Math.*, 13: 1–50, 2000.

C. H. FitzGerald, C. A. Micchelli and A. Pinkus. Functions that preserve families of positive semidefinite matrices. *Linear Algebra Appl.*, 221: 83–102, 1995.

J. B. Gao, C. J. Harris and S. R. Gunn. On a class of support vector kernels based on frames in function Hilbert spaces. *Neural Comput.*, 13: 1975–1994, 2001.

L. Grafakos. *Classical and Modern Fourier Analysis*. Prentice Hall, New Jersey, 2004.

B. Grünbaum and G. C. Shephard. *Tilings and Patterns*. W. H. Freeman and Company, New York, 1989.

G. Kimeldorf and G. Wahba. Some results on Tchebycheffian spline functions. *J. Math. Anal. Appl.*, 33: 82–95, 1971.

S. Mallat. Multiresolution approximations and wavelet orthonormal bases of $L^2(R)$. *Trans. Amer. Math. Soc.*, 315: 69–87, 1989.

S. Mallat. *A Wavelet Tour of Signal Processing*. 2nd Edition, Academic Press, San Diego, CA, 1998.

Y. Meyer. *Wavelets and Operators*. Cambridge University Press, Cambridge, 1992.

C. A. Micchelli and M. Pontil. A function representation for learning in Banach spaces. In *Proceeding of the 17th Annual Conference on Learning Theory* (COLT 04), pages 255–269, Banff, Alberta, 2004.

C. A. Micchelli and M. Pontil. Learning the kernel function via regularization. *Journal of Machine Learning Research*, 6: 1099–1125, 2005.

C. A. Micchelli and M. Pontil. On learning vector-valued functions. *Neural Comput.*, 17: 177–204, 2005.

C. A. Micchelli, Y. Xu and P. Ye. Cucker Smale learning theory in Besov spaces. In *Advances in Learning Theory: Methods, Models and Applications*, pages 47–68, IOS Press, Amsterdam, The Netherlands, 2003.

C. A. Micchelli, Y. Xu and H. Zhang. Universal kernels. *Journal of Machine Learning Research*, 7: 2651–2667, 2006.

S. Mukherjee, P. Niyogi, T. Poggio and R. Rifkin. Learning theory: stability is sufficient for generalization and necessary and sufficient for empirical risk minimization. *Adv. Comput. Math.*, 25: 161–193, 2006.

R. Opfer. Multiscale kernels. *Adv. Comput. Math.*, 25: 357–380, 2006.

R. Opfer. Tight frame expansions of multiscale reproducing kernels in Sobolev spaces. *Appl. Comput. Harmon. Anal.*, 20: 357–374, 2006.

A. Rakotomamonjy and S. Canu. Frames, reproducing kernels, regularization and learning. *Journal of Machine Learning Research*, 6: 1485–1515, 2005.

A. Rakotomamonjy, X. Mary and S. Canu. Non-parametric regression with wavelet kernels. *Appl. Stoch. Models Bus. Ind.*, 21: 153–163, 2005.

W. Rudin. *Real and Complex Analysis*. 3rd Edition, McGraw-Hill, New York, 1987.

B. Schölkopf, R. Herbrich and A. J. Smola. A generalized representer theorem. In *Proceeding of the 14th Annual Conference on Computational Learning Theory and the 5th European Conference on Computational Learning Theory*, pages 416–426, Springer-Verlag, London, UK, 2001.

B. Schölkopf and A. J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, Cambridge, Mass, 2002.

J. Shawe-Taylor and N. Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, Cambridge, 2004.

S. Smale and D. X. Zhou. Estimating the approximation error in learning theory. *Anal. Appl.*, 1: 17–41, 2003.

S. Smale and D. X. Zhou. Shannon sampling and function reconstruction from point values. *Bull. Amer. Math. Soc.*, 41: 279–305, 2004.

S. Smale and D. X. Zhou. Learning theory estimates via integral operators and their approximations. *Constr. Approx.*, 26: 153–172, 2007.

I. Steinwart and C. Scovel. Fast rates for support vector machines using Gaussian kernels. In *Proceeding of the 18th Annual Conference on Learning Theory* (COLT 05), pages 279–294, Bertinoro, 2005.

V. N. Vapnik. *Statistical Learning Theory*. Wiley, New York, 1998.

G. Wahba. Support vector machines, reproducing kernel Hilbert spaces and the randomized GACV. In *Advances in Kernel Methods–Support Vector Learning*, pages 69–86, MIT Press, Cambridge, Mass, 1999.

C. Walder, K. I. Kim and B. Schölkopf. Sparse multiscale Gaussian process regression. Technical Report No. TR-162, Max Planck Institute for Biological Cybernetics, 2007.

C. Walder, B. Schölkopf and O. Chapelle. Implicit surface modelling with a globally regularised basis of compact support. *Computer Graphics Forum*, 25: 635–644, 2006.

Y. Ying and D. X. Zhou. Learnability of Gaussians with flexible variances. *Journal of Machine Learning Research*, 8: 249–276, 2007.

R. M. Young. *An Introduction to Nonharmonic Fourier Series*. Academic Press, New York, 1980.

B. Yu and H. Zhang. The Bedrosian identity and homogeneous semi-convolution equations. *J. Integral Equations Appl*., accepted, 2006.

T. Zhang. Statistical behavior and consistency of classification methods based on convex risk minimization. *Ann. Statis*., 32: 56–85, 2004.

D. X. Zhou. Density problem and approximation error in learning theory. *Preprint*, 2003.

Journal of Machine Learning Research 8 (2007) 2121-2123 Submitted 11/06; Revised 5/07; Published 9/07

A Complete Characterization of a Family of Solutions to a Generalized Fisher Criterion

Marco Loog[*] LOOG@DIKU.DK
Datalogical Institute
University of Copenhagen
Universitetsparken 1
DK-2100 Copenhagen Ø, Denmark

Editor: Marina Meila

Abstract

Recently, Ye (2005) suggested yet another optimization criterion for discriminant analysis and proposed a characterization of the family of solutions to this objective. The characterization, however, merely describes a part of the full solution set, that is, it is not *complete* and therefore not at all a characterization. This correspondence first gives the correct characterization and afterwards compares it to Ye's.

Keywords: linear discriminant analysis, Fisher criterion, small sample, characterization

1. Classical and New Criteria

Given N feature vectors of dimensionality n, a linear reduction of dimensionality, based on classical Fisher LDA, determines an $n \times d$ transformation matrix $\mathbf{L}$ that, for a given $d < K$, K the number of classes, maximizes the so-called Fisher criterion: $F(\mathbf{A}) = \mathrm{tr}((\mathbf{A}^t\mathbf{S}_W\mathbf{A})^{-1}(\mathbf{A}^t\mathbf{S}_B\mathbf{A}))$ or, equivalently, $F_0(\mathbf{A}) = \mathrm{tr}((\mathbf{A}^t\mathbf{S}_T\mathbf{A})^{-1}(\mathbf{A}^t\mathbf{S}_B\mathbf{A}))$. Here, $\mathbf{S}_B := \sum_{i=1}^K p_i(\mathbf{m}_i - \overline{\mathbf{m}})(\mathbf{m}_i - \overline{\mathbf{m}})^t$, $\mathbf{S}_W := \sum_{i=1}^K p_i\mathbf{S}_i$, and $\mathbf{S}_T = \mathbf{S}_B + \mathbf{S}_W$. The matrices $\mathbf{S}_B$, $\mathbf{S}_W$, and $\mathbf{S}_T$ are the so-called between-class, pooled within-class, and total covariance matrices. In addition, $\mathbf{m}_i$ is the mean vector of class i, p_i is the prior of class i, and the overall mean $\overline{\mathbf{m}}$ equals $\sum_{i=1}^k p_i\mathbf{m}_i$. Finally, $\mathbf{S}_i$ is the covariance matrix of class i.

A solution to these optimization problems can be obtained by means of a generalized eigenvalue decomposition, which Fukunaga (1990) relates to a simultaneous diagonalization of the two matrices involved (see also Campbell and Atchley, 1981). More common is it to apply a standard eigenvalue decomposition to $\mathbf{S}_T^{-1}\mathbf{S}_B$ (or $\mathbf{S}_W^{-1}\mathbf{S}_B$), resulting in an equivalent set of eigenvectors. The d columns of the optimal solution $\mathbf{L}$ are simply taken to equal the d eigenvectors corresponding to the d largest eigenvalues. It is known that this solution is not unique and the full class can be obtained by multiplying $\mathbf{L}$ to the right with nonsingular $d \times d$ matrices (see Fukunaga, 1990).

Clearly, if the total covariance $\mathbf{S}_T$ is singular, neither the generalized nor the standard eigenvalue decomposition can be readily employed. Directly or indirectly, the problem is that the matrix inverse $\mathbf{S}_T^{-1}$ does not exist, which is the typical situation when dealing with small samples. In an attempt to overcome this problem, Ye (2005) introduced a different criterion that is defined as

$$F_1(\mathbf{A}) = \mathrm{tr}((\mathbf{A}^t\mathbf{S}_T\mathbf{A})^+(\mathbf{A}^t\mathbf{S}_B\mathbf{A}))\,, \tag{1}$$

∗. Also at Nordic Bioscience Imaging, Hovegade 207, DK-2730 Herlev, Denmark.

where $^+$ denotes taking the Moore-Penrose generalized inverse of a matrix. Like for F_0, an optimal transform $\mathbf{L}$ is one that maximizes the objective F_1. Again, this solution is not unique.

2. Correct Characterization

For the full characterization of the set of solutions to Equation (1), initially the problem is looked at from a geometrical point of view (cf., Campbell and Atchley, 1981). It is assumed that the number of samples N is smaller than or equal to the feature dimensionality n. In the undersampled case, it is clear that all data variation occurs in an $N-1$-dimensional subspace of the original space.

To start with, a PCA of the data is carried out and the first $N-1$ principal components are rotated to the first $N-1$ axes of the n-dimensional space by means of a rotation matrix $\mathbf{R}$. This matrix consists of all (normalized) eigenvectors of $\mathbf{S}_T$ taken as its columns. After this rotation, new total and between-class covariance matrices, $\mathbf{S}'_T = \mathbf{R}^t\mathbf{S}_T\mathbf{R}$ and $\mathbf{S}'_B = \mathbf{R}^t\mathbf{S}_B\mathbf{R}$, are obtained. These matrices can be partitioned as follows: $\mathbf{S}'_T = \left(\begin{smallmatrix}\Sigma_T & \mathbf{0}\\ \mathbf{0} & \mathbf{0}\end{smallmatrix}\right)$ and $\mathbf{S}'_B = \left(\begin{smallmatrix}\Sigma_B & \mathbf{0}\\ \mathbf{0} & \mathbf{0}\end{smallmatrix}\right)$, where Σ_T and Σ_B are $N-1\times N-1$ covariance matrices and Σ_T is nonsingular and diagonal by construction. The between-class variation is obviously restricted to the $N-1$-dimensional subspace in which the total data variation takes place, therefore a same partitioning of $\mathbf{S}'_B$ is possible. However, the covariance submatrix Σ_B is not necessarily diagonal, neither does it have to be nonsingular. Basically, the PCA-based rotation $\mathbf{R}$ converts the initial problem into a more convenient one, splitting up the original space in an $N-1$-dimensional one in which "everything interesting" takes place and a remaining $n-N+1$-dimensional subspace in which "nothing happens at all".

Now consider F_1 in this transformed space and take a general $n\times d$ transformation matrix $\mathbf{A}$, which is partitioned in a way similar to the covariance matrices, that is,

$$\mathbf{A} = \begin{pmatrix}\mathbf{X}\\ \mathbf{Y}\end{pmatrix}. \tag{2}$$

Here, $\mathbf{X}$ is an $N-1\times d$-matrix and $\mathbf{Y}$ is of size $n-N+1\times d$. Taking this definition, the following holds (cf., Ye, 2005):

$$\begin{aligned}
F_1(\mathbf{A}) &= \mathrm{tr}((\mathbf{A}^t\mathbf{S}'_T\mathbf{A})^+(\mathbf{A}^t\mathbf{S}'_B\mathbf{A})) = \mathrm{tr}\left(\left(\begin{pmatrix}\mathbf{X}\\ \mathbf{Y}\end{pmatrix}^t\begin{pmatrix}\Sigma_T & \mathbf{0}\\ \mathbf{0} & \mathbf{0}\end{pmatrix}\begin{pmatrix}\mathbf{X}\\ \mathbf{Y}\end{pmatrix}\right)^+\left(\begin{pmatrix}\mathbf{X}\\ \mathbf{Y}\end{pmatrix}^t\begin{pmatrix}\Sigma_B & \mathbf{0}\\ \mathbf{0} & \mathbf{0}\end{pmatrix}\begin{pmatrix}\mathbf{X}\\ \mathbf{Y}\end{pmatrix}\right)\right)\\
&= \mathrm{tr}\left(\begin{pmatrix}\mathbf{X}^t\Sigma_T\mathbf{X} & \mathbf{0}\\ \mathbf{0} & \mathbf{0}\end{pmatrix}^+\begin{pmatrix}\mathbf{X}^t\Sigma_B\mathbf{X} & \mathbf{0}\\ \mathbf{0} & \mathbf{0}\end{pmatrix}\right) = \mathrm{tr}\left(\begin{pmatrix}(\mathbf{X}^t\Sigma_T\mathbf{X})^{-1} & \mathbf{0}\\ \mathbf{0} & \mathbf{0}\end{pmatrix}\begin{pmatrix}\mathbf{X}^t\Sigma_B\mathbf{X} & \mathbf{0}\\ \mathbf{0} & \mathbf{0}\end{pmatrix}\right)\\
&= \mathrm{tr}((\mathbf{X}^t\Sigma_T\mathbf{X})^{-1}(\mathbf{X}^t\Sigma_B\mathbf{X})) = F_0(\mathbf{X}).
\end{aligned}$$

From this it is immediate that a matrix $\mathbf{A}$ maximizes F_1 if and only if the submatrix $\mathbf{X}$ maximizes the original Fisher criterion in the lower-dimensional subspace. Moreover, if $\mathbf{L}$ is such a matrix maximizing F_1 in the PCA-transformed space, it is easy to check that $\mathbf{R}^{-1}\mathbf{L} = \mathbf{R}^t\mathbf{L}$ provides a solution to the original, general problem that has not been preprocessed by means of a PCA (see also Fukunaga, 1990). A characterization of the complete family of solutions can now be given.

Let $\Lambda\in\mathbb{R}^{N-1\times d}$ be an optimal solution of $F_0(\mathbf{X}) = \mathrm{tr}((\mathbf{X}^t\Sigma_T\mathbf{X})^{-1}(\mathbf{X}^t\Sigma_B\mathbf{X}))$. As already noted in Section 1, the full set of solutions is given by $\mathscr{F} = \{\Lambda\mathbf{Z}\in\mathbb{R}^{N-1\times d}\,|\,\mathbf{Z}\in\mathrm{GL}_d(\mathbb{R})\}$, where $\mathrm{GL}_d(\mathbb{R})$ denotes the general linear group of $d\times d$ invertible matrices. The previous paragraph essentially demonstrates that if $\mathbf{X}\in\mathscr{F}$, $\mathbf{A}$ in Equation (2) maximizes F_1. The matrix $\mathbf{Y}$ can be chosen ad

libitum. Now, the latter provides the solution in the PCA-transformed space and to solve the general problem we need to take the rotation back to the original space into account. All in all, this leads to the following complete family of solutions $\mathscr{L}$, maximizing F_1 in the original space:

$$\mathscr{L} = \left\{ \mathbf{R}^{\mathrm{t}} \begin{pmatrix} \Lambda \mathbf{Z} \\ \mathbf{Y} \end{pmatrix} \in \mathbb{R}^{n \times d} \,\middle|\, \mathbf{Z} \in \mathrm{GL}_d(\mathbb{R}), \mathbf{Y} \in \mathbb{R}^{n-N+1 \times d} \right\}, \tag{3}$$

where $\Lambda = \mathrm{argmax}_{\mathbf{X}} \mathrm{tr}((\mathbf{X}^{\mathrm{t}} \Sigma_T \mathbf{X})^{-1} (\mathbf{X}^{\mathrm{t}} \Sigma_B \mathbf{X}))$ and $\mathbf{R}^{\mathrm{t}}$ takes care of the rotation back.

3. Original Characterization

Though not noted by Ye (2005), his attempt to characterize the full set of solutions of Equation (1) is based on a simultaneous diagonalization of the three covariance matrices $\mathbf{S}_B$, $\mathbf{S}_W$, and $\mathbf{S}_T$ that is similar to the ideas described by Campbell and Atchley (1981) and Fukunaga (1990). Moreover, Golub and Van Loan (Theorem 8.7.1. 1996) can be readily applied to demonstrate that such simultaneous diagonalization is possible in the small sample setting. After the diagonalization step, partitioned between-class, pooled within-class, and total covariance matrices are considered. This partitioning is similar to the one employed in the previous section, which does not enforce matrices to be diagonal however.

In the subsequent optimization step, the classical Fisher criterion is maximized basically in the appropriate subspace, comparable to the approach described above, but in a mildly more involved and concealed way. For this, matrices of the form $\mathbf{R}^{\mathrm{t}} \binom{\mathbf{X}}{\mathbf{Y}}$ are considered, consider Equations (2) and (3). However, $\mathbf{Y}$ is simply the null matrix and the family of solutions $\mathscr{L}'$ provided is limited to

$$\mathscr{L}' = \left\{ \mathbf{R}^{\mathrm{t}} \begin{pmatrix} \Lambda \mathbf{Z} \\ \mathbf{0} \end{pmatrix} \in \mathbb{R}^{n \times d} \,\middle|\, \mathbf{Z} \in \mathrm{GL}_d(\mathbb{R}) \right\}.$$

Obviously, this is far from a complete characterization, especially when $N - 1 \ll n$ which is, for instance, typically the case for the data sets considered by Ye (2005).

Generally, the utility of a dimensionality reduction criterion, without additional constrains, depends on the efficiency over the full set of solutions. As Ye (2005) only considers two very specific instances from the large class of possibilities, it is unclear to what extent the new criterion really provides an efficient way of performing a reduction of dimensionality.

References

N. A. Campbell and W. R. Atchley. The geometry of canonical variate analysis. *Systematic Zoology*, 30(3):268–280, 1981.

K. Fukunaga. *Introduction to Statistical Pattern Recognition*. Academic Press, New York, 1990.

G. H. Golub and C. F. Van Loan. *Matrix Computations*. The Johns Hopkins University Press, third edition, 1996.

J. Ye. Characterization of a family of algorithms for generalized discriminant analysis on undersampled problems. *Journal of Machine Learning Research*, 6:483–502, 2005.

Journal of Machine Learning Research 8 (2007) 2125-2167 Submitted 11/06; Revised 4/07; Published 9/07

Transfer Learning via Inter-Task Mappings for Temporal Difference Learning

Matthew E. Taylor MTAYLOR@CS.UTEXAS.EDU
Peter Stone PSTONE@CS.UTEXAS.EDU
Yaxin Liu YXLIU@CS.UTEXAS.EDU
Department of Computer Sciences
The University of Texas at Austin
Austin, Texas 78712-1188

Editor: Michael L. Littman

Abstract

Temporal difference (TD) learning (Sutton and Barto, 1998) has become a popular reinforcement learning technique in recent years. TD methods, relying on function approximators to generalize learning to novel situations, have had some experimental successes and have been shown to exhibit some desirable properties in theory, but the most basic algorithms have often been found slow in practice. This empirical result has motivated the development of many methods that speed up reinforcement learning by modifying a task for the learner or helping the learner better generalize to novel situations. This article focuses on generalizing *across tasks*, thereby speeding up learning, via a novel form of transfer using handcoded task relationships. We compare learning on a complex task with three function approximators, a cerebellar model arithmetic computer (CMAC), an artificial neural network (ANN), and a radial basis function (RBF), and empirically demonstrate that directly transferring the *action-value function* can lead to a dramatic speedup in learning with all three. Using *transfer via inter-task mapping* (TVITM), agents are able to learn one task and then markedly reduce the time it takes to learn a more complex task. Our algorithms are fully implemented and tested in the RoboCup soccer Keepaway domain.

This article contains and extends material published in two conference papers (Taylor and Stone, 2005; Taylor et al., 2005).

Keywords: transfer learning, reinforcement learning, temporal difference methods, value function approximation, inter-task mapping

1. Introduction

Machine learning has traditionally been limited to training and testing on the same distribution of problem instances. However, humans are able to learn to perform well in complex tasks by utilizing principles learned in previous tasks. Few current machine learning methods are able to transfer knowledge between pairs of tasks, and none are able to transfer between a broad range of tasks to the extent that humans are. This article presents a new method for *transfer learning* in the *reinforcement learning* (RL) framework using *temporal difference* (TD) learning methods (Sutton and Barto, 1998), whereby an agent can learn faster in a *target task* after training on a different, typically less complex, *source task*.

TD learning methods have shown some success in many reinforcement learning tasks because of their ability to learn where there is limited prior knowledge and minimal environmental feedback.

However, the basic unenhanced TD algorithms, such as Q-Learning (Watkins, 1989) and Sarsa (Rummery and Niranjan, 1994; Singh and Sutton, 1996), have been found slow to produce near-optimal behaviors in practice. Many techniques exist (Selfridge et al., 1985; Colombetti and Dorigo, 1993; Asada et al., 1994) which attempt, with more or less success, to speed up the learning process. Section 9 will discuss in depth how our transfer learning method differs from other existing methods and can potentially be combined with them if desired.

In this article we introduce *transfer via inter-task mapping* (TVITM), whereby a TD learner trained on one task with *action-value function* RL can learn faster when training on another task with related, but different, state and action spaces. TVITM thus enables faster TD learning in situations where there are two or more similar tasks. This transfer formulation is analogous to a human being told how a novel task is related to a known task, and then using this relation to decide how to perform the novel task. The key technical challenge is mapping an action-value function—the expected return or value of taking a particular action in a particular state—in one representation to a meaningful action-value function in another, typically larger, representation. It is this transfer functional which defines transfer in the TVITM framework.

In stochastic domains with continuous state spaces, agents will rarely (if ever) visit the same state twice. It is therefore necessary for learning agents to use *function approximation* when estimating the action-value function. Without some form of approximation, an agent would only be able to predict a value for states that it had previously visited. In this work we are primarily concerned with a different kind of generalization. Instead of finding similarities between different states, we focus on exploiting similarities between different *tasks*.

The primary contribution of this article is an existence proof that there are domains in which it is possible to construct a mapping between tasks and thereby speed up learning by transferring an action-value function. This approach may seem counterintuitive initially: the action-value function is the learned information which is directly tied to the particular task it was learned in. Nevertheless, we will demonstrate the efficacy of using TVITM to speed up learning in agents across tasks, irrespective of the representation used by the function approximator. Three different function approximators (as defined in Section 4.3), a CMAC, an ANN, and an RBF, are used to learn a single reinforcement learning problem. We will compare their effectiveness and demonstrate why TVITM is promising for future transfer studies.

The remainder of this article is organized as follows. Section 2 formally defines TVITM. Section 3 gives an overview of the tasks over which we quantitatively test our transfer method. Section 4 gives details of learning in our primary domain, robot soccer Keepaway. Section 5 describes how we perform transfer in our selected tasks. Sections 6 and 7 present the results of our experiments. Section 8 discusses some of their implications and future work. Section 9 details other related work while contrasting our methods and Section 10 concludes.

2. Transfer via Inter-Task Mapping

TVITM is defined for value function reinforcement learners. Thus, to formally define how to use our transfer method we first briefly review the general reinforcement learning framework that conforms to the generally accepted notation for *Markov decision processes* (MDP) (Puterman, 1994).

In an MDP, there is some set of possible perceptions of the current state of the world, S, and a learner has an initial starting state, $s_{initial}$. An agent's knowledge of the current state of its environment from observation, $s \in S$ is a vector of k *state variables*, so that $s = \langle x_1, x_2, \ldots, x_k \rangle$. There is a

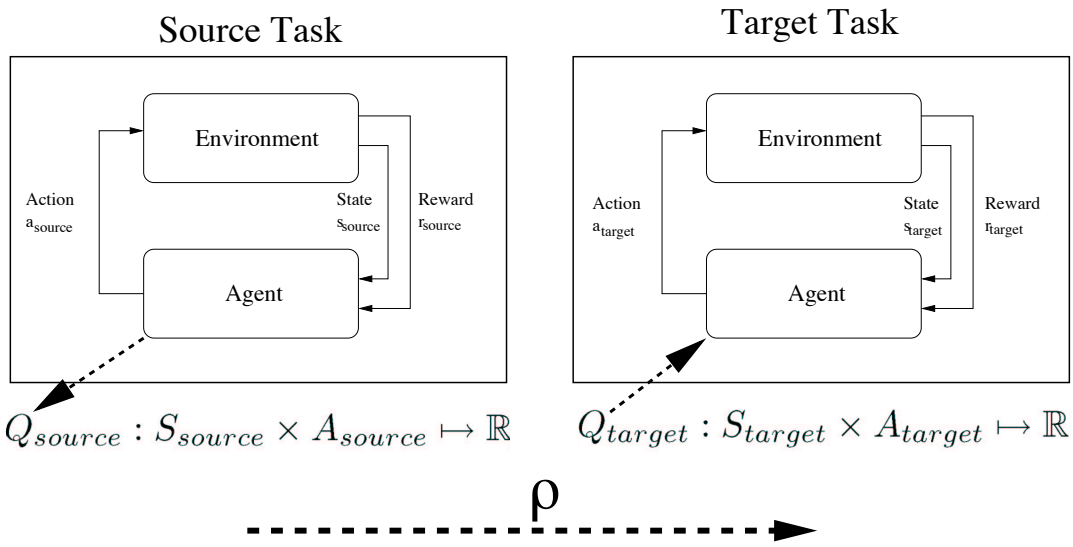


Figure 1: ρ is a functional that transforms a state-action function Q from one task so that it is applicable in a second task with different state and action spaces.

set of actions, A, which the agent can perform. The reward function, $R : S \mapsto \mathbb{R}$, maps each state of the environment to a single number which is the instantaneous reward achieved for reaching the state. The transition function, $T : S \times A \mapsto S$, takes a state and an action and returns the state of the environment after the action is performed. If transitions are non-deterministic the transition function is a probability distribution function. A learner is able to sense the current state, s, and typically knows A and what state variables comprise S. However, it does not know R, how it is rewarded for moving between states, or T, how actions move the agent between states.

A learner chooses which action to take in a given perceived environmental state by using a policy, $\pi : S \mapsto A$. π is modified by the learner over time to improve performance, the expected total reward accumulated, and it completely defines the behavior of the learner in an environment. In the general case the policy can be stochastic. The success of an agent is determined by how well it maximizes the total reward it receives in the long run while acting under some policy π. An *optimal policy*, π^*, is a policy that maximizes the expectation of this value. Any reasonable learning algorithm attempts to modify π over time so that the agent's performance approaches that of π^* in the limit. Value function reinforcement learning relies on learning a value function $V : S \mapsto \mathbb{R}$ so that the learner is able to estimate the total discounted reward that would be accumulated from moving to state s and then following the current policy π. In practice, the action-value function $Q : S \times A \mapsto \mathbb{R}$ is often learned, which frees the learner from having to explicitly model the transition function. If the action-value function is optimal (i.e., $Q = Q^*$), π^* can be followed by always selecting the optimal action a, which is the action with the largest value of $Q(s,a)$ in the current state.

In this article we consider the general case where the state features in the source and target tasks are different ($S_{source} \neq S_{target}$), and/or the actions in the source and target tasks are different ($A_{source} \neq A_{target}$). To use the learned action-value function from the source task $Q_{(source,final)}$ as the initial action-value function for a TD learner in a target task, we must transform the action-value function so that it can be directly applied to the new state and action space. This transformed action-

value function may not provide immediate improvement over acting randomly in the target task, but it should bias the learner so that it is able to learn the target task faster than if it were learning without transfer.

A transfer functional $\rho(Q)$ will allow us to apply a policy in a new task (see Figure 1). The policy transform functional ρ needs to modify the action-value function so that it accepts S_{target} as inputs and allows for A_{target} to be outputs. A policy generally selects the action which is believed to accumulate the largest expected total reward; the problem of transforming a policy between two tasks therefore reduces to transforming the action-value function. Defining ρ to do this correctly is the key technical challenge to enable general TVITM.

2.1 Constructing a Transfer Functional

Given an arbitrary pair of unknown tasks and no experience in the pair of tasks, one could not hope to correctly define ρ, the transfer functional (for example, there are certainly pairs of tasks which have no relationship and thus mastery in one task would not lead to improved performance in the other). For our transfer method to succeed, not only must the two tasks be related, but we should be able to characterize *how* they are related. We represent these relations as a pair of inter-task mappings, denoted χ_X and χ_A. State variables in the target task are mapped via χ_X to the most similar state variable in the source task:

$$\chi_X(x_{i,target}) = x_{j,source}.$$

Similarly, χ_A maps each action in the target task to the most similar action in the source tasks:

$$\chi_A(a_{i,target}) = a_{j,source}.$$

χ_X and χ_A, mappings from the target task to the source task, are used to construct ρ, a transfer functional from the source task to the target task (see Figure 2). Note that χ_X and χ_A are defined only once for a pair of tasks, while multiple ρs (one for each type of function approximator employed by our learning agents), are constructed from this single pair of inter-task mappings. In this article we take χ_X and χ_A as given; learning them autonomously is an important goal of future work.

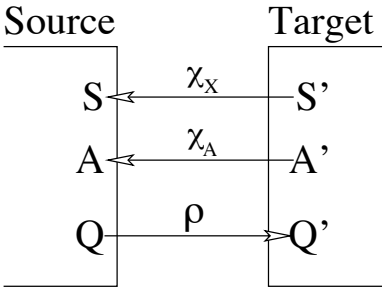


Figure 2: χ_X and χ_A are mappings from a target to a source task; ρ maps an action-value function from a source to a target task.

Thus, given χ_X, χ_A, and a learned action-value function Q_{source}, we can create an initial action-value function Q_{target}. The details of ρ depend on the particular function approximators used in the source and target task. In Sections 5.3 and 5.4 we construct three different ρ functionals from χ_X and χ_A for the RoboCup Soccer Keepaway domain.

It may seem counterintuitive that low-level action-value function information is able to speed up learning across different tasks. Often transfer techniques attempt to abstract knowledge so that it is applicable to more general tasks. For instance, an agent could be trained to balance a pole on a cart and then be asked to balance a pair of poles on a cart. An example of abstract knowledge in this domain would be things like "avoid hitting the end of the track," "it is better to have the pole near vertical," etc. Instead of trying to transfer higher level information about a source task into a target task, we instead focus on information contained in individual weights within function approximators. In this example, such weights which would contain specific information such as how fast to move the cart to the left when a pole was at a particular angle. Weights that encode this type of low level knowledge are the most task-specific part of the learner's knowledge, but it is exactly these domain-dependant details that allow us to achieve significant speedups on similar tasks.

2.2 Evaluation of Transfer

There are many possible ways to measure the effectiveness of transfer, including:

1. Asymptotic Performance: Measure the performance after convergence in the target task.
2. Initial Performance: Measure the initial performance in the target task.
3. Total Reward: Measure the total accumulated reward during training in the target task.
4. Area Ratio: Measure the area between the transfer and non-transfer learning curves.
5. Time-to-Threshold: Measure the time needed to reach a performance threshold in the target task.

This section discusses these five different testing criteria and argues that the time-to-threshold metric is most appropriate for evaluating TVITM in our experimental domain.

One could examine the asymptotic performance of a learned policy. Such a metric would compare the average reward achieved after learning both with and without transfer. Leveraging source task knowledge may allow a learner to reach a higher asymptote, but it may be difficult to tell when the learner has converged, and convergence may take prohibitively long. Additionally, in applications of reinforcement learning we are often interested in the time required, not simply the performance of a learner with infinite time. Lastly, it is not uncommon for different learners to converge to the same asymptotic performance on a given task, making them indistinguishable in terms of the asymptotic performance metric.

A second measure of transfer is to look at the initial performance in a target task. Learned source task knowledge may be able to improve initial target task performance relative to learning the target task without transfer. While such an initial performance boost is appealing, we argue in Section 6 that this goal may often be infeasible to achieve in practice. Further, because we are primarily interested in the learning process of agents in pairs of tasks, it makes sense to concentrate on the rate of learning in the target task.

A third possible measure is that of the total reward accumulated during training. By measuring the total reward over some amount of training, we are able to quantify how much reward the agent accumulates in a certain amount of time. Transfer may allow an agent to accumulate more reward in

the target task when compared to the non-transfer case; better initial performance and faster learning would help agents achieve more on-line reward. TD methods are not guaranteed to converge with function approximation and even when they do, learners do not always converge to the same performance levels. If the time considered is long enough, a learning method which achieves very fast learning will "lose" to a learning method which learns very slowly but eventually plateaus at a slightly higher performance level. Thus this metric is most appropriate for tasks that have a defined time limit for learning. However, it is more common to think of learning until some performance is reached (if ever), rather than specifying the amount of time, computational complexity, or sample complexity *a priori*.

A fourth measure of transfer efficacy is that of the ratio of the areas defined by two learning curves. Consider two learning curves: one that uses transfer, and one that does not. Assuming that the transfer learner is able to learn faster or reach a higher performance, the area under the transfer curve will be greater than the area under the non-transfer curve. The ratio

$$r = \frac{\textit{area under curve with transfer - area under curve without transfer}}{\textit{area under curve without transfer}}$$

gives us a metric for how much transfer improves learning. This metric is most appropriate if the same eventual performance is achieved, or there is a predetermined time for the task. Otherwise the ratio will directly depend on the length of time considered for the two curves. In the tasks we consider, the learners that use transfer and the learners that learn without transfer do not always plateau to the same performance, nor is there a defined task length.

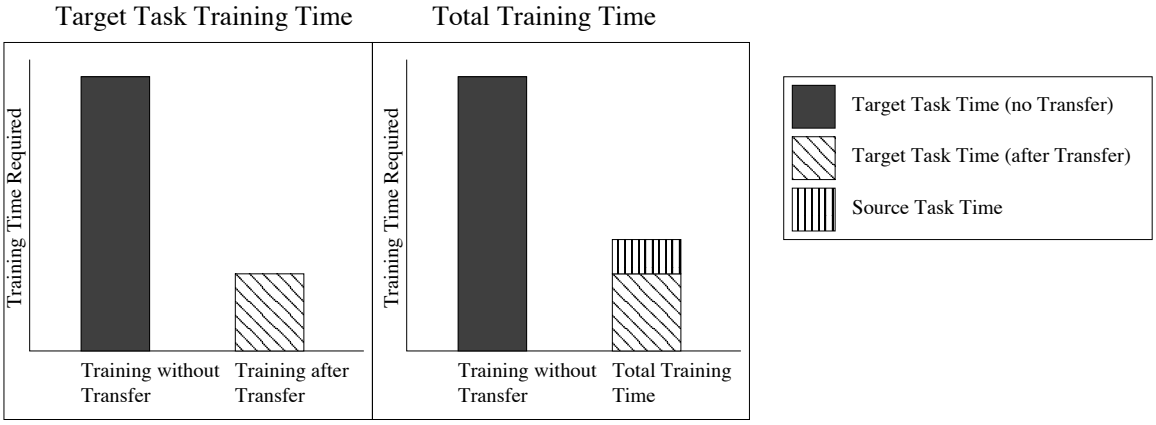


Figure 3: In this article we evaluate transfer by both considering the training time in the target task (left) and by considering the total time spent training in both tasks (right).

For these reasons we use the time-to-threshold metric. After preliminary experiments are conducted, thresholds for analysis are chosen such that all trials must learn for some amount of time before reaching the performance threshold, and most trials are able to eventually reach the threshold. We will show in Section 6 that given a $Q_{(source,final)}$, the training time for the learner in the target task to reach some performance threshold decreases when initializing $Q_{(target,initial)}$ with $\rho(Q_{(source,final)})$. This criterion is relevant when the source task is given and is of interest in its own right or if $Q_{(source,final)}$ can be used repeatedly to speed up multiple related tasks (see Figure 3).

A stronger measure of success that we will also use is that the training time for *both* tasks using TVITM is shorter than the training time to learn just the target task without transfer. This criterion is relevant when the source task is created for the sole purpose of speeding up learning with transfer and $Q_{(source,final)}$ is not reused.

3. Testbed Domains

This section introduces the Keepaway task, the testbed domain where we empirically evaluate our transfer method, and use as a running example throughout the rest of the article. We also introduce the Knight Joust, a task which we will later use as a supplemental source task from which to transfer into Keepaway.

3.1 The Keepaway Task

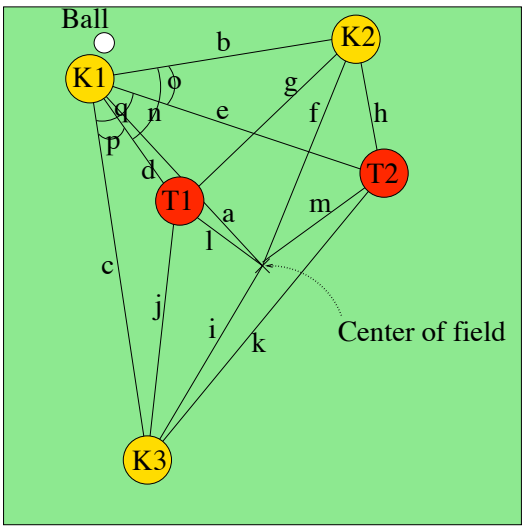
RoboCup simulated soccer is well understood, as it has been the basis of multiple international competitions and research challenges. The multiagent domain incorporates noisy sensors and actuators, as well as enforcing a hidden state so that agents only have a partial world view at any given time. While previous work has attempted to use machine learning to learn the full simulated soccer problem (Andre and Teller, 1999; Riedmiller et al., 2001), the complexity and size of the problem have so far proven intractable. However, many of the RoboCup subproblems have been isolated and solved using machine learning techniques, including the task of playing Keepaway. By focusing on the smaller task of Keepaway we are able to use reinforcement learning to learn an action-value function for a more complex task, establish that TVITM provides considerable benefit, and hold the required computational resources to manageable levels.

Since late 2002, the *Keepaway* task has been part of the official release of the open source RoboCup Soccer Server used at RoboCup (starting with version 9.1.0). Agents in the simulator (Noda et al., 1998) receive visual perceptions every 150 *msec* indicating the relative distance and angle to visible objects in the world, such as the ball and other agents. They may execute a primitive, parameterized action such as `turn(`*angle*`)`, `dash(`*power*`)`, or `kick(`*power*`,`*angle*`)` every 100 *msec*. Thus the agents must sense and act asynchronously. Random noise is injected into all sensations and actions. Individual agents must be controlled by separate processes, with no inter-agent communication permitted other than via the simulator itself, which enforces communication bandwidth and range constraints. Full details of the simulator are presented in the server manual (Chen et al., 2003).

When started in a special mode, the simulator enforces the rules of the Keepaway task, as described below, instead of the rules of full soccer. In particular, the simulator places the players at their initial positions at the start of each *episode* and ends an episode when the ball leaves the play region or is taken away. In this mode, the simulator also informs the players when an episode has ended and produces a log file with the duration of each episode.

Keepaway is a subproblem of RoboCup simulated soccer in which one team—the *keepers*—attempts to maintain possession of the ball within a limited region while another team—the *takers*—attempts to steal the ball or force it out of bounds, ending an episode. Whenever the takers take possession or the ball leaves the region, the episode ends and the players are reset for another episode (with the keepers being given possession of the ball again). Standard parameters of the task include the size of the region, the number of keepers, and the number of takers. Other parameters such as player speed, player kick speed, player vision capabilities, sensor noise, and actuator noise,

are all adjustable. This paper will use standard settings with the exception of a set of experiments in Section 7.1 that uses different kick speed actuators. Figure 4 shows a diagram of 3 keepers and 2 takers (3 vs. 2).[1]



a. dist(K1,C)
b. dist(K1,K2)
c. dist(K1,K3)
d. dist(K1,T1)
e. dist(K1,T2)
f. dist(K2,C)
g. dist(K2.T1)
h. dist(K2,T2)
i. dist(K3,C)
j. dist(K3,T1)
k. dist(K3,T2)
l. dist(T1,C)
m. dist(T2,C)
n. ang(K2,K1,T1)
o. ang(K2,K1,T2)
p. ang(K3,K1,T1)
q. ang(K3,K1,T2)

Figure 4: This diagram depicts the distances and angles used to construct the 13 state variables used for learning with 3 keepers and 2 takers. Relevant objects are the 5 players and the center of the field, C. All 13 state variables are enumerated later in Table 1.

When Keepaway was introduced as a testbed (Stone and Sutton, 2002), a standard task was defined. All our experiments are run on a code base derived from version 0.6 of the benchmark Keepaway implementation[2] (Stone et al., 2006) and the RoboCup Soccer Server version 9.4.5.

Our setup is similar to past research in Keepaway (Stone et al., 2005), which showed that Sarsa with CMAC function approximation can learn well in this domain. On a $25m \times 25m$ field, three keepers are initially placed near three corners of the field and a ball is placed near one of the keepers. The two takers are placed in the fourth corner. When the episode starts, the three keepers attempt to keep control of the ball by passing among themselves and moving to open positions. The keeper with the ball has the option to either pass the ball to one of its two teammates or to hold the ball. In this task $A = \{$*hold, pass to closest teammate, pass to second closest teammate*$\}$. S is defined by 13 state variables, as shown in Figure 4. When a taker gains control of the ball or the ball is kicked out of the field's bounds the episode is finished. The reward to the learning algorithm is the number of time steps the ball remains in play after an action is taken. After an episode ends, the next starts with a random keeper placed near the ball.

3.2 Knight Joust

Knight Joust is a variation on a previously introduced task (Taylor and Stone, 2007) situated in the grid world domain. In this task the player begins on one end of a $25m \times 25m$ board, the opponent begins on the other, and the players alternate moves. The player's goal is to reach the opposite

1. Flash files illustrating the task are available at `http://www.cs.utexas.edu/~AustinVilla/sim/Keepaway/`.
2. Released players are available at `http://www.cs.utexas.edu/~AustinVilla/sim/Keepaway/`.

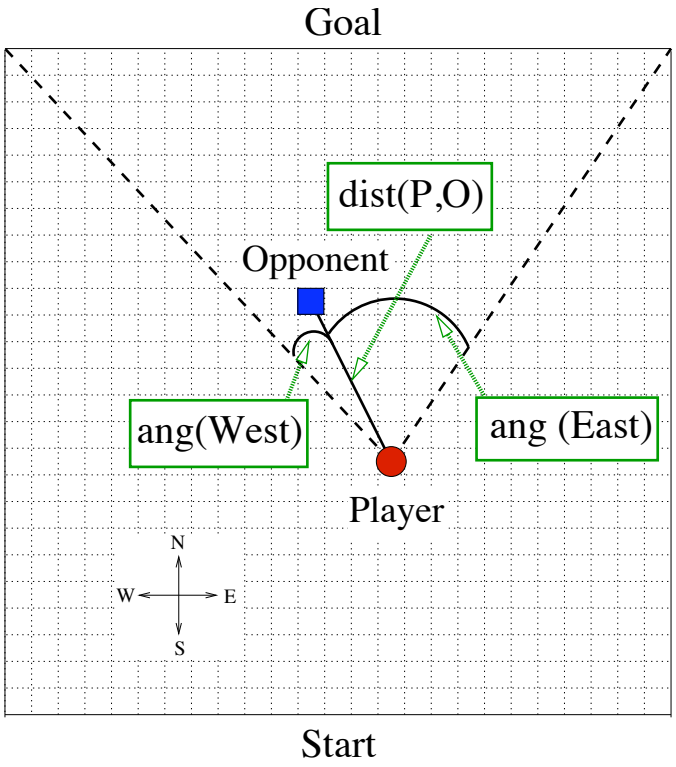


Figure 5: Knight Joust: The player attempts to reach the goal end of a a 25 $\times$ 25 grid-world while the opponent attempts to touch the player.

end of the board without being touched by the opponent (see Figure 5); the episode ends if the player reaches the goal line or the opponent is on the same square as the player. The state space is discretized into 1m squares and there is no noise in the perception. The player's state variables are composed of the distance from the player to the opponent, and two angles which describe how much of the goal line is viewable by the player.

The player may deterministically move one square North or perform a knight's jump where the player moves one square North and two West or two East: $A = \{Forward, Jump_W, Jump_E\}$. The opponent follows a fixed stochastic policy which allows it to move in any of the 8 directions. Given the start state and size of the board, an opponent that acted optimally would always prevent the player from reaching the goal line. In order to allow the player to reach the goal line with a learned policy, we restrict the opponent's motion so that 10% of the time, when it attempts to move East or West, it fails. 20% of the time, when it attempts to move North, it fails (the opponent never fails when it attempts to move South). These movement failure probabilities were selected after initial experiments showed that this opponent policy generally prevented the player from reaching the goal before training but allowed the player to reach the goal line with a high probability after learning with Sarsa. The opponent's policy is as follows:

if opponent is E of player **then**
  Move W with probability 0.9
else if opponent is W of player **then**
  Move E with probability 0.9
if opponent is N of player **then**
  Move S with probability 1.0
else if opponent is S of player **then**
  Move N with probability 0.8

The player receives a reward of +20 every time it takes the forward action, 0 if either knight jump action is taken, and an additional +20 upon reaching the goal line. The player uses Sarsa with a Q-value table to learn in this task. While this task is quite different from Keepaway, there are some similarities, such as favoring larger distances between player and opponent. This domain is much simpler than Keepaway and an agent takes roughly 20 seconds of wall-clock time (roughly 50,000 episodes) to plateau in our Java-based simulation.

4. Learning Keepaway

TVITM aims to improve learning in the target task based on prior learning in the source, and therefore a prerequisite is that both source and target tasks are learnable. In this section we outline how tasks in the Keepaway domain are learned using Sarsa.

4.1 Sarsa

Sarsa is a TD method that learns to estimate the action-value function by backing up the received rewards through time. Sarsa is an acronym for State Action Reward State Action, describing the 5-tuple needed to perform the update: $(s_t, a_t, r, s_{t+1}, a_{t+1})$, where s_t, a_t are the the agent's current state and action, r is the immediate reward the agent receives from the environment, and s_{t+1}, a_{t+1} are the agent's subsequent state and chosen action. After each action, action values are updated according to the following rule:

$$Q(s_t, a_t) \leftarrow (1-\alpha)Q(s_t, a_t) + \alpha(r + Q(s_{t+1}, a_{t+1})) \tag{1}$$

where α is the learning rate. Note that if the task is non-episodic we need to include an extra discount factor to weigh immediate rewards more heavily than future rewards.

Like other TD methods, Sarsa estimates the value of a given state-action pair by bootstrapping off the estimates of other such pairs. In particular, the value of a given state-action pair (s_t, a_t) can be estimated as $r + Q(s_{t+1}, a_{t+1})$, which is the value of the subsequent state-action pair (s_{t+1}, a_{t+1}) plus the immediate reward received during the transition. Sarsa's update rule takes the old action-value estimate $Q(s_t, a_t)$, and moves it incrementally closer towards this new estimate. The learning rate parameter α controls the size of these increments. Ideally, these action-value estimates will become more accurate over time and the agent's policy will steadily improve.

4.2 Framing the RL Problem

As described by Stone et al. (2005), the Keepaway problem maps fairly directly onto the discrete-time, episodic, reinforcement-learning framework. As a way of incorporating domain knowledge,

the learners choose not from the simulator's primitive actions but from a set of higher-level macro-actions implemented as part of the player. These macro-actions can last more than one time step and the keepers have opportunities to make decisions only when an on-going macro-action terminates. To handle such situations, it is convenient to treat the problem as a *semi-Markov decision process*, or SMDP (Puterman, 1994; Bradtke and Duff, 1995). The agents make decisions at discrete SMDP time steps (when macro-actions are initiated and terminated).

The keepers learn in a constrained policy space: they have the freedom to decide which action to take only when in possession of the ball. A keeper in possession may either hold the ball or pass to one of its teammates. Therefore the number of actions from which the keeper with the ball may choose is equal to the number of keepers in the task. Keepers not in possession of the ball are required to execute the Receive macro-action in which the player who can reach the ball the fastest goes to the ball and the remaining players follow a handcoded strategy to try to get open for a pass.

When training the keepers, the behavior of the takers is "hard-wired" and relatively simple. The two takers that are closest to the ball go directly toward it. Note that a single keeper can hold the ball indefinitely from a single taker by constantly keeping its body between the ball and the taker. The remaining takers, if present, try to block open passing lanes.

The keepers learn which action to take when in possession of the ball by using episodic SMDP Sarsa(λ) (Sutton and Barto, 1998), to learn their task.[3] The episode consists of a sequence of states, macro-actions, and rewards. We choose episode duration as the performance measure for this task: the keepers attempt to maximize it while the the takers try to minimize it. Since we want the keepers to maintain possession of the ball for as long as possible, the reward in the Keepaway task is simply the number of time steps the ball remains in play after a macro-action is initiated. Learning attempts to discover an optimal action-value function that maps state-action pairs to expected time steps until the episode will end.

As more players are added to the task, Keepaway becomes harder for the keepers because the field becomes more crowded. As more takers are added there are more players to block passing lanes and chase down any errant passes. As more keepers are added, the keeper with the ball has more passing options but the average pass distance is shorter. This reduced distance forces more passes and often leads to more errors because of the noisy actuators and sensors. For this reason, keepers in 4 vs. 3 (i.e., 4 keepers and 3 takers) take longer to learn an optimal control policy than in 3 vs. 2. The average episode length of the best policy for a constant field size also decreases when adding an equal number of keepers and takers. The time needed to learn a policy with performance roughly equal to a handcoded solution roughly doubles as each additional keeper and taker is added (Stone et al., 2005). In our experiments we set the agents to have a 360° field of view. Although agents do also learn with a more realistic 90° field of view, allowing the agents to see 360° speeds up the rate of learning, enabling more experiments. Additionally, 360° vision also increases the learned hold times in comparison to learning with the limited 90° vision.

For the purposes of this article, it is particularly important to note the state variables and action possibilities used by the learners. The keepers' states consist of distances and angles of the keepers $K_1 - K_n$, the takers $T_1 - T_m$, and the center of the playing region C (see Figure 4 and Table 1). Keepers and takers are ordered by increasing distance from the ball, leading to an indexical representation. Note that as the number of keepers n and the number of takers m increase, the number of state variables also increases so that the more complex state can be fully described. S must

3. In previous experiments we found that setting $\lambda = 0$ produced the best learning results and "Sarsa(0)" is synonymous with "Sarsa."

3 vs. 2 State Variables

State Variable	Description
$dist(K_1,C)$	Distance from keeper with ball to center of field
$dist(K_1,K_2)$	Distance from keeper with ball to closest teammate
$dist(K_1,K_3)$	Distance from keeper with ball to second closest teammate
$dist(K_1,T_1)$	Distance from keeper with ball to closest taker
$dist(K_1,T_2)$	Distance from keeper with ball to second closest taker
$dist(K_2,C)$	Distance from closest teammate to center of field
$dist(K_3,C)$	Distance from second closest teammate to center of field
$dist(T_1,C)$	Distance from closest taker to center of field
$dist(T_2,C)$	Distance from second closest taker to center of field
$\text{Min}(dist(K_2,T_1), dist(K_2,T_2))$	Distance from nearest teammate to its nearest taker
$\text{Min}(dist(K_3,T_1), dist(K_3,T_2))$	Distance from second nearest teammate to its nearest taker
$\text{Min}(ang(K_2,K_1,T_1),$ $ang(K_2,K_1,T_2)$	Angle of passing lane from keeper with ball to closest teammate
$\text{Min}(ang(K_3,K_1,T_1),$ $ang(K_3,K_1,T_2)$	Angle of passing lane from keeper with ball to second closest teammate

Table 1: This table lists all state variables used for representing the state of 3 vs. 2 Keepaway. Note that the state is ego-centric for the keeper with the ball and rotationally invariant.

change (e.g., there are more distances to players to account for) and $|A|$ increases as there are more teammates for the keeper with possession of the ball to pass to.

4.3 Function Approximation

Continuous state variables combined with noise necessitate some form of function approximation for the action-value function: an agent will rarely visit the same state twice, with the possible exception of an initial start state. In this article we use three distinct function approximators and show that all are able to learn Keepaway, as well as use our transfer methodology (see Figure 6). In one implementation, we use linear tile-coding function approximation, also known as a CMAC (cerebellar model arithmetic computer), which has been successfully used in many reinforcement learning systems (Albus, 1981), including past Keepaway research (Stone et al., 2005). A second uses radial basis function approximation (RBF) (Sutton and Barto, 1998). The third implementation uses artificial neural networks (ANN), another method for function approximation that has had some notable past successes (Tesauro, 1994; Crites and Barto, 1996).

A CMAC takes arbitrary groups of continuous state variables and lays infinite, axis-parallel tilings over them. Using this method we are able to discretize the continuous state space by using tilings while maintaining the capability to generalize via multiple overlapping tilings. Increasing the tile widths allows better generalization while increasing the number of tilings allows more accurate representations of smaller details. The number of tiles and width of the tilings are handcoded: this sets the center, c_i, of each tile and dictates which state values will activate which tiles. The function approximation is learned by changing how much each tile contributes to the output of the function approximator. Thus, the output from the CMAC is the computed sum:

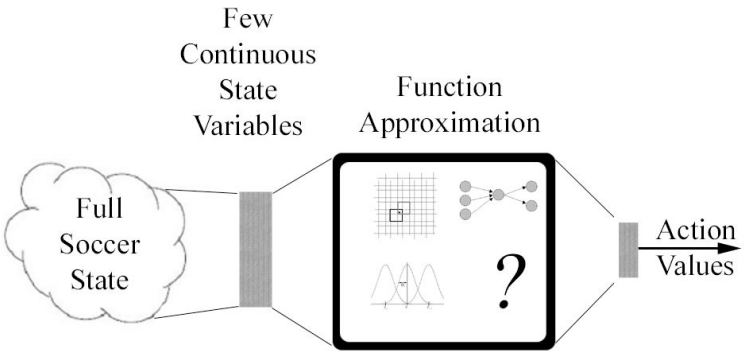


Figure 6: Function approximation is necessary for agents interacting with a continuous world. This article examines three different function approximators for Keepaway but many different methods could in principle be used by a transfer learner.

$$\hat{f}(x) = \sum_i w_i f_i(x) \tag{2}$$

but only tiles which are activated by the current state feature contribute to the sum:

$$f_i(x) = \begin{cases} 1, & \text{if tile i is activated} \\ 0, & \text{otherwise.} \end{cases}$$

By default, all the CMAC's weights are initialized to zero. This approach to function approximation in the RoboCup soccer domain has been detailed previously (Stone et al., 2005). We use one-dimensional tilings so that each state variable is tiled independently, but the principles apply in the n-dimensional case. For each variable, 32 tilings were overlaid, each offset from the others by by $\frac{1}{32}$ of a tile width. For each tiling, the current state activates a single tile. In 3 vs. 2, there are 32 tiles active for each state variable and $13 \times 32 = 416$ tiles activated in total. The tile widths are defined so that the distance state features have a width of roughly 3.0 meters and tiles for angle state features are roughly 10.0 degrees. In this work we do not vary these settings but set them to agree with past work.

RBF function approximation is a generalization of the tile coding idea to continuous functions (Sutton and Barto, 1998) and their application in Keepaway have been introduced elsewhere (Stone et al., 2006). When considering a single state variable, an RBF approximator is a linear function approximator:

$$\hat{f}(x) = \sum_i w_i f_i(x) \tag{3}$$

where the basis functions have the form:

$$f_i(x) = \phi(|x - c_i|) \tag{4}$$

x is the value of the current state variable, c_i is the center of feature i (which is unchanged from the CMAC, Equation 2), and w_i represents weights that can be modified over time by a learning algorithm. Here we set the features to be evenly spaced Gaussian radial basis functions, where:

$$\phi(x) = \exp(-\frac{x^2}{2\sigma^2}). \tag{5}$$

The σ parameter controls the width of the Gaussian function and therefore the amount of generalization over the state space. We set σ to 0.25, which roughly spans the width of three CMAC tiles, after running experiments with $\sigma = 1.0, 0.5, 0.25$ and observing that the learning rates were not dramatically effected.

As we did with the CMAC, we again assume that the state variables are independent and thus have one set of linearly tiled RBFs for each state variable. Similar to the CMAC implementation, all state variables are tiled independently and there are 32 tilings for each state variable. The RBFs in every tiling are spaced so that their centers correspond to the centers of CMAC tiles. We use Equations 3-5 to calculate Q-values of a state s. Because σ specifies that the spread of a RBF is roughly 3 CMAC tiles, each 3 vs. 2 state will thus be computed from approximately $3 \times 13 \times 32 = 1248$ weights in total. All weights w_i are initially set to zero, but over time learning updates changes the values of the weights so that the resulting Q-values more closely predict the true returns, as specified by Equation 1.

The ANN function approximator similarly allows a learner to approximate the action-value function, given a set of continuous, real valued, state variables. Each input to the ANN is set to the value of a state variable and the output corresponds to an action. Activations of the output nodes correspond to Q values. We use a fully-connected feedforward network with a single hidden layer of 20 sigmoid units for all our tasks. The output layer nodes are linear and return the currently predicted $Q(s,a)$ for each action. Weights were initialized with uniformly random numbers chosen from $[0, 1.0]$. We had also tried initializing the weights uniformly to 0 and from $[0, 0.01]$, with little effect on learning rates. This network topology was selected after testing 7 different sizes of hidden layers, from 5 to 30 hidden units. Again, the learning rate did not seem to be strongly affected by this parameter. The network is trained using standard backpropagation where the error signal to modify weights is generated by the Sarsa algorithm, as with the other function approximators.

4.4 Learning 3 vs. 2 Keepaway

To learn 3 vs. 2 Keepaway as a source task for transfer, all weights in the CMAC and RBF function approximators are initially set to zero; every initial state-action value is thus zero and our action-value function is uniform. All weights and biases in the 13-20-3 feedforward ANN are set to small random numbers to encourage faster backprop training (Mehrotra et al., 1997) but the initial action-value is still nearly uniform. As training progresses, the weights of the function approximators are changed by Sarsa so that the average hold time of the keepers increases.

In our experiments we set the learning rate, α, to be 0.1 for the CMAC function approximator, as in previous experiments. α was 0.05, and 0.125 for the RBF and ANN function approximators, respectively. These values were determined after trying approximately five different learning rates for each function approximator. The exploration rate, ε, was set to 0.01 (1%) in all experiments and λ was set to 0, which we selected to be consistent with past work (Stone et al., 2005).

4.5 Learning 4 vs. 3 Keepaway and 5 vs. 4 Keepaway without Transfer

Holding the field size constant we now add an additional keeper and an additional taker to generate the 4 vs. 3 task. All three takers still start in a single corner. Three keepers start in each of the other three corners and the fourth keeper begins an episode at the center of the field. R and T are effectively unchanged from 3 vs. 2 Keepaway, but now $A = \{$*hold, pass to closest teammate, pass*

to second closest teammate, pass to third closest teammate}, and S is made up of 19 state variables due to the added players.

It is also important to point out that the addition of an extra taker and keeper in 4 vs. 3 results in a qualitative change in the task. In 3 vs. 2 both takers must go towards the ball as two takers are needed to capture the ball from the keeper. However, the third taker is now free to roam the field and attempt to intercept passes. This necessarily changes the keeper behavior as one teammate is often blocked from receiving a pass by this new taker. Furthermore, adding a keeper in the center of the field changes the start state significantly as now the keeper that starts with the ball has a teammate that is closer to itself, but is also closer to the takers.

In order to quantify how fast an agent in 4 vs. 3 learns, we set a target performance of 10.0 seconds for ANN learners, while CMAC and RBF learners have a target of 11.5 seconds. These threshold times are chosen so that learners are able to consistently attain the performance level without transfer, but players using TVITM must also learn and do not initially perform above the threshold. CMAC and RBF learners are able to learn better policies than the ANN learners and thus have higher threshold values. When a group of four CMAC keepers has learned to hold the ball from the three takers for an average of 11.5 seconds over 1,000 episodes we say that the keepers have sufficiently learned the 4 vs. 3 task. Thus agents learn until the on-line reward of the keepers, averaged over 1,000 episodes, with exploration, passes a set threshold.[4] In 4 vs. 3, it takes a set of four keepers using CMAC function approximators 30.8 simulator hours (roughly 15 hours of wall-clock time, or 12,000 episodes) on average to learn to hold the ball for 11.5 seconds when training without transfer. By comparison, in 3 vs. 2, it takes a set of three keepers using CMAC function approximators 5.5 hours on average to learn to hold the ball for 11.5 seconds when training without transfer.

The ANN used in 4 vs. 3 is a 19-20-4 feedforward network.[5] The ANN learners do not learn as quickly nor achieve as high a performance before learning plateaus and therefore we use a threshold of 10.0 seconds. (After training four keepers using ANN function approximation without transfer in 4 vs. 3 for over 80 hours, the average hold time was only 10.3 seconds.)

5 vs. 4 is harder than 4 vs. 3 for the same reasons that 4 vs. 3 is more difficult than 3 vs. 2. In 5 vs. 4 three keepers are again placed in three corners and the two remaining keepers are placed in the middle of the $25m \times 25m$ field. All four takers are placed in the fourth corner. There are now five actions: {*hold, pass to closest teammate, pass to second closest teammate, pass to third closest teammate, pass to fourth closest teammate*}, and 25 state variables. In 5 vs. 4, it takes a set of five keepers using CMAC function approximators 59.9 hours (roughly 24,000 episodes) on average to learn to hold the ball for 11.5 seconds when training without transfer. In this paper we investigate the 5 vs. 4 problem only with the CMAC function approximator.

5. Transfer via Inter-Task Mapping in Keepaway

Having introduced our testbed domain and baseline learning approaches, we can now show how TVITM is performed in Keepaway, utilizing terminology described in Section 2. Recall that TVITM

4. We begin each trial by following the initial policy for 1,000 episodes without learning (and therefore without counting this time towards the learning time). This enables us to assign a well-defined initial performance when we begin learning because there already exist 1,000 episodes to average over.
5. Again, other networks with different numbers of hidden units were tried, but the differences in learning times were not significant.

relies on a functional ρ that is able to transfer an action-value function from a source task into a target task with different state and action spaces. ρ is built from the inter-task mappings χ_X and χ_A, and thus this section begins by defining these two mappings and then describing how they are used to generate different ρs.

In the Keepaway domain, A and S are determined by the current Keepaway task and thus differ from instance to instance. $s_{initial}$, R, and T, though formally different, are effectively constant across tasks. When S and A change, $s_{initial}$, R, and T change by definition because they are functions defined over S and A, but in practice R is always defined as $+1$ for every time step that the keepers maintain possession, and $s_{initial}$ and T are always defined by the RoboCup soccer simulation.

5.1 Defining χ_X and χ_A for 4 vs. 3 Keepaway and 3 vs. 2 Keepaway

In the Keepaway domain we are able to intuit the inter-task mappings between states and actions in the two tasks based on our knowledge of the domain. Our choice for the mappings is supported by empirical evidence in Section 6 showing that using these mappings do allow us to construct transfer functions that successfully reduce training time. In general, the transform may not be so straightforward, but experimenting in a domain where it is easily defined allows us to focus on showing the benefits of transfer. This article demonstrates that transfer can be successful when a mapping is available, while we leave it to future work to show how to best construct (or learn) such a transform.

We define χ_A, the inter-task mapping between actions in the two tasks, by identifying actions that have similar effects on the world state in both tasks. For the 3 vs. 2 and 4 vs. 3 tasks, the action "Hold ball" is equivalent because this action has a similar effect on the world in both tasks. Likewise, the action "Pass to closest keeper" is analogous in both tasks, as is "Pass to second closest keeper." We map the novel target action "Pass to third closest keeper" to "Pass to second closest keeper" in the source task.

The state variable mapping, χ_X, is handled with a similar strategy. Each of the 19 state variables in the 4 vs. 3 task is mapped to a similar state variable in the 3 vs. 2 task. For instance, "Distance to closest keeper" is the same in both tasks. "Distance to second closest keeper" in the target task is similar to "Distance to second closest keeper" in the source task. "Distance to third closest keeper" in the target task is also mapped to "Distance to second closest keeper" in the source task. See Table 2 for a full description of χ_X.

Now that χ_A and χ_X are defined, relating the state variables and actions in a target task to the state variables and actions in a source task, we can use them to construct ρs for different internal representations. The functionals will transfer the learned action-value function from the source task into the target task. We denote these functionals as ρ_{CMAC}, ρ_{RBF}, and ρ_{ANN} for the CMAC, RBF, and ANN function approximators, respectively.

5.2 Defining χ_X and χ_A for 4 vs. 3 Keepaway and Knight Joust

The Knight Joust task is less similar to 4 vs. 3 Keepaway than 3 vs. 2 Keepaway is. There are many fewer state variables, a less similar transition function, and a very different reward structure. However, we will show later that information from Knight Joust can significantly improve the performance of Keepaway players because very basic information, such as that it is desirable to maximize the distance to the opponent, will initially cause the players to perform better than acting randomly.

Description of χ_X Mapping from 4 vs. 3 to 3 vs. 2

4 vs. 3 state variable	3 vs. 2 state variable
$dist(K_1,C)$	$dist(K_1,C)$
$dist(K_1,K_2)$ $dist(K_1,K_3)$ $\mathbf{dist(K_1,K_4)}$	$dist(K_1,K_2)$ $dist(K_1,K_3)$ $dist(K_1,K_3)$
$dist(K_1,T_1)$ $dist(K_1,T_2)$ $\mathbf{dist(K_1,T_3)}$	$dist(K_1,T_1)$ $dist(K_1,T_2)$ $dist(K_1,T_2)$
$dist(K_2,C)$ $dist(K_3,C)$ $\mathbf{dist(K_4,C)}$	$dist(K_2,C)$ $dist(K_3,C)$ $dist(K_3,C)$
$dist(T_1,C)$ $dist(T_2,C)$ $\mathbf{dist(T_3,C)}$	$dist(T_1,C)$ $dist(T_2,C)$ $dist(T_2,C)$
$\text{Min}(dist(K_2,T_1), dist(K_2,T_2), \mathbf{dist(K_2,T_3)})$ $\text{Min}(dist(K_3,T_1), dist(K_3,T_2), \mathbf{dist(K_3,T_3)})$ $\text{Min}(\mathbf{dist(K_4,T_1)}, \mathbf{dist(K_4,T_2)}, \mathbf{dist(K_4,T_3)})$	$\text{Min}(dist(K_2,T_1), dist(K_2,T_2))$ $\text{Min}(dist(K_3,T_1), dist(K_3,T_2))$ $\text{Min}(dist(K_3,T_1), dist(K_3,T_2))$
$\text{Min}(ang(K_2,K_1,T_1), ang(K_2,K_1,T_2),$ $\mathbf{ang(K_2,K_1,T_3)})$ $\text{Min}(ang(K_3,K_1,T_1), ang(K_3,K_1,T_2),$ $\mathbf{ang(K_3,K_1,T_3)})$ $\text{Min}(\mathbf{ang(K_4,K_1,T_1)}, \mathbf{ang(K_4,K_1,T_2)},$ $\mathbf{ang(K_4,K_1,T_3)})$	$\text{Min}(ang(K_2,K_1,T_1), ang(K_2,K_1,T_2))$ $\text{Min}(ang(K_3,K_1,T_1), ang(K_3,K_1,T_2))$ $\text{Min}(ang(K_3,K_1,T_1), ang(K_3,K_1,T_2))$

Table 2: This table describes the mapping between states in 4 vs. 3 to states in 3 vs. 2. The distance between a and b is denoted as $dist(a,b)$; the angle made by a, b, and c, where b is the vertex, is denoted by $ang(a,b,c)$; and values not present in 3 vs. 2 are in bold. Relevant points are the center of the field C, keepers K_1-K_4, and takers T_1-T_3, where players are ordered by increasing distance from the ball.

Table 3 describes the inter-task mappings used to transfer between Knight Joust and 4 vs. 3 Keepaway. Our hypothesis was that the Knight Joust player would learn to move North when possible and jump to the side when necessary, which could be similar to holding the ball in Keepaway when possible and passing when necessary.

5.3 Constructing ρ_{CMAC} and ρ_{RBF}

The CMAC function approximator takes a state and an action and returns the expected long-term reward. The learner can evaluate each possible action for the current state and then use π to choose one. We construct a ρ_{CMAC} and use it so that when the learner considers a 4 vs. 3 action, the weights for the activated tiles are not zero but instead are initialized by $Q_{(3vs2,final)}$. To accomplish this, we copy weights learned in the source CMAC into weights in a newly initialized target CMAC, using χ_X and χ_A. Algorithm 1 describes the process in detail.

χ_x: 4 vs. 3 to Knight Joust

4 vs. 3 state variable	Knight Joust state variable
$dist(K_1, T_1)$	$dist(P, O)$
$\text{Min}(ang(K_2, K_1, T_1), ang(K_2, K_1, T_2), ang(K_2, K_1, T_3))$	$ang(West)$
$\text{Min}(ang(K_3, K_1, T_1), ang(K_3, K_1, T_2), ang(K_3, K_1, T_3))$	$ang(East)$
$\text{Min}(ang(K_4, K_1, T_1), ang(K_4, K_1, T_2), ang(K_4, K_1, T_3))$	$ang(East)$
All other Keepaway variables	$\emptyset$

χ_A: 4 vs. 3 to Knight Joust

4 vs. 3 action	Knight Joust action
Hold Ball	Forward
Pass to closest teammate	$Jump_W$
Pass to second closest teammate	$Jump_E$
Pass to third closest teammate	$Jump_E$

Table 3: This table describes the mapping between state variables and actions from 4 vs. 3 to Knight Joust. Note that the we have made Jump West in the Knight Joust correspond to passing to K_2 and Jump East correspond to passing to K_3, but either is reasonable, as long as the state variables and actions are consistent.

Note that this target CMAC will initially be unable to distinguish between some states and actions because the inter-task mappings allow duplication of values. For instance, the weights corresponding to the tiles that are activated for the "Pass to second closest teammate" in the source task are copied into the weights for the tiles that are activated to evaluate the "Pass to second closest teammate" action and the "Pass to third closest teammate" in the target task. The 4 vs. 3 agents are initially unable to distinguish between these two actions. In other words, because the values for the weights corresponding to the two 4 vs. 3 actions are the same, $Q_{(4vs3,initial)}$ will evaluate both actions as having the same expected return. The 4 vs. 3 agents will therefore have to learn to differentiate these two actions as they learn in the target task.

Algorithm 1 APPLICATION OF ρ_{CMAC}

for each non-zero weight, w_i in the source CMAC **do**
  $x_{source} \leftarrow$ value of state variable corresponding to tile i
  $a_{source} \leftarrow$ action corresponding to i
  for each value x_{target} such that $\chi_x(x_{target}) = x_{source}$ **do**
    for each value a_{target} such that $\chi_A(a_{target}) = a_{source}$ **do**
      $j \leftarrow$ the tile in the target CMAC activated by x_{target}, a_{target}
      $w_j \leftarrow w_i$
$w_{Average} \leftarrow$ average value of all non-zero weights in the target CMAC
for each weight w_j in the target CMAC **do**
  if $w_j = 0$ **then**
    $w_j \leftarrow w_{Average}$

As a final step (Algorithm 1, lines 8–11), any weights which have not been initialized by ρ_{CMAC} are set to the average value of all initialized weights. The 3 vs. 2 training was likely not exhaustive and therefore some weights which may be used in 4 vs. 3 would otherwise remain uninitialized. Tiles which correspond to every value in the new 4 vs. 3 state vector have thus been initialized to values determined via training in 3 vs. 2 and can therefore be considered in the computation. This averaging effect is discussed further in Section 6 and has the effect of allowing agents in the target task to learn faster.

ρ_{RBF} is constructed similarly to ρ_{CMAC}. The main difference between the RBF and CMAC function approximators are how weights are summed together to produces values, but the weights have similar structure in both function approximators. For a given state variable, a CMAC sums one weight per tiling. An RBF differs in that it sums multiple weights for each tiling, where weights are multiplied by the Gaussian function $\phi(x - c_i)$. Thus when using ρ_{RBF} we copy weights following the same schema as in ρ_{CMAC} in Algorithm 1.

5.4 Constructing ρ_{ANN}

To construct a (fully connected, feedforward) neural network for the 4 vs. 3 target task, the 13-20-3 network from 3 vs. 2 is first augmented by adding 6 inputs and 1 output node. The weights connecting inputs 1–13 to the hidden nodes are copied over from the 13-20-3 network. Likewise, the weights from hidden nodes to outputs 1–3 are copied over to the 19-20-4 network. Weights from inputs 14-19 to the hidden nodes correspond to the new state variables and are copied over from the analogous 3 vs. 2 state variable, according to χ_x. The weights from the hidden nodes to the novel output are copied over from the analogous 3 vs. 2 action, according to χ_a. Every weight in the 19-20-4 network is therefore set to an initial value based on the trained 13-20-3 network. Algorithm 2 describes this process in detail. We define the function ψ to map nodes in the two networks:

$$\psi(n) = \begin{cases} \chi_x(n), & \text{if } n \text{ is an input} \\ \chi_a(n), & \text{if } n \text{ is an output} \\ \delta(n), & \text{if } n \text{ is a hidden node} \end{cases}$$

where a function δ represents the correspondence between these hidden nodes ($\delta(h_{target}) = h_{source}$). In our case the number of hidden nodes used are the same in both tasks. Therefore, in practice ψ("n^{th} hidden node in the source network") = "n^{th} hidden node in the target network."

Whereas ρ_{CMAC} and ρ_{RBF} copied many weights (hundreds or thousands, where increasing the amount of 3 vs. 2 training will increase the number of learned non-zero weights), ρ_{ANN} always copies the same number of weights regardless of training. In fact, ρ_{ANN} initializes only 140 new weights (in addition to the 320 weights that existed in 3 vs. 2) in the 4 vs. 3 representation and is therefore in some sense simpler than the other ρs.

Algorithm 2 APPLICATION OF ρ_{ANN}

1: **for** each pair of nodes n_i, n_j in ANN_{target} **do**
2: **if** link($\psi(n_i), \psi(n_j)$) exists in ANN_{source} **then**
3: Set link(n_i, n_j) in ANN_{target} to have weight of link($\psi(n_i), \psi(n_j)$) in ANN_{source}

5.5 Q-value Reuse

The three ρs previously introduced are specific to particular function approximators. In this section we introduce a different approach, *Q-value Reuse*, to transfer between a source and target. Rather than initialize a function approximator in the target task with values learned in the source task, we instead reuse the entire learned source task's Q-values. A copy of the source task's function approximator is retained so that it can calculate the source task's Q-values for any state, action pair: $Q_{sourceFA} : S \times A \mapsto \mathbb{R}$. When computing Q-values for the target task, we first map the target task state and action to the source task's state and action via the inter-task mappings. The computed Q-value is a combination of the output of the source task's saved function approximator and the target task's current function approximator:

$$Q(s,a) = Q_{sourceFA}(\chi_X(s), \chi_A(a)) + Q_{targetFA}(s,a)$$

Sarsa updates in the target task are computed as normal, but only the target function approximator's weights are eligible for updates. Note that if $\chi_X(s)$ or $\chi_A(a)$ were undefined for a certain s,a pair in the target task, $Q(s,a)$ would equal $Q_{targetFA}(s,a)$.

Q-value Reuse may be considered a type of *reward shaping* (Colombetti and Dorigo, 1993; Mataric, 1994): we are able to directly use the expected rewards from the source task to bias the learner in the target task. This method has two advantages. First, it is not function-approximator specific, and could, in theory, be used to transfer between different function approximators as well as between different tasks. Second, there is no initialization step needed between learning the two tasks. However, drawbacks include an increased lookup time and larger memory requirements. Such requirements will grow linearly in the number of transfer steps; while they are not substantial with a single source task, they may become prohibitive when using multiple source tasks or when performing doing multi-step transfer (such as shown later in Section 7.2).

6. Experimental Results: 3 vs. 2 Keepaway to 4 vs. 3 Keepaway

This section discusses the results of our transfer experiments between the 3 vs. 2 and 4 vs. 3 Keepaway tasks using our two metrics, training time reduction in the target task and total training time reduction. Section 6.1 shows the success of transfer when the 3 vs. 2 is used as a source task to learn 4 vs. 3. Section 6.2 includes additional analysis of these results. Section 6.3 demonstrates transfer between 3 vs. 2 and 4 vs. 3 CMAC players using Q-value Reuse.

6.1 Transferring via ρ from 3 vs. 2 Keepaway into 4 vs. 3 Keepaway

Having constructed three ρs that transform the learned action-value functions, we can now set $Q_{(4vs3,initial)} = \rho(Q_{(3vs2,final)})$ between Keepaway agents with CMAC, RBF, or ANN function approximation. We do not claim that these initial action-value functions are correct (and empirically they are not), but instead that the constructed action-value functions allow the learners to more quickly discover a better-performing policy.

In this section we show the results of learning 4 vs. 3 Keepaway, both without transfer and after using TVITM with varying amounts of 3 vs. 2 training. Analyses of learning times required to reach

CMAC Learning Results

# 3 vs. 2 Episodes	Ave. 3 vs. 2 Time	Ave. 4 vs. 3 Time	Ave. Total Time	Std. Dev.
0	0	30.84	30.84	4.72
10	0.03	24.99	25.02	4.23
50	0.12	19.51	19.63	3.65
100	0.25	17.71	17.96	4.70
250	0.67	16.98	**17.65**	4.82
500	1.44	17.74	19.18	4.16
1000	2.75	16.95	19.70	5.5
3000	9.67	9.12	18.79	2.73
6000	21.65	**8.56**	30.21	2.98

Table 4: Results showing that learning Keepaway with a CMAC and applying transfer via inter-task mapping reduces training time (in simulator hours) for CMAC players. Minimum learning times for reaching the 11.5 second threshold are bold. As source task training time increases, the required target task training time decreases. The total training time is minimized with a moderate amount of source task training.

threshold performance levels[6] show that agents utilizing CMAC, RBF, and ANN function approximation are all able to learn faster in the target task by using ρ_{CMAC}, ρ_{RBF}, and ρ_{ANN}, respectively.

Tables 4 and 5 show learning times to reach a threshold performance and verify that a CMAC, an RBF, and an ANN successfully allow independent players to learn to hold the ball from opponents when learning without transfer; agents utilizing these three function approximation methods are able to successfully attain the 4 vs. 3 threshold performance.[7]

This result shows that a CMAC is more efficient than an ANN trained with backprop, another obvious choice. We posit that this difference is due to the CMAC's property of *locality*. When a particular CMAC weight for one state variable is updated during training, the update affects the output value of the CMAC for other nearby state variable values. The width of the CMAC tiles determines the generalization effect and outside of this tile width, the change has no effect. Contrast this with the non-locality of an ANN. Every weight is used for the calculation of an action-value function, regardless of how close two inputs are in state space. Any update to a weight in the ANN must necessarily change the final output of the network for every set of inputs. Therefore it may take the ANN longer to settle into an effective configuration. Furthermore, the ANNs use many fewer weights than the CMAC and RBF learners, which may have allowed for faster learning at the cost of reduced performance of the final policy.

The RBF function approximator had the best performance of the three when learning without transfer (i.e., the top row of each table). The RBF shares the CMAC's locality benefits, but is also able to generalize more smoothly due to the Gaussian summation of weights.

To test the effect of using transfer with a learned 3 vs. 2 action-value function, we train a set of keepers for a number of 3 vs. 2 episodes, save the function approximator's weights ($Q_{(3vs2,final)}$)

6. Our results hold for other threshold times as well, provided that the threshold is not initially reached without training and that learning will enable the keepers' performance to eventually cross the threshold.
7. All times reported in this article refer to simulator time, which is roughly twice that of the wall clock time. We only report sample complexity and not computational complexity; the running time for our learning methods is negligible compared to that of the RoboCup Soccer Server.

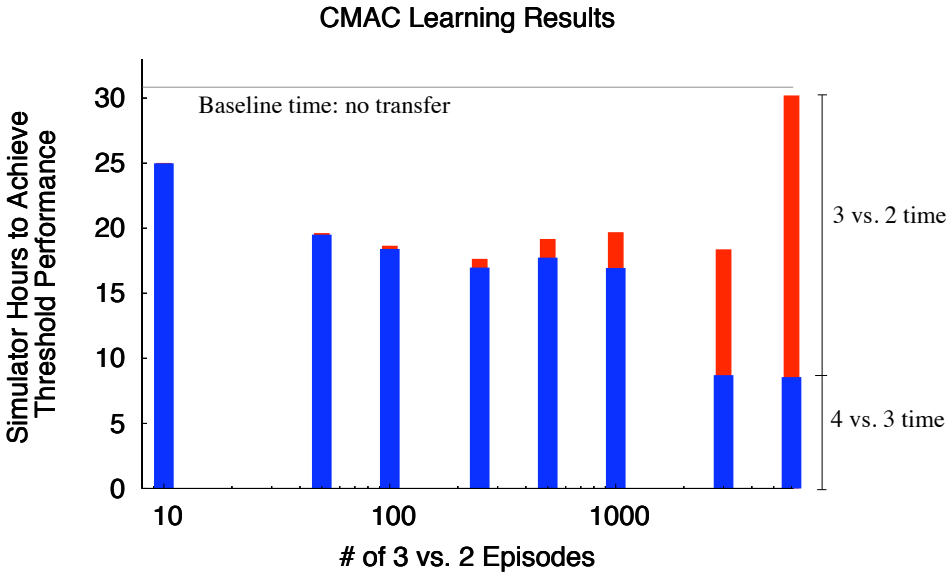


Figure 7: A graph of Table 4 where the x-axis uses a logarithmic scale. The thin bars show the amount of time spent training in the source task, the thick bars show the amount of time spent training in the target task, and their sum represents the total time. The target task training time is reduced as more time is spent training in the source task. The total time is minimized when using a moderate amount of source task training.

from a random 3 vs. 2 keeper, and use the weights to initialize all four keepers[8] in 4 vs. 3 so that $Q_{(4vs3,initial)} \leftarrow \rho(Q_{(3vs2,final)})$. Then we train on the 4 vs. 3 Keepaway task until the average hold time for 1,000 episodes is greater than some performance threshold. Recall that in section 4.5 we specify a threshold of 11.5 seconds in the case of CMAC and RBF function approximators and 10.0 seconds for ANNs as neural network agents were unable to learn as effectively.

To determine if Keepaway players using CMAC function approximation can benefit from transfer, we compare the time it takes agents to learn the target task after transferring from the source task with the time it takes to learn the target task without transfer. The result tables show different amounts of source task training time, where the minimal learning times are in bold. The top row of each table represents learning the task without transfer and thus any column with transfer times lower than the top row shows beneficial transfer. Our second goal of transfer would be met if the total training time in both tasks with transfer was less than learning without transfer in the target task. Table 4 reports the average time spent training in 4 vs. 3 with CMAC function approximation to achieve an 11.5 second average hold time after different amounts of 3 vs. 2 training. Column two reports the time spent training on 4 vs. 3 while the third column shows the total time to train 3 vs. 2 and 4 vs. 3. As can be seen from the table, spending time training in the simpler 3 vs. 2 domain can cause the learning time for 4 vs. 3 to decrease. To overcome the high amounts of noise in our evaluation we run at least 25 independent trials for each data point reported.

8. We do so under the hypothesis that the policy of a single keeper represents all of the keepers' learned knowledge. Though in theory the keepers could be learning different policies that interact well with one another, so far there is no evidence that they do. One pressure against such specialization is that the keepers' start positions are randomized. There appears to be specialization when each keeper starts in the same location every episode.

RBF and ANN Learning Results

# of 3 vs. 2 Episodes	Ave. RBF 4 vs. 3 Time	Ave. RBF Total Time	Standard Deviation	Ave. ANN 4 vs. 3 Time	Ave. ANN Total Time	Standard Deviation
0	19.52	19.52	6.03	33.08	33.08	16.14
10	18.99	19.01	6.88	19.28	**19.31**	9.37
50	19.22	19.36	5.27	22.24	22.39	11.13
100	18.00	18.27	5.59	23.73	24.04	9.47
250	18.00	18.72	7.57	22.80	23.60	12.42
500	16.56	18.12	5.94	19.12	20.73	8.81
1,000	**14.30**	**17.63**	3.34	**16.99**	20.19	9.53
3,000	14.48	26.34	5.71	17.18	27.19	10.68

Table 5: Results from learning Keepaway with different amounts of 3 vs. 2 training time (in simulator hours) indicates that ρ_{RBF} and ρ_{ANN} can reduce training time for RBF players (11.5 second threshold) and ANN players (10.0 second threshold). Minimum learning times for each method are in bold.

The potential of TVITM is evident in Table 4 and Figure 7. To analyze these results, we conduct a number of Student's t-tests to determine if the differences between the distributions of learning times for the different settings are significant. These tests confirm that the differences in the distributions of 4 vs. 3 training times when using TVITM are statistically significant ($p < 0.05$) when compared to training 4 vs. 3 without transfer. Not only is the time to train the 4 vs. 3 task decreased when we first train on 3 vs. 2, but the total training time is less than the time to train 4 vs. 3 without transfer. We can therefore conclude that in the Keepaway domain, training first on a simpler source task can increase the rate of learning enough that the total training time is decreased when using a CMAC function approximator. It is not obvious how to choose the amount of time to spend learning the source task to minimize the total time and this an optimization will be left for future work (see Section 8).

Analogous experiments for Keepaway players using RBF and neural network function approximation are presented in Table 5. Again, successful transfer is demonstrated as both the transfer agents' target task training time and the transfer agent's total training time are less than the time required to learn the target task without transfer. All numbers reported are averaged over at least 25 independent trials; both 4 vs. 3 time and total time can be reduced with TVITM. For the RBF players, all TVITM 4 vs. 3 results using at least 500 3 vs. 2 episodes show a statistically significant difference from those that learn without transfer ($p < 0.05$), while the learning trials that used less than 500 source task episodes did not significantly reduce the target task training time. The difference in all 4 vs. 3 training times for the ANN players between using TVITM and training without transfer is statistically significant ($p < 0.05$).

The RBF function approximator yielded the best learning rates for 3 vs. 2 Keepaway, followed by the CMAC function approximator, and lastly the ANN trained with backpropagation. However, TVITM provided the least percentage speedup to the RBF agents. One possible hypothesis is that transfer is less useful to the best learners. One explanation is that if a particular representation is poorly suited for a task, transfer may be able to provide proportionally more speedup because it is that much further from an "optimal learner." Nonetheless, while some function approximators get more or less benefit from TVITM, it is clear that all three are able learn the target task faster with the

Ablation Studies with ρ_{CMAC}

Transfer Functional	# of 3 vs. 2 Episodes	Ave. 4 vs. 3 Time	Standard Deviation
No Transfer	0	30.84	4.72
ρ_{CMAC}	100	17.71	4.70
ρ_{CMAC}	1000	16.95	5.5
ρ_{CMAC}	3000	9.12	2.73
$\rho_{CMAC,\ No\ Averaging}$	100	25.68	4.21
$\rho_{CMAC,\ No\ Averaging}$	3000	9.53	2.28
only averaging	100	19.06	6.85
only averaging	3000	10.26	2.42
$\rho_{CMAC,\ Ave\ Source}$	1000	15.67	4.31

Table 6: Results showing that transfer with the full ρ_{CMAC} outperforms using ρ_{CMAC} without the final averaging step, using only the averaging step of ρ_{CMAC}, and when averaging weights in the source task before transferring the weights.

technique, and that more training in the source task generally reduces the time needed to learn the target task.

6.2 Understanding ρ_{CMAC}'s Benefit

To better understand how TVITM uses ρ_{CMAC} to reduce the required training time in the target task, and to isolate the effects of its various components, this section details a number of supplemental experiments.[9]

To help understand how ρ_{CMAC} enables transfer we isolate its two components. We first ablate the functional so that the final averaging step (Algorithm 1, lines 8–11), which places the average weight into all zero weights, is removed. We anticipated that the benefit from transfer would be increasingly degraded, relative to using the actual ρ_{CMAC}, as fewer numbers of training episodes in the source task were used. The resulting 4 vs. 3 training times were all shorter than training without transfer, but longer than when the averaging step was incorporated. The relative benefit of our ablated ρ_{CMAC} is greater after greater numbers of source task episodes; the averaging step appears to have given initial values to weights in the state/action space that have never been visited with low numbers of source episodes and thus imparts some bias in the target task even with very little 3 vs. 2 training. Over time more of the state space in the source task is explored and thus our ablated functional performs quite well. This result shows that the averaging step is most useful with less source task training, but becomes less so as more source experience is accumulated (see Table 6 for result details).

If we perform *only* the averaging step from ρ_{CMAC} on learners trained in the *target* task, we can determine how important this step is to our method's effectiveness. Applying the averaging step

9. Informal experiments showed that the CMAC and RBF transfer results were qualitatively similar, which is reasonable given the two function approximator's many similarities. Thus we expect that the supplemental experiments in this section would yield qualitatively similar results if we used RBFs rather than CMACs. While our results demonstrate that all three function approximators can successfully transfer knowledge, we focus our supplementary experiments on CMAC function approximation so that our transfer work can be directly comparable to previous work in Keepaway, which also used CMACs (Stone et al., 2005).

Time required for CMAC 4 vs. 3 players to reach 11.5 sec. hold time

Initial CMAC weight	Ave. Learning Time	Standard Deviation
0	30.84	4.72
0.5	35.03	8.68
1.0	N/A	N/A
Each weight randomly selected from the uniform distribution from [0,1.0]	28.01	6.93

Table 7: 10 independent trials are averaged for different values for initial CMAC weights. None of the trials with initial weights of 1.0 were able to reach the 11.5 threshold within 45 hours, and thus are shown as N/A above.

causes the total training time to decrease below that of training 4 vs. 3 without transfer, but again the training times are longer than running ρ_{CMAC} on weights trained in 3 vs. 2. This result confirms that both parts of ρ_{CMAC} contribute to reducing 4 vs. 3 training time and that training on 3 vs. 2 is more beneficial for reducing the required 4 vs. 3 training time than training on 4 vs. 3 and applying ρ_{CMAC} (see Table 6 for result details).

The averaging step in ρ_{CMAC} is defined so that the average weight in the *target* CMAC overwrites all zero-weights. We also conducted a set of 30 trials which modified ρ_{CMAC} so that the average weight in the *source* CMAC is put into all zero-weights in the target CMAC, which is possible when agents in the source task know that their saved weights will be used for TVITM. Table 6 shows that when the weights are averaged in the source task ($\rho_{CMAC,\ Ave\ Source}$) the performance is not statistically different ($p < 0.05$ from TVITM when averaging in the target task (See Table 4).

To verify that the 4 vs. 3 CMAC players were benefiting from TVITM and not from having non-zero initial weights, we initialized CMAC weights uniformly to 0.5 in one set of experiments, 1.0 uniformly in a second set of experiments, and then to random numbers uniformly distributed from 0.0-1.0 in a third set of experiments. We do so under the assumption that 0.0, 0.5, and 1.0 are all reasonable initial values for weights (although in practice 0.0 is most common). The learning time was never statistically better than learning with weights initialized to zero, and in some experiments the non-zero initial weights decreased the speed of learning. Haphazardly initializing CMAC weights may hurt the learner but systematically setting them through TVITM is beneficial. Thus we conclude that the benefit of transfer is not a byproduct of our initial setting of weights in the CMAC (see Table 7 for result details).

To further test the sensitivity of the ρ_{CMAC} function, we change it in two different ways. We first defined $\rho_{modified}$ by modifying χ_A so that instead of mapping the novel target task action "Pass to second third keeper" into the action "Pass to second closest keeper," we instead map the novel action into "Hold ball." Now $Q_{4vs3,initial}$ will initially evaluate "pass to third closest keeper" and "hold ball" as equivalent for all states. Second, we modify χ_A and χ_X so that state variables and actions not present in 3 vs. 2 are not initialized in the target task. Using these new inter-task mappings, we construct ρ_{3vs2}, a functional which copies over information learned in 3 vs. 2 exactly but assigns the average weight to all novel state variables and actions in 4 vs. 3.

When using this $\rho_{modified}$ to initialize weights in 4 vs. 3, the total training time increased relative to the normal ρ_{CMAC} but still outperformed training without transfer. Similarly, ρ_{3vs2} is able to outperform learning without transfer, but underperforms the full ρ_{CMAC}, particularly for higher amounts of training in the source task.

Testing Sub-optimal Inter-task Mappings

Transfer Functional	# of 3 vs. 2 Episodes	Ave. 4 vs. 3 Time	Standard Deviation
ρ_{CMAC}	100	17.71	4.70
ρ_{CMAC}	3000	9.12	2.73
$\rho_{modified}$	100	21.74	6.91
$\rho_{modified}$	3000	10.33	3.21
ρ_{3vs2}	100	18.90	3.73
ρ_{3vs2}	3000	12.00	5.38

Table 8: Results showing that transfer with the full ρ_{CMAC} outperforms using sub-optimal or incomplete inter-task mappings.

Choosing non-optimal inter-task mappings when constructing ρ seems to have a detrimental, but not necessarily disastrous, effect on the training time. This result shows that the structure of ρ is indeed important to the success of transfer (see Table 8 for result details).

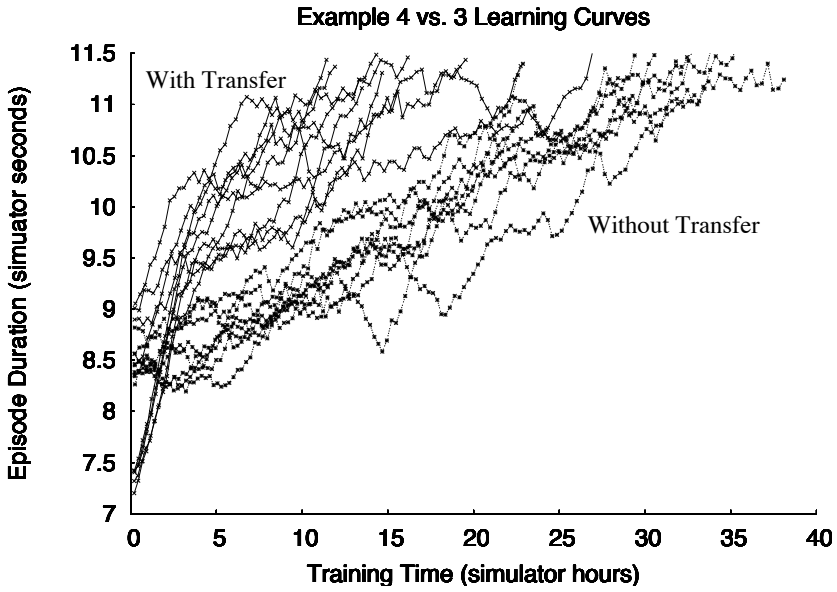


Figure 8: Representative learning curves, showing that transfer via inter-task mapping does not significantly increase the performance of the initial policy in 4 vs. 3, but enables faster learning by biasing the learner towards a productive part of the function approximator's weight space. Eight 4 vs. 3 learning curves without transfer are compared to eight learning curves in 4 vs. 3 after transferring from 250 episodes of 3 vs. 2.

Interestingly, when the CMACs' weights are loaded into the keepers in 4 vs. 3, the initial hold times of the keepers do not differ significantly from those of keepers with uninitialized CMACs (i.e., CMACs where all weights are initially set to zero). The information contained in the function approximators' weights prime the 4 vs. 3 keepers to more quickly learn their task by biasing

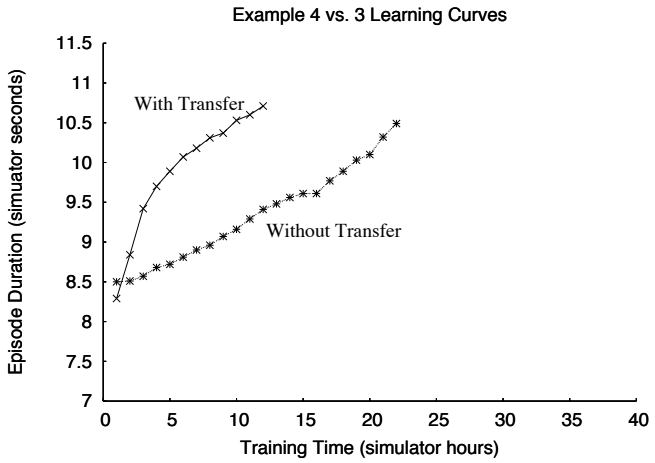


Figure 9: The average performance from the learners in Figure 8 shows a clear benefit from using transfer.

Initial Performance in 4 vs. 3 with CMAC Function Approximation

# of 3 vs. 2 Episodes	Ave. Performance (sec.)	Standard Deviation
0	8.46	0.17
1000	8.92	1.48
6000	9.24	1.15

Table 9: This graph shows the difference in initial performance between 4 vs. 3 players with and without transfer. 40 independent trials are averaged for each setting and the differences in initial performance (i.e., initial episode lengths) are small. A Student's t-test shows that 8.46 and 8.92 are not statistically different while 8.46 and 9.24 are ($p < 0.05$).

their search, even though the knowledge we transfer is of limited initial value. See Figure 8 for representative learning curves and Table 9 for result details.[10]

TVITM relies on effectively reusing learned data in the target task. We hypothesized that successfully leveraging this data may be effected by ε, Sarsa's exploration parameter, which balances exploration with exploitation. Recall that we had initially chosen an exploration rate of 0.01 (1%) to be consistent with past research. Table 10 shows the results of learning 3 vs. 2 Keepaway with $\varepsilon = 0.01$ for 1,000 episodes, utilizing ρ_{CMAC}, and then learning 4 vs. 3 Keepaway with various settings for ε. The results show that of these 4 additional settings for ε, only $\varepsilon = 0.05$ is statistically better than the default rate of 0.01. To further explore this last result we ran a series of 30 trials of learning 4 vs. 3 from scratch with the value of $\varepsilon = 0.05$ and found that there was a significant difference from learning 4 vs. 3 from scratch with $\varepsilon = 0.01$. Thus the speedup for this particular setting of ε in transfer, relative to the default value, is explained by the increased learning speed

10. The more similar the source and target tasks are, the more of an immediate performance improvement we would expect to see. For example, in the degenerate case where the source and target task are identical, the initial performance in the target task will be equivalent to the final performance in the source task. However, in such a situation, reducing the total time—our more difficult transfer goal—would prove *impossible*.

Varying the Exploration in 4 vs. 3

ε in 4 vs. 3	# 3 vs. 2 Episodes	Ave. 4 vs. 3 Time	Standard Deviation
0.001	1000	22.06	10.52
0.005	1000	19.22	8.31
0.01 (default)	1000	16.95	5.5
0.05	1000	12.84	2.55
0.1	1000	18.40	5.70
0.01 (default)	0	30.84	4.72
0.05	0	17.57	2.59

Table 10: The first five rows detail experiments where 3 vs. 2 is first learned with $\varepsilon = 0.01$ and then transfer is used to speed up learning in 4 vs. 3. 30 independent trials are averaged for each setting of ε in the target task. The last two rows show the results of learning 4 vs. 3 without transfer for two settings of ε. These results show that the amount of exploration in the target task affects learning speed both with and without transfer.

without transfer. This experiment does suggest, however, that the previously determined value of $\varepsilon = 0.01$ is not optimal for Sarsa with CMAC function approximation in the Keepaway domain.

From these supplemental results we conclude:

1. Both parts of ρ_{CMAC}—copying weights based on χ_x and χ_A, and the final averaging step—contribute to the success of TVITM. The former gives more benefit after more training is completed in the source task and the second helps when less knowledge is gained in the source task before transfer.

2. Using ρ_{CMAC} is superior to weights initialized to zero (training without transfer), as well as weights initialized to 0.5, 1.0 and [0,1.0], three other reasonable initial settings.

3. A suboptimal or incomplete transfer functional, such as $\rho_{modified}$ and ρ_{3vs2}, allows TVITM to speed up learning, but not as much as the more correct ρ_{CMAC}.

4. Players initialized by TVITM in the source task do not initially outperform uninitialized players in the target task, but are able to learn faster.

6.3 Transferring via Q-value Reuse from 3 vs. 2 Keepaway into 4 vs. 3 Keepaway

In the previous sections we showed that TVITM was capable of transferring from 3 vs. 2 into 4 vs. 3 by using ρ_{CMAC}, ρ_{RBF}, and ρ_{ANN}. In this section we use TVITM with Q-value Reuse (Section 5.5) between CMAC players in 3 vs. 2 and 4 vs. 3. Recall that Q-value Reuse directly uses a learned function approximator from the source task when calculating Q-values in the target task.

Table 11 shows the results of using Q-value Reuse. Each transfer experiment shows the average of 30 independent trials. Both the 4 vs. 3 and total times are statistically different from learning without transfer ($p < 0.05$, via Student's t-tests). As when using ρ_{CMAC} for transfer (Table 4), more 3 vs. 2 episodes correspond to a decrease in the time required for 4 vs. 3 players to reach the 11.5 second threshold performance.

The reduction in transfer efficacy, relative to using ρ_{CMAC}, is due to the averaging step in ρ_{CMAC}. As we showed in the previous section, this averaging step has an impact on the target task learning

Q-value Reuse between CMAC players

# of 3 vs. 2 Episodes	Ave. 4 vs. 3 Time	Ave. Total Time	Standard Deviation
0	30.84	30.84	4.72
10	28.18	28.21	5.04
50	28.0	28.13	5.18
100	26.8	27.06	5.88
250	24.02	24.69	6.53
500	22.94	24.39	4.36
1,000	22.21	**24.05**	4.52
3,000	**17.82**	27.39	3.67

Table 11: Results from learning 3 vs. 2 with CMAC players for different numbers of episodes and then utilizing the learned 3 vs. 2 CMAC directly while learning 4 vs. 3. Minimum learning times for reaching the 11.5 second threshold are bold.

times. However, in Q-value Reuse we treat the source task function approximator as a "black box" and thus do not permute its values, nor use it to set the initial values of the target task's function approximator. These results suggest that if the source and target function approximators are different, Q-value Reuse may be appropriate. However, if memory is limited, running time is critical, and/or multiple transfer steps are involved (such as transferring from 3 vs. 2 to 4 vs. 3, and then from 4 vs. 3 to 5 vs. 4), then using a ρ is preferable.

7. Experimental Results: Different Transfer Tasks

In this section of the article we show that TVITM can work between a variety of different source/target task pairs. Section 7.1 presents results of transfer between 3 vs. 2 and 4 vs. 3 agents with different abilities. Section 7.2 demonstrates that transfer can also be used to reduce both target and total training time for 5 vs. 4 Keepaway, and gives some initial results for 6 vs. 5 Keepaway, demonstrating that TVITM can scale to more complex tasks. Section 7.3 shows the results of transferring from two variants of 3 vs. 2 into 4 vs. 3 to demonstrate how the relatedness of source and target tasks effect the efficacy of TVITM. Lastly, Section 7.4 shows that TVITM can successfully transfer between the Knight Joust task and 4 vs. 3, two tasks with very different characteristics.

7.1 3 vs. 2 Keepaway and 4 vs. 3 Keepaway with Differing Player Abilities

The results in Section 6.1 show that Q-values learned in 3 vs. 2 can be successfully used to speed up learning in 4 vs. 3. In this section we test how robust TVITM is to changes in the agent's abilities. In addition to changing the number of players between the source and target tasks, other variations such as the size of the field, wind, and player ability can be modified. It is a qualitatively different challenge to use TVITM to speed up learning between two tasks where the agents' actions have different effects (i.e., T has been modified so that the actions are qualitatively different) in addition to different state and action spaces. We choose to test the robustness of TVITM by changing the passing

Learning Results with Different Actuators

# of 3 vs. 2 Episodes	3 vs. 2 Time	3 vs. 2 Actuator Accurate?	Ave. 4 vs. 3 Time	Standard Deviation	4 vs. 3 Actuator Accurate?
0	0	N/A	30.84	16.14	Yes
500	1.44	Yes	17.74	4.16	Yes
3000	9.67	Yes	9.12	2.73	Yes
0	0	N/A	54.15	6.13	No
500	1.23	No	37.3	9.24	No
3000	8.36	No	29.86	9.20	No
500	1.37	Yes	37.54	7.48	No
3000	9.45	Yes	24.17	5.54	No
500	1.3	No	18.46	3.93	Yes
3000	8.21	No	13.57	3.64	Yes

Table 12: Results showing transfer via inter-task mapping benefits CMAC players utilizing ρ_{CMAC} with two kinds of actuators. These results demonstrate that transfer can succeed even when actions in the source and target tasks are qualitatively different. The results in rows 1–3 are from Table 4.

actuators on some sets of agents so that the passes are less accurate.[11] We show in this section that TVITM speeds up learning, relative to learning without transfer, in the following scenarios:

1. Learning 4 vs. 3 with damaged passing actuators after transferring from 3 vs. 2 players with damaged passing actuators.

2. Learning 4 vs. 3 with a normal passing actuators after transferring from 3 vs. 2 players with damaged passing actuators.

3. Learning 4 vs. 3 with damaged passing actuators after transferring from 3 vs. 2 players with normal passing actuators.

Accurate CMAC players learning without transfer in 4 vs. 3 take only 30.1 hours to reach the threshold performance level (row 1 of Table 12). When we allow sets of CMAC keepers to learn 4 vs. 3 without transfer while using the less accurate pass mechanism, the average time to reach an average performance of 11.5 seconds is 54.2 hours (row 4 of Table 12). We are also able to use the same ρ_{CMAC} to speed up learning in the target task when both the target and source tasks have inaccurate actuators (rows 5 and 6). These two results, as well as all other 4 vs. 3 transfer learning times in this table, are statistically significant when compared to learning the relevant 4 vs. 3 task without transfer ($p < 0.05$).

Now consider that we would like to learn the 4 vs. 3 target task with inaccurate passing, but that we have already trained some 3 vs. 2 keepers that learned using an accurate pass action in the source task. As we can see in the third group (i.e., rows 7 and 8) of Table 12, even though the players in the source task have different actuators than in the target task, transfer is able to significantly speed up learning compared to not using transfer.

11. Actuators are changed in the benchmark players by changing the pass action from the default "PassNormal" to "PassFast" which increases the speed of the pass by 50%, reducing accuracy.

This result confirms that the same ρ will allow TVITM to transfer between tasks where not only have S and A changed, but the effect of the actions have also changed qualitatively. This situation is of practical import as well, as many robotic systems experience gradual degradation in performance over time due to wear and tear. If a set of robots with worn down actuators are available, they may still be able to benefit from action-value function transfer of Q-values from learners that have fresh actuators. Alternately, if a set of agents have learned a task and then later want to learn another task but have damaged their actuators since learning the source task, transfer may still increase the speed of learning.

We also perform the inverse experiment where agents in the source task have inaccurate actuators and agents in the target task have normal actuators. We perform TVITM after 500 and 3,000 episodes of 3 vs. 2 with inaccurate passing to initialize the Q-values of agents in 4 vs. 3 with accurate passing. The final two rows in Table 12 again shows using this transfer is a significant improvement over learning without transfer. Thus a fielded agent with worn down actuators would be able to successfully transfer its learned action-value function to agents whose actuators were undamaged. Interestingly, transferring from source keepers that have accurate actuators is more effective than transferring from source keepers that have inaccurate actuators both when the target task has accurate actuators and when it has inaccurate actuators. We posit that this is because it is easier to learn with accurate actuators, which means that more useful information exists to be transfered.

We first showed in Section 6 that transfer from 3 vs. 2 keepers with accurate pass actuators to 4 vs. 3 keepers with accurate pass actuators was successful. In this subsection we demonstrate that transfer works when actuators are inaccurate in both the source and target tasks. It also speeds learning in the target task when transferring from inaccurate 3 vs. 2 players to accurate 4 vs. 3 players or from accurate 3 vs. 2 players to inaccurate 4 vs. 3 players.

Combined, our results show that TVITM is able to speed up learning in multiple target tasks with different state and action spaces, and even when the agents have somewhat different actuators in the two tasks.

7.2 Scaling up to Larger Keepaway Tasks

In this section we show that our method can also be used to speed up the 5 vs. 4 Keepaway task, which provides evidence for scalability to larger tasks. The 5 vs. 4 task is more difficult than the 4 vs. 3 task, as discussed in Section 4.5. In addition to using 4 vs. 3 to speed up learning in 5 vs. 4, we show that TVITM can be used *twice* to learn the 3 vs. 2, 4 vs. 3, and 5 vs. 4 tasks in succession.

Results in Table 13 show that TVITM scales to the 5 vs. 4 Keepaway task. In 5 vs. 4 we say that the task has been learned when the 5 keepers are able to hold the ball for an average of 9.0 seconds over 1,000 episodes. ρ_{CMAC} can be formulated by extending χ_X and χ_A so that they can transfer the action-value function from 4 vs. 3 to 5 vs. 4, analogous to the way it transfers values from 3 vs. 2 to 4 vs. 3. These results are shown in rows 2 and 3 of Table 13.

χ_X and χ_A can also be formulated so that we can use TVITM to speed up 5 vs. 4 after learning 3 vs. 2. For instance, the the target task actions "Pass to Second Closest Teammate", "Pass to Third Closest Teammate", and "Pass to Fourth Closest Teammate" are mapped to the source task action "Pass to Second Closest Teammate." Table 13, rows 4 and 5, show that this mapping formulation is successful. In fact, there is more benefit than transferring from 4 vs. 3. We posit that this is due to the fact that it is easier to learn more in the simpler target task and this outweighs the fact that it is less related to 5 vs. 4 than to 4 vs. 3. Another way to understand this is that in a fixed amount of

CMAC Learning Results in 5 vs. 4

# of 3 vs. 2 Episodes	# of 4 vs. 3 Episodes	Ave. 5 vs. 4 Time	Ave. Total Time	Standard Deviation
0	0	22.58	22.58	3.46
0	500	13.44	14.60	7.82
0	1000	9.66	12.02	4.50
500	0	6.76	**8.18**	1.90
1000	0	6.70	9.66	2.12
500	500	**6.19**	8.86	1.26

Table 13: Results showing that learning Keepaway with a CMAC and applying transfer via inter-task mapping can reduce training time (in simulator hours) for CMAC in 5 vs. 4 with a target performance of 9.0 seconds. All numbers are averaged over at least 25 independent trials.

experience, players in 3 vs. 2 are able to update more weights than 4 vs. 3, measured as a percentage of the total possible number of weights used in the task.

A final refinement is to use a two-step application of TVITM so that 3 vs. 2 runs first. This learned action-value function is used as the initial action-value function in 4 vs. 3 after applying ρ_{CMAC}, and after training the final 4 vs. 3 action-value function is used as the initial action-value function for 5 vs. 4. Using this procedure (Table 13, bottom row) we find that the time to learn 5 vs. 4 is reduced to roughly 27% of learning without transfer. A t-test confirms that the differences between all 5 vs. 4 training times shown are statistically significant ($p < 0.05$) when compared to learning without transfer.

These results clearly show that TVITM allows 5 vs. 4 Keepaway to be learned faster after training on 4 vs. 3 and/or 3 vs. 2. They also suggest that a multi-step process where tasks are made incrementally more challenging may produce faster learning times than a single application of TVITM. As a final result, a similar χ_X, χ_A, and ρ_{CMAC} can be constructed to significantly speed up learning in 6 vs. 5 as well (which also takes place on a $25m \times 25m$ field), as shown in Table 14.

Transfer in 6 vs. 5 with CMAC Function Approximation

# of 5 vs. 4 Episodes	Ave. 6 vs. 5 Time	Ave Total Time	Standard Deviation
0	22.85	22.85	1.71
1000	9.38	11.53	2.38

Table 14: 10 independent trials are averaged for learning 6 vs. 5 with and without transfer from 5 vs. 4. The threshold performance time is 8.0 seconds. A Student's t-test confirms that the difference is statistically significant ($p < 0.05$).

7.3 Variants of 3 vs. 2 Keepaway for Transfer into 4 vs. 3 Keepaway

In this section we introduce two novel variants of the 3 vs. 2 Keepaway task to show how TVITM with ρ_{CMAC} can fail to improve performance relative to learning without transfer.

We first modify 3 vs. 2 so that the reward is defined as $+1$ for each action, rather than $+1$ for each timestep. Players in the *3 vs. 2 Flat Reward* task can still learn to increase the average episode

time. We hypothesized that the changes in reward structure would prevent TVITM from successfully improving performance in the standard 4 vs. 3 task because of the different reward structure.

We next modify 3 vs. 2 so that the reward is defined to be -1 for each timestep. The task of *3 vs. 2 Giveaway* is thus very different from Keepaway. Given the available actions, the optimal action for players is for the player closest to the takers to hold the ball until the takers captures it. We hypothesized that using TVITM from Giveaway to 4 vs. 3 Keepaway would produce *negative transfer*, where the required target task training time is increased by using transfer.

Table 15 shows the results of using these two 3 vs. 2 source task variants and compares them to using Keepaway as a source task and to learning without transfer. Transfer from the Flat Reward tasks gives a benefit relative to learning without transfer, but not nearly as much as transferring from 3 vs. 2 Keepaway. Students t-tests determine that transfer after 3,000 episodes of Giveaway is significantly slower than learning without transfer.

Transfer into 4 vs. 3 with ρ_{CMAC}: different source tasks

Source Task	# of 3 vs. 2 Episodes	Ave. 4 vs. 3 Time	Ave. Total Time	Std. Dev.
none	0	30.84	30.84	4.72
Keepaway	1000	16.95	19.70	5.5
Keepaway	3000	9.12	18.79	2.73
Flat Reward	1000	25.11	27.62	6.31
Flat Reward	3000	19.42	28.03	8.62
Giveaway	1000	27.05	28.58	10.71
Giveaway	3000	32.94	37.10	8.96

Table 15: Results compare transferring from three different source tasks. Each line is an average of 30 independent trials. The 3 vs. 2 Flat Reward task improves performance relative to learning without transfer, but less than when transferring from Keepaway. The 3 vs. 2 Giveaway task can decrease 4 vs. 3 performance when it is used as a source task.

7.4 Transferring from Knight Joust to 4 vs. 3 Keepaway

In this section we show that TVITM can successfully transfer between the gridworld Knight Joust task and 4 vs. 3 Keepaway. We use a variant of ρ_{CMAC} to transfer the learned weights, because Knight Joust is learned with a tabular function approximator rather than a CMAC. This representation choice results in changes to the syntax of ρ_{CMAC}, as described in Algorithm 3. Note that this new variant of the transfer functional is not necessitated by the novel target task and that if we learned Knight Joust with a CMAC, the original ρ_{CMAC} would be sufficient for transfer between Knight Joust and 4 vs. 3.

The results in Table 16 report the average of 30 independent trials. The 4 vs. 3 transfer times (in simulator hours) are statistically different from learning without transfer (determined via Student's t-tests). Recall that the wall-clock time of the Knight Joust simulator is negligible and thus, in practice, the 4 vs. 3 time is the same as the total time.

The reader will notice that the number of source task episodes used in these experiments is much larger than other experiments in this paper. The reason for this is two-fold. First, we learn Knight Joust with tabular function approximation, which is significantly slower to learn than a CMAC, for instance, because there is no generalization. Secondly, because the wall-clock time requirements

Algorithm 3 APPLICATION OF ρ_{CMAC} FROM TABULAR FUNCTION APPROXIMATION

$n_{source} \leftarrow$ number of variables in source task
for each non-zero Q-value, q_i in the source task's Q-table **do**
  $a_{source} \leftarrow$ action corresponding to q_i
  for each state variable, x_{source}, in source task **do**
    for each value x_{target} such that $\chi_X(x_{target}) = x_{source}$ **do**
      for each value a_{target} such that $\chi_A(a_{target}) = a_{source}$ **do**
        $j \leftarrow$ the tile in the target CMAC activated by x_{target}, a_{target}
        $w_j \leftarrow (q_i / n_{source})$
$w_{Average} =$ average value of all non-zero weights in the target CMAC
for each weight w_j in the target CMAC **do**
  if $w_j = 0$ **then**
    $w_j \leftarrow w_{Average}$

Transfer from Knight Joust into 4 vs. 3

# of Knight Joust Episodes	Ave. 4 vs. 3 Time	Standard Deviation
0	30.84	4.72
25,000	24.24	16.18
50,000	18.90	13.20

Table 16: Results from using Knight Joust to speed up learning in 4 vs. 3 Keepaway. Knight Joust is learned with Q-learning and tabular function approximation and Keepaway players are learned using Sarsa with CMAC function approximation. Both transfer times are significantly less than learning without transfer, as determined via Student's t-tests ($p < 0.05$).

for this domain were so small, we felt justified in allowing the source task learners run until learning plateaued (which takes roughly 50,000 episodes).

The main importance of these results is showing that TVITM can successfully transfer between different tasks with very different dynamics. Keepaway has stochastic actions, is partially observable, and uses a continuous state space. In contrast, Knight Joust has no stochasticity in the player's actions, is fully observable, and has a discrete state space.

8. Discussion and Future Work

We consider the research reported in this article to be a first step towards autonomous transfer learning. In particular, the results presented in this article serve mainly as an existence proof that TVITM *can* be effective for speeding up learning in a target task after training in a source task. As such, it opens up the door for future research that is necessary to build it into a fully general autonomous transfer method. Specifically,

1. Can χ_X and χ_A be learned?

2. When concerned with the total training time of both tasks, what is the optimal amount of training time to spend in the source task?

3. How can an agent determine if two tasks are related so that transfer will be able to impart some advantage?

In this work we construct ρs for learners from a pair of χ_X, χ_A inter-task mappings for a given pair of tasks. There has been initial progress in learning such mappings, as discussed in the next section. In the future we intend to make this process more automatic or completely autonomous.

Another question that arises from this work is: if the ultimate goal is to learn the target task in the minimal amount of time, how can one determine the optimal amount of training in the source task automatically based on task characteristics? While it is clear from our results that spending more time learning 3 vs. 2 often decreases the amount of time it takes to learn 4 vs. 3 Keepaway, it is unclear how to determine the number of 3 vs. 2 episodes to use *a priori* when the goal is to minimize overall training time. It is likely that such a calculation or heuristic will have to consider the structure of the two tasks, how they are related, and the specifics of the ρ used.

A fundamental difficulty of transfer learning is determining whether and how two tasks are similar. Here we only consider the case where the agents are directed to use previous experience to speed up learning in a new task. In practice, it may be the case that the agent has no experience that is relevant and the optimal approach is to simply learn the target task without transfer.

Consider the problem of learners that have trained in multiple tasks and have built up different action-value functions for each of those tasks. When a new task is presented to the learners, they must now decide from which task to perform the transfer. This situation is analogous to presenting a human a solution to a problem and then ask how to solve a related problem. Research has shown (Gick and Holyoak, 1980) that humans are not good at this type of *analogy* problem. For instance, Gick and Holyoak presented subjects with a source task and a solution. When they later presented a similar target task, most people were initially unable to solve the target problem. However, told to use a strategy similar to that used for a past problem, 90% of the subjects were able to discover the analogy and solve the target problem. It is likely that computerized learners will have similar problems deciding if two tasks are related at all, particularly when agents have built up experience on many different kinds of problems.

Our TVITM methodology relies on being able to find an inter-task mapping between similar states and actions. The mapping should identify state variables and actions that have similar effects on the long-term discounted reward. If, for instance, the Keepaway task were changed so that instead of receiving a reward of $+1$ at every time step, you received a $+10$, the ρ could be trivially modified so that all the weights were multiplied by 10. However, if the reward structure is more significantly changed, to that of *Giveaway* for instance, ρ would need to be dramatically changed, if it could be formulated at all.[12]

We hypothesize that the main requirement for TVITM to successfully transfer is that, on average, at least one of the following is true:

1. The best learned actions in the source task, for a given state, be mapped to the best action in the target task via the inter-task mappings.

2. The average Q-values learned for states are of the correct magnitude in the trained target task's function approximator.

12. The "obvious" solution of multiplying all weights by -1 would not work, for instance. In Keepaway a keeper typically learns to hold the ball until a taker comes within roughly $6m$. Thus, if all weights from this policy were multiplied by -1, the keepers would continually pass the ball until a taker came within $6m$. These Giveaway episodes would last much longer than simply forcing the first keeper to the ball to always hold, which is very easily learned.

The first condition will work to bias the learner so that the best actions in the target task are chosen more often, even if these actions' Q-values are incorrect. The second condition will make learning faster because smaller adjustments to the function approximators' weights will be needed to reach their optimal values, even if the optimal actions are not initially chosen. In this work, an example of the first condition being met is that a keeper learns to hold the ball in the source task until forced to pass. Hold is often the correct action in both 3 vs. 2 and 4 vs. 3 when the takers are far away from the ball. The second condition is also met between 3 vs. 2 and 4 vs. 3 by virtue of similar reward structures and roughly similar episode lengths. If either of these conditions were not true, the transfer functional we employed would have to account for the differences (or suffer from reduced transfer efficacy).

It is important to recognize that domain knowledge contained in χ_X and χ_A is required to generate an effective ρ. As our experiments show, simply copying weights without respecting the inter-task mapping is not a viable method of transfer, as our function approximator representations necessarily differ between the two tasks due to changes in S and A. Simply putting the average value of the 3 vs. 2 weights into the 4 vs. 3 function approximator does not give nearly as much benefit as using a ρ which explicitly handles the different state and action values. Likewise, when we used $\rho_{modified}$ (introduced in Section 6.2), which copied the values for the weights corresponding to the some of the state variables incorrectly, learning in 4 vs. 3 was significantly slower. These results suggest that a ρ which is able to leverage inter-task similarities will outperform more simpleminded ρs.

In this work we have defined χ_X and χ_A so that the state variables and actions are mapped independently. This formulation was sufficient for all the source task and target task pairs considered in this work. However, there are likely tasks where these two mappings are intertwined. For instance, it could be that the actions map differently depending on the agent's current location in state space. We would like to investigate how often such an interdependence would be beneficial, and how our formulation could be enhanced to account for such added complexity.

It is almost certainly possible to find pairs of tasks for which no ρ exists, where transfer would provide no benefit or even hinder learning. It is also possible to think of a pair of tasks for which transferring knowledge should be able to provide a benefit but that an intuitive ρ that allows speedup in learning cannot be found due to the complexity of the domain. This article focuses on providing an existence proof: we show that we are able to construct ρs for the Keepaway domain and that they provide significant benefits to learning. A main goal for our future research is to allow ρ to be constructed automatically between a given pair of tasks, potentially by learning χ_X and χ_A.

9. Related Work

The concept of seeding a learned behavior with some initial simple behavior is not new. The psychological concept of shaping (Skinner, 1953) is well understood and many researchers have applied the idea to machine learning as well as animal training. There have been approaches to simplifying reinforcement learning by manipulating the transition function, the agent's initial state, and/or the reward function, as reviewed in the following paragraphs.

Past research confirms that if two tasks are closely related the learned policy from a source task can be used to provide a good initial policy for a target task. For example, Selfridge et al. (1985) showed that the 1-D pole balancing task could be made harder over time by shortening the length of the pole and increasing its mass; when the learner was first trained on a longer and lighter pole it learned to succeed faster in the harder task with different dynamics and transition function. This

method thus changes the transition function T between pairs of tasks but leaves S, the velocity and angle of the pole, and A, move right or move left, unmodified.

Learning from easy missions (Asada et al., 1994) allows a human to change the start state of the learner, $s_{initial}$, making the task incrementally harder. Starting the learner near the exit of a maze and gradually allowing the learner to start further and further from the goal is a demonstration of this. This kind of direction allows the learner to spend less total time learning to perform the final task. Our work differs from these two methods because we allow the modification of S and A between tasks, rather than only T or $s_{initial}$.

Transfer of learning (Singh, 1992) applies specifically to temporally sequential subtasks. Using compositional learning, a large task may be broken down into subtasks that are easier to learn and have distinct beginning and termination conditions. However, the subtasks must all be very similar in that they have the same state spaces, action spaces, and environment dynamics. The reward functions R are allowed to differ. TVITM does not have the restriction that tasks must be divided into subtasks with these characteristics and, again, we have the additional flexibility of changing S and A between the source and target tasks.

Another successful idea, *reward shaping* (Colombetti and Dorigo, 1993; Mataric, 1994), also contrasts with TVITM. In reward shaping, learners are given an artificial problem which will allow the learner to train faster than if they had trained on the actual problem with different environmental rewards, R. However, the policy that is learned is designed to work on the original task as well as the artificial one. TVITM differs in intent in that we aim to transfer behaviors from existing, relevant tasks which can have different state and action spaces, rather than creating artificial problems which are easier for the agent to learn. In the RoboCup soccer domain, all the different Keepaway tasks may occur during the full task of simulated soccer. For instance, there may be times when three teammates on defense must keep the ball from two opponent forwards until another player comes into passing range. We therefore argue that we train on useful tasks rather than just simpler variations of a real task. Using TVITM, learners are able to take previously learned behaviors from related tasks and apply that behavior to harder tasks that can have different state and action spaces.

While these four methods allow the learner to spend less total time training, they rely on modification of the transition function, the initial start state, or the reward function to create artificial problems to train on. We contrast this with TVITM where we allow the state and/or action spaces to change between actual tasks. This added flexibility permits TVITM to be applied to a wider range of domains and tasks than the other aforementioned methods. Furthermore, TVITM does not preclude the modification of the transition function, the start state, or the reward function and can therefore be combined with other methods if desired.

In some problems where subtasks are clearly defined by state features, the subtasks can be automatically identified (Drummond, 2002) and leveraged to increase learning rates. This method is only directly applicable to tasks in which features clearly define subtasks. Furthermore, if the shape of the various regions in the value function are too complex and the smoothness assumption is violated too often, the algorithm to automatically detect subtasks will fail.

Learned subroutines have been successfully transfered in a hierarchical reinforcement learning framework (Andre and Russell, 2002). By analyzing two tasks, subroutines may be identified which can be directly reused in a target task that has a slightly modified state space. The learning rate for the target task can be substantially increased by duplicating the local sub-policy. This work can be thought of as another example in which ρ has been successfully constructed, but in a very different way.

Imitation is another technique which may transfer knowledge from one learner to another (Price and Boutilier, 2003). However, there is the assumption that "the mentor and observer have similar abilities" and thus may not be directly applicable when the number of dimensions of the state space changes or the agents have a qualitatively different action set. Other research (Fern et al., 2004) has shown that it is possible to learn policies for large-scale planning tasks that generalize across different tasks in the same domain. Using this method, researchers are able to speed up learning in different tasks without explicitly transferring any knowledge, as the policy is defined for the planning domain rather than a specific task.

Another related approach (Guestrin et al., 2003) uses linear programming to determine value functions for classes of similar agents. Rather than treating the different agents independently, all agents in the same class use a single value function. The target task is assumed to have similar transition functions and rewards for each class of agent. Thus the authors can directly insert the class-based value subfunctions into agents in the new task, even though there are a different number of objects (and thus different state and action spaces). Although no learning is performed in the new world, the previously learned value functions may still perform better than a baseline hand-coded strategy. However, as the authors themselves state, the technique will not perform well in heterogeneous environments or domains with "strong and constant interactions between many objects (e.g., RoboCup)." Our work is further differentiated as we continue learning in the target task after performing transfer. While the initial performance in the new domain may be increased after loading learned action-value functions compared to learning without transfer, we have found that the primary benefit is an increased learning rate.

The technique of *autonomous shaping* (Konidaris and Barto, 2006) may prove to be useful for transfer learning, particularly in agents that have many sensors. In this work the authors show that reward shaping may be learned on-line by the agent. If a later task has a similar reward structure and actions, the learned reward shaping will help the agent initially have a much higher performance than if it were learning without transfer. For instance, if a signal device (a beacon) is near the goal state, the agent may learn a shaping reward that gives an internal reward for approaching the beacon even if the environmental reward is zero. This work does not directly address how to handle novel actions (specifically, actions which are not in the source task's *agent space*). Additionally, while the authors limit themselves to transfer between "reward-linked" tasks, no method is given for determining if a sequence of tasks are reward-linked and a learned shaping function will not necessarily be useful in a given target task.

Policies from different tasks can also be learned and then used to speed up the current task (Fernandez and Veloso, 2006). This technique relies on having a set of tasks where S, A, and T are constant, but the goal state moves. When the agent is placed in a new task, it can learn to either exploit a past policy, exploit the policy that it is currently learning, or explore. This technique is currently restrictive in that it requires S, A, and T to be unchanged between the set of tasks, that there be one goal state, and that all rewards other than the goal state be zero. However, the idea of building a library of policies may be a critical one for transfer learning. For instance, once an agent has trained in multiple Keepaway tasks it would ideally be able to recall any of these policies if it were placed in the same task or use the most similar of them to speed up the current task.

Automatically generated advice can also be used to speed up learning in transfer (Torrey et al., 2005). This method allows a RL learner to build up a model for the source task and then extract general advice. A human then provides a translation for this advice into the new task, similar to our ρ. The modified advice is then used as constraints in a knowledge-based support vector regression

method. Instead of setting the initial Q-values as in our work, the advice sets relative preferences for different actions in different states and thus may work when the reward structure of the tasks change. One apparent shortcoming of this work is that the initial performance and learning rate of the agents is only slightly improved relative to learning without transfer, but the results suggest that advice from one task may be able to help learn a second, related, task. It is also worth noting that while we cannot directly compare the performance of our two methods because we have not implemented the "Breakaway" domain, the relative speedup in learning from TVITM is much greater than that obtained from automatically generating advice.

Wilson et al. (2007) take a different approach by showing that a prior may be transferred in a hierarchical Bayesian reinforcement learning setting. In this work, the authors consider a multitask setting where the goal is to be able to learn quickly on an MDP drawn from a fixed but unknown distribution of MDPs. Learning on subsequent tasks shows a clear performance on a novel task drawn from this distribution but no attempt is made to reduce the total training time.

This work has demonstrated that inter-task mappings can be used to transfer action-value functions. Taylor et al. (2007) demonstrates that the same mappings can be used to transfer policies, learned with a genetic algorithm, between tasks. Rules have also been successfully used (Taylor and Stone, 2007) in conjunction with inter-task mappings as a way of transferring between tasks that used different RL learning methods. Using such a translation mechanism allows, for instance, transfer between a source task policy search learner and a value function target task learner.

There have also been recent advances in learning relationships between related RL tasks, a topic beyond the scope of the current article. Liu and Stone (2006) define *qualitative dynamic Bayes networks* which summarize the effect of actions on different state variables. After networks for the source and target tasks are defined by hand, a graph mapping technique can be used to automatically find inter-task mappings between the two tasks. Taylor et al. (2007) show that it is also possible to learn both χ_A and χ_X by using a classification technique. To enable such a method, the method must be provided *task-independent objects* which describe an object in a task with a set number of state variables. This assumption is also leveraged in other work (Soni and Singh, 2006), but only the state-feature mapping is learned (the action mapping χ_A is hand-coded). *AtEase* (Talvitie and Singh, 2007) is an algorithm that generates a number of possible state-feature mappings and then uses a multi-armed bandit approach to select the best mapping. However, the action mapping (χ_A) is assumed, and learning is not allowed in the target task after an appropriate mapping is selected.

10. Conclusions

This article describes the implementation and results from learning Keepaway with Sarsa, a standard TD method, and three different function approximators. We introduce the transfer via inter-task mapping method for speeding up reinforcement learning and give empirical evidence in the Keepaway domain of its usefulness. Rather than utilizing abstract knowledge, this transfer method is able to leverage the weights from function approximators specifying action-value functions, a very task-specific form of knowledge. We first give formulations of how to define transfer functionals for the different function approximators, or re-use learned weights via Q-value Reuse, from a single pair of inter-task mappings. We proceed to show that agents using all three function approximation methods can learn to reach a target performance faster in the target task. Additionally, we show that the total training time can be reduced using TVITM when compared to simply learning the final task without transfer.

We give further evidence that TVITM is useful for speeding up learning by utilizing the 5 vs. 4 Keepaway task, which suggests that this method will scale up to even more complex problems. We have shown that the TVITM method is robust to some changes in the transition function, such as when the effectiveness of actuators in the two tasks differ. This flexibility may prove critical when transferring behavior between agents situated in the real world, where environmental conditions may cause sensors and actuators to have different behaviors at different times.

We introduce a novel variant of Knight Joust, a gridworld task, and demonstrate that transfer between it and Keepaway is effective despite substantial qualitative differences in the two tasks. We also show how transfer efficacy is reduced when the source task and target task are less related, such as when using 3 vs. 2 Flat Reward or 3 vs. 2 Giveaway as source tasks.

When considered as a whole, the experiments presented in this article establish that TVITM can be used successfully for transferring action-value functions between tasks and reducing training time. This article therefore constitutes a first step towards a fully general and autonomous transfer method within the RL framework.

Acknowledgments

We would like to thank Gregory Kuhlmann for his help with Keepaway experiments described in this article, Cynthia Matuszek and Shimon Whiteson for useful discussions, and the anonymous reviewers for their detailed and constructed comments. This research was supported in part by NSF CAREER award IIS-0237699, NSF award EIA-0303609, and DARPA grant HR0011-04-1-0035.

References

James S. Albus. *Brains, Behavior, and Robotics*. Byte Books, Peterborough, NH, 1981.

David Andre and Stuart J. Russell. State abstraction for programmable reinforcement learning agents. In *Proc. of the Eighteenth National Conference on Artificial Intelligence*, pages 119–125, 2002.

David Andre and Astro Teller. Evolving team Darwin United. In Minoru Asada and Hiroaki Kitano, editors, *RoboCup-98: Robot Soccer World Cup II*, pages 346–351. Springer Verlag, Berlin, 1999.

Minoru Asada, Shoichi Noda, Sukoya Tawaratsumida, and Koh Hosoda. Vision-based behavior acquisition for a shooting robot by using a reinforcement learning. In *Proc. of IAPR/IEEE Workshop on Visual Behaviors-1994*, pages 112–118, 1994.

Steven J. Bradtke and Michael O. Duff. Reinforcement learning methods for continuous-time Markov decision problems. In G. Tesauro, D. Touretzky, and T. Leen, editors, *Advances in Neural Information Processing Systems*, volume 7, pages 393–400, San Mateo, CA, 1995. Morgan Kaufmann.

Mao Chen, Ehsan Foroughi, Fredrik Heintz, Spiros Kapetanakis, Kostas Kostiadis, Johan Kummeneje, Itsuki Noda, Oliver Obst, Patrick Riley, Timo Steffens, Yi Wang, and Xiang Yin. Users manual: RoboCup soccer server manual for soccer server version 7.07 and later, 2003. Available at `http://sourceforge.net/projects/sserver/`.

Marco Colombetti and Marco Dorigo. Robot Shaping: Developing Situated Agents through Learning. Technical Report TR-92-040, International Computer Science Institute, Berkeley, CA, 1993.

Robert H. Crites and Andrew G. Barto. Improving elevator performance using reinforcement learning. In D. S. Touretzky, M. C. Mozer, and M. E. Hasselmo, editors, *Advances in Neural Information Processing Systems 8*, pages 1017–1023, Cambridge, MA, 1996. MIT Press.

Chris Drummond. Accelerating reinforcement learning by composing solutions of automatically identified subtasks. *Journal of Artificial Intelligence Research*, 16:59–104, 2002.

Alan Fern, Sungwook Yoon, and Robert Givan. Approximate policy iteration with a policy language bias. In Sebastian Thrun, Lawrence Saul, and Bernhard Schölkopf, editors, *Advances in Neural Information Processing Systems 16*. MIT Press, Cambridge, MA, 2004.

Fernando Fernandez and Manuela Veloso. Probabilistic policy reuse in a reinforcement learning agent. In *Proceedings of the 5th International Conference on Autonomous Agents and Multiagent Systems*, pages 720–727, 2006.

Mary L. Gick and Keith J. Holyoak. Analogical problem-solving. *Cognitive Psychology*, 12:306–355, 1980.

Carlos Guestrin, Daphne Koller, Chris Gearhart, and Neal Kanodia. Generalizing plans to new environments in relational mdps. In *International Joint Conference on Artificial Intelligence (IJCAI-03)*, Acapulco, Mexico, August 2003.

George Konidaris and Andrew Barto. Autonomous shaping: Knowledge transfer in reinforcement learning. In *Proceedings of the 23rd International Conference on Machine Learning*, pages 489–496, 2006.

Yaxin Liu and Peter Stone. Value-function-based transfer for reinforcement learning using structure mapping. In *Proceedings of the Twenty-First National Conference on Artificial Intelligence*, pages 415–20, July 2006.

Maja J. Mataric. Reward functions for accelerated learning. In *International Conference on Machine Learning*, pages 181–189, 1994.

Kishan Mehrotra, Chilukuri K. Mohan, and Sanjay Ranka. *Elements of Artificial Neural Networks*. MIT Press, Cambridge, MA, USA, 1997. ISBN 0-262-13328-8.

Itsuki Noda, Hitoshi Matsubara, Kazuo Hiraki, and Ian Frank. Soccer server: A tool for research on multiagent systems. *Applied Artificial Intelligence*, 12:233–250, 1998.

Bob Price and Craig Boutilier. Accelerating reinforcement learning through implicit imitation. *Journal of Artificial Intelligence Research*, 19:569–629, 2003.

Martin L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc., 1994. ISBN 0471619779.

Martin Riedmiller, Author Merke, David Meier, Andreas Hoffman, Alex Sinner, Ortwin Thate, and Ralf Ehrmann. Karlsruhe brainstormers—a reinforcement learning approach to robotic soccer. In Peter Stone, Tucker Balch, and Gerhard Kraetszchmar, editors, *RoboCup-2000: Robot Soccer World Cup IV*, pages 367–372. Springer Verlag, Berlin, 2001.

Gavin Rummery and Mahesan Niranjan. On-line Q-learning using connectionist systems. Technical Report CUED/F-INFENG-RT 116, Engineering Department, Cambridge University, 1994.

Oliver G. Selfridge, Richard S. Sutton, and Andrew G. Barto. Training and tracking in robotics. In *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, pages 670–672, 1985.

Satinder P. Singh. Transfer of learning by composing solutions of elemental sequential tasks. *Machine Learning*, 8:323–339, 1992.

Satinder P. Singh and Richard S. Sutton. Reinforcement learning with replacing eligibility traces. *Machine Learning*, 22:123–158, 1996.

Burrhus F. Skinner. *Science and Human Behavior*. Colliler-Macmillian, 1953. ISBN 0029290406.

Vishal Soni and Satinder Singh. Using homomorphisms to transfer options across continuous reinforcement learning domains. In *Proceedings of the Twenty First National Conference on Artificial Intelligence*, July 2006.

Peter Stone and Richard S. Sutton. Keepaway soccer: a machine learning testbed. In Andreas Birk, Silvia Coradeschi, and Satoshi Tadokoro, editors, *RoboCup-2001: Robot Soccer World Cup V*, volume 2377 of *Lecture Notes in Artificial Intelligence*, pages 214–223. Springer Verlag, Berlin, 2002.

Peter Stone, Richard S. Sutton, and Gregory Kuhlmann. Reinforcement learning for RoboCup-soccer keepaway. *Adaptive Behavior*, 13(3):165–188, 2005.

Peter Stone, Gregory Kuhlmann, Matthew E. Taylor, and Yaxin Liu. Keepaway soccer: From machine learning testbed to benchmark. In Itsuki Noda, Adam Jacoff, Ansgar Bredenfeld, and Yasutake Takahashi, editors, *RoboCup-2005: Robot Soccer World Cup IX*, volume 4020, pages 93–105. Springer Verlag, Berlin, 2006.

Richard S. Sutton and Andrew G. Barto. *Introduction to Reinforcement Learning*. MIT Press, 1998. ISBN 0262193981.

Erik Talvitie and Satinder Singh. An experts algorithm for transfer learning. In *Proceedings of the Twentieth International Joint Conference on Artificial Intelligence*, 2007.

Matthew E. Taylor and Peter Stone. Behavior transfer for value-function-based reinforcement learning. In Frank Dignum, Virginia Dignum, Sven Koenig, Sarit Kraus, Munindar P. Singh, and Michael Wooldridge, editors, *The Fourth International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 53–59, New York, NY, July 2005. ACM Press.

Matthew E. Taylor and Peter Stone. Cross-domain transfer for reinforcement learning. In *Proceedings of the Twenty-Fourth International Conference on Machine Learning*, June 2007.

Matthew E. Taylor, Peter Stone, and Yaxin Liu. Value functions for RL-based behavior transfer: A comparative study. In *Proceedings of the Twentieth National Conference on Artificial Intelligence*, July 2005.

Matthew E. Taylor, Shimon Whiteson, and Peter Stone. Transfer via inter-task mappings in policy search reinforcement learning. In *The Sixth International Joint Conference on Autonomous Agents and Multiagent Systems*, May 2007.

Gerald Tesauro. TD-Gammon, a self-teaching backgammon program, achieves master-level play. *Neural Computation*, 6(2):215–219, 1994.

Lisa Torrey, Trevor Walker, Jude Shavlik, and Richard Maclin. Using advice to transfer knowledge acquired in one reinforcement learning task to another. In *Proceedings of the Sixteenth European Conference on Machine Learning*, 2005.

Christopher J. C. H. Watkins. *Learning from Delayed Rewards*. PhD thesis, King's College, Cambridge, UK, 1989.

Aaron Wilson, Alan Fern, Soumya Ray, and Prasad Tadepalli. Multi-task reinforcement learning: a hierarchical bayesian approach. In *ICML '07: Proceedings of the 24th international conference on Machine learning*, pages 1015–1022, New York, NY, USA, 2007. ACM Press.

Journal of Machine Learning Research 8 (2007) 2169-2231 Submitted 6/06; Revised 3/07; Published 10/07

Proto-value Functions: A Laplacian Framework for Learning Representation and Control in Markov Decision Processes

Sridhar Mahadevan MAHADEVA@CS.UMASS.EDU
Department of Computer Science
University of Massachusetts
Amherst, MA 01003, USA

Mauro Maggioni MAURO.MAGGIONI@DUKE.EDU
Department of Mathematics and Computer Science
Duke University
Durham, NC 27708

Editor: Carlos Guestrin

Abstract

This paper introduces a novel *spectral* framework for solving Markov decision processes (MDPs) by jointly learning representations and optimal policies. The major components of the framework described in this paper include: (i) A general scheme for constructing representations or *basis functions* by diagonalizing symmetric *diffusion* operators (ii) A specific instantiation of this approach where global basis functions called *proto-value functions* (PVFs) are formed using the eigenvectors of the *graph Laplacian* on an undirected graph formed from state transitions induced by the MDP (iii) A three-phased procedure called *representation policy iteration* comprising of a sample collection phase, a representation learning phase that constructs basis functions from samples, and a final parameter estimation phase that determines an (approximately) optimal policy within the (linear) subspace spanned by the (current) basis functions. (iv) A specific instantiation of the RPI framework using least-squares policy iteration (LSPI) as the parameter estimation method (v) Several strategies for scaling the proposed approach to large discrete and continuous state spaces, including the *Nyström* extension for out-of-sample interpolation of eigenfunctions, and the use of *Kronecker sum factorization* to construct compact eigenfunctions in product spaces such as factored MDPs (vi) Finally, a series of illustrative discrete and continuous control tasks, which both illustrate the concepts and provide a benchmark for evaluating the proposed approach. Many challenges remain to be addressed in scaling the proposed framework to large MDPs, and several elaboration of the proposed framework are briefly summarized at the end.

Keywords: Markov decision processes, reinforcement learning, value function approximation, manifold learning, spectral graph theory

1. Introduction

This paper introduces a novel *spectral* framework for solving Markov decision processes (MDPs) (Puterman, 1994) where both the underlying representation or basis functions and (approximate) optimal policies within the (linear) span of these basis functions are simultaneously learned. This framework addresses a major open problem not addressed by much previous work in the field of *approximate dynamic programming* (Bertsekas and Tsitsiklis, 1996) and *reinforcement learning*

(Sutton and Barto, 1998), where the set of "features" or *basis functions* mapping a state s to a k-dimensional real vector $\phi(s) \in \mathbb{R}^k$ is usually hand-engineered.

The overall framework can be summarized briefly as follows. The underlying task environment is modeled as an MDP, where the system dynamics and reward function are typically assumed to be unknown. An agent explores the underlying state space by carrying out actions using some policy, say a random walk. Central to the proposed framework is the notion of a *diffusion model* (Coifman et al., 2005a; Kondor and Lafferty, 2002): the agent constructs a (directed or undirected) graph connecting states that are "nearby". In the simplest setting, the diffusion model is defined by the *combinatorial graph Laplacian* matrix $L = D - W$, where W is a symmetrized weight matrix, and D is a diagonal matrix whose entries are the row sums of W.[1] Basis functions are derived by diagonalizing the Laplacian matrix L, specifically by finding its "smoothest" eigenvectors that correspond to the smallest eigenvalues. Eigenvectors capture large-scale temporal properties of a transition process. In this sense, they are similar to value functions, which reflect the accumulation of rewards over the long run. The similarity between value functions and the eigenvectors of the graph Laplacian sometimes can be remarkable, leading to a highly compact encoding (measured in terms of the number of basis functions needed to encode a value function). Laplacian basis functions can be used in conjunction with a standard "black box" parameter estimation method, such as Q-learning (Watkins, 1989) or least-squares policy iteration (LSPI) (Lagoudakis and Parr, 2003) to find the best policy representable within the space of the chosen basis functions.

While the overall goal of learning representations is not new within the context of MDPs—it has been addressed by Dayan (1993) and Drummond (2002) among others—our approach is substantially different from previous work. The fundamental idea is to construct basis functions for solving MDPs by diagonalizing symmetric diffusion operators on an empirically learned graph representing the underlying state space. A diffusion model is intended to capture information flow on a graph or a *manifold*.[2] A simple diffusion model is a random walk on an undirected graph, where the probability of transitioning from a vertex (state) to its neighbor is proportional to its degree, that is $P_r = D^{-1}W$ (Chung, 1997). As we will see in Section 3, the combinatorial Laplacian operator L defined in the previous paragraph is closely related spectrally to the random walk operator P_r. A key advantage of diffusion models is their simplicity: it can be significantly easier to estimate a "weak" diffusion model, such as the undirected random walk P_r or the combinatorial Laplacian L, than to learn the true underlying transition matrix P^π of a policy π.

The proposed framework can be viewed as automatically generating subspaces on which to project the value function using spectral analysis of operators on graphs. This differs fundamentally from many past attempts at basis function generation, for example tuning the parameters of pre-defined basis functions (Menache et al., 2005; Kveton and Hauskrecht, 2006), dynamically allocating new parametric basis functions based on state space trajectories (Kretchmar and Anderson, 1999), or generating basis functions using the Bellman error in approximating a specific value function (Keller et al., 2006; Petrik, 2007; Parr et al., 2007; Patrascu et al., 2002). The main contribution

1. Section 9 describes how to generalize this simple diffusion model in several ways, including *directed* graphs where the symmetrization is based on the *Perron* vector or the leading eigenvector associated with the largest eigenvalue (Chung, 2005), state-action diffusion models where the vertices represent state-action pairs, and diffusion models for temporally extended actions.
2. Intuitively, a manifold is a (finite or infinite) set that looks "locally Euclidean", in that an invertible mapping can be defined from a neighborhood around each element of the set to $\mathbb{R}^n$. There are technical conditions that additionally need to be satisfied for a manifold to be *smooth*, as explained in Lee (2003).

of this paper is to show how to construct *novel* non-parametric basis functions whose shapes reflect the geometry of the environment.

In this paper, basis functions are constructed from spectral analysis of diffusion operators where the resulting representations are constructed without explicitly taking rewards into account. This approach can be contrasted with recent approaches that explicitly use reward information to generate basis functions (Keller et al., 2006; Petrik, 2007; Parr et al., 2007). There are clear advantages and disadvantages to these two approaches. In the non-reward based approach, basis functions can be more easily transferred across MDPs in situations where an agent is required to solve multiple tasks defined on a common state (action) space. Furthermore, basis functions constructed using spectral analysis reflect global geometric properties, such as bottlenecks and symmetries in state spaces, that are invariant across multiple MDPs on the same state (action) space. Finally, in the full control learning setting studied here, the agent does not initially know the true reward function or transition dynamics, and building representations based on estimating these quantities introduces another potential source of error. It is also nontrivial to learn accurate transition models, particularly in continuous MDPs. However, in other settings such as planning, where the agent can be assumed to have a completely accurate model, it is entirely natural and indeed beneficial to exploit reward or transition dynamics in constructing basis functions. In particular, it is possible to design algorithms for basis function generation with provable performance guarantees (Parr et al., 2007; Petrik, 2007), although these theoretical results are at present applicable only to the more limited case of evaluating a *fixed* policy. The proposed framework can in fact be easily extended to use reward information by building reward-based diffusion models, as will be discussed in more detail in Section 9.

Since eigenvectors of the graph Laplacian form "global" basis functions whose support is the entire state space, each eigenvector induces a real-valued mapping over the state space. Consequently, we can view each eigenvector as a "proto-value" function (or PVF), and the set of PVFs form the "building blocks" of all value functions on a state space (Mahadevan, 2005a). Of course, it is easy to construct a complete orthonormal set of basis functions spanning all value functions on a graph: the unit vectors themselves form such a basis, and indeed, any collection of $|S|$ random vectors can (with high probability) be orthonormalized so that they are of unit length and "perpendicular" to each other. The challenge is to construct a *compact* basis set that is *efficient* at representing value functions with as *few* basis functions as possible. Proto-value functions differ from these obvious choices, or indeed other more common parametric choices such as radial basis functions (RBFs), polynomial bases, or CMAC, in that they are associated with the spectrum of the Laplacian which has an intimate relationship to the large-scale geometry of a state space. The eigenvectors of the Laplacian also provide a *systematic* organization of the space of functions on a graph, with the "smoothest" eigenvectors corresponding to the smallest eigenvalues (beginning with the constant function associated with the zero eigenvalue). By projecting a given value function on the space spanned by the eigenvectors of the graph Laplacian, the "spatial" content of a value function is mapped into a "frequency" basis, a hallmark of classical "Fourier" analysis (Mallat, 1989).

It has long been recognized that traditional parametric function approximators may have difficulty accurately modeling value functions due to nonlinearities in an MDP's state space (Dayan, 1993). Figure 1 illustrates the problem with a simple example.[3] In particular, as Dayan (1993) and Drummond (2002) among others have noted, states close in Euclidean distance may have values that are very far apart (e.g., two states on opposite sides of a wall in a spatial navigation task). While

3. Further details of this environment and similar variants are given in Section 2.1 and Section 4.2.

there have been several attempts to fix the shortcomings of traditional function approximators to address the inherent nonlinear nature of value functions, these approaches have lacked a sufficiently comprehensive and broad theoretical framework (related work is discussed in more detail in Section 8). We show that by rigorously formulating the problem of value function approximation as *approximating real-valued functions on a graph or manifold using a diffusion model*, a more general solution emerges that not only has broader applicability than these previous methods, but also enables a novel framework called *Representation Policy Iteration* (RPI) (Mahadevan, 2005b) where representation learning is interleaved with policy learning. The RPI framework consists of an outer loop that learns basis functions from sample trajectories, and an inner loop that consists of a control learner that finds improved policies.

Figure 2 shows a set of samples produced by doing a random walk in the inverted pendulum task. In many continuous control tasks, there are often physical constraints that limit the "degrees of freedom" to a lower-dimensional manifold, resulting in motion along highly constrained regions of the state space. Instead of placing basis functions uniformly in all regions of the state space, the proposed framework recovers the underlying manifold by building a graph based on the samples collected over a period of exploratory activity. The basis functions are then computed by diagonalizing a diffusion operator (the Laplacian) on the space of functions on the graph, and are thereby customized to the manifold represented by the state (action) space of a particular control task. In discrete MDPs, such as Figure 1, the problem is one of compressing the space of (value) functions $\mathbb{R}^{|S|}$ (or $\mathbb{R}^{|S|\times|A|}$ for action-value functions). In continuous MDPs, such as Figure 2, the corresponding problem is compressing the space of square-integrable functions on $\mathbb{R}^2$, denoted as $\mathbb{L}^2(\mathbb{R}^2)$. In short, the problem is one of dimensionality reduction not in the data space, but on the space of functions on the data.[4]

Both the discrete MDP shown in Figure 1 and the continuous MDP shown in Figure 2 have "inaccessible" regions of the state space, which can be exploited in focusing the function approximator to accessible regions. Parametric approximators, as typically constructed, do not distinguish between accessible and inaccessible regions. Our approach goes beyond modeling just the reachable state space, in that it also models the local *non-uniformity* of a given region. This non-uniform modeling of the state space is facilitated by constructing a graph operator which models the local density across regions. By constructing basis functions adapted to the non-uniform density and geometry of the state space, our approach extracts significant topological information from trajectories. These ideas are formalized in Section 3.

The additional power obtained from knowledge of the underlying state space graph or manifold comes at a potentially significant cost: the manifold representation needs to be learned, and furthermore, basis functions need to be computed from it. Although our paper demonstrates that eigenvector-type basis functions resulting from a diffusion analysis of graph-based manifolds can solve standard benchmark discrete and continuous MDPs, the problem of efficiently learning manifold representations of *arbitrary* MDPs is beyond the scope of this introductory paper. We discuss a number of outstanding research questions in Section 9 that need to be addressed in order to develop a more complete solution.

One hallmark of Fourier analysis is that the basis functions are localized in frequency, but not in time (or space). Hence, the eigenfunctions of the graph Laplacian are localized in frequency by being associated with a specific eigenvalue λ, but their support is in general the whole graph. This

4. The graph Laplacian induces a smoothing prior on the space of functions of a graph that can formally be shown to define a data-dependent reproducing kernel Hilbert space (Scholkopf and Smola, 2001).

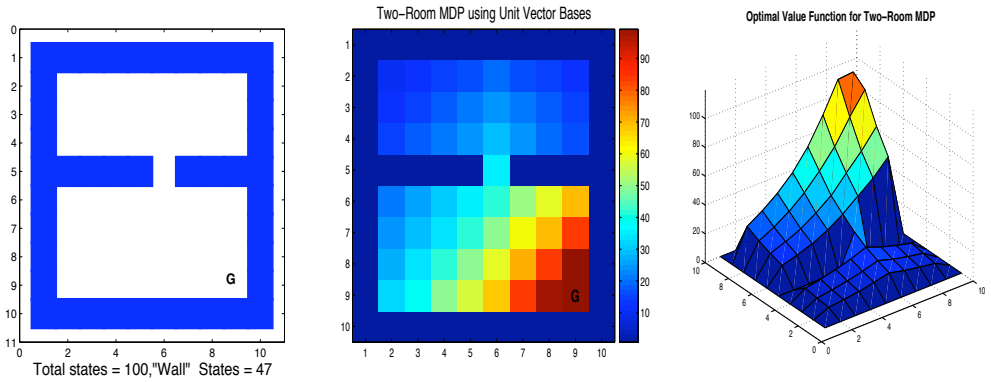


Figure 1: It is difficult to approximate nonlinear value functions using traditional parametric function approximators. Left: a "two-room" environment with 100 total states, divided into 57 accessible states (including one doorway state), and 43 inaccessible states representing exterior and interior walls (which are "one state" thick). Middle: a $2D$ view of the optimal value function for the two-room grid MDP, where the agent is (only) rewarded for reaching the state marked G by $+100$. Access to each room from the other is only available through a central door, and this "bottleneck" results in a strongly nonlinear optimal value function. Right: a $3D$ plot of the optimal value function, where the axes are reversed for clarity.

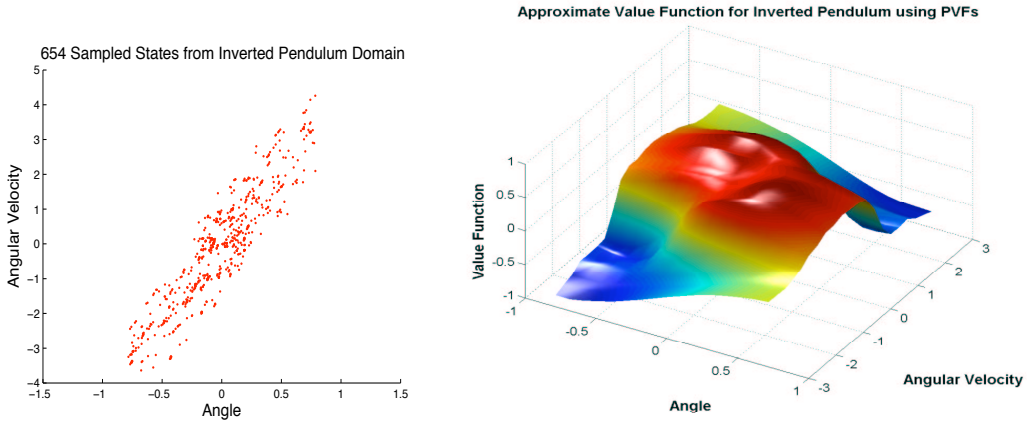


Figure 2: Left: Samples from a series of random walks in an inverted pendulum task. Due to physical constraints, the samples are largely confined to a narrow region. The proto-value function framework presented in this paper empirically models the underlying manifold in such continuous control tasks, and derives customized basis functions that exploit the unique structure of such point-sets in $\mathbb{R}^n$. Right: An approximation of the value function learned by using PVFs.

global characteristic raises a natural computational concern: can Laplacian bases be computed and represented compactly in large discrete and continuous spaces?[5] We will address this problem in particular cases of interest: large factored spaces, such as grids, hypercubes, and tori, lead naturally to *product spaces* for which the Laplacian bases can be constructed efficiently using *tensor products*. For continuous domains, by combining low-rank approximations and the *Nyström* interpolation method, Laplacian bases can be constructed quite efficiently (Drineas and Mahoney, 2005). Finally, a variety of other techniques can be used to sparsify Laplacian bases, including graph partitioning (Karypis and Kumar, 1999), matrix sparsification (Achlioptas et al., 2002), and automatic Kronecker matrix factorization (Van Loan and Pitsianis, 1993). Other sources of information can be additionally exploited to facilitate sparse basis construction. For example, work on hierarchical reinforcement learning surveyed in Barto and Mahadevan (2003) studies special types of MDPs called semi-Markov decision processes, where actions are temporally extended, and value functions are decomposed using the hierarchical structure of a task. In Section 9, we discuss how to exploit such additional knowledge in the construction of basis functions.

This research is related to recent work on manifold and spectral learning (Belkin and Niyogi, 2004; Coifman and Maggioni, 2006; Roweis and Saul, 2000; Tenenbaum et al., 2000). A major difference is that our focus is on solving Markov decision processes. Value function approximation in MDPs is related to regression on graphs (Niyogi et al., 2003) in that both concern approximation of real-valued functions on the vertices of a graph. However, value function approximation is fundamentally different since target values are initially unknown and must be determined by solving the Bellman equation, for example by iteratively finding a fixed point of the Bellman backup operator. Furthermore, the set of samples of the manifold is not given a priori, but is determined through active exploration by the agent. Finally, in our work, basis functions can be constructed multiple times by interleaving policy improvement and representation learning. This is in spirit similar to the design of kernels adapted to regression or classification tasks (Szlam et al., 2006).

The rest of the paper is organized as follows. Section 2 gives a quick summary of the main framework called Representation Policy Iteration (RPI) for jointly learning representations and policies, and illustrates a simplified version of the overall algorithm on the small two-room discrete MDP shown earlier in Figure 1. Section 3 motivates the use of the graph Laplacian from several different points of view, including as a spectral approximation of transition matrices, as well as inducing a smoothness regularization prior that respects the topology of the state space through a data-dependent kernel. Section 4 describes a concrete instance of the RPI framework using least-squares policy iteration (LSPI) (Lagoudakis and Parr, 2003) as the underlying control learning method, and compares PVFs with two parametric bases—radial basis functions (RBFs) and polynomials—on small discrete MDPs. Section 5 describes one approach to scaling proto-value functions to large discrete product space MDPs, using the Kronecker sum matrix factorization method to decompose the eigenspace of the combinatorial Laplacian. This section also compares PVFs against RBFs on the *Blockers* task (Sallans and Hinton, 2004). Section 6 extends PVFs to continuous MDPs using the *Nyström* extension for interpolating eigenfunctions from sampled states to novel states. A detailed evaluation of PVFs in continuous MDPs is given in Section 7, including

5. The global nature of Fourier bases have been exploited in other areas, for example they have led to significantly improved algorithms for learning boolean functions (Jackson, 1995). Circuit designers have discovered fast algorithms for converting state-based truth-table and decision diagram representations of boolean functions into Fourier representations using the Hadamard transformation (Thornton et al., 2001). The eigenvectors of the graph Laplacian on boolean hypercubes form the columns of the Hadamard matrix (Bernasconi, 1998).

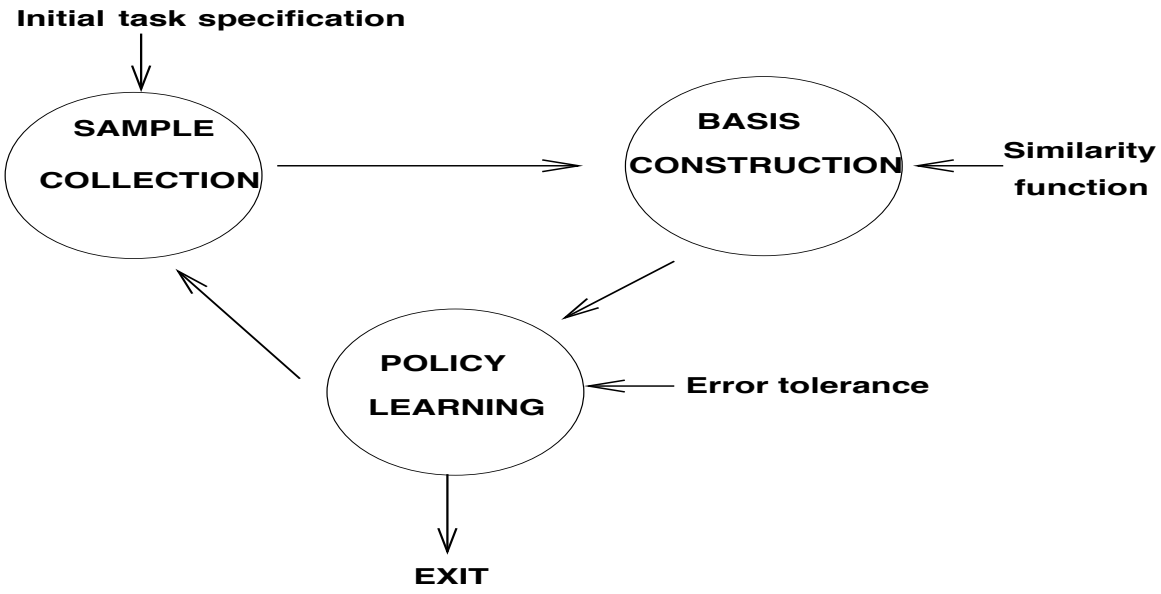


Figure 3: Flowchart of the unified approach to learning representation and behavior.

the inverted pendulum, the mountain car, and the Acrobot tasks (Sutton and Barto, 1998). Section 8 contains a brief description of previous work. Section 9 discusses several ongoing extensions of the proposed framework, including Kronecker product matrix factorization (Johns et al., 2007), multiscale diffusion wavelet bases (Mahadevan and Maggioni, 2006), and more elaborate diffusion models using directed graphs where actions are part of the representation (Johns and Mahadevan, 2007; Osentoski and Mahadevan, 2007).

2. Overview of The Framework

This section contains a brief summary of the overall framework, which we call *Representation Policy Iteration* (RPI) (Mahadevan, 2005b).[6] Figure 3 illustrates the overall framework. There are three main components: sample collection, basis construction, and policy learning. Sample collection requires a task specification, which comprises of a domain simulator (or alternatively a physically embodied agent like a robot), and an initial policy. In the simplest case, the initial policy can be a random walk, although it can also reflect a more informative hand-coded policy. The second phase involves constructing the bases from the collected samples using a diffusion model, such as an undirected (or directed) graph. This process involves finding the eigenvectors of a symmetrized graph operator such as the graph Laplacian. The final phase involves estimating the "best" policy representable in the span of the basis functions constructed (we are primarily restricting our attention to linear architectures, where the value function is a weighted linear combination of the bases). The entire process can then be iterated.

Figure 4 specifies a more detailed algorithmic view of the overall framework. In the sample collection phase, an initial random walk (perhaps guided by an informed policy) is carried out to

6. The term "Representation Policy Iteration" is used to succinctly denote a class of algorithms that jointly learn basis functions and policies. In this paper, we primarily use LSPI as the control learner, but in other work we have used control learners such as Q(λ) (Osentoski and Mahadevan, 2007).

obtain samples of the underlying manifold on the state space. The number of samples needed is an empirical question which will be investigated in further detail in Section 5 and Section 6. Given this set of samples, in the representation learning phase, an undirected (or directed) graph is constructed in one of several ways: two states can be connected by a unit cost edge if they represent temporally successive states; alternatively, a local distance measure such as k-nearest neighbor can be used to connect states, which is particularly useful in the experiments on continuous domains reported in Section 7. From the graph, proto-value functions are computed using one of the graph operators discussed below, for example the combinatorial or normalized Laplacian. The smoothest eigenvectors of the graph Laplacian (that is, associated with the smallest eigenvalues) are used to form the suite of proto-value functions. The number of proto-value functions needed is a model selection question, which will be empirically investigated in the experiments described later. The encoding $\phi(s) : S \rightarrow \mathbb{R}^k$ of a state is computed as the value of the k proto-value functions on that state. To compute a state action encoding, a number of alternative strategies can be followed: the figure shows the most straightforward method of simply replicating the length of the state encoding by the number of actions and setting all the vector components to 0 except those associated with the current action. More sophisticated schemes are possible (and necessary for continuous actions), and will be discussed in Section 9.

At the outset, it is important to point out that the algorithm described in Figure 4 is one of many possible designs that combine the learning of basis functions and policies. In particular, the RPI framework is an *iterative* approach, which interleaves the two phases of generating basis functions by sampling trajectories from policies, and then subsequently finding improved policies from the augmented set of basis functions. It may be possible to design alternative frameworks where basis functions are learned *jointly* with learning policies, by attempting to optimize some cumulative measure of optimality. We discuss this issue in more depth in Section 9.

2.1 Sample Run of RPI on the Two-Room Environment

The result of running the algorithm is shown in Figure 5, which was obtained using the following specific parameter choices.

- The state space of the two room MDP is as shown in Figure 1. There are 100 states totally, of which 43 states are inaccessible since they represent interior and exterior walls. The remaining 57 states are divided into 1 doorway state and 56 interior room states. The agent is rewarded by $+100$ for reaching state 89, which is the last accessible state in the bottom right-hand corner of room 2. In the $3D$ value function plots shown in Figure 5, the axes are reversed to make it easier to visualize the value function plot, making state 89 appear in the top left (diagonally distant) corner.

- 3463 samples were collected using off-policy sampling from a random walk of 50 episodes, each of length 100 (or terminating early when the goal state was reached).[7] Four actions (compass direction movements) were possible from each state. Action were stochastic. If a movement was possible, it succeeded with probability 0.9. Otherwise, the agent remained in

7. Since the approximated value function shown in Figure 5 is the result of a specific set of random walk trajectories, the results can vary over different runs depending on the number of times the only rewarding (goal) state was reached. Section 4.2 contains more detailed experiments that measures the learned policy over multiple runs.

`RPI` $(\pi_m, T, N, \varepsilon, k, O, \mu, \mathcal{D})$:

// π_m: Policy at the beginning of trial m
// T: Number of initial random walk trials
// N: Maximum length of each trial
// ε : Convergence condition for policy iteration
// k: Number of proto-value basis functions to use
// O: Type of graph operator used
// μ: Parameter for basis adaptation
// $\mathcal{D}$: Initial set of samples

Sample Collection Phase

1. **Off-policy or on-policy sampling:** Collect a data set of samples $\mathcal{D}_m = \{(s_i, a_i, s_{i+1}, r_i), \ldots\}$ by either randomly choosing actions (off-policy) or using the supplied initial policy (on-policy) for a set of T trials, each of maximum N steps (terminating earlier if it results in an absorbing goal state), and add these transitions to the complete data set $\mathcal{D}$.
2. **(Optional) Subsampling step:** Form a subset of samples $\mathcal{D}_s \subseteq \mathcal{D}$ by some subsampling method such as random subsampling or trajectory subsampling. For episodic tasks, optionally prune the trajectories stored in $\mathcal{D}_s$ so that only those that reach the absorbing goal state are retained.

Representation Learning Phase

3. Build a diffusion model from the data in $\mathcal{D}_s$. In the simplest case of discrete MDPs, construct an undirected weighted graph G from $\mathcal{D}$ by connecting state i to state j if the pair (i, j) form temporally successive states $\in S$. Compute the operator O on graph G, for example the normalized Laplacian $\mathcal{L} = D^{-\frac{1}{2}}(D - W)D^{-\frac{1}{2}}$.
4. Compute the k smoothest eigenvectors of O on the graph G. Collect them as columns of the basis function matrix Φ, a $|S| \times k$ matrix. The state action bases $\phi(s, a)$ can be generated from rows of this matrix by duplicating the state bases $\phi(s)$ $|A|$ times, and setting all the elements of this vector to 0 except for the ones corresponding to the chosen action.[a]

Control Learning Phase

5. Using a standard parameter estimation method (e.g., Q-learning or LSPI), find an ε-optimal policy π that maximizes the action value function $Q^\pi = \Phi w^\pi$ within the linear span of the bases Φ using the training data in $\mathcal{D}$.
6. **Optional:** Set the initial policy π_{m+1} to π and call `RPI` $(\pi_{m+1}, T, N, \varepsilon, k, O, \mu, \mathcal{D})$.

a. In large continuous and discrete MDPs, the basis matrix Φ need not be explicitly formed and the features $\phi(s, a)$ can be computed "on demand" as will be explained later.

Figure 4: This figure shows a generic algorithm for combining the learning of representation (or basis functions) from spectral analysis of random walks, and estimation of policies within their linear span. Elaborations of this framework will be studied in subsequent sections.

the same state. When the agent reaches state 89, it receives a reward of 100, and is randomly reset to an accessible interior state.

- An undirected graph was constructed from the sample transitions, where the weight matrix W is simply the adjacency $(0,1)$ matrix. The graph operator used was the normalized Laplacian $\mathcal{L} = D^{-\frac{1}{2}} L D^{-\frac{1}{2}}$, where $L = D - W$ is referred to as the combinatorial Laplacian (these graph operators are described in more detail in Section 3).

- 20 eigenvectors corresponding to the smallest eigenvalues of $\mathcal{L}$ (duplicated 4 times, one set for each action) are chosen as the columns of the state action basis matrix Φ. For example, the first four eigenvectors are shown in Figure 5. These eigenvectors are *orthonormal*: they are normalized to be of length 1 and are mutually perpendicular. Note how the eigenvectors are sensitive to the geometric structure of the overall environment. For example, the second eigenvector allows partitioning the two rooms since it is negative for all states in the first room, and positive for states in the second room. The third eigenvector is non-constant over only one of the rooms. The connection between the Laplacian and regularities such as symmetries and bottlenecks is discussed in more detail in Section 3.6.

- The parameter estimation method used was least-squares policy iteration (LSPI), with $\gamma = 0.8$. LSPI is described in more detail in Section 4.1.

- The optimal value function using unit vector bases and the approximation produced by 20 PVFs are compared using the $2D$ array format in Figure 6.

In the remainder of this paper, we will evaluate this framework in detail, providing some rationale for why the Laplacian bases are adept at approximating value functions, and demonstrating how to scale the approach to large discrete MDPs as well as continuous MDPs.

3. Representation Learning by Diffusion Analysis

In this section, we discuss the graph Laplacian, specifically motivating its use as a way to construct basis functions for MDPs. We begin with a brief introduction to MDPs, and then describe the spectral analysis of a restricted class of MDPs where the transition matrix is diagonalizable. Although this approach is difficult to implement for general MDPs, it provides some intuition into why eigenvectors are a useful way to approximate value functions. We then introduce the graph Laplacian as a symmetric matrix, which acts as a surrogate for the true transition matrix, but which is easily diagonalizable. It is possible to model non-symmetric actions and policies using more sophisticated symmetrization procedures (Chung, 2005), and we postpone discussion of this extension to Section 9. There are a number of other perspectives to view the graph Laplacian, namely as generating a data-dependent reproducing kernel Hilbert space (RKHS) (Scholkopf and Smola, 2001), as well as a way to generate nonlinear embeddings of graphs. Although a full discussion of these perspectives is beyond this paper, they are worth noting in order to gain deeper insight into the many remarkable properties of the Laplacian.

3.1 Brief Overview of MDPs

A discrete Markov decision process (MDP) $M = (S, A, P^a_{ss'}, R^a_{ss'})$ is defined by a finite set of discrete states S, a finite set of actions A, a transition model $P^a_{ss'}$ specifying the distribution over future states

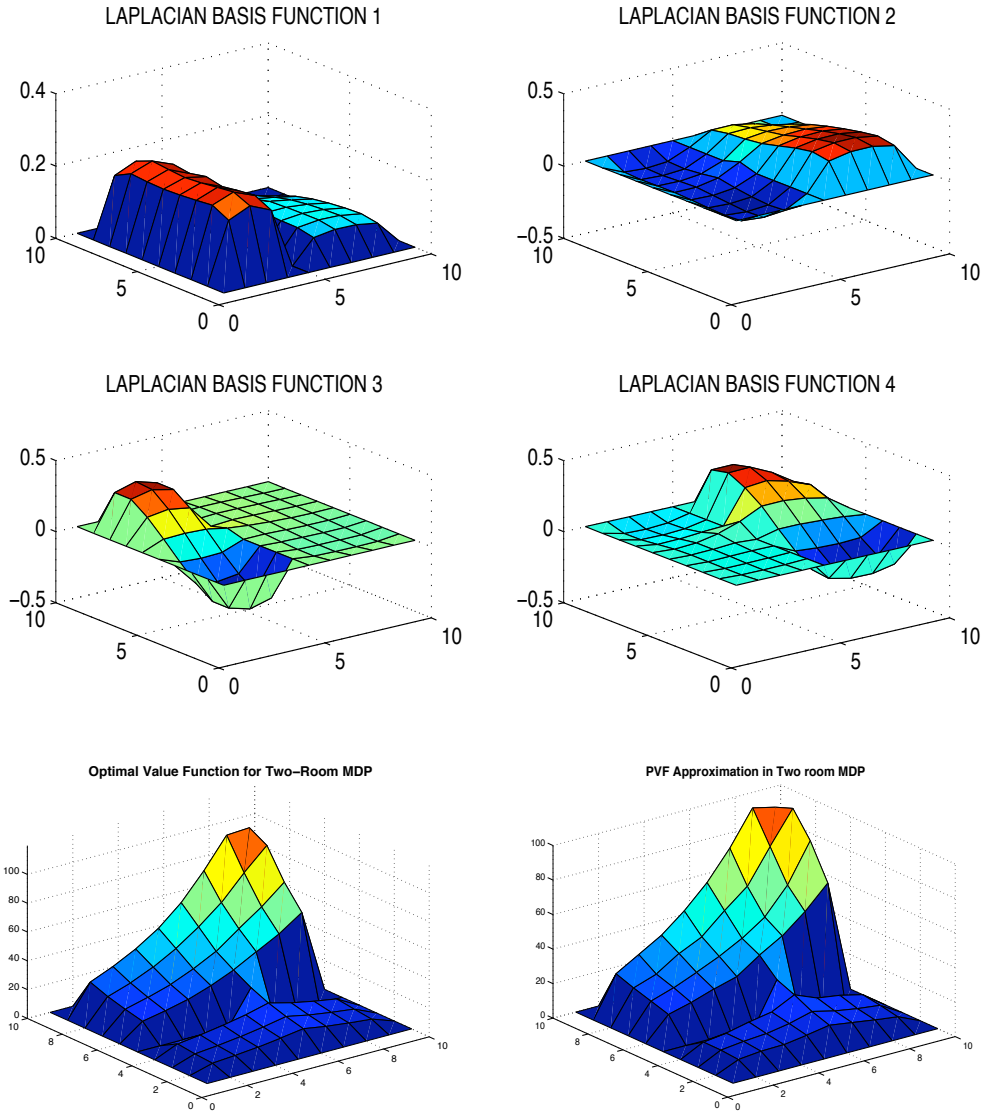


Figure 5: Top: proto-value functions formed from the four "smoothest" eigenvectors of the normalized graph Laplacian in a two-room MDP of 100 states. Bottom left: the optimal value function for a 2 room MDP, repeated from Figure 1 for comparative purposes. Bottom right: the approximation produced by the RPI algorithm using 20 proto-value functions, computed as the eigenvectors of the normalized graph Laplacian on the adjacency graph. The nonlinearity represented by the walls is clearly captured.

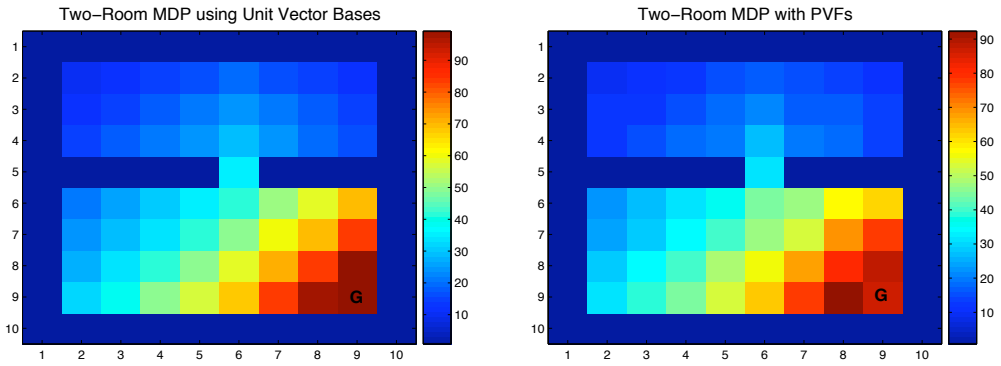


Figure 6: Left: the optimal value function for the two-room MDP using unit vector bases. Right: approximation with 20 PVFs using the RPI algorithm.

s' when an action a is performed in state s, and a corresponding reward model $R^a_{ss'}$ specifying a scalar cost or reward (Puterman, 1994). In continuous Markov decision processes, the set of states $\subseteq \mathbb{R}^d$. Abstractly, a value function is a mapping $S \rightarrow \mathbb{R}$ or equivalently (in discrete MDPs) a vector $\in \mathbb{R}^{|S|}$. Given a policy $\pi : S \rightarrow A$ mapping states to actions, its corresponding value function V^π specifies the expected long-term discounted sum of rewards received by the agent in any given state s when actions are chosen using the policy. Any optimal policy π^* defines the same unique optimal value function V^* which satisfies the nonlinear constraints

$$V^*(s) = T^*(V^*(s)) = \max_a \left(R_{sa} + \gamma \sum_{s' \in S} P^a_{ss'} V^*(s') \right),$$

where $R_{sa} = \sum_{s' \in S} P^a_{ss'} R^a_{ss'}$ is the expected immediate reward. Value functions are mappings from the state space to the expected long-term discounted sum of rewards received by following a fixed (deterministic or stochastic) policy π. Here, T^* can be viewed as an *operator* on value functions, and V^* represents the fixed point of the operator T^*. The value function V^π associated with following a (deterministic) policy π can be defined as

$$V^\pi(s) = T(V^\pi(s)) = R_{s\pi(s)} + \gamma \sum_{s' \in S} P^{\pi(s)}_{ss'} V^\pi(s').$$

Once again, T is an operator on value functions, whose fixed point is given by V^π. Value functions in an MDP can be viewed as the result of rewards " diffusing" through the state space, governed by the underlying system dynamics. Let P^π represent an $|S| \times |S|$ transition matrix of a (deterministic) policy $\pi : S \rightarrow A$ mapping each state $s \in S$ to a desired action $a = \pi(s)$. Let R^π be a (column) vector of size $|S|$ of rewards. The value function associated with policy π can be defined using the Neumann series:

$$V^\pi = (I - \gamma P^\pi)^{-1} R^\pi = \left(I + \gamma P^\pi + \gamma^2 (P^\pi)^2 + \ldots \right) R^\pi. \quad (1)$$

3.2 Approximation of Value Functions

It is obviously difficult to represent value functions exactly on large discrete state spaces, or in continuous spaces. Consequently, there has been much study of approximation architectures for representing value functions (Bertsekas and Tsitsiklis, 1996). Value functions generally exhibit two key properties: they are typically *smooth*, and they reflect the underlying state space geometry. A fundamental contribution of this paper is the use of an approximation architecture that exploits a new notion of smoothness, not in the traditional sense of Euclidean space, but smoothness on the state space graph. The notion of smooth functions on graphs can be formalized using the Sobolev norm (Mahadevan and Maggioni, 2006). In addition, value functions usually reflect the geometry of the environment (as illustrated in Figure 5). Smoothness derives from the fact that the value at a given state $V^\pi(s)$ is always a function of values at "neighboring" states. Consequently, it is natural to construct basis functions for approximating value functions that share these two properties.[8]

Let us define a set of *basis functions* $F_\Phi = \{\phi_1, \ldots, \phi_k\}$, where each basis function represents a "feature" $\phi_i : S \rightarrow \mathbb{R}$. The basis function matrix Φ is an $|S| \times k$ matrix, where each column is a particular basis function evaluated over the state space, and each row is the set of all possible basis functions evaluated on a particular state. Approximating a value function using the matrix Φ can be viewed as projecting the value function onto the column space spanned by the basis functions ϕ_i,

$$V^\pi \approx \hat{V}^\pi = \Phi w^\pi = \sum_i w_i^\pi \phi_i \ .$$

Mathematically speaking, this problem can be rigorously formulated using the framework of best approximation in inner product spaces (Deutsch, 2001). In fact, it is easy to show that the space of value functions represents a Hilbert space, or a complete inner product space (Van Roy, 1998). For simplicity, we focus on the simpler problem of approximating a fixed policy π, which defines a Markov chain where ρ^π represents its invariant (long-term) distribution. This distribution defines a Hilbert space, where the inner product is given by

$$\langle V_1, V_2 \rangle_\pi = \sum_{s \in S} V_1^\pi(s) V_2^\pi(s) \rho^\pi(s).$$

The "length" or norm in this inner product space is defined as $\|V\|_\pi = \sqrt{\langle V, V \rangle_\pi}$. Value function approximation can thus be formalized as a problem of best approximation in a Hilbert space (Deutsch, 2001). It is well known that if the basis functions ϕ_i are *orthonormal* (unit-length and mutually perpendicular), the best approximation of the value function V^π can be expressed by its projection onto the space spanned by the basis functions, or more formally

$$V^\pi \approx \sum_{i \in I} \langle V^\pi, \phi_i \rangle_\pi \ \phi_i,$$

where I is the set of indices that define the basis set. In finite MDPs, the best approximation can be characterized using the weighted least-squares projection matrix

$$M_\Phi^\pi = \Phi (\Phi^T D_{\rho^\pi} \Phi)^{-1} \Phi^T D_{\rho^\pi},$$

8. For low values of the discount factor γ, it is possible to construct highly non-smooth value functions, which decay rapidly and are not influenced by nonlinearities in state space geometry. In many problems of interest, however, the discount factor γ needs to be set close to 1 to learn a desirable policy.

where D_{ρ^π} is a diagonal matrix whose entries represent the distribution ρ^π. We know the Bellman "backup" operator T defined above has a fixed point $V^\pi = T(V^\pi)$. Many standard parameter estimation methods, including LSPI (Lagoudakis and Parr, 2003) and LSTD (Bradtke and Barto, 1996), can be viewed as finding an approximate fixed point of the operator T

$$\hat{V}^\pi = \Phi w^\pi = M_\phi^\pi \left(T(\Phi w^\pi)\right).$$

It can be shown that the operator T is a contraction mapping, where

$$\|TV_1 - TV_2\|_\pi \leq \gamma \|V_1 - V_2\|_\pi .$$

A natural question that arises is whether we can quantify the error in value function approximation under a set of basis functions F_ϕ. Exploiting the contraction property of the operator T under the norm defined by the weighted inner product, it can be shown that the "distance" between the true value function V^π and the fixed point $\hat{V}^\pi$ can be bounded in terms of the distance between V^π and its projection onto the space spanned by the basis functions (Van Roy, 1998):

$$\|V^\pi - \hat{V}^\pi\|_\pi \leq \frac{1}{1-\kappa^2} \|V^\pi - M_\phi^\pi V^\pi\|_\pi ,$$

where κ is the contraction rate defined by Bellman operator T in conjunction with the weighted least-squares projection.

The problem of value function approximation in control learning is significantly more difficult, in that it involves finding an approximate fixed point of the initially unknown operator T^*. One standard algorithm for control learning is *approximate policy iteration* (Bertsekas and Tsitsiklis, 1996), which interleaves an *approximate policy evaluation* step of finding an approximation of the value function $\hat{V}^{\pi_k}$ associated with a given policy π_k at stage k, with a *policy improvement* step of finding the greedy policy associated with $\hat{V}^{\pi_k}$. Here, there are two sources of error introduced by approximating the exact value function, and approximating the policy. We will describe a specific type of approximate policy iteration method—the LSPI algorithm (Lagoudakis and Parr, 2003)—in Section 4, which uses a least-squares approach to approximate the action-value function. An additional problem in control learning is that the standard theoretical results for approximate policy iteration are often expressed in terms of the maximum (normed) error, whereas approximation methods are most naturally formulated as projections in a least-squared normed space. There continues to be work on developing more useful weighted least-square bounds, although these currently assume the policy is exactly representable (Munos, 2003, 2005). Also, it is possible to design approximation methods that directly carry out max-norm projections using linear programming, although this work usually assumes the transition dynamics is known (Guestrin et al., 2001),

Our approach to the problem of control learning involves finding a suitable set of basis functions by diagonalizing a learned diffusion model from sample trajectories, and to use projections in the Hilbert space defined by the diffusion model for policy evaluation and improvement. We first introduce the *Fourier* approach of finding basis functions by diagonalization, and then describe how diffusion models are used as a substitute for transition models. In Section 9, we will return to discuss other approaches (Petrik, 2007; Parr et al., 2007), where the Bellman operator T is used more directly in finding basis functions.

3.3 Spectral Analysis of Transition Matrices

In this paper, the orthogonal basis functions are constructed in the Fourier tradition by diagonalizing an operator (or matrix) and finding its *eigenvectors*. We motivate this approach by first assuming that the eigenvectors are constructed directly from a (known) state transition matrix P^π and show that if the reward function R^π is known, the eigenvectors can be selected nonlinearly based on expanding the value function V^π on the eigenvectors of the transition matrix. Petrik (2007) develops this line of reasoning, assuming that P^π and R^π are both known, and that P^π is diagonalizable. We describe this perspective in more detail below as it provides a useful motivation for why we use diagonalizable diffusion matrices instead. One subclass of diagonalizable transition matrices are those corresponding to *reversible* Markov chains (which will turn out to be useful below). Although transition matrices for general MDPs are *not* reversible, and their spectral analysis is more delicate, it will still be a useful starting point to understand diffusion matrices such as the graph Laplacian.[9] If the transition matrix P^π is diagonalizable, there is a complete set of eigenvectors $\Phi^\pi = (\phi_1^\pi, \ldots \phi_n^\pi)$ that provides a change of basis in which the transition matrix P^π is representable as a diagonal matrix. For the sub-class of diagonalizable transition matrices represented by reversible Markov chains, the transition matrix is not only diagonalizable, but there is also an orthonormal basis. In other words, using a standard result from linear algebra, we have

$$P^\pi = \Phi^\pi \Lambda^\pi (\Phi^\pi)^T,$$

where Λ^π is a diagonal matrix of *eigenvalues*. Another way to express the above property is to write the transition matrix as a sum of *projection matrices* associated with each eigenvalue:

$$P^\pi = \sum_{i=1}^{n} \lambda_i^\pi \phi_i^\pi (\phi_i^\pi)^T,$$

where the eigenvectors ϕ_i^π form a complete orthogonal basis (i.e., $\| \phi_i^\pi \|_2 = 1$ and $\langle \phi_i^\pi, \phi_j^\pi \rangle = 0, i \neq j$). It readily follows that powers of P^π have the same eigenvectors, but the eigenvalues are raised to the corresponding power (i.e., $(P^\pi)^k \phi_i^\pi = (\lambda_i^\pi)^k \phi_i^\pi$). Since the basis matrix Φ spans all vectors on the state space S, we can express the reward vector R^π in terms of this basis as

$$R^\pi = \Phi^\pi \alpha^\pi, \tag{2}$$

where α^π is a vector of scalar weights. For high powers of the transition matrix, the projection matrices corresponding to the largest eigenvalues will dominate the expansion. Combining Equation 2

9. In Section 9, we discuss extensions to more general non-reversible MDPs.

with the Neumann expansion in Equation 1, we get

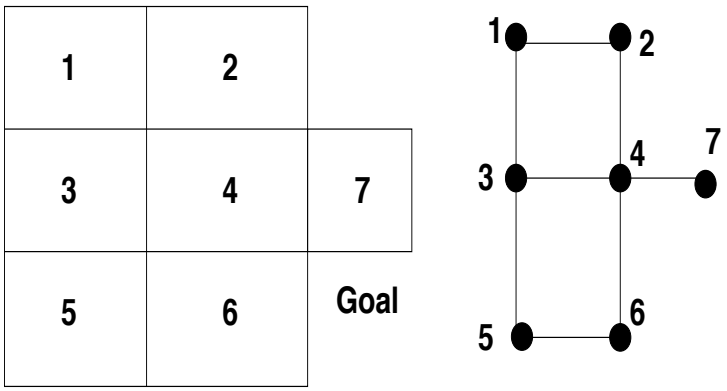
$$\begin{aligned} V^\pi &= \sum_{i=0}^{\infty} (\gamma P^\pi)^i \Phi^\pi \alpha^\pi \\ &= \sum_{i=0}^{\infty} \sum_{k=1}^{n} \gamma^i (P^\pi)^i \phi_k^\pi \alpha_k^\pi \\ &= \sum_{k=1}^{n} \sum_{i=0}^{\infty} \gamma^i (\lambda_k^\pi)^i \phi_k^\pi \alpha_k^\pi \\ &= \sum_{k=1}^{n} \frac{1}{1-\gamma\lambda_k^\pi} \phi_k^\pi \alpha_k^\pi \\ &= \sum_{k=1}^{n} \beta_k \phi_k^\pi, \end{aligned}$$

where we used the property that $(P^\pi)^i \phi_j^\pi = (\lambda_j^\pi)^i \phi_j^\pi$. Essentially, the value function V^π is represented as a linear combination of eigenvectors of the transition matrix. In order to provide the most efficient approximation, we can truncate the summation by choosing some small number $m < n$ of the eigenvectors, preferably those for whom β_k is large. Of course, since the reward function is not known, it might be difficult to pick a priori those eigenvectors that result in the largest coefficients. A simpler strategy instead is to focus on those eigenvectors for whom the coefficients $\frac{1}{1-\gamma\lambda_k^\pi}$ are the largest. In other words, we should pick the eigenvectors corresponding to the *largest* eigenvalues of the transition matrix P^π (since the spectral radius is 1, the eigenvalues closest to 1 will dominate the smaller ones):

$$V^\pi \approx \sum_{k=1}^{m} \frac{1}{1-\gamma\lambda_k^\pi} \phi_k^\pi \alpha_k^\pi, \tag{3}$$

where we assume the eigenvalues are ordered in non-increasing order, so λ_1^π is the largest eigenvalue. If the transition matrix P^π and reward function R^π are both known, one can of course construct basis functions by diagonalizing P^π and choosing eigenvectors "out-of-order" (that is, pick eigenvectors with the largest β_k coefficients above). Petrik (2007) shows a (somewhat pathological) example where a linear spectral approach specified by Equation 3 does poorly when the reward vector is chosen such that it is orthogonal to the first k basis functions. It is an empirical question whether such pathological reward functions exhibit themselves in more natural situations. The repertoire of discrete and continuous MDPs we study seem highly amenable to the linear spectral decomposition approach. However, we discuss various approaches for augmenting PVFs with reward-sensitive bases in Section 9.

The spectral approach of diagonalizing the transition matrix is problematic for several reasons. One, the transition matrix P^π cannot be assumed to be symmetric, in which case one has to deal with complex eigenvalues (and eigenvectors). Second, we cannot assume that the transition matrix is known. Of course, one can always use samples of the underlying MDP generated by exploration to estimate the transition matrix, but the number of samples needed may be large. Finally, in control learning, the policy keeps changing, causing one to have to reestimate the transition matrix. What one would ideally like to have is a *surrogate* model that is easier to estimate than a full transition matrix, is always diagonalizable, and results in smooth basis functions that capture large scale geometry. *Diffusion models* serve to fulfill this role, as discuss next.



$$P_r = \begin{bmatrix} 0 & 0.5 & 0.5 & 0 & 0 & 0 & 0 \\ 0.5 & 0 & 0 & 0.5 & 0 & 0 & 0 \\ 0.33 & 0 & 0 & 0.33 & 0.33 & 0 & 0 \\ & & & \cdots & & & \end{bmatrix}$$

Figure 7: Top: A simple diffusion model given by an undirected unweighted graph connecting each state to neighbors that are reachable using a single (reversible) action. Bottom: first three rows of the random walk matrix $P_r = D^{-1}W$. P_r is not symmetric, but it has real eigenvalues and eigenvectors since it is spectrally related to the normalized graph Laplacian.

3.4 From Transition Matrices to Diffusion Models

We now develop a line of analysis where a graph is induced from the state space of an MDP, by sampling from a policy such as a random walk. Let us define a weighted graph $G = (V, E, W)$, where V is a finite set of vertices, and W is a weighted adjacency matrix with $W(i, j) > 0$ if $(i, j) \in E$, that is it is possible to reach state i from j (or vice-versa) in a single step. A simple example of a diffusion model on G is the random walk matrix $P_r = D^{-1}W$. Figure 7 illustrates a random walk diffusion model. Note the random walk matrix $P_r = D^{-1}W$ is not symmetric. However, it can be easily shown that P_r defines a *reversible* Markov chain, which induces a Hilbert space with respect to the inner product defined by the invariant distribution ρ:

$$\langle f, g \rangle_\rho = \sum_{v \in V} f(i) g(i) \rho(i).$$

In addition, the matrix P_r can be shown to be self-adjoint (symmetric) with respect to the above inner product, that is

$$\langle P_r f, g \rangle_\rho = \langle f, P_r g \rangle_\rho.$$

Consequently, the matrix P_r can be shown to have real eigenvalues and orthonormal eigenvectors, with respect to the above inner product.

The random walk matrix $P_r = D^{-1}W$ is called a *diffusion model* because given any function f on the underlying graph G, the powers of $P_r^t f$ determine how quickly the random walk will "mix" and converge to the long term distribution (Chung, 1997). It can be shown that the stationary distribution

of a random walk on an undirected graph is given by $\rho(v) = \frac{d_v}{vol(G)}$, where d_v is the degree of vertex v and the "volume" $vol(G) = \sum_{v \in G} d_v$. Even though the random walk matrix P_r can be diagonalized, for computational reasons, it turns out to be highly beneficial to find a symmetric matrix with a closely related spectral structure. This is essentially the graph Laplacian matrix, which we now describe in more detail.

3.5 The Graph Laplacian

For simplicity, assume the underlying state space is represented as an undirected graph $G = (V, E, W)$, where V is the set of vertices, E is the set of edges where $(u, v) \in E$ denotes an undirected edge from vertex u to vertex v. The more general case of directed graphs is discussed in Section 9.3. The *combinatorial Laplacian* L is defined as the operator $L = D - W$, where D is a diagonal matrix called the *valency* matrix whose entries are row sums of the weight matrix W. The first three rows of the combinatorial Laplacian matrix for the grid world MDP in Figure 7 is illustrated below, where we assume a unit weight on each edge:

$$L = \begin{bmatrix} 2 & -1 & -1 & 0 & 0 & 0 & 0 \\ -1 & 2 & 0 & -1 & 0 & 0 & 0 \\ -1 & 0 & 3 & -1 & -1 & 0 & 0 \\ & & & \cdots & & & \end{bmatrix}.$$

Comparing the above matrix with the random walk matrix in Figure 7, it may seem like the two matrices have little in common. Surprisingly, we will show that there is indeed an intimate connection between the random walk matrix and the Laplacian. The Laplacian has many attractive spectral properties. It is both symmetric as well as positive semi-definite, and hence its eigenvalues are not only all real, but also non-negative. It is useful to view the Laplacian as an *operator* on the space of functions $\mathcal{F} : V \to \mathbb{R}$ on a graph. In particular, it can be easily shown that

$$Lf(i) = \sum_{j \sim i} (f(i) - f(j)),$$

that is the Laplacian acts as a *difference* operator. On a two-dimensional grid, the Laplacian can be shown to essentially be a discretization of the continuous Laplace operator

$$\frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2},$$

where the partial derivatives are replaced by finite differences.

Another fundamental property of the graph Laplacian is that projections of functions on the eigenspace of the Laplacian produce the smoothest global approximation respecting the underlying graph topology. More precisely, let us define the inner product of two functions f and g on a graph as $\langle f, g \rangle = \sum_u f(u)g(u)$.[10] Then, it is easy to show that

$$\langle f, Lf \rangle = \sum_{u \sim v} w_{uv}(f(u) - f(v))^2,$$

10. For simplicity, here we consider the unweighted inner product ignoring the invariant distribution ρ induced by a random walk.

where this so-called *Dirichlet sum* is over the (undirected) edges $u \sim v$ of the graph G, and w_{uv} denotes the weight on the edge. Note that each edge is counted only once in the sum. From the standpoint of regularization, this property is crucial since it implies that rather than smoothing using properties of the ambient Euclidean space, smoothing takes the underlying manifold (graph) into account.

To make the connection between the random walk operator P_r introduced in the previous section, and the Laplacian, we need to introduce the *normalized Laplacian* (Chung, 1997), which is defined as

$$\mathcal{L} = D^{-\frac{1}{2}} L D^{-\frac{1}{2}}.$$

To see the connection between the normalized Laplacian and the random walk matrix $P_r = D^{-1}W$, note the following identities:

$$\begin{aligned} \mathcal{L} &= D^{-\frac{1}{2}} L D^{-\frac{1}{2}} = I - D^{-\frac{1}{2}} W D^{-\frac{1}{2}}, \\ I - \mathcal{L} &= D^{-\frac{1}{2}} W D^{-\frac{1}{2}}, \\ D^{-\frac{1}{2}}(I - \mathcal{L}) D^{\frac{1}{2}} &= D^{-1} W. \end{aligned}$$

Hence, the random walk operator $D^{-1}W$ is similar to $I - \mathcal{L}$, so both have the same eigenvalues, and the eigenvectors of the random walk operator are the eigenvectors of $I - \mathcal{L}$ point-wise multiplied by $D^{-\frac{1}{2}}$. We can now provide a rationale for choosing the eigenvectors of the Laplacian as basis functions. In particular, if λ_i is an eigenvalue of the random walk transition matrix P_r, then $1 - \lambda_i$ is the corresponding eigenvalue of $\mathcal{L}$. Consequently, in the expansion given by Equation 3, we would select the eigenvectors of the normalized graph Laplacian corresponding to the *smallest* eigenvalues.

The normalized Laplacian $\mathcal{L}$ also acts as a *difference* operator on a function f on a graph, that is

$$\mathcal{L} f(u) = \frac{1}{\sqrt{d_u}} \sum_{v \sim u} \left(\frac{f(u)}{\sqrt{d_u}} - \frac{f(v)}{\sqrt{d_v}} \right) w_{uv}.$$

The difference between the combinatorial and normalized Laplacian is that the latter models the degree of a vertex as a local measure. In Section 7, we provide an experimental evaluation of the different graph operators for solving continuous MDPs.

Building on the Dirichlet sum above, a standard *variational* characterization of eigenvalues and eigenvectors views them as the solution to a sequence of minimization problems. In particular, the set of eigenvalues can be defined as the solution to a series of minimization problems using the Rayleigh quotient (Chung, 1997). This provides a variational characterization of eigenvalues using projections of an arbitrary function $g : V \to \mathcal{R}$ onto the subspace $\mathcal{L}g$. The quotient gives the eigenvalues and the functions satisfying orthonormality are the eigenfunctions:

$$\frac{\langle g, \mathcal{L} g \rangle}{\langle g, g \rangle} = \frac{\langle g, D^{-\frac{1}{2}} L D^{-\frac{1}{2}} g \rangle}{\langle g, g \rangle} = \frac{\sum_{u \sim v} (f(u) - f(v))^2 w_{uv}}{\sum_u f^2(u) d_u},$$

where $f \equiv D^{-\frac{1}{2}} g$. The first eigenvalue is $\lambda_0 = 0$, and is associated with the constant function $f(u) = \mathbf{1}$, which means the first eigenfunction $g_o(u) = \sqrt{D}\,\mathbf{1}$ (for an example of this eigenfunction, see top left plot in Figure 5). The first eigenfunction (associated with eigenvalue 0) of the combinatorial Laplacian is the constant function $\mathbf{1}$. The second eigenfunction is the infimum over all functions

$g : V \rightarrow \mathcal{R}$ that are perpendicular to $g_o(u)$, which gives us a formula to compute the first non-zero eigenvalue λ_1, namely

$$\lambda_1 = \inf_{f \perp \sqrt{D}\mathbf{1}} \frac{\sum_{u \sim v} (f(u) - f(v))^2 w_{uv}}{\sum_u f^2(u) d_u}.$$

The Rayleigh quotient for higher-order basis functions is similar: each function is perpendicular to the subspace spanned by previous functions (see top four plots in Figure 5). In other words, the eigenvectors of the graph Laplacian provide a systematic organization of the space of functions on a graph that respects its topology.

3.6 Proto-Value Functions and Large-Scale Geometry

We now formalize the intuitive notion of why PVFs capture the large-scale *geometry* of a task environment, such as its symmetries and bottlenecks. A full discussion of this topic is beyond the scope of this paper, and we restrict our discussion here to one interesting property connected to the automorphisms of a graph. Given a graph $G = (V, E, W)$, an *automorphism* π of a graph is a bijection $\pi : V \rightarrow V$ that leaves the weight matrix invariant. In other words, $w(u,v) = w(\pi(u), \pi(v))$. An automorphism π can be also represented in matrix form by a permutation matrix Γ that commutes with the weight matrix:

$$\Gamma W = W \Gamma.$$

An immediate consequence of this property is that automorphisms leave the valency, or degree of a vertex, invariant, and consequently, the Laplacian is invariant under an automorphism. The set of all automorphisms forms a non-Abelian group, in that the group operation is non-commutative. Let x be an eigenvector of the combinatorial graph Laplacian L. Then, it is easy to show that Γx must be an eigenvector as well for any automorphism Γ. This result follows because

$$L\Gamma x = \Gamma L x = \Gamma \lambda x = \lambda \Gamma x.$$

A detailed graph-theoretic treatment of the connection between symmetries of a graph and its spectral properties are provided in books on algebraic and spectral graph theory (Chung, 1997; Cvetkovic et al., 1980, 1997). For example, it can be shown that if the permuted eigenvector Γx is independent of the original eigenvector x, then the corresponding eigenvalue λ has geometric multiplicity $m > 1$. More generally, it is possible to exploit the theory of linear representations of groups to construct compact basis functions on symmetric graphs, which have found applications in the study of complex molecules such as "buckyballs" (Chung and Sternberg, 1992). It is worth pointing out that these ideas extend to continuous manifolds as well. The use of the Laplacian in constructing representations that are *invariant* to group operations such as translation is a hallmark of work in harmonic analysis (Gurarie, 1992).

Furthermore, considerable work in spectral graph theory as well as its applications in AI uses the properties of the *Fiedler* eigenvector (the eigenvector associated with the smallest non-zero eigenvalue), such as its sensitivity to bottlenecks, in order to find clusters in data or segment images (Shi and Malik, 2000; Ng et al., 2002). To formally explain this, we briefly review spectral geometry. The *Cheeger* constant h_G of a graph G is defined as

$$h_G(S) = \min_S \frac{|E(S,\tilde{S})|}{\min(\text{vol } S, \text{vol } \tilde{S})}.$$

Here, S is a subset of vertices, $\tilde{S}$ is the complement of S, and $E(S,\tilde{S})$ denotes the set of all edges (u,v) such that $u \in S$ and $v \in \tilde{S}$. The volume of a subset S is defined as vol $S = \sum_{x \in S} d_X$. Consider the problem of finding a subset S of states such that the edge boundary ∂S contains as few edges as possible, where

$$\partial S = \{(u,v) \in E(G) : u \in S \text{ and } v \notin S\}.$$

The relation between ∂S and the Cheeger constant is given by

$$|\partial S| \geq h_G \text{ vol } S.$$

In the two-room grid world task illustrated in Figure 1, the Cheeger constant is minimized by setting S to be the states in the first room, since this will minimize the numerator $E(S,\tilde{S})$ and maximize the denominator $min(\text{vol } S, \text{vol } \tilde{S})$. A remarkable inequality connects the Cheeger constant with the spectrum of the graph Laplacian operator. This theorem underlies the reason why the eigenfunctions associated with the second eigenvalue λ_1 of the graph Laplacian captures the geometric structure of environments, as illustrated in Figure 5.

Theorem 1 *(Chung, 1997): Define λ_1 to be the first (non-zero) eigenvalue of the normalized graph Laplacian operator $\mathcal{L}$ on a graph G. Let h_G denote the Cheeger constant of G. Then, we have $2h_G \geq \lambda_1 > \frac{h_G^2}{2}$.*

In the context of MDPs, our work explores the construction of representations that similarly exploit large-scale geometric features, such as symmetries and bottlenecks. In other words, we are evaluating the hypothesis that such representations are useful in solving MDPs, given that topology-sensitive representations have proven to be useful across a wide variety of problems both in machine learning specifically as well as in science and engineering more generally.

4. Representation Policy Iteration

In this section, we begin the detailed algorithmic analysis of the application of proto-value functions to solve Markov decision processes. We will describe a specific instantiation of the RPI framework described previously, which comprises of an outer loop for learning basis functions and an inner loop for estimating the optimal policy representable within the given set of basis functions. In particular, we will use least-square policy iteration (LSPI) as the parameter estimation method. We will analyze three variants of RPI, beginning with the most basic version in this section, and then describing two extensions of RPI to continuous and factored state spaces in Section 5 and Section 6.

4.1 Least-Squares Approximation of Action Value Functions

The basics of Markov decision processes as well as value function approximation was briefly reviewed in Section 3. Here, we focus on action-value function approximation, and in particular, describe the LSPI method (Lagoudakis and Parr, 2003). In action-value learning, the goal is to approximate the true action-value function $Q^\pi(s,a)$ for a policy π using a set of basis functions

$\phi(s,a)$ that can be viewed as doing dimensionality reduction on the space of functions. The true action value function $Q^\pi(s,a)$ is a vector in a high dimensional space $\mathbb{R}^{|S|\times|A|}$, and using the basis functions amounts to reducing the dimension to $\mathbb{R}^k$ where $k \ll |S| \times |A|$. The approximated action value is thus

$$\hat{Q}^\pi(s,a;w) = \sum_{j=1}^{k} \phi_j(s,a) w_j,$$

where the w_j are weights or parameters that can be determined using a least-squares method. Let Q^π be a real (column) vector $\in \mathbb{R}^{|S|\times|A|}$. $\phi(s,a)$ is a real vector of size k where each entry corresponds to the basis function $\phi_j(s,a)$ evaluated at the state action pair (s,a). The approximate action-value function can be written as $\hat{Q}^\pi = \Phi w^\pi$, where w^π is a real column vector of length k and Φ is a real matrix with $|S| \times |A|$ rows and k columns. Each row of Φ specifies all the basis functions for a particular state action pair (s,a), and each column represents the value of a particular basis function over all state action pairs. The least-squares fixed-point approximation tries to find a set of weights w^π under which the projection of the backed up approximated Q-function $T_\pi \hat{Q}^\pi$ onto the space spanned by the columns of Φ is a fixed point, namely

$$\hat{Q}^\pi = \Phi(\Phi^T\Phi)^{-1}\Phi^T(T_\pi \hat{Q}^\pi),$$

where T_π is the Bellman backup operator. It can be shown (Lagoudakis and Parr, 2003) that the resulting solution can be written in a *weighted* least-squares form as $Aw^\pi = b$, where the A matrix is given by

$$A = \left(\Phi^T D_\rho^\pi (\Phi - \gamma P^\pi \Phi)\right),$$

and the b column vector is given by

$$b = \Phi^T D_\rho^\pi R,$$

where D_ρ^π is a diagonal matrix whose entries reflect varying "costs" for making approximation errors on (s,a) pairs as a result of the nonuniform distribution $\rho^\pi(s,a)$ of visitation frequencies. A and b can be estimated from a database of transitions collected from some source, for example, a random walk. The A matrix and b vector can be estimated as the sum of many rank-one matrix summations from a database of stored samples.

$$\begin{aligned} \tilde{A}^{t+1} &= \tilde{A}^t + \phi(s_t,a_t)\left(\phi(s_t,a_t) - \gamma\phi(s_t',\pi(s_t'))\right)^T, \\ \tilde{b}^{t+1} &= \tilde{b}^t + \phi(s_t,a_t) r_t, \end{aligned}$$

where (s_t,a_t,r_t,s_t') is the t^{th} sample of experience from a trajectory generated by the agent (using some random or guided policy). Once the matrix A and vector b have been constructed, the system of equations $Aw^\pi = b$ can be solved for the weight vector w^π either by taking the inverse of A (if it is of full rank) or by taking its pseudo-inverse (if A is rank-deficient). This defines a specific policy since $\hat{Q}^\pi = \Phi w^\pi$. The process is then repeated, until convergence (which can be defined as when the $\mathbb{L}^2$- normed difference between two successive weight vectors falls below a predefined threshold ε). Note that in succeeding iterations, the A matrix will be different since the policy π has changed. Figure 8 describes a specific instantiation of RPI, using LSPI as the control learning method.

`RPI` $(\pi_m, T, N, \varepsilon, k, O, \mu, \mathcal{D})$:

// π_m: Policy at the beginning of trial m
// T: Number of initial random walk trials
// N: Maximum length of each trial
// ε : Convergence condition for policy iteration
// k: Number of proto-value basis functions to use
// O: Type of graph operator used
// μ: Parameter for basis adaptation

Sample Collection Phase

1. See Figure 4 on page 2177.

Representation Learning Phase

2. See Figure 4 on page 2177.

Control Learning Phase (LSPI)

3. Initialize $w^0 \in \mathbb{R}^k$ to a random vector.
4. **Repeat** the following steps:
   (a) Set $i \leftarrow i+1$. Using the stored transitions $(s_t, a_t, s'_t, a'_t, r_t) \in \mathcal{D}$, compute the matrix A and vector b as follows:

$$\begin{aligned} \tilde{A}^{t+1} &= \tilde{A}^t + \phi(s_t, a_t)\left(\phi(s_t, a_t) - \gamma\phi(s'_t, \pi(s_t))\right)^T. \\ \tilde{b}^{t+1} &= \tilde{b}^t + \phi(s_t, a_t) r_t. \end{aligned}$$

   (b) Solve the linear system of equations $\tilde{A}w^i = \tilde{b}$ using any standard method.[a]
   (c) **Optional basis adaptation step:** Prune the basis matrix Φ by discarding basis functions (columns) whose coefficients are smaller than μ.
   (d) **until** $\|w^i - w^{i+1}\|^2 \leq \varepsilon$.
5. Set $\pi_{m+1}(s) = \text{argmax}_{a \in A} \hat{Q}^i(s,a)$ where $\hat{Q}^i = \Phi w^i$ is the ε-optimal approximation to the optimal value function within the linear span of basis functions Φ.
6. **Optional:** Repeat the above procedure by calling `RPI` $(\pi_{m+1}, T, N, \varepsilon, k, O, \mu, \mathcal{D})$.

a. If A is of full rank, it can be inverted, otherwise if it is rank-deficient, the pseudo-inverse of A can be used. It is possible to avoid matrix inversion entirely by using the incremental *Sherman-Woodbury-Morrison* method (Lagoudakis and Parr, 2003).

Figure 8: Pseudo-code of the representation policy iteration (RPI) using the least-squares policy iteration (LSPI) fix-point method as the control learning component.

4.2 Evaluating RPI on Simple Discrete MDPs

In this section, we evaluate the effectiveness of PVFs using small discrete MDPs such as the two-room discrete MDP used above, before proceeding to investigate how to scale the framework to larger discrete and continuous MDPs.[11] PVFs are evaluated along a number of dimensions, including the number of bases used, and its relative performance compared to parametric bases such as polynomials and radial basis functions. In subsequent sections, we will probe the scalability of PVFs on larger more challenging MDPs.

Two-room MDP: The two-room discrete MDP used here is a 100 state MDP, where the agent is rewarded for reaching the top left-hand corner state in Room 2. As before, 57 states are reachable, and the remaining states are exterior or interior wall states. The state space of this MDP was shown earlier in Figure 1. Room 1 and Room 2 are both rectangular grids connected by a single door. There are four (compass direction) actions, each succeeding with probability 0.9, otherwise leaving the agent in the same state. The agent is rewarded by 100 for reaching the goal state (state 89), upon which the agent is randomly reset back to some starting (accessible) state.

Number of Basis Functions: Figure 9 evaluates the learned policy by measuring the number of steps to reach the goal, as a function of the number of training episodes, and as the number of basis functions is varied (ranging from 10 to 35 for each of the four actions). The results are averaged over 10 independent runs, where each run consisted of a set of training episodes of a maximum length of 100 steps, where each episode was terminated if the agent reached the absorbing goal state. Around 20 basis functions (per action) were sufficient to get close to optimal behavior, and increasing the number of bases to 35 produced a marginal improvement. The variance across runs is fairly small for 20 and 35 bases, but relatively large for smaller numbers of bases (not shown for clarity). Figure 9 also compares the performance of PVFs with unit vector bases (table lookup), showing that PVFs with 25 bases closely tracks the performance of unit vector bases on this task. Note that we are measuring performance in terms of the number of steps to reach the goal, averaged over a set of 10 runs. Other metrics could be plotted as well, such as the total discounted reward received, which may be more natural. However, our point is simply to show that there are significant differences in the quality of the policy learned by PVFs with that learned by the other parametric approximators, and these differences are of such an order that they will clearly manifest themselves regardless of the metric used.

Comparison with Parametric Bases: One important consideration in evaluating PVFs is how they compare with standard parametric bases, such as radial basis functions and polynomials. As Figure 1 suggests, parametric bases as conventionally formulated may have difficulty representing highly nonlinear value functions in MDPs such as the two room task. Here, we test whether this poor performance can be ameliorated by varying the number of basis functions used. Figure 9 evaluates the effectiveness of polynomial bases and radial basis functions in the two room MDP. In polynomial bases, a state i is mapped to the vector $\phi(i) = (1, i, i^2, \ldots i^{k-1})$ for k basis functions—this architecture was studied by (Koller and Parr, 2000; Lagoudakis and Parr, 2003).[12] In RBFs, a state i

11. In Section 9, we describe more sophisticated diffusion models for grid-world tasks in the richer setting of semi-Markov decision processes (SMDPs), using directed state-action graphs with temporally extended actions, such as "exiting a room", modeled with distal edges (Osentoski and Mahadevan, 2007).

12. The performance of polynomial bases gets worse for higher degrees, partly due to the numerical instability caused by taking large powers of state indices.

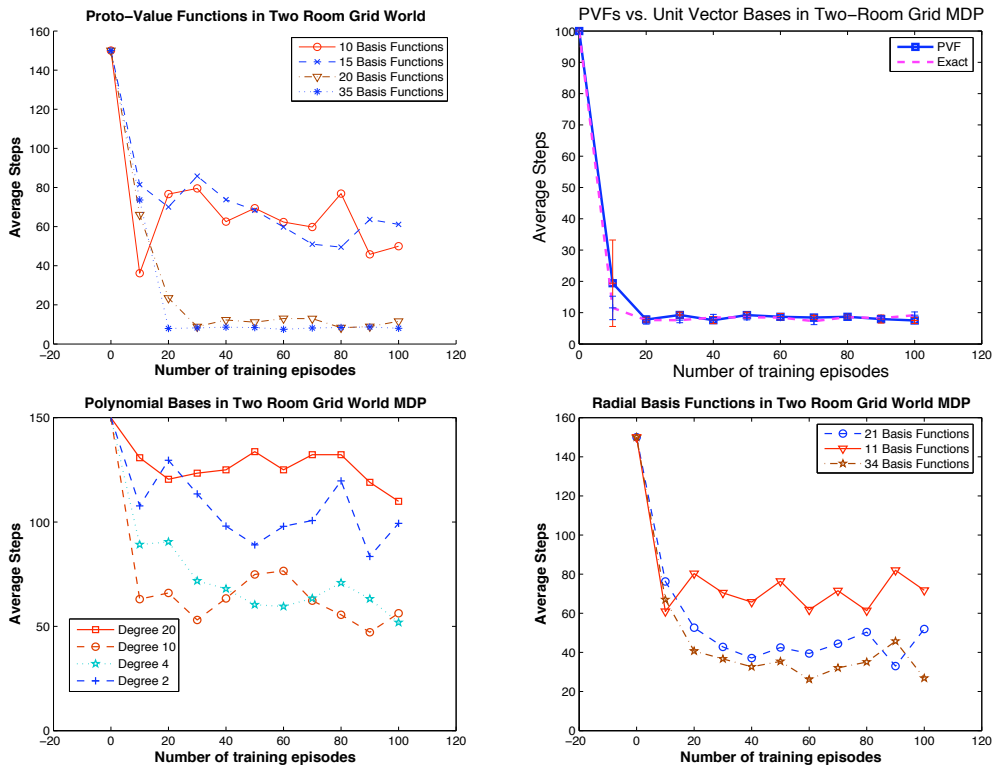


Figure 9: This experiment contrasts the performance of Laplacian PVFs (top left) with unit vector bases (top right), handcoded polynomial basis functions (bottom left) and radial basis functions (bottom right) on a 100 state two-room discrete MDP. Results are averaged over 10 runs. The performance of PVFs (with 25 bases) closely matches that of unit vector bases, and is considerably better than both polynomials and RBFs on this task.

is mapped to $\phi_j(i) = \exp^{-\frac{(i-j)^2}{2\sigma^2}}$, where j is the center of the RBF basis function. In the experiments shown, the basis centers were placed equidistantly from each other along the 100 states. The results show that both parametric bases under these conditions performed worse than PVFs in this task. [13]

Additional Results: Figure 10 shows an experiment on a larger 15×15 two-room MDP, with the same dynamics and goal structure as the smaller 10×10 two-room MDP. In this environment, there were a total of 225 states, with 157 of these being accessible interior states, and the remaining 68 representing "wall" states. Results are shown only for PVFs in this domain. The plotted result is averaged over 10 independent learning runs. As the number of PVFs is increased, the variance reduces and the performance significantly improves.

Figure 11 shows an additional experiment on a four-room MDP, where the agent is tasked to reach the state marked G. Results are shown only for PVFs in this domain. The plotted result is

13. Our results do not contradict any theoretical findings regarding the generality of RBFs or systems of orthogonal polynomials, since such results generally pertain to their asymptotic performance. Our evaluation of polynomials and RBFs gauges their performance on particular parameter settings.

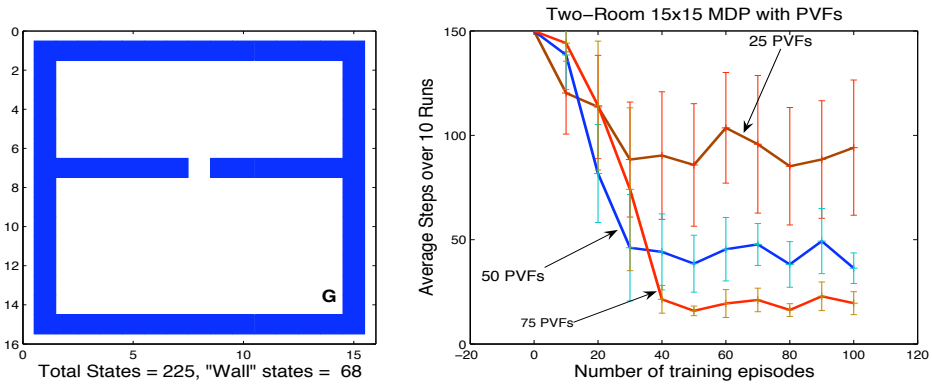


Figure 10: This figure shows results on a larger 15×15 two-room grid world MDP of 225 total states. The dynamics are identical to the two-room MDP. The results shown are using $25 - 75$ PVFs.

averaged over 10 independent learning runs. Here, the agent was trained on sample random walk trajectories that terminated in goal state G.

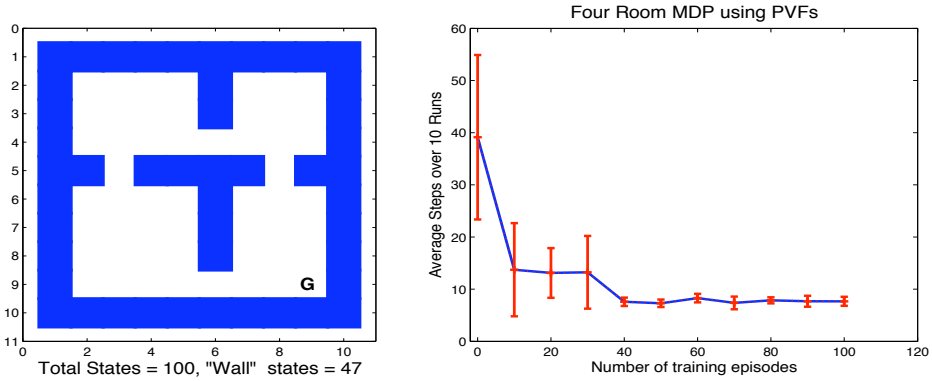


Figure 11: This figure shows results on a four-room grid world MDP of 100 total states. The dynamics are identical to the two-room MDP. The results shown are using 25 PVFs.

5. Scaling Proto-Value Functions: Product Spaces

Thus far, we have restricted our discussion of proto-value functions to small discrete MDPs. In this and the next section, we explore the issue of scaling the Laplacian framework to larger discrete and continuous domains. Computing and storing proto-value functions in large continuous or discrete domains can be intractable: spectral analysis of the state space graph or diagonalization of the policy transition matrix can be an infeasible eigenvector computation in large domains, even if the matrices are inherently sparse. To address this scaling issue, we explore a number of approaches,

from exploiting the large-scale regular structure of product spaces described in this section, to the use of sparsification through sampling for continuous states described in the next section.

In this section, we describe a general framework for scaling proto-value functions to large *factored* discrete spaces using properties of product spaces, such as grids, cylinders, and tori. A crucial property of the graph Laplacian is that its embeddings are highly regular for structured graphs (see Figure 13). We will explain the reason for this property below, and how to exploit it to construct compact encodings of Laplacian bases. We should also distinguish the approach described in this section, which relies on an *exact* Kronecker decomposition of the Laplacian eigenspace in product spaces, with the *approximate* Kronecker decomposition for arbitrary MDPs described in Section 9. The approach described here is applicable only to MDPs where the state space can be represented as the Kronecker sum of simpler state spaces (this notion will be defined more precisely below, but it covers many standard MDPs like grids). More generally, the weight matrices for arbitrary MDPs can also be factorized, although using the Kronecker *product*, where, however, the factorization is an approximation (Van Loan and Pitsianis, 1993).

5.1 Product Spaces: Complex Graphs from Simple Ones

Building on the theory of graph spectra (Cvetkovic et al., 1980), we now describe a hierarchical framework for efficiently computing and compactly storing proto-value functions. Many RL domains lead to *factored* representations where the state space is generated as the Cartesian product of the values of state variables (Boutilier et al., 1999). Consider a hypercube Markov decision process with d dimensions, where each dimension can take on k values. The size of the resulting state space is $O(k^d)$, and the size of each proto-value function is $O(k^d)$. Using the hierarchical framework presented below, the hypercube can be viewed as the *Kronecker sum* of d path or chain graphs, each of whose transition matrix is of size (in the worst case) $O(k^2)$. Now, each factored proto-value function can be stored in space $O(dk^2)$, and the cost of spectral analysis greatly reduces as well. Even greater savings can be accrued since usually only a small number of basis functions are needed relative to the size of a state space. We present detailed experimental results on a large factored multiagent domain of $> 10^6$ states, where proto-value functions are constructed from diagonalizing Laplacian matrices of size only 100×100, a huge computational savings! Figure 12 illustrates the idea of scaling proto-value functions to large product spaces.[14]

Following Cvetkovic et al. (1980), various compositional schemes can be defined for constructing complex graphs from simpler graphs. We focus on compositions that involve the Kronecker (or the tensor) sum of graphs. Let $G_1, \ldots, G_n$ be n undirected graphs whose corresponding vertex and edge sets are specified as $G_i = (V_i, E_i)$. The *Kronecker sum graph* $G = G_1 \oplus \ldots \oplus G_n$ has the vertex set $V = V_1 \times \ldots V_n$, and edge set $E(u,v) = 1$, where $u = (u_1, \ldots, u_n)$ and $v = (v_1, \ldots, v_n)$, if and only if u_k is adjacent to v_k for some $u_k, v_k \in V_k$ and all $u_i = v_i, i \neq k$. For example, the grid graph illustrated in Figure 12 is the *Kronecker sum* of two path graphs; the hypercube is the Kronecker sum of three or more path graphs.

The Kronecker sum graph can also be defined using operations on the component adjacency matrices. If A_1 is a (p,q) matrix and A_2 is a (r,s) matrix, the Kronecker product matrix [15] $A = A_1 \otimes A_2$ is a (pr, qs) matrix, where $A(i,j) = A_1(i,j) * A_2$. In other words, each entry of A_1 is replaced by

14. Even greater reduction in the size of PVFs can be realized by exploiting the group invariance property of Laplacian operators, as described in Section 3.6. In particular, the graphs shown in Figure 12 have large automorphism groups, which can be exploited in significantly reducing the size of the corresponding Laplacian eigenspaces.

15. The Kronecker product of two matrices is often also referred to as the *tensor product* in the literature (Chow, 1997).

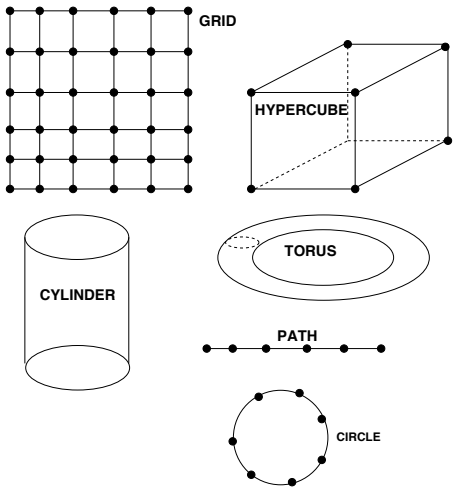


Figure 12: The spectrum and eigenspace of structured state spaces, including grids, hypercubes, cylinders, and tori, can be efficiently computed from "building block" subgraphs, such as paths and circles. Applied to MDPs, this hierarchical framework greatly reduces the computational expense of computing and storing proto-value functions.

the product of that entry with the entire A_2 matrix. The Kronecker sum of two graphs $G = G_1 \oplus G_2$ can be defined as the graph whose adjacency matrix is the Kronecker sum $A = A_1 \otimes I_2 + A_2 \otimes I_1$, where I_1 and I_2 are the identity matrices of size equal to number of rows (or columns) of A_1 and A_2, respectively. The main result that we will exploit is that the eigenvectors of the Kronecker product of two matrices can be expressed as the Kronecker products of the eigenvectors of the component matrices. The following result is well-known in the literature (Graham, 1981).

Theorem 2 *Let A and B be full rank square matrices of size $r \times r$ and $s \times s$, respectively, whose eigenvectors and eigenvalues can be written as*

$$Au_i = \lambda_i u_i,\ 1 \leq i \leq r \qquad\qquad Bv_j = \mu_j v_j,\ 1 \leq j \leq s.$$

Then, the eigenvalues and eigenvectors of the Kronecker product $A \otimes B$ and Kronecker sum $A \oplus B$ are given as

$$\begin{aligned} (A \otimes B)(u_i \otimes v_j) &= \lambda_i \mu_j (u_i \otimes v_j) \\ (A \oplus B)(u_i \otimes v_j) = (A \otimes I_s + I_r \otimes B)(u_i \otimes v_j) &= (\lambda_i + \mu_j)(u_i \otimes v_j). \end{aligned}$$

The proof of this theorem relies on the following identity regarding Kronecker products of matrices: $(A \otimes B)(C \otimes D) = (AC) \otimes (BD)$ for any set of matrices where the products AC and BD are well defined. We denote the set of eigenvectors of an operator $\mathcal{T}$ by the notation $X(\mathcal{T})$ and its spectrum by $\Sigma(\mathcal{T})$. A standard result that follows from the above theorem shows that the combinatorial

graph Laplacian of a Kronecker sum of two graphs can be computed from the Laplacian of each subgraph.[16]

Theorem 3 *If $L_1 = L(G_1)$ and $L_2 = L(G_2)$ are the combinatorial Laplacians of graphs $G_1 = (V_1, E_1, W_1)$ and $G_2 = (V_2, E_2, W_2)$, then the spectral structure of the combinatorial Laplacian $L(G)$ of the Kronecker sum of these graphs $G = G_1 \oplus G_2$ can be computed as*

$$(\Sigma(L), X(L)) = \{\lambda_i + \kappa_j, l_i \otimes k_j\},\ 1 \leq i \leq |V_1|, 1 \leq j \leq |V_2|,$$

where λ_i is the i^{th} eigenvalue of L_1 with associated eigenvector l_i and κ_j is the j^{th} eigenvalue of L_2 with associated eigenvector k_j.

The proof is omitted, but fairly straightforward by exploiting the property that the Laplace operator acts on a function by summing the difference of its value at a vertex with those at adjacent vertices. Figure 13 illustrates this theorem, showing that the eigenvectors of the combinatorial Laplacian produce a regular embedding of a grid in 2D as well as a cylinder in 3D. These figures were generated as follows. For the grid shown on the left, the eigenvectors were generated as the Kronecker product of the eigenvectors of the combinatorial Laplacian for two chains of size 10. The figure shows the embedding of the grid graph where each state was embedded in $\mathbb{R}^2$ using the second and third smallest eigenvector. For the cylinder on the right, the eigenvectors were generated as the Kronecker product of the eigenvectors of the combinatorial Laplacian for a 10 state closed chain and a 5 state open chain. The embedding of the cylinder shown on the right was produced using the third and fourth eigenvector of the combinatorial Laplacian.

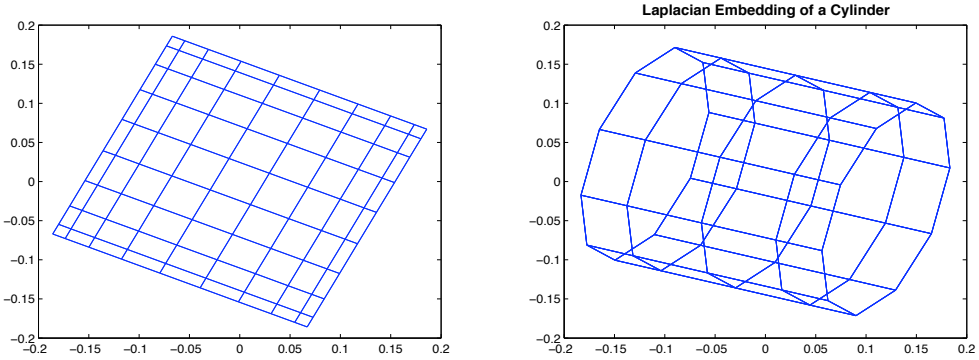


Figure 13: Left: this figure shows an embedding in $\mathbb{R}^2$ of a 10×10 grid world environment using "low-frequency" (smoothest) eigenvectors of the combinatorial Laplacian, specifically those corresponding to the second and third smallest eigenvalues. Right: the embedding of a "cylinder" graph using two low-order eigenvectors (3^{rd} and 4^{th}) of the combinatorial Laplacian. The cylinder graph is the Kronecker sum of a closed and open chain graph.

16. In contrast, the normalized Laplacian is not well-defined under sum, but has a well-defined semantics for the Kronecker or direct product of two graphs. The Kronecker product can also be used as a general method to approximate any matrix by factorizing it into the product of smaller matrices. We discuss the use of this approach to scaling PVFs in Section 9.

For the combinatorial Laplacian, the constant vector $\mathbf{1}$ is an eigenvector with associated eigenvalue $\lambda_0 = 0$. Since the eigenvalues of the Kronecker sum graph are the sums of the eigenvalues of the individual graphs, 0 will be an eigenvalue of the Laplacian of the sum graph as well. Furthermore, for each eigenvector v_i, the Kronecker product $v_i \otimes \mathbf{1}$ will also be an eigenvector of the sum graph. One consequence of these properties is that geometry is well preserved, so for example the combinatorial Laplacian produces well-defined embeddings of structured spaces. Figure 13 shows the embedding of a cylinder (Kronecker sum of a closed and open chain) under the combinatorial Laplacian.

5.2 Factored Representation Policy Iteration for Structured Domains

We derive the update rule for a factored form of RPI (and LSPI) for structured domains when the basis functions can be represented as Kronecker products of elementary basis functions on simpler state spaces. Basis functions are *column* eigenvectors of the diagonalized representation of a graph operator, whereas embeddings $\phi(s)$ are *row* vectors representing the first k basis functions evaluated on state s. By exploiting the property that $(A \otimes B)^T = A^T \otimes B^T$, it follows that embeddings for structured domains can be computed as the Kronecker products of embeddings for the constituent state components. As a concrete example, a grid world domain of size $m \times n$ can be represented as a graph $G = G_m \oplus G_n$ where G_m and G_n are *path graphs* of size m and n, respectively. The basis functions for the entire grid world can be written as the Kronecker product $\phi(s) = \phi_m(s^r) \otimes \phi_n(s^c)$, where $\phi_m(s^r)$ is the basis (eigen)vector derived from a path graph of size m (in particular, the row s^r corresponding to state s in the grid world), and $\phi_n(s^c)$ is the basis (eigen)vector derived from a path graph of size n (in particular, the column s^c corresponding to state s in the grid world).

Extending this idea to state action pairs, the basis function $\phi(s,a)$ can written as $e_I(a) \otimes \phi(s)$, where $e_I(a)$ is the unit vector corresponding to the index of action a (e.g., action a_1 corresponds to $e_1 = [1,0,\ldots]^T$). Actually, the full Kronecker product is not necessary if only a relatively small number of basis functions are needed. For example, if 50 basis functions are to be used in a $10 \times 10 \times 10$ hypercube, the full state embedding is a vector of size 1000, but only the first 50 terms need to be computed. Such savings imply proto-value functions can be efficiently computed even in very large structured domains. For a factored state space $s = (s^1, \ldots, s^m)$, we use the notation s^i to denote the value of the i^{th} component. We can restate the update rules for factored RPI and LSPI as follows:

$$\begin{aligned} \tilde{A}^{t+1} &= \tilde{A}^t + \phi(s_t,a_t)\left(\phi(s_t,a_t) - \gamma\phi(s'_t,\pi(s'_t))\right)^T \\ &= \tilde{A}^t + e_{I(a_t)} \otimes \prod_{\otimes} \phi_i(s^i_t) \\ &\times \left(e_{I(a_t)} \prod_{\otimes} \phi_i(s^i_t) - \gamma e_{I(\pi(s'_t))} \otimes \prod_{\otimes} \phi_i(s'^i_t) \right)^T . \end{aligned}$$

The corresponding update equation for the reward component is:

$$\tilde{b}^{t+1} = \tilde{b}^t + \phi(s_t,a_t) r_t = \tilde{b}^t + r_t e_{I(a_t)} \otimes \prod_{\otimes} \phi_i(s^i_t).$$

5.3 Experimental Results

To illustrate the Kronecker factorization presented in the previous section, we begin with a simple MDP. Figure 14 shows the results of using the factored RPI algorithm on a 10×10 grid world

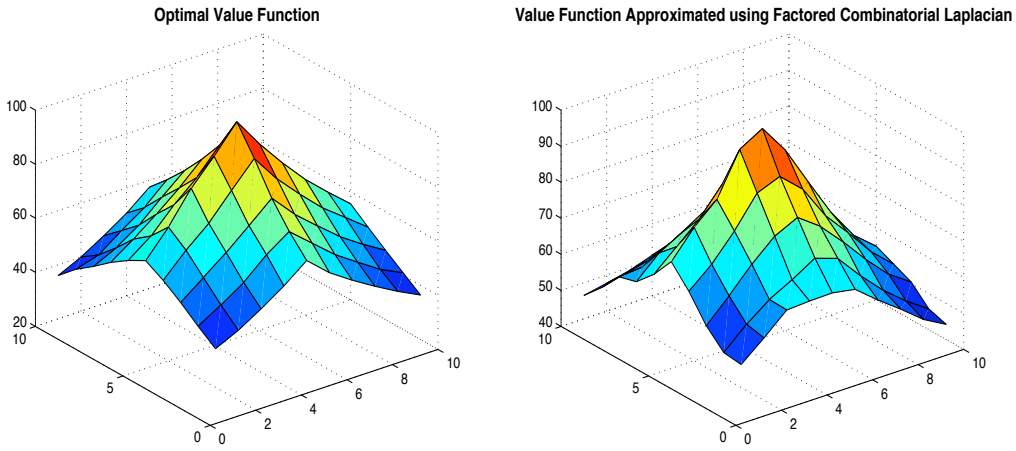


Figure 14: Left: the exact value function on a 10×10 grid world with a reward of $+100$ at the center. Right: a factored (combinatorial) Laplacian approximation using basis functions constructed by taking Kronecker products of basis functions for chain graphs (of length corresponding to row and column sizes).

domain. There are four (compass direction) actions, each of which succeeds with probability 0.9. Any "illegal" action (going "north" from the first row) leaves the agent in the same state. The only reward of $+100$ is received for reaching the center of the grid. The discount factor was set at $\gamma = 0.9$. If a "flat" approach was used, each basis function is a vector of size 100 and requires diagonalizing a Laplacian matrix of size 100×100. The factored PVFs are computed as the Kronecker product of the PVFs on a 10 node chain graph, which requires both significantly smaller space of size $10 \times k$ for k basis functions, and much less computational effort (diagonalizing a Laplacian of size 10×10). These computational savings obviously magnify in larger grid world domains. In a grid world with 10^6 states, "flat" proto-value functions require $k \times 10^6$ space and time proportional to $(10^6)^3$ to be computed, whereas the factored basis functions only require space $k \times 10^3$ to store with much less computational cost to find.

5.4 The Blocker Task

We now present a detailed study using a much larger factored multiagent domain called the "Blockers" task, which was first proposed by Sallans and Hinton (2004). This task, illustrated in Figure 15, is a cooperative multiagent problem where a group of agents try to reach the top row of a grid, but are prevented in doing so by "blocker" agents who move horizontally on the top row. If any agent reaches the top row, the entire team is rewarded by +1; otherwise, each agent receives a negative reward of -1 on each step. The agents always start randomly placed on the bottom row of the grid, and the blockers are randomly placed on the top row. The blockers remain restricted to the top row, executing a fixed strategy. The overall state space is the Cartesian product of the location of each agent. Our experiments on the blocker domain include more difficult versions of the task not studied in Sallans and Hinton (2004) specifically designed to test the scalability of the Kronecker product bases to "irregular" grids whose topology deviates from a pure hypercube or toroid. In the

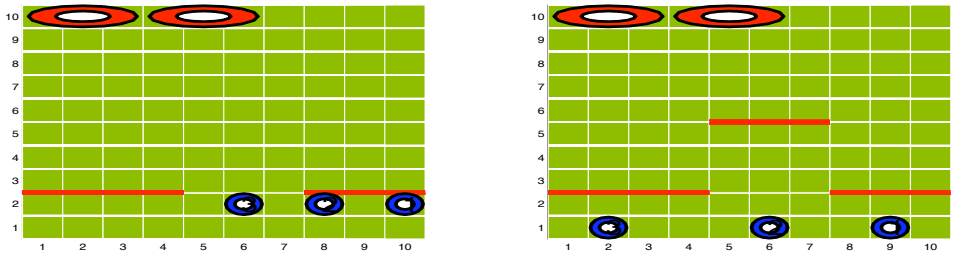


Figure 15: Two versions of the blocker domain are shown, each generating a state space of $> 10^6$ states. Interior walls shown create an "irregular" factored MDP whose overall topology can be viewed as a "perturbed" variant of a pure product of grids or cylinders (for the "wrap-around" case).

first variant, shown on the left in Figure 15, horizontal interior walls extend out from the left and right side walls between the second and third row. In the second variant, an additional interior wall is added in the middle as shown on the right.[17]

The basis functions for the overall Blocker state space were computed as Kronecker products of the basis functions over each agent's state space. Each agent's state space was modeled as a grid (as in Figure 14) or a cylinder (for the "wrap-around" case). Since the presence of interior walls obviously violates the pure product of cylinders or grids topology, each individual agent's state space was learned from a random walk. The overall basis functions were then constructed as Kronecker products of Laplacian basis functions for each learned (irregular) state grid.

Figure 16 compares the performance of the factored Laplacian bases with a set of radial basis functions (RBFs) for the first Blocker domain (shown on the left in Figure 15). The width of each RBF was set at $\frac{2|S_a|}{k}$ where $|S_a|$ is the size of each individual agent's grid, and k is the number of RBFs used. The RBF centers were uniformly spaced. The results shown are averages over 10 learning runs. On each run, the learned policy is measured every 25 training episodes. Each episode begins with a random walk of a maximum of 70 steps (terminating earlier if the top row was reached). After every 25 such episodes, RPI is run on all the samples collected thus far. The learned policy is then tested over 500 test episodes. The graphs plot the average number of steps to reach the goal. The experiments were conducted on both "normal" grids (not shown) and "wrap around" cylindrical grids. The results show that RBFs converge faster, but learn a worse policy. The factored Laplacian bases converge slower than RBFs, but learn a substantially better policy. Figure 16 also shows results for the second Blocker domain (shown on the right in Figure 15 with both side and interior middle walls), comparing 100 factored Laplacian bases with a similar number of RBFs. The results show a significant improvement in performance of the factored Laplacian bases over RBFs.

In terms of both space and time, the factored approach greatly reduces the computational complexity of finding and storing the Laplacian bases. A worst-case estimate of the size of the full Laplacian matrix is $O(|S|^2)$. Diagonalizing a $|S| \times |S|$ symmetric matrix and finding k eigenvectors requires time $O(k|S|^2)$ and $O(k|S|)$ space. Instantiating these general estimates for the Blocker

17. In the Blocker domain, the interior walls are modeled as having "zero width", and hence all 100 states in each grid remain accessible, unlike the two-room environment.

domain, let n refer to the number of rows and columns in each agent's state space ($n = 10$ in our experiments), and k refer to the number of basis functions ($k = 100$ in our experiments). Then, the size of the state space is $|S| = (n^2)^3$, implying that the non-factored approach requires $O(k(n^2)^3)$ space and $O(k(n^6)^2)$ time, whereas the factored approach requires $O(kn^2)$ space and $O(k(n^2)^2)$ time. Note these are worse-case estimates. The Laplacian matrix is in fact highly sparse in the Blocker domain, requiring far less than $O(|S|^2)$ space to be stored. In fact, even in such a deterministic MDP where the Laplacian matrix can be stored in $O(|S|)$ space, the non-factored approach will still take $O(kn^3)$ space and $O(kn^6)$ time, whereas the factored approach takes $O(kn)$ space and $O(kn^2)$ time.

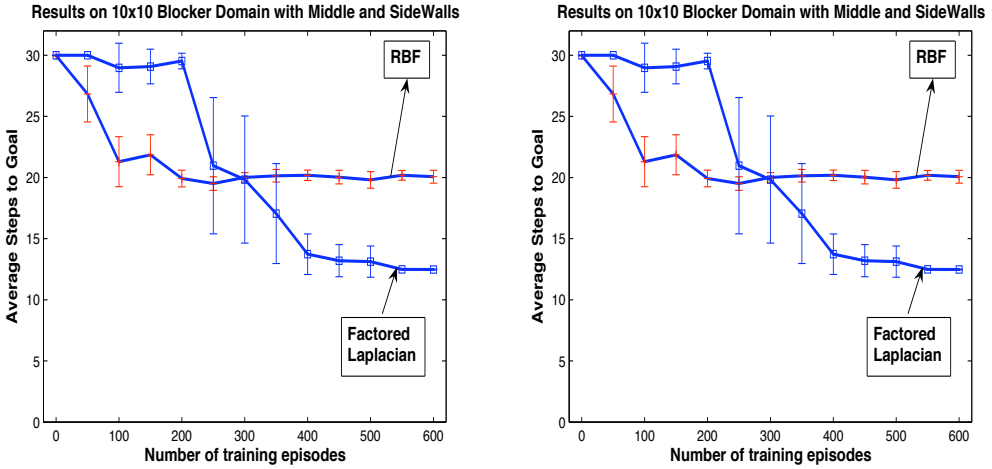


Figure 16: Comparison of factored (Laplacian) PVF basis functions with hand coded radial basis functions (RBF) on a 10×10 "wrap-around" grid with 3 agents and 2 blockers of $> 10^6$ states. Both approaches were tested using 100 basis functions. The plots show performance of PVFs against RBFs on the two blocker domains in Figure 15.

6. Scaling Proto-Value Functions: Continuous Domains

Thus far, the construction of proto-value functions was restricted to discrete MDPs. We now show how proto-value functions can be constructed for continuous MDPs, which present significant challenges not encountered in discrete state spaces. The eigenfunctions of the Laplacian can only be computed and stored on sampled real-valued states, and hence must be interpolated to novel states. We apply the Nyström interpolation method. While this approach has been studied previously in kernel methods (Williams and Seeger, 2000) and spectral clustering (Belongie et al., 2002), our work represents the first detailed study of the Nyström method for learning control, as well as a detailed comparison of graph normalization methods (Mahadevan et al., 2006).

There is a rich and well-developed theory of the Laplace operator on manifolds, which we can only briefly summarize here. The Laplace-Beltrami operator has been extensively studied in the general setting of Riemannian manifolds (Rosenberg, 1997). Riemannian manifolds have been actively studied recently in machine learning in several contexts, namely in the context of designing new types of kernels for supervised machine learning (Lafferty and Lebanon, 2005) and faster policy gradient methods using the natural Riemannian gradient on a space of parametric policies (Kakade, 2002; Bagnell and Schneider, 2003; Peters et al., 2003).

The Laplacian on Riemannian manifolds and its eigenfunctions (Rosenberg, 1997), which form an orthonormal basis for square-integrable functions on the manifold (Hodge's theorem), generalize Fourier analysis to manifolds. Historically, manifolds have been applied to many problems in AI, for example configuration space planning in robotics, but these problems assume a model of the manifold is known (Latombe, 1991; Lavalle, 2006), unlike here where only samples of a manifold are given.

6.1 Nyström Extension

To learn policies on continuous MDPs, it is necessary to be able to extend eigenfunctions computed on a set of points $\in \mathbb{R}^n$ to new unexplored points. We describe here the Nyström method, which can be combined with iterative updates and randomized algorithms for low-rank approximations. The Nyström method interpolates the value of eigenvectors computed on sample states to novel states, and is an application of a classical method used in the numerical solution of integral equations (Baker, 1977). The eigenfunction problem can be stated as

$$\int_D K(x,y)\phi(y)dy = \lambda\phi(x), \forall x \in D, \tag{4}$$

where D can be any domain, for example, $\mathbb{R}$. Using the standard quadrature approximation, the above integral can be written as

$$\int_D K(x,y)\phi(y)dy \approx \sum_{i=1}^{n} w_i k(x,s_i)\hat{\phi}(s_i), \tag{5}$$

where w_i are the quadrature weights, s_i are n selected sample points, and $\hat{\phi}$ is an approximation to the true eigenfunction. Combining Equation 4 and Equation 5 gives us

$$\sum_{i=1}^{n} w_i k(x,s_i)\hat{\phi}(s_i) = \hat{\lambda}\hat{\phi}(x).$$

By letting x denote any set of n points, for example the set of quadrature points s_i itself, the kernel $k(s_i,s_j)$ becomes a symmetric matrix. This enables computing the approximate eigenfunction at any new point as

$$\hat{\phi}_m(x) = \frac{1}{\hat{\lambda}} \sum_{i=1}^{n} w_i k(x,s_i)\hat{\phi}_m(s_i). \tag{6}$$

Let us instantiate Equation 6 in the context of the normalized Laplacian $\mathcal{L} = I - D^{-\frac{1}{2}} W D^{-\frac{1}{2}}$. First, note that if λ_i is an eigenvalue of $\mathcal{L}$, then $1-\lambda_i$ is the corresponding eigenvalue of the diffusion matrix $D^{-\frac{1}{2}} W D^{-\frac{1}{2}}$. Applying the the Nyström extension for computing the eigenfunctions of the normalized Laplacian $\mathcal{L}\phi_i = \lambda_i \phi_i$, we get the equation

$$\phi_i(x) = \frac{1}{1-\lambda_i} \sum_{y \sim x} \frac{w(x,y)}{\sqrt{d(x)d(y)}}\, \phi_i(y),$$

where $d(z) = \sum_{y \sim z} w(z,y)$, and x is a new vertex in the graph. Note that the weights $w(x,y)$ from the new state x to its nearest neighbors y in the previously stored samples is determined at "run time"

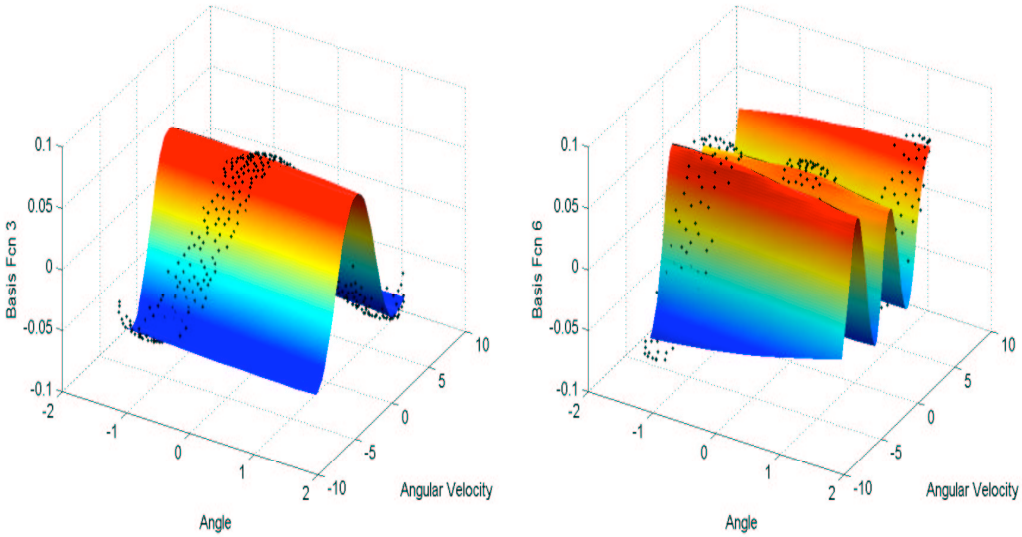


Figure 17: This figure illustrates the Nyström interpolation method for extending eigenfunctions on samples to new states. Left: the 3rd eigenvector of the Laplacian plotted on a set of samples (shown as filled dots) drawn from a random walk in the inverted pendulum domain, as well as its Nyström interpolated values. Right: the Nyström interpolated 6th eigenvector illustrated the entire state space as well as on the actual samples (again shown as filled dots).

using the same nearest neighbor weighting algorithm used to compute the original weight matrix W. An extensive discussion of the Nyström method is given in Drineas and Mahoney (2005), and more details of its application to learning control in MDPs are given in Mahadevan et al. (2006).

Figure 17 illustrates the basic idea. Note that the Nyström method does *not* require recalculating eigenvectors—in essence, the embedding of a new state is computed by averaging over the already computed embeddings of "nearby" states. In practice, significant speedups can be exploited by using the following optimizations. We have empirically observed that roughly only 10% of the overall samples needed for learning a good policy are necessary to construct basis functions. Once the bases is defined over these sub-sampled states, the Nyström extended embeddings of the remaining 90% of training samples needs to be calculated only once, and henceforth can be cached during repeated runs of policy iteration. During testing, the Nyström embeddings of novel states encountered must be computed, but since the eigenvectors are defined over a relatively small core set of sample states, the extensions can be computed very efficiently using a fast nearest neighbor algorithm.[18]

6.2 Representation Policy Iteration for Continuous Domains

Figure 18 presents the modified RPI algorithm for continuous Markov decision processes. The core of the algorithm remains the same as before, but there are important differences from the discrete case. First, the proto-value functions are computed on a subsampled set of states, for two reasons:

18. In our experiments, we used the TSTOOLS MATLAB nearest neighbor package.

RPI $(\pi_m, T, N, Z, \varepsilon, k, O, \mathcal{D})$:

// π_m: Initial policy
// T: Number of initial random walk trials
// N: Maximum length of each trial
// ε : Convergence condition for policy iteration
// k: Number of proto-value basis functions to use
// O: Type of graph operator used
// $\mathcal{D}$: Data set of transitions

Sample Collection Phase

1. See Figure 4 on page 2177.

Representation Learning Phase

2. Build an undirected weighted graph G from the set of subsampled transitions $\mathcal{D}_s \subseteq \mathcal{D}$ using the method described in Section 6.4 on graph construction from point sets $\in \mathbb{R}^n$. Compute the operator O on graph G as discussed in Section 6.4.
3. Compute the k "smoothest" eigenvectors of O on the sub-sampled graph $\mathcal{D}_s$, and collect them as columns of the basis function matrix Φ, a $|\mathcal{D}_s| \times k$ matrix. The embedding of a state action pair $\phi(s,a)$ where $s \in \mathcal{D}_s$ is given as $e_a \otimes \phi(s)$, where e_a is the unit vector corresponding to action a, $\phi(s)$ is the s^{th} row of Φ, and $\otimes$ is the Kronecker product.

Control Learning Phase:

4. See Figure 8 on page 2191. For all transitions involving a state $s \notin \mathcal{D}_s$, its embedding is computed using the Nyström extension described in Section 6.1.
5. **Optional:** Repeat the above procedure by calling RPI $(\pi_{m+1}, T, N, \varepsilon, k, O, \mu, \mathcal{D})$.

Figure 18: Pseudo-code of the representation policy iteration algorithm for continuous MDPs.

the number of samples needed to compute the proto-value functions is much less than that needed to learn a good policy using RPI, as the experiments in Section 7 reveal. In Figure 18, $\mathcal{D}_Z$ denotes the subsampled set of states. The choice of the subsampling method can make a significant difference, as explained below. The second major difference is the use of the Nyström method to extend proto-value functions from the samples stored in $\mathcal{D}_Z$ to all the states visited during the initial random walk (denoted $\mathcal{D}$ in Figure 18), as well as new states encountered during the testing of a learned policy.

6.3 Sampling from Point Sets $\in \mathbb{R}^n$

One challenge in continuous MDPs is how to choose a subset of samples from which a graph can be built and proto-value functions computed. The set of samples collected during the course of exploration can be very large, and a much smaller set of samples is usually sufficient to learn proto-value functions. Many ways of constructing a subsample from the overall sample can be devised. The simplest method is of course to randomly subsample from the complete set, but this might not be the most efficient way of using the samples. Figure 19 illustrates two methods for subsampling in

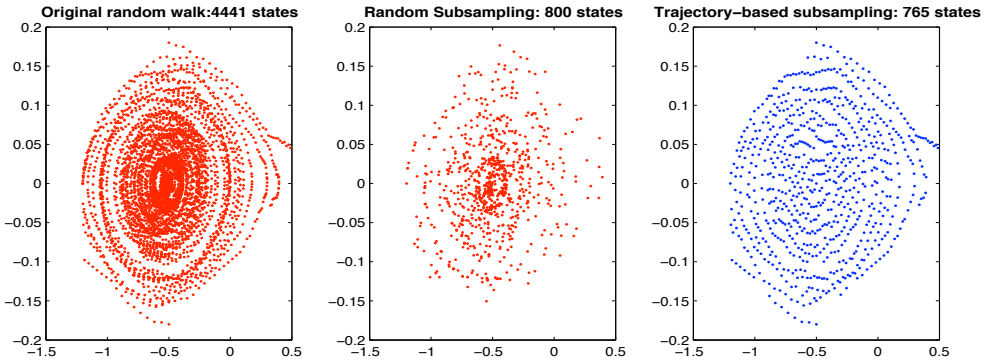


Figure 19: The problem of subsampling is illustrated in the mountain car domain. On the left is shown the original states visited during a random walk. In the middle is the subsampled data using a random subsampling algorithm. On the right is a trajectory based subsampling method.

the mountain car domain, including random subsampling and trajectory-based subsampling. The trajectory-based algorithm follows a greedy strategy: starting with the null set, add samples to the subset that are not within a specified distance to any sample currently in the subset. A maximal subset is returned when no more samples can be added. The trajectory-based method also tries to retain "important" samples, such as goal states or states with high reward. Note that the random subsampling method clearly loses important information about the trajectory, which is nicely retained by the trajectory method.

More formally, the trajectory based subsampling algorithm works as follows. We define an ε-net of points in $\mathcal{S}'$ to be a subset $\mathcal{S}''$ such that no two points are closer than ε, and that for every point y in $\mathcal{S}'$, there is a point in $\mathcal{S}''$ which is not farther than ε from y. One can construct a (random) ε-net in $\mathcal{S}'$ as follows. Pick $x_0 \in \mathcal{S}'$ at random. By induction, for $k \geq 1$ suppose $x_0, x_1, \ldots, x_k$ have been picked so that the distance between any pair is larger than ε. If

$$R_k := \mathcal{S}' \setminus (\cup_{l=1}^{k} B_\varepsilon(x_l))$$

is empty, stop, otherwise pick a point x_{k+1} in R_k. By definition of R_k the distance between x_{k+1} and any of the points $x_0, \ldots, x_k$ is not smaller than ε. When this process stops, say after k^* points have been selected, for any $y \in \mathcal{S}'$ we can find a point in $\mathcal{S}''$ not farther than ε, for otherwise $y \in R_{k^*}$ and the process would not have stopped.

In the experiments reported in Section 7, where states are continuous vectors $\in \mathbb{R}^n$, typically $< 10\%$ of the transitions in the original set of random walks are necessary to learn an adequate set of basis functions. For example, in the mountain car task, around 700 samples are sufficient to form the basis functions, whereas usually > 7000 samples are needed to learn a close to optimal policy.[19]

19. In Section 9, we describe how Kronecker factorization can be used to significantly compress the size of the basis matrices.

6.4 Graph Construction from Point Sets $\in \mathbb{R}^n$

Given a data set $\{x_i\}$ in $\mathbb{R}^n$, we can associate different weighted graphs to this point set. There are different choices of edges and for any such choice there is a choice of weights on the edges. In the experiments below, the following construction was used. Edges were inserted between a pair of states x_i and x_j if:

- x_j is among the k nearest neighbors of x_i, where $k > 0$ is a parameter.

Weights were assigned to the edges in the following way:

- $W(i,j) = \alpha(i) e^{-\frac{||x_i - x_j||^2_{\mathbb{R}^n}}{\sigma}}$, where $\sigma > 0$ is a parameter, and α a weight function to be specified.

Observe that for undirected graphs, since x_j can be among the K nearest neighbors of x_i but x_i may not be among the K nearest neighbors of x_j, the above construction will still yield asymmetric weight matrices. We used an additional symmetrization step where we replaced the weight matrix W constructed by the symmetric $W + W^T$. If the states $\{x_i\}$ are drawn uniformly from a Riemannian manifold, then it is shown in Belkin and Niyogi (2004) that the above construction, with $\alpha = 1$, approximates the continuous Laplace-Beltrami operator on the underlying manifold. If $\{x_i\}$ is not drawn uniformly from the manifold, as it typically happens in MDPs when the space is explored by an agent, it is shown in Lafon (2004) that a pre-processing normalization step can (must) be performed that yields the weight function α, so that the above construction yields an approximation to the Laplace-Beltrami operator. Various ways of normalizing the weight matrix were explored in our experiments in Section 7. In particular, we compared the normalized Laplacian $\mathcal{L} = D^{-\frac{1}{2}}(D - W)D^{-\frac{1}{2}}$ and the combinatorial Laplacian, $L = D - W$ operators.

7. Fully Interleaved Representation and Policy Learning: Continuous MDPs

In this section, we present a detailed analysis of fully interleaved representation and policy learning on continuous MDPs. By "fully interleaved", we mean that the overall learning run is divided into a set of discrete episodes of sample collection, basis construction, and policy learning. At the end of each episode, a set of additional samples is collected using either a random walk (off-policy) or the currently best performing policy (on-policy), and then basis functions are then recomputed and a new policy is learned. In all the experiments below, the trajectory based method was used to build the graph from which proto-value functions were learned. We discuss alternate approaches for interleaving basis function generation and control learning in Section 9.

7.1 Three Control Tasks

We explored the effectiveness and stability of proto-value functions in three continuous domains—the Acrobot task, the inverted pendulum task, and the mountain car task—that have long been viewed as benchmarks in the field. These three domains are now described in more detail.

The Inverted Pendulum: The inverted pendulum problem requires balancing a pendulum of unknown mass and length by applying force to the cart to which the pendulum is attached. We used the implementation described in Lagoudakis and Parr (2003). The state space is defined by two variables: θ, the vertical angle of the pendulum, and $\dot{\theta}$, the angular velocity of the pendulum. The three actions are applying a force of -50, 0, or 50 Newtons. Uniform noise from -10 and 10 is added

to the chosen action. State transitions are defined by the nonlinear dynamics of the system, and depend upon the current state and the noisy control signal, u.

$$\ddot{\theta} = \frac{g\sin(\theta) - \alpha ml\dot{\theta}^2\sin(2\theta)/2 - \alpha\cos(\theta)u}{4l/3 - \alpha ml\cos^2(\theta)},$$

where g is gravity, 9.8 m/s^2, m is the mass of the pendulum, 2.0 kg, M is the mass of the cart, 8.0 kg, l is the length of the pendulum, .5 m, and $\alpha = 1/(m+M)$. The simulation time step is set to 0.1 seconds. The agent is given a reward of 0 as long as the absolute value of the angle of the pendulum does not exceed $\pi/2$. If the angle is greater than this value the episode ends with a reward of -1. The discount factor was set to 0.95. The maximum number of episodes the pendulum was allowed to balance was fixed at 3000 steps. Each learned policy was evaluated 10 times.

Mountain Car: The goal of the *mountain car* task is to get a simulated car to the top of a hill as quickly as possible (Sutton and Barto, 1998). The car does not have enough power to get there immediately, and so must oscillate on the hill to build up the necessary momentum. This is a minimum time problem, and thus the reward is -1 per step. The state space includes the position and velocity of the car. There are three actions: full throttle forward (+1), full throttle reverse (-1), and zero throttle (0). Its position, x_t and velocity $\dot{x}_t$, are updated by

$$x_{t+1} = \text{bound}[x_t + \dot{x}_{t+1}]$$

$$\dot{x}_{t+1} = \text{bound}[\dot{x}_t + 0.001a_t + -0.0025, cos(3x_t)],$$

where the bound operation enforces $-1.2 \leq x_{t+1} \leq 0.6$ and $-0.07 \leq \dot{x}_{t+1} \leq 0.07$. The episode ends when the car successfully reaches the top of the mountain, defined as position $x_t >= 0.5$. In our experiments we allow a maximum of 500 steps, after which the task is terminated without success. The discount factor was set to 0.99.

The Acrobot Task: The Acrobot task (Sutton and Barto, 1998) is a two-link under-actuated robot that is an idealized model of a gymnast swinging on a highbar. The only action available is a torque on the second joint, discretized to one of three values (positive, negative, and none). The reward is -1 for all transitions leading up to the goal state. The detailed equations of motion are given in Sutton and Barto (1998). The state space for the Acrobot is 4-dimensional. Each state is a 4-tuple represented by $(\theta_1, \dot{\theta}_1, \theta_2, \dot{\theta}_2)$. θ_1 and θ_2 represent the angle of the first and second links to the vertical, respectively, and are naturally in the range $(0, 2\pi)$. $\dot{\theta}_1$ and $\dot{\theta}_2$ represent the angular velocities of the two links. Notice that angles near 0 are actually very close to angles near 2π due to the rotational symmetry in the state space.

Figure 20 plots the Acrobot state space projected onto the subspace spanned by the two joint angles θ_1 and θ_2. This subspace is actually a torus. To approximate computing distances on the torus, the original states were projected upwards to a higher dimensional state space $\subset \mathbb{R}^6$ by mapping each angle θ_i to $(\sin(\theta_i), \cos(\theta_i))$. Thus, the overall state space is now $(\sin(\theta_1), \cos(\theta_1), \dot{\theta}_1, \sin(\theta_2), \cos(\theta_2), \dot{\theta}_2)$. The motivation for this remapping is that now Euclidean distances in this augmented space better approximate local distances on the torus. In fact, ignoring the wrap-around nature of the Acrobot state space by simply using a local Euclidean distance metric on the four-dimensional state space results in significantly poorer performance. This example illustrates how overall global knowledge of the state space, just like in the Blockers domain, is valuable in designing a better local

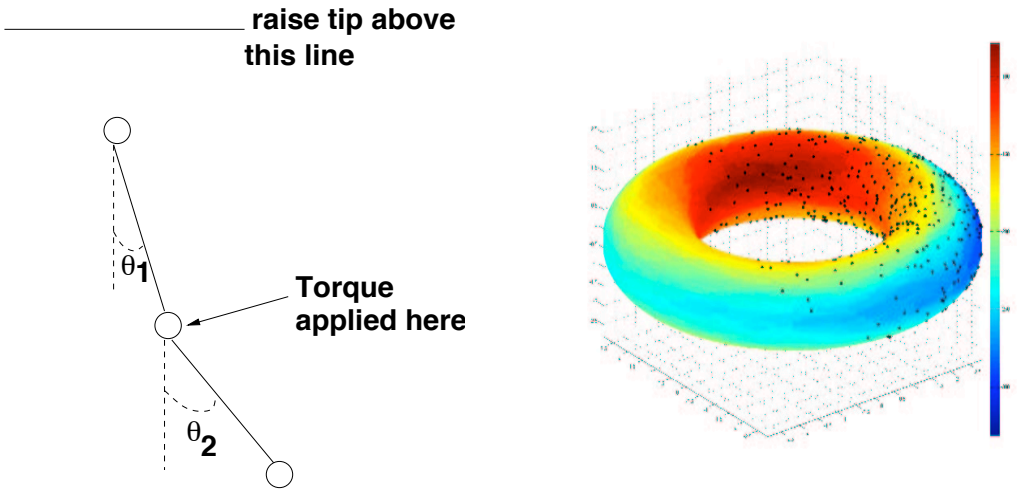


Figure 20: The state space of the Acrobot (shown on the left) exhibits rotational symmetries. The figure on the right plots its projection onto the subspace of $\mathbb{R}^2$ spanned by the two joint angles θ_1 and θ_2, which can be visualized as a torus. The angular velocities $\dot{\theta}_1$ and $\dot{\theta}_2$ were set to 0 for this plot. The points shown on the torus are subsampled states from a random walk. The colors indicate the value function, with red (darker) regions representing states with higher values.

distance function for learning PVFs. This domain serves to reemphasize that basis construction is dependent on a good choice of a local distance metric.

7.2 RPI with Off-Policy Sampling

In the first set of experiments, we used off-policy random walks in Step 1 of the sample collection phase in the RPI algorithm since we wanted to compare the effects of different parameter choices (graph operator, number of nearest neighbors, number of bases) using the *same* set of samples. In Section 7.4 we will see that significantly better results were obtained using a modified form of on-policy sampling. Table 1 summarizes the range of parameters over which the RPI algorithm was tested in these domains. The results for the following experiments were (median) averaged over 30 runs. To avoid clutter, variances are shown only on selected plots.

As Table 1 reveals, the type of off-policy sample collection used in the experiments below varied, from a long series of short random walks (inverted pendulum) to a short series of long random walks (Acrobot). In particular, in the inverted pendulum, samples were collected using a series of short random walks, typically of length < 20 before the episode terminated because the pole was dropped. This simple strategy was sufficient to explore the underlying manifold. By contrast, in the mountain car domain, longer random walks were needed to explore the manifold. One reason for this difference is the nature of the underlying manifold: the samples in the inverted pendulum are in a relatively narrow region around the 45 degree line. In contrast, the samples in the mountain car domain are distributed across a wider region of the state space. Finally, in the Acrobot domain, the random walks were very long, terminating when the goal state was reached.

Another difference in sample collection in these domains was in initialization. In the inverted pendulum and Acrobot domains, the initial state was always set the same, with the pole starting from the vertical position at rest, or the arm at rest. In the mountain car domain, however, starting the car from a position of rest at the bottom of the hill produced poorer results than starting from the bottom with the velocities initialized randomly. The experiments reported below scaled the raw state variables to make the dimensions of each variable more commensurate. The scaling used is shown in Table 1.

While performance in all three domains is measured by the number of steps, note that for the Acrobot and mountain car task, lower numbers indicate better performance since we are measuring the steps to reach the goal. In the inverted pendulum, however, since we are measuring the number of steps that the pole remained upright, higher numbers indicate better performance.

Local Distance Metric: In the first experiment, illustrated in Figure 21, the effect of varying the local distance metric used in constructing the graph Laplacian was evaluated, from a low setting of $k = 10$ nearest neighbors to a higher setting of $k = 50$ nearest neighbors. All the plots in the figure show median-averaged plots over 30 learning runs. Variances are not shown to avoid clutter. The effect of varying k was most pronounced in the inverted pendulum domain, with less tangible results in the mountain car and Acrobot domains. Note that in the inverted pendulum domain, the differences between $k = 25$ and $k = 50$ are negligible, and the corresponding runs tightly overlap.

Number of Basis Functions: Figure 22 varied the number of proto-value functions used. Here, there were significant differences, and the results reveal a nonlinear relationship between the number of PVFs used and the best performance. In the Acrobot task, the best results were obtained for 25 and 100 PVFs, and significantly poorer results for 50 PVFs. In the inverted pendulum domain, 10 PVFs was significantly better than using 30 PVFs, but was closely matched by using 60 PVFs. Finally, in the mountain car domain, 30 PVFs produced the best results, followed by 50 PVFs and a setting of 10 PVFs produced the worst results.

Type of Graph Operator: Figure 23 investigates the effect of varying the graph operator in the three domains. The two operators compared were the normalized Laplacian $\mathcal{L} = I - D^{-\frac{1}{2}} W D^{-\frac{1}{2}}$ and the combinatorial Laplacian $L = D - W$. In both the Acrobot and mountain car domains, the normalized Laplacian operator produced significantly better results than the combinatorial Laplacian. However, in the inverted pendulum domain, the combinatorial Laplacian was better than the normalized Laplacian operator. These results suggest an interesting dependence between the graph operator and the type of manifold. Note that in both the Acrobot and mountain car domains, the manifold is significantly more spread out spatially than the inverted pendulum task.

7.3 RPI with On-Policy Sampling

As noted earlier, the performance of PVFs can be improved using a modified form of on-policy sampling in Step 1 of the sample collection phase in the RPI algorithm. Specifically, we kept track of the best-performing policy (in terms of the overall performance measure of the number of steps). If the policy learned in the current round of RPI improved on the best-performing policy thus far, samples were collected in the next iteration of RPI using the newly learned policy (which was then viewed as the best performing policy in subsequent runs). Otherwise, samples were collected using an off-policy random walk. We also found that using shorter episodes of sample collection in between rounds of representation construction and policy estimation also produced better results.

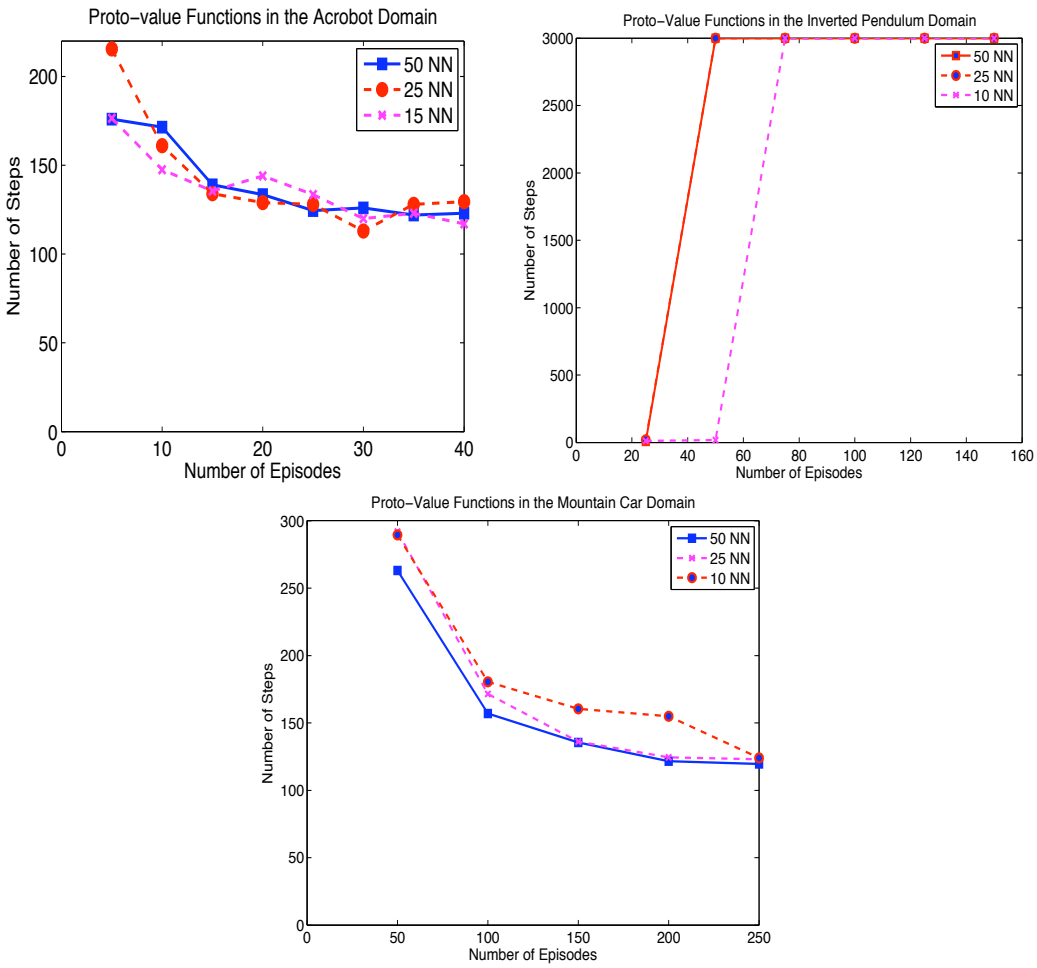


Figure 21: Performance of PVFs on the Acrobot, inverted pendulum, and mountain car domains as a function of the number of nearest neighbors used to compute the graph Laplacian. Results are median averages over 30 learning runs. In all three domains, the graph operator used was the normalized Laplacian. For the Acrobot domain, the number of PVFs was set at 100, whereas in the mountain car and inverted pendulum tasks, the number of PVFs was set to 30.

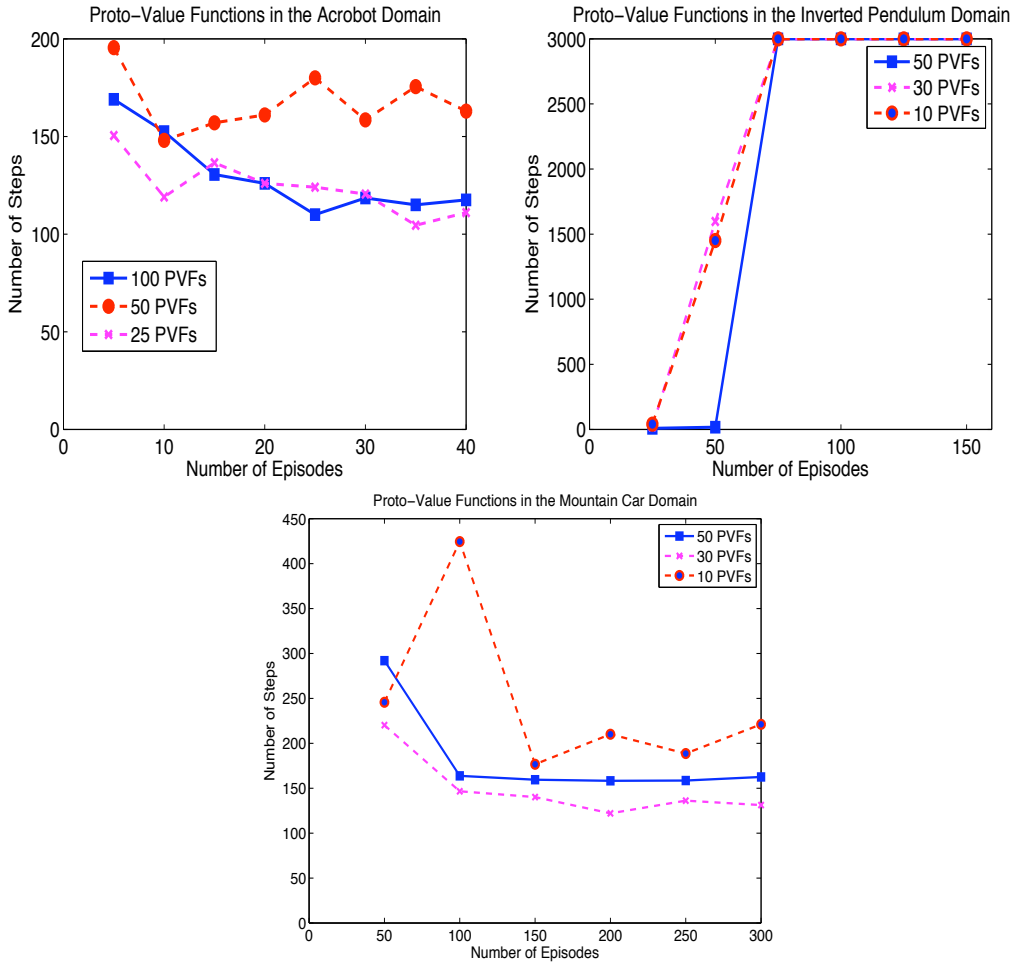


Figure 22: Performance of PVFs on the Acrobot, inverted pendulum, and mountain car domains as a function of the number of basis functions. Results shown are median averages over 30 learning runs. In all three domains, the normalized Laplacian was used as the graph operator. The number of nearest neighbors $k = 25$ in the Acrobot and inverted pendulum domains, and $k = 30$ in the mountain car domain.

Figure 24 shows the results of these two modifications in the Acrobot domain, whereas Figure 25 and Figure 27 show the corresponding results from the inverted pendulum and mountain car domains. Comparing these results with the corresponding off-policy results in Figure 21, Figure 22, and Figure 23 shows significantly faster convergence of PVFs in all three domains.

7.4 Comparing PVFs with RBFs on Continuous MDPs

In this section, we compare the performance of PVFs with radial basis functions (RBFs), which are a popular choice of basis functions for both discrete and continuous MDPS. We restrict our comparison of PVFs and RBFs in this section to the inverted pendulum and mountain car domains. To

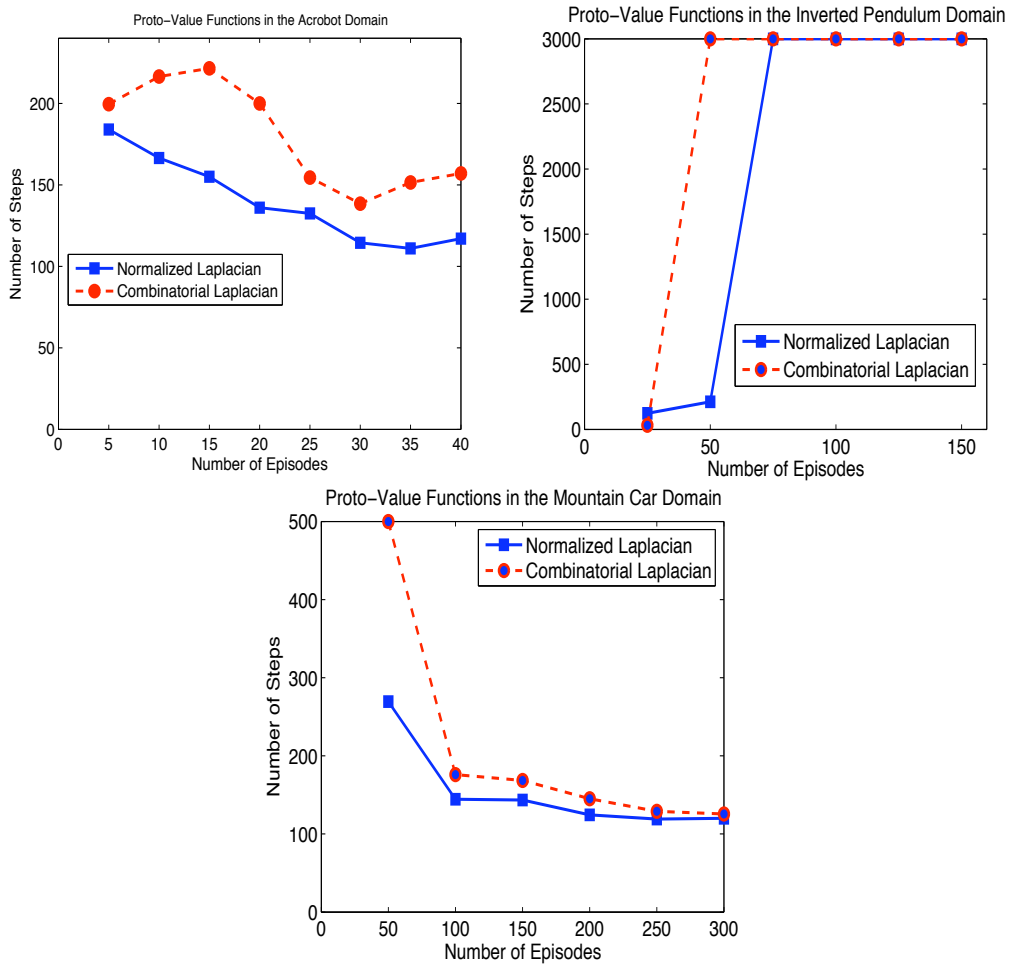


Figure 23: Performance of PVFs in the Acrobot, inverted pendulum, and mountain car domains as a function of the graph operator. Results shown are median averages over 30 learning runs. In the Acrobot task, 100 PVFs were used, whereas 30 basis functions were used in the mountain car task, and 10 basis functions were used in the inverted pendulum task.

choose a suitable set of parameters for RBFs, we initially relied on the values chosen in the published study of LSPI for the inverted pendulum domain (Lagoudakis and Parr, 2003). However, we found that by tuning the kernel widths, we were able to significantly improve the performance of RBFs over that previously reported in their experiments. Table 2 shows the parameters of the RBF used in the comparisons below. Generally speaking, the results demonstrate that PVFs are significantly quicker to converge, by almost a factor of two in both the inverted pendulum and mountain car domains. Asymptotically, both approaches to converge to the same result. We emphasize that these comparisons are meant to be *suggestive*, and not definitive. For example, we did not fine tune the centers of the RBF bases, or incorporate the scaling factors used in the experiments with PVFs. Our goal here is to provide a reasonable set of benchmarks to compare PVFs against, commensurate with that shown in earlier studies using such parametric approximators.

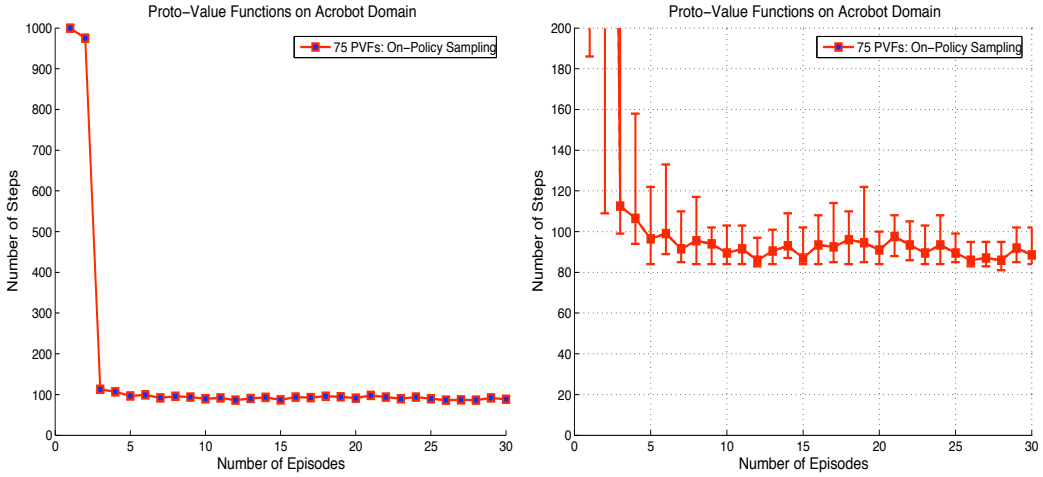


Figure 24: Performance of PVFs with on-policy sampling in the Acrobot task. The plot on the left shows the median average number of steps to goal averaged over 30 runs. The plot on the right shows the variance, after scaling the y axis to magnify the plot.

Inverted Pendulum: We begin by comparing the performance of PVFs with a linear RBF approximation architecture for the inverted pendulum domain. Figure 25 plots the effect of varying the kernel width for RBFs in the inverted pendulum domain (left plot). It is seen that the best results are obtained for a kernel width $\sigma = 0.25$. We compare a varying number of RBF architectures with using 15 PVFs in Figure 25 (right plot). PVFs converge significantly faster to the final goal of balancing the pendulum for 3000 steps: PVFs take 20 trials to converge, but RBFs take roughly twice as long. Figure 26 plots the variance across 100 learning runs for both PVFs and RBFs, showing that PVFs not only converge faster, but also have significantly less variance.

Mountain Car: As with the inverted pendulum, we were able to improve the performance of RBFs by fine-tuning the kernel width, although the differences are less significant than in the inverted pendulum domain. Figure 27 plots the effect of varying the kernel width for RBFs using 13 basis functions in the mountain car domain (left plot). We also found increasing the number of RBF basis functions above 13 worsened their performance. The figure also plots the best performing RBF architecture (13 basis functions) compared with the PVF approach (25 basis functions). Given sufficient training experience, both converge to approximately the same result, although PVFs seem to converge to a slightly better result. However, as with the inverted pendulum results, PVFs converge significantly quicker, and clearly outperform RBFs for smaller numbers of samples.

Figure 28 shows the variances over 30 runs for both PVFs and RBFs in the mountain car domain. As in the inverted pendulum, we note that PVFs clearly converge more quickly to a more stable performance than RBFs, although the differences are not as dramatic as in the inverted pendulum domain.

Parameter	Inverted Pendulum	Mountain Car	Acrobot
Episodes T	(20 to 160)	(50 to 300)	(5 to 40)
Episode Length N	≤ 20	≤ 70	≤ 800
Nearest neighbors ω	$\{10, 25, 50\}$	$\{10, 25, 50\}$	$\{25, 50, 100\}$
Number of PVFs k	$\{10, 30, 60\}$	$\{10, 30, 50\}$	$\{25, 50, 100\}$
Graph Operator O	(Norm., Comb.)	(Norm., Comb.)	(Norm., Comb.)
Scaling	$(3\theta, \dot{\theta})$	$(x, 3\dot{x})$	$(\theta_1, \theta_2, 0.5\dot{\theta_1}, 0.3\dot{\theta_2})$

Table 1: Parameter values (as defined in Figure 18) for Acrobot, inverted pendulum and mountain car domains. Comb. and Norm. refer to the combinatorial and normalized Laplacian operators.

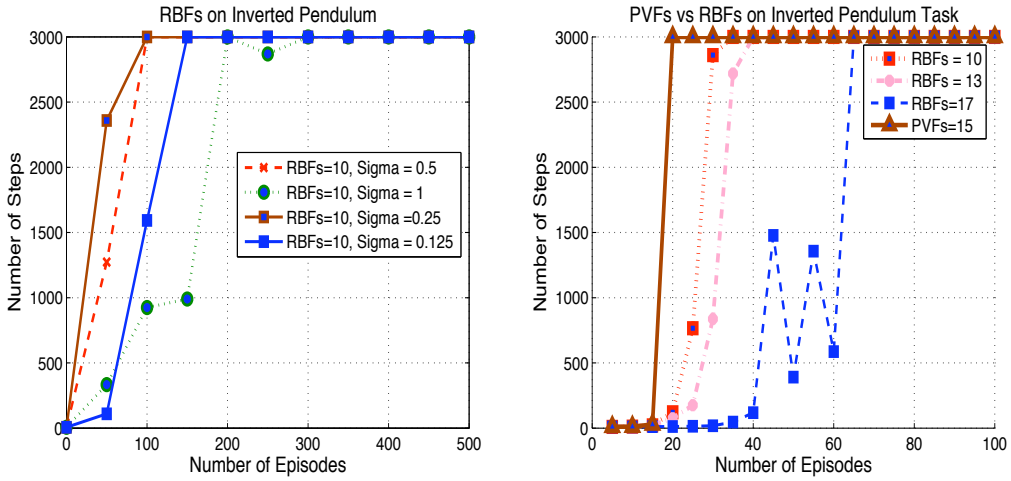


Figure 25: Left: The performance of a linear parametric RBF architecture is analyzed for varying kernel widths in the inverted pendulum domain. Right: A comparison of 15 PVFs with several choices of RBFs on the inverted pendulum task, focusing on the initial 100 episodes averaged over 100 runs.

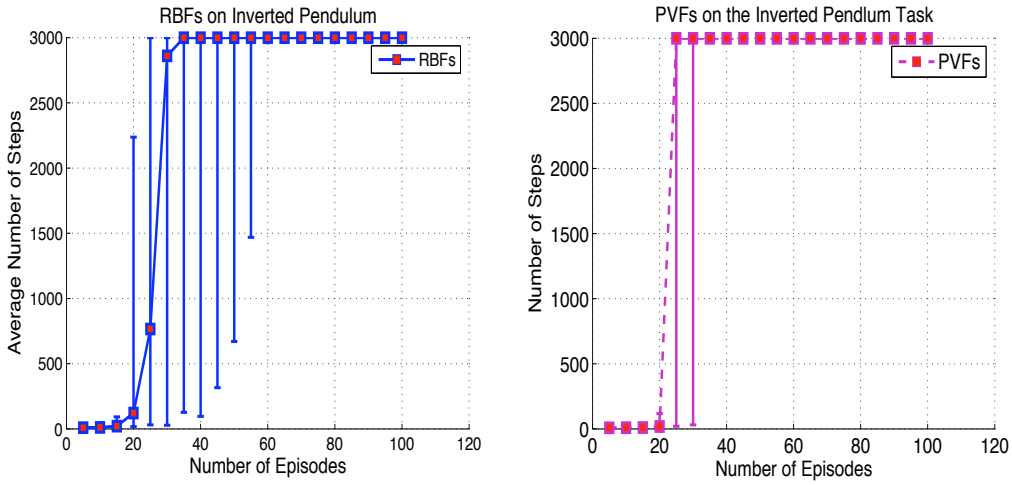


Figure 26: This plot shows that PVFs (right) have significantly less variance compared to RBFs (left) in the inverted pendulum task. Both plots show median-averaged number of steps the pole was balanced over 100 learning runs.

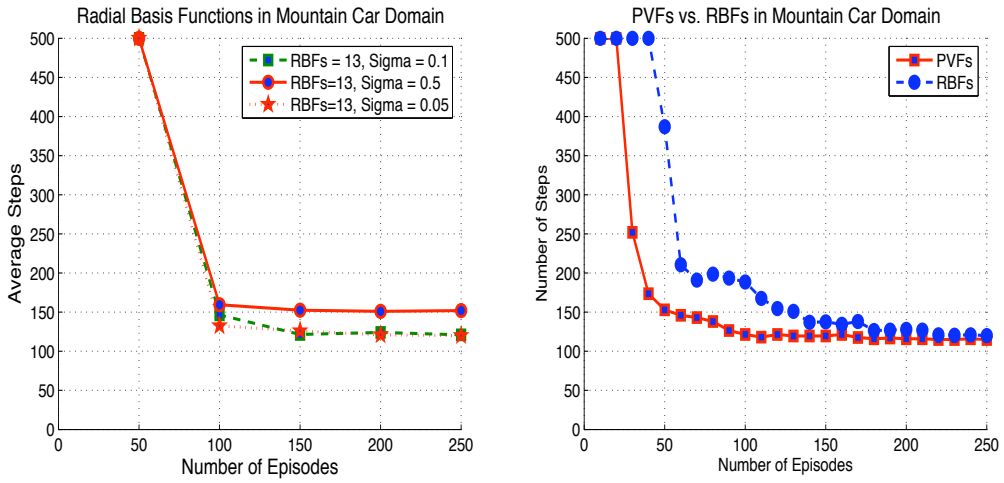


Figure 27: Left: The performance of a linear parametric RBF architecture is analyzed for varying kernel widths in the mountain car domain. Right: A comparison of 25 PVFs and 13 RBFs on the mountain car task. Higher number of RBFs produced worse results.

Number of RBFs	Inverted Pendulum RBF Parameters
10	3 x-axis, 3 y-axis, $\sigma = 1, 0.5, 0.25, 0.125$
13	4 x-axis, 3 y-axis, $\sigma = 0.25$
17	4 x-axis, 4 y-axis, $\sigma = 0.25$
Number of RBFs	Mountain Car RBF Parameters
13	4 x-axis, 3 y-axis, $\sigma = 0.5, 0.1, 0.05$

Table 2: RBF parameter settings for inverted pendulum and mountain car experiments.

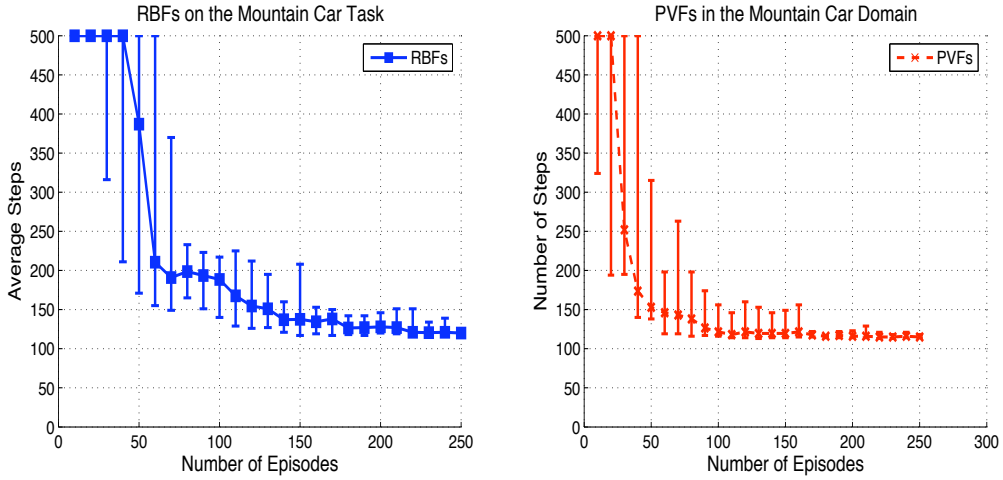


Figure 28: Left: The variance in performance of a linear parametric RBF architecture is analyzed over 30 learning runs in the mountain car domain. Right: Variance across 30 runs for PVFs in the mountain car task.

8. Related Work

In this section, we briefly review related work, beginning with methods for approximating value functions, followed by a description of past research on representation learning, concluding with a short summary of recent work on manifold and spectral learning.

8.1 Value Function Approximation

Value function approximation has been studied by many researchers. Bertsekas and Tsitsiklis (1996) provide an authoritative review. Parametric approaches using linear architectures, such as radial basis functions (Lagoudakis and Parr, 2003), and nonlinear architectures, such as neural networks (Tesauro, 1992), have been extensively explored. However, most approaches (with notable exceptions discussed below) are based on a fixed parametric architecture, and a parameter estimation method is used to approximate value functions, such as temporal-difference learning (Sutton and Barto, 1998; Tsitsiklis and Van Roy, 1997), least squares projection (Bradtke and Barto, 1996; Boyan, 1999; Nedic and Bertsekas, 2003; Lagoudakis and Parr, 2003), and linear programming (de Farias, 2003; Guestrin et al., 2003). There has also been significant work on non-parametric

methods for approximating value functions, including nearest neighbor methods (Gordon, 1995) and kernel density estimation (Ormoneit and Sen, 2002). Although our approach is also non-parametric, it differs from kernel density estimation and nearest neighbor techniques by extracting a distance measure through modeling the underlying graph or manifold. Non-parametric kernel methods based on Hilbert spaces have also been applied to value function approximation, including support vector machines (Dietterich and Wang, 2002) and Gaussian processes (Engel et al., 2003; Rasmussen and Kuss, 2004). Note that in this approach, the kernel is largely hand-engineered, such as the Gaussian kernel. Our approach can be viewed as extending this work using an automatically generated data-dependent graph or diffusion kernel (Kondor and Vert, 2004). There are interesting connections between the graph Laplacian matrix and covariance matrices (Ben-Chen and Gotsman, 2005).

8.2 Representation Learning

The problem of learning representations has a long history in AI. Amarel (1968) was an early pioneer, advocating the study of representation learning through global state space analysis. Amarel's ideas motivated much subsequent research on representation discovery (Subramanian, 1989; Utgoff and Stracuzzi, 2002), and many methods for discovering global state space properties like "bottlenecks" and "symmetries" have been studied (McGovern, 2002; Ravindran and Barto, 2003; Mannor et al., 2004). However, this past research lacked a formal framework showing how the geometrical analysis of a state space analysis can be transformed into representations for approximating value functions, a hallmark of our approach.

There have been several attempts at overcoming the limitations of traditional function approximators, such as radial basis functions. In particular, it has been recognized that Euclidean smoothing methods do not incorporate geometric constraints intrinsic to the environment: states close in Euclidean distance may be far apart on the manifold. Dayan (1993) proposed the idea of building *successor representations*. While this approach was restricted to policy evaluation in simple discrete MDPs, and did not formally build on manifold or graph-theoretic concepts, the idea of constructing representations that are faithful to the underlying dynamics of the MDP was a key motivation underlying this work. Drummond (2002) also pointed out the nonlinearities that value functions typically exhibit, and used techniques from computer vision to detect nonlinearities. Neither of these studies formulated the problem of value function approximation as approximating functions on a graph or manifold, and both were restricted to discrete MDPs. There have been several attempts to dynamically allocate basis functions to regions of the state space based on the nonuniform occupancy probability of visiting a region (e.g., Kretchmar and Anderson, 1999), but these methods do not construct the basis functions adaptively. Finally, there has also been research on finding common structure among the set of value functions on a given state space, where only the goal location is changed (Foster and Dayan, 2002), assuming a probabilistic generative (mixture) model of a value function, and using maximum likelihood estimation techniques. Proto-value functions can be viewed similarly as the building block of the set of value functions on a given state space, except that they are constructed without the need to make such parametric assumptions.

8.3 Manifold and Spectral Learning

This research also builds on recent work on manifold and spectral learning, including diffusion maps (Coifman et al., 2005a,b,c), ISOMAP (Tenenbaum et al., 2000), LLE (Roweis and Saul, 2000), and Laplacian eigenmaps (Belkin and Niyogi, 2004; Jones et al., 2007). One major difference is that

these methods have largely (but not exclusively) been applied to nonlinear dimensionality reduction and semi-supervised learning on graphs, whereas our work focuses on approximating (real-valued) value functions on graphs. Although related to regression on graphs (Niyogi et al., 2003), the problem of value function approximation is fundamentally different: the set of target values is not known a priori, but must be inferred through an iterative process of computing an approximate fixed point of the Bellman backup operator, and projecting these iterates onto subspaces spanned by the basis functions. Furthermore, value function approximation introduces new challenges not present in supervised learning or dimensionality reduction: the set of samples is not specified a priori, but must be collected through *active* exploration of the state space.

9. Discussion and Future Research

The fundamental contribution of this paper is an algorithmic framework called RPI that combines the learning of representations (basis functions) and policies. RPI is based on some specific design choices, and we have naturally restricted our description of the framework to the simplest settings. The scope of RPI can easily be extended to more general situations. Many extensions of the framework are being actively explored, and we briefly summarize these ongoing investigations.

9.1 Analysis of RPI and Variants

RPI is based on a two-phased procedure, where basis functions are learned from spectral analysis of trajectories generated by simulating policies, and improved policies are found by a control learning algorithm using the newly generated basis functions. Section 7 evaluated both the *off-policy* setting, where basis functions were learned purely from random walks, as well as the *on-policy* setting, where additional samples were generated from newly learned improved policies and combined with the random-walk samples. In both approaches, a smaller subset of samples were extracted using a subsampling method described in Section 6.3. Many questions remain to be addressed about the specific properties of architectures like RPI as well as other related architectures that combine the learning of representation and behavior. We summarize some key issues that need to be addressed in future research:

- How can we modify the design of RPI, so that basis functions are learned *simultaneously* with the learning of policies? Recent work on Bellman-error basis functions (Keller et al., 2006; Petrik, 2007; Parr et al., 2007) suggests an alternative approach where basis functions are learned *in-situ* during the policy evaluation phase itself, by explicitly modeling the error in approximating the value function using the Bellman residual. In such approaches, the basis functions generated are very sensitive to a specific reward function, whose shapes reflect the error in approximating a given value function. Can such in-situ basis-function learners be combined with offline approaches such as RPI, where basis functions are generated using a more global analysis of the state space as a whole, to yield more robust provably optimal control learners? For example, Petrik (2007) proposes combining reward-specific Krylov bases with Laplacian bases as a way of integrating localized high-frequency reward-specific bases with more global long-term eigenvector bases such as PVFs. We discuss below other approaches for integrating local vs. global basis functions, such as diffusion wavelets.
- Is it possible to specify optimality metrics for basis function generation, similar to metrics used in control learning such as maximizing the cumulative long-term discounted sum of

rewards (or average reward)? How can the cost of learning basis functions be amortized over multiple problems? Does this tradeoff suggest a way to balance the learning of reward-based and reward-independent basis functions?

- What are the pros and cons of off-policy sampling vs. on-policy sampling in designing the outer loop of RPI? For example, is it possible to construct problems where on-policy sampling results in oscillation, as samples are increasingly generated from policies that visit increasingly restricted portions of the state space? In the experiments in Section 7, newly generated samples are combined with previously generated samples to avoid overfitting basis functions to narrow regions of the state space, but this strategy may be computationally expensive in large MDPs.

- Under what assumptions can RPI be shown to converge? It is clear from the experiments presented in Section 7 that RPI converges extremely quickly in problems like the inverted pendulum, whereas in other problems such as the mountain car or Acrobot, convergence takes significantly longer. Can we characterize more formally conditions on the underlying state (action) manifold under which RPI can be shown to reliably converge?

9.2 Combining Nonparametric Graph-based and Parametric Basis Functions

Proto-value functions are given information about the underlying state space manifold in terms of the underlying graph that captures non-local smoothness, whereas parametric bases generally make fairly broad uniformity assumptions about the underlying state space topology. It is reasonable to try to combine the graph-based approach with parametric methods, such as RBFs, to combine the advantages of the two approaches. For example, geodesic Gaussian kernels (Sugiyama et al., 2007) are based on learning a graph of the underlying MDP from random walks, and using the shortest path between any two states as the distance metric for a set of RBFs defined on the graph. The Gaussian exponential term in the RBF approximator can be shown to be the solution of a *diffusion kernel* (Kondor and Lafferty, 2002) or *heat kernel* (Chung, 1997) defined by a differential equation, whose solution can be expressed as a matrix exponential function of the graph Laplacian. Interestingly, matrix exponentials can serve as *generators* of manifold structures called *Lie groups* (Baker, 2001), of which some interesting varieties are rotation and motion groups discussed in more detail in Section 9.8. The Laplacian can also be viewed as an inverse covariance matrix (Ben-Chen and Gotsman, 2005), defining a smoothing prior on the space of functions, which can be contrasted with other priors such as Gaussian processes (Rasmussen and Kuss, 2004; Rasmussen and Williams, 2006). It is possible to combine the graph Laplacian smoothness functional with other parametric smoothing kernels using manifold regularization methods (Belkin et al., 2006).

9.3 Proto-Value Functions From Directed Graphs

In this paper, we constructed PVFs by diagonalizing a symmetric diffusion operator on an undirected graph. This approach can be readily generalized to more elaborate diffusion models which capture asymmetry of actions using *directed* graphs. In particular, PVFs can be constructed by diagonalizing the directed graph Laplacian (Chung, 2005), which is defined as

$$L_D = D_\phi - \frac{D_\phi P + P^T D_\phi}{2},$$

where D_ϕ is a diagonal matrix whose entries are given by $\phi(v)$, the *Perron* vector or leading eigenvector associated with the spectral radius of the transition matrix P specifying the directed random walk on G. For a strongly connected directed graph G, the Perron-Frobenius theorem can be applied to show that the transition matrix is irreducible and non-negative, and consequently the leading eigenvector associated with the largest (real) eigenvalue must have all positive components $\phi(v) > 0$. In an initial study (Johns and Mahadevan, 2007), we have found that the directed graph Laplacian can result in a significant improvement over the undirected Laplacian in some discrete and continuous MDPs. For example, in a modified two-room task where there are two "one-way" doors leading from one room to the other, PVFs constructed from the directed Laplacian significantly outperformed the non-directional PVFs constructed from undirected graphs for certain locations of the goal state (e.g., near one of the one-way doors). Directed PVFs also appeared to yield improvements in some continuous control tasks, such as the inverted pendulum.

9.4 Scaling PVFs by Kronecker Product Factorization

Proto-value functions can be made more compact using a variety of sparsification methods, some of which have been explored in the literature on kernel methods. These include matrix sparsification techniques (Achlioptas et al., 2002), low-rank approximation techniques (Frieze et al., 1998), graph partitioning (Karypis and Kumar, 1999), and Kronecker product approximation (Van Loan and Pitsianis, 1993). We discuss one specific approach that we have implemented for continuous MDPs, and that has given us promising results (Johns et al., 2007). A random walk weight matrix $P_r = D^{-1}W$ constructed through the methods specified above in Section 6 can be approximated by a Kronecker product of two smaller stochastic matrices P_a and P_b, which minimizes the Frobenius norm of the error:

$$f(P_a, P_b) = \min \left(\| P_r - P_a \otimes P_b \|_F \right).$$

We have implemented the approach specified in Van Loan and Pitsianis (1993) to construct two smaller stochastic matrices whose Kronecker product approximates the original random walk matrix P_r. [20] To ensure that the decomposed matrices are not only stochastic, but also diagonalizable, which the Kronecker factorization procedure does not guarantee, we incorporate an additional step using the Metropolis Hastings algorithm (Billera and Diaconis, 2001) to make the smaller matrices P_a and P_b *reversible*. Then, the PVFs for the original random walk matrix P_r can be approximated as the Kronecker product of the PVFs of the factorized smaller reversible matrices P_a^r and P_b^r (since the smaller matrices are reversible, they can also be symmetrized using the normalized Laplacian, which makes the numerical task of computing their eigenvectors much simpler). In an initial study (Johns et al., 2007), we have been able to significantly reduce the size of the random walk weight matrices for the inverted pendulum, mountain car, and the Acrobot tasks with modest loss in performance compared to the full matrix. For example, in the Acrobot task, the original basis matrix is compressed by a factor of $36:1$, which resulted in a policy slightly worse than the original larger basis matrix. One important point to emphasize is that the full basis matrix never needs to be stored

20. It is important to distinguish this approach from the Kronecker decomposition approach described in Section 5, where the factorization was not an approximation, but an exact decomposition assuming the overall state space was a product space. Here, the Kronecker factorization can be applied to arbitrary weight matrices, but the decomposition is an approximation.

or computed in constructing the state embeddings from the smaller matrices. The factorization can be carried out recursively as well, leading to a further reduction in the size of the basis matrices.

9.5 Multiscale Diffusion Wavelet Bases

In this paper, proto-value functions were constructed by diagonalization, that is by finding eigenvectors, of a symmetrized diffusion operator such as the Laplacian on an undirected graph. Formally, such eigenvectors are essentially global *Fourier* bases and their properties have been extensively studied in Euclidean spaces (Mallat, 1989). One well-known limitation of global Laplacian bases is that they are poor at representing piecewise linear (value) functions. We have extended the approach presented in this paper to construct multiscale diffusion bases, using the recently proposed *diffusion wavelet* framework (Coifman and Maggioni, 2006; Bremer et al., 2006). Diffusion wavelets provide an interesting alternative to global Fourier eigenfunctions for value function approximation, since they encapsulate all the traditional advantages of wavelets (Mallat, 1989): basis functions have compact support, and the representation is inherently hierarchical since it is based on multi-resolution modeling of processes at different spatial and temporal scales. In Mahadevan and Maggioni (2006) we compare the performance of diffusion wavelet bases and Laplacian bases on a variety of simple MDPs. In Maggioni and Mahadevan (2006), we present an efficient direct method for policy evaluation by using the multiscale diffusion bases to invert the Bellman matrix $I - \gamma P^{\pi}$. We are currently exploring faster methods of constructing multiscale diffusion wavelet bases.

9.6 Policy and Reward-Sensitive PVFs

In the PVF framework presented above, basis functions are constructed without taking rewards into account. This restriction is not intrinsic to the approach, and reward or policy information when available can easily be incorporated into the construction of PVFs. One recent approach studied in Petrik (2007) assumes that the reward function R^{π} and policy transition matrix P^{π} are known, and combines Laplacian PVF bases with *Krlyov bases*. This approach is restricted to *policy evaluation*, which consists of solving the system of linear equations

$$(I - \gamma P^{\pi})V^{\pi} = R^{\pi}.$$

This equation is of the well-studied form $Ax = b$, and Krylov bases are used extensively in the solution of such linear systems of equations. The Krylov space is defined as the space spanned by the vectors

$$\left(b \; Ab \; A^2 b \; \ldots A^{m-1} b\right).$$

The use of Krylov bases to compress the *belief* space of a partially-observable Markov decision process (POMDP) is investigated in Poupart and Boutilier (2003), which explores how to exploit the factored representation of the transition dynamics specified by a dynamic Bayes net. As discussed earlier, Keller et al. (2006) and Parr et al. (2007) both investigate constructing reward-sensitive basis functions by explicitly estimating the error in approximating the value function using the *Bellman residual*. These approaches can also be combined with Laplacian PVFs in several ways, for example by combining low-frequency Laplacian bases with the more high-frequency reward-specific Krylov bases, or by using the estimated Bellman residuals to set the weights of the graph.

A more direct way to incorporate reward-sensitive information into PVFs is to modify the weight matrix W to take into account the *gradient* of the value function to be approximated. Formally, this approach is similar to estimating a function by knowing not only its values at sample points, but also its gradient. Of course, any errors in the estimation of such gradients will then be reflected in the weight matrix, and such an approach is not also without some drawbacks. While making bases sensitive to rewards can lead to superior results, if the reward function or policy is modified, reward-sensitive basis functions would need to be re-learned. In comparison, reward-independent bases may be more generally applicable across different tasks.

9.7 Learning State Action PVFs

In our paper, the basis functions $\phi(s)$ are originally defined over states, and then extended to state action pairs $\phi(s,a)$ by duplicating the state embedding $|A|$ times and "zeroing" out elements of the state-action embedding corresponding to actions not taken. That is, $\phi(s,a) = \phi(s) \otimes I_a$ where I_a is a vector indicator function for action a (all elements of I_a are 0 except for the chosen action). This construction is somewhat wasteful, especially in domains where the number of actions can vary significantly from one state to another. We have recently implemented PVFs on *state action* graphs, where vertices represent state action pairs. Thus, the pair (s,a) is connected by an edge to the pair (s',a') if action a in state s resulted in state s' from which action a' was next attempted. State action graphs are naturally highly directional, and we used the directed Laplacian to compute basis functions over state action graphs. Our initial results (Osentoski and Mahadevan, 2007) show that state action bases can significantly improve the performance of PVFs in discrete MDPs.

9.8 Group-Theoretic Methods for Constructing Proto-Value Functions

As we discussed earlier in Section 3.6, there is a long tradition in mathematics of constructing representations that are invariant under a group operator, including Fourier and wavelet transforms (Mallat, 1989). One interesting extension is to exploit the properties of linear (matrix) representations of groups to construct compact PVFs. In particular, many of the continuous MDPs we studied, including the inverted pendulum and the Acrobot, define continuous manifolds that have been extensively studied in mathematics (Baker, 2001) and robotics (Lavalle, 2006). In addition, the product spaces described in Section 5 generate graphs with large automorphism groups, which can be exploited in reducing the size of their associated Laplacian eigenspaces.

To make this more concrete, consider the set of points generated by a rotation of a rigid object in $\mathbb{R}^2$. This manifold can be modeled as a *Lie* (matrix) group called $SO(2)$, which stands for special orthogonal group of order 2. This rotation group is defined by all orthogonal matrices whose determinant is 1. Rotations and translations in $\mathbb{R}^2$ can be represented by another Lie group called $SE(2)$ (special Euclidean group). Finally, problems like the Acrobot task are instances of *kinematic chains*, which can be modeled by products of $SE(2)$ matrices. These groups generalize correspondingly to higher dimensions. Note that $SE(n)$ groups are non-Abelian because rotations do not commute with translations—the order matters! A detailed overview of Fourier analysis on non-Abelian groups is given in Chirikjian and Kyatkin (2001), with an emphasis on rotation and motion groups useful in robotics. An interesting direction for future work is to exploit such group representations to construct compact PVFs.

9.9 Proto-Value Functions for Semi-Markov Decision Processes

Proto-value functions provide a way of constructing function approximators for hierarchical reinforcement learning (Barto and Mahadevan, 2003), as well as form a theoretical foundation for some recent attempts to automate the learning of task structure in hierarchical reinforcement learning, by discovering "symmetries" or "bottlenecks" (McGovern, 2002; Ravindran and Barto, 2003; Mannor et al., 2004; Şimşek et al., 2005). In particular, Şimşek et al. (2005) use the second eigenvector of the discrete graph Laplacian operator $I - D^{-1}W$ to find bottlenecks in (undirected) state space graphs. Ravindran and Barto (2003) explore the use of group homomorphisms on state action spaces to abstract semi-MDPs, which can be combined with PVFs as a way of solving large SMDPs.

Another direction that we have begun exploring is to construct PVFs for temporally extended actions, such as "exiting a room". These temporally extended actions result in longer "distal" edges connecting non-adjacent vertices (such as the vertices corresponding to interior states in a room with those representing the "door" state). Our initial results reported in Osentoski and Mahadevan (2007) suggest that constructing PVFs over state-action graphs using these distal edges can significantly improve the performance over PVFs constructed over state graphs with only primitive actions.

9.10 Theoretical Analysis

Theoretical guarantees on the efficiency of proto-value functions in approximating value functions are being investigated. Some results follow immediately from the construction of proto-value functions. For example, it can be shown easily that the approximation produced by projecting a given function on a graph on the subspace spanned by the smallest k proto-value functions produces *globally* the smoothest approximation taking the graph or manifold into account (Mahadevan and Maggioni, 2006). There are also classical results on the efficiency of Fourier bases for approximating smooth functions in a Sobolev space (Mallat, 1989), which can be carried over to the discrete case of graphs. Belkin and Niyogi (2005) and Hein et al. (2007) study the sampling conditions under which the various graph Laplacians converge to the Laplace-Beltrami operator on the underlying manifold. For example, Hein et al. (2007) show that under non-uniform sampling conditions, the random walk Laplacian converges to a weighted Laplace-Beltrami operator. These results need to be combined with exploration techniques to investigate the conditions under which these sampling conditions can be met in the context of MDPs. We are also currently exploring the stability of the subspaces defined by proto-value functions using the tools of matrix perturbation theory (Stewart and Sun, 1990; Sato, 1995), which quantifies the degree to which small perturbations of (positive definite) matrices lead to bounded changes in the spectrum and eigenspace as well.

9.11 Transfer Across Tasks

Proto-value functions are learned not from rewards, but from the topology of the underlying state space (in the "off-policy" case). Consequently, they suggest a solution to the well-known problem of transfer in reinforcement learning (Mahadevan, 1992; Sherstov and Stone, 2005). One key advantage of proto-value functions is that they provide a theoretically principled approach to transfer, which respects the underlying state (action) space manifold. We have recently begun to investigate a framework called *proto-transfer* learning to explore the transfer of learned representations from one task to another (in contrast to transferring learned policies) (Ferguson and Mahadevan, 2006).

10. Summary

This paper describes a novel spectral framework for learning both representation and control in Markov decision processes, where basis functions called proto-value functions are constructed by diagonalization of a symmetric diffusion operator learned from samples collected during a random walk of the underlying state space. Proto-value functions can be defined in several ways: this paper focused principally on using the graph Laplacian on undirected graphs. Eigenfunctions of the graph Laplacian provide geometrically customized basis functions that capture large-scale properties such as bottlenecks and symmetries. Projections of a value function onto the eigenfunctions of the graph Laplacian provide the globally smoothest approximation that respects the underlying graph or manifold. A general algorithmic framework called representation policy iteration (RPI) was presented consisting of three components: sample collection, basis function construction, and control learning. A specific instance of RPI was described that uses the least-squares policy iteration (LSPI) method as the underlying control learner. Several directions for scaling the approach were described, including Kronecker sum matrix factorization for large factored MDPs, and sparse sampling combined with the Nystrom̈ interpolation method for continuous MDPs. Detailed experimental results were provided using benchmark discrete and continuous MDPs, which evaluated the effectiveness of the proto-value function approach, and compared their performance to handcoded parametric function approximators, such as polynomials and radial basis functions. Many extensions of the proposed framework are possible, and a few promising directions were elaborated.

Acknowledgments

We are indebted to the anonymous reviewers and the action editor for their detailed comments on an earlier draft of this paper. The first author would like to thank members of the PVF group—Kimberly Ferguson, Jeff Johns, Vimal Mathew, Sarah Osentoski, Marek Petrik, Ilya Scheidwasser, Andrew Stout, and Chang Wang—for their valuable input. We are also grateful to Kimberly Ferguson, Jeff Johns, and Sarah Osentoski for carrying out some of the experiments. We would like to thank Andrew Barto and other members of the Autonomous Learning Laboratory for their feedback. Support for this research was provided in part by the National Science Foundation under grants IIS-0534999 and DMS-0650413.

References

D. Achlioptas, F. McSherry, and B. Scholkopff. Sampling techniques for kernel methods. In *Proceedings of the* 14^{th} *International Conference on Neural Information Processing Systems (NIPS)*, pages 335–342. MIT Press, 2002.

S. Amarel. On representations of problems of reasoning about actions. In Donald Michie, editor, *Machine Intelligence 3*, volume 3, pages 131–171. Elsevier/North-Holland, 1968.

J. Bagnell and J. Schneider. Covariant policy search. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1019–1024, 2003.

A. Baker. *Matrix Groups: An Introduction to Lie Group Theory*. Springer, 2001.

C. Baker. *The Numerical Treatment of Integral Equations*. Oxford: Clarendon Press, 1977.

A. Barto and S. Mahadevan. Recent advances in hierarchical reinforcement learning. *Discrete Event Systems Journal*, 13:41–77, 2003.

M. Belkin and P. Niyogi. Towards a theoretical foundation for Laplacian-based manifold methods. In *Proceedings of the International Conference on Computational Learning Theory (COLT)*, pages 486–500, 2005.

M. Belkin and P. Niyogi. Semi-supervised learning on Riemannian manifolds. *Machine Learning*, 56:209–239, 2004.

M. Belkin, P. Niyogi, and V. Sindhwani. Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *Journal of Machine Learning Research*, 7:2399–2434, 2006.

S. Belongie, C. Fowlkes, F. Chung, and J. Malik. Spectral partitioning with indefinite kernels using the Nyström extension. In *Proceedings of the* 7^{th} *European Conference on Computer vision*, pages 531–542, 2002.

M. Ben-Chen and C. Gotsman. On the optimality of spectral compression of mesh data. *ACM Transactions on Graphics*, 24(1), 2005.

A. Bernasconi. *Mathematical Techniques for Analysis of Boolean Functions*. PhD thesis, University of Pisa, 1998.

D. Bertsekas and J. Tsitsiklis. *Neuro-Dynamic Programming*. Athena Scientific, Belmont, Massachusetts, 1996.

L. Billera and P. Diaconis. A geometric interpretation of the Metropolis-Hasting algorithm. *Statistical Science*, 16:335–339, 2001.

C. Boutilier, T. Dean, and S. Hanks. Decision-theoretic planning: Structural assumptions and computational leverage. *Journal of Artificial Intelligence Research*, 11:1–94, 1999.

J. A. Boyan. Least-squares temporal difference learning. In *Proceedings of the 16th International Conference on Machine Learning*, pages 49–56. Morgan Kaufmann, San Francisco, CA, 1999.

S. Bradtke and A. Barto. Linear least-squares algorithms for temporal difference learning. *Machine Learning*, 22:33–57, 1996.

J. Bremer, R. Coifman, M.Maggioni, and A. Szlam. Diffusion wavelet packets. *Applied and Computational Harmonic Analysis*, 21(1):95–112, July 2006.

G. Chirikjian and A. Kyatkin. *Engineering Applications of Noncommutative Harmonic Analysis*. CRC Press, 2001.

T. Chow. The Q-spectrum and spanning trees of tensor products of bipartite graphs. *Proceedings of the American Mathematical Society*, 125(11):3155–3161, 1997.

F. Chung. *Spectral Graph Theory*. Number 92 in CBMS Regional Conference Series in Mathematics. American Mathematical Society, 1997.

F Chung. Laplacians and the Cheeger inequality for directed graphs. *Annals of Combinatorics*, 9 (1):1–19, April 2005.

F. Chung and S. Sternberg. Laplacian and vibrational spectra for homogeneous graphs. *Journal of Graph Theory*, 16(6):605–627, 1992.

R. Coifman and M. Maggioni. Diffusion wavelets. *Applied and Computational Harmonic Analysis*, 21(1):53–94, July 2006.

R. Coifman, S. Lafon, A. Lee, M. Maggioni, B. Nadler, F. Warner, and S. Zucker. Geometric diffusions as a tool for harmonic analysis and structure definition of data. part i: Diffusion maps. *Proceedings of National Academy of Science.*, 102(21):7426–7431, May 2005a.

R. Coifman, S. Lafon, A. Lee, M. Maggioni, B. Nadler, Frederick Warner, and Steven Zucker. Geometric diffusions as a tool for harmonic analysis and structure definition of data. part ii: Multiscale methods. *Proceedings of the National Academy of Science*, 102(21):7432–7437, May 2005b.

R. Coifman, M. Maggioni, S. Zucker, and I. Kevrekidis. Geometric diffusions for the analysis of data from sensor networks. *Curr Opin Neurobiol*, 15(5):576–84, October 2005c.

D. Cvetkovic, M. Doob, and H. Sachs. *Spectra of Graphs: Theory and Application*. Academic Press, 1980.

D. Cvetkovic, P. Rowlinson, and S. Simic. *Eigenspaces of Graphs*. Cambridge University Press, 1997.

P. Dayan. Improving generalisation for temporal difference learning: The successor representation. *Neural Computation*, 5:613–624, 1993.

D. de Farias. The linear programming approach to approximate dynamic programming. In *Learning and Approximate Dynamic Programming: Scaling up to the Real World*. John Wiley and Sons, 2003.

F. Deutsch. *Best Approximation In Inner Product Spaces*. Canadian Mathematical Society, 2001.

T. Dietterich and X. Wang. Batch value function approximation using support vectors. In *Proceedings of Neural Information Processing Systems*. MIT Press, 2002.

P Drineas and M W Mahoney. On the Nyström method for approximating a Gram matrix for improved kernel-based learning. *J. Machine Learning Research*, 6:2153–2175, 2005.

C. Drummond. Accelerating reinforcement learning by composing solutions of automatically identified subtasks. *Journal of AI Research*, 16:59–104, 2002.

Y. Engel, S. Mannor, and R. Meir. Bayes meets Bellman: The Gaussian process approach to temporal difference learning. In *Proceedings of the 20th International Conference on Machine Learning*, pages 154–161. AAAI Press, 2003.

K. Ferguson and S. Mahadevan. Proto-transfer learning in Markov decision processes using spectral methods. In *International Conference on Machine Learning (ICML) Workshop on Transfer Learning*, 2006.

D. Foster and P. Dayan. Structure in the space of value functions. *Machine Learning*, 49:325–346, 2002.

A Frieze, R Kannan, and S Vempala. Fast Monte Carlo algorithms for finding low-rank approximations. In *Proceedings of the 39th annual IEEE symposium on foundations of computer science*, pages 370–378, 1998.

G. Gordon. Stable function approximation in dynamic programming. Technical Report CMU-CS-95-103, Department of Computer Science, Carnegie Mellon University, 1995.

A. Graham. *Kronecker Products and Matrix Calculations: With Applications*. Ellis Horwood, 1981.

C. Guestrin, D. Koller, and R. Parr. Max-norm projections for factored Markov decision processes. In *Proceedings of the 15th IJCAI*, 2001.

C. Guestrin, D. Koller, R. Parr, and S. Venkataraman. Efficient solution algorithms for factored MDPs. *Journal of AI Research*, 19:399–468, 2003.

D. Gurarie. *Symmetries and Laplacians: Introduction to Harmonic Analysis, Group Representations and Laplacians*. North-Holland, 1992.

M. Hein, J. Audibert, and U. von Luxburg. Graph Laplacians and their convergence on random neighborhood graphs. *Journal of Machine Learning Research*, 8:1325–1368, 2007.

J. Jackson. *The Harmonic Sieve: A Novel Application of Fourier Analysis to Machine Learning Theory and Practice*. PhD thesis, Carnegie-Mellon University, 1995.

J. Johns and S. Mahadevan. Constructing basis functions from directed graphs for value function approximation. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 385–392. ACM Press, 2007.

J. Johns, S. Mahadevan, and C. Wang. Compact spectral bases for value function approximation using Kronecker factorization. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, 2007.

P. Jones, M. Maggioni, and R. Schul. Universal parametrizations via eigenfunctions of the Laplacian and heat kernels. Submitted, 2007.

S. Kakade. A Natural Policy Gradient. In *Proceedings of Neural Information Processing Systems*. MIT Press, 2002.

G. Karypis and V. Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM Journal of Scientific Computing*, 20(1):359–392, 1999.

P. Keller, S. Mannor, and D Precup. Automatic basis function construction for approximate dynamic programming and reinforcement learning. In *Proceedings of the* 22^{nd} *International Conference on Machine Learning (ICML)*, pages 449–456. MIT Press, 2006.

D. Koller and R. Parr. Policy iteration for factored MDPs. In *Proceedings of the 16th Conference on Uncertainty in AI*, pages 326–334, 2000.

R. Kondor and J. Lafferty. Diffusion kernels on graphs and other discrete input spaces. In *Proceedings of the 19th International Conference on Machine Learning*, pages 315–322, 2002.

R. Kondor and R. Vert. Diffusion kernels. In *Kernel Methods in Computational Biology*. MIT Press, 2004.

R. Kretchmar and C. Anderson. Using temporal neighborhoods to adapt function approximators in reinforcement learning. In *International Work Conference on Artificial and Natural Neural Networks*, pages 488–496, 1999.

S. Kveton and M. Hauskrecht. Learning basis functions in hybrid domains. In *Proceedings of the Twentieth National Conference on Artificial Intelligence*, 2006.

J. Lafferty and G. Lebanon. Diffusion kernels on statistical manifolds. *Journal of Machine Learning Research*, 6:129–163, 2005.

S. Lafon. *Diffusion Maps and Geometric Harmonics*. PhD thesis, Yale University, Dept of Mathematics & Applied Mathematics, 2004.

M. Lagoudakis and R. Parr. Least-squares policy iteration. *Journal of Machine Learning Research*, 4:1107–1149, 2003.

J. C. Latombe. *Robot Motion Planning*. Kluwer Academic Press, 1991.

S. Lavalle. *Planning Algorithms*. Cambridge University Press, 2006.

J. M. Lee. *Introduction to Smooth Manifolds*. Springer, 2003.

M. Maggioni and S. Mahadevan. Fast direct policy evaluation using multiscale analysis of Markov Diffusion Processes. In *Proceedings of the 23rd international conference on Machine learning*, pages 601–608, New York, NY, USA, 2006. ACM Press.

S. Mahadevan. Proto-Value Functions: Developmental Reinforcement Learning. In *Proceedings of the International Conference on Machine Learning*, pages 553–560, 2005a.

S. Mahadevan. Enhancing transfer in reinforcement learning by building stochastic models of robot actions. In *Proceedings of the Ninth International Conference on Machine Learning, Aberdeen, Scotland*, pages 290–299, 1992.

S. Mahadevan. Representation policy iteration. In *Proceedings of the 21th Annual Conference on Uncertainty in Artificial Intelligence (UAI-05)*, pages 372–37. AUAI Press, 2005b.

S. Mahadevan and M. Maggioni. Value function approximation with Diffusion Wavelets and Laplacian Eigenfunctions. In *Proceedings of the Neural Information Processing Systems (NIPS)*. MIT Press, 2006.

S. Mahadevan, M. Maggioni, K. Ferguson, and S. Osentoski. Learning representation and control in continuous markov decision processes. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, 2006.

S. Mallat. A theory for multiresolution signal decomposition: The wavelet representation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 11(7):674–693, 1989. ISSN 0162-8828.

S. Mannor, I. Menache, A. Hoze, and U. Klein. Dynamic abstraction in reinforcement learning via clustering. In *International Conference on Machine Learning*, 2004.

A. McGovern. *Autonomous Discovery of Temporal Abstractions from Interactions with an Environment*. PhD thesis, University of Massachusetts, Amherst, 2002.

N. Menache, N. Shimkin, and S. Mannor. Basis function adaptation in temporal difference reinforcement learning. *Annals of Operations Research*, 134:215–238, 2005.

R. Munos. Error bounds for approximate value iteration. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pages 1006–1011, 2005.

R. Munos. Error bounds for approximate policy iteration. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 560–567, 2003.

A. Nedic and D. Bertsekas. Least-squares policy evaluation algorithms with linear function approximation. *Discrete Event Systems Journal*, 13, 2003.

A. Ng, M. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. In *NIPS*, 2002.

P. Niyogi, I. Matveeva, and M. Belkin. Regression and regularization on large graphs. Technical report, University of Chicago, Nov. 2003.

D. Ormoneit and S. Sen. Kernel-based reinforcement learning. *Machine Learning*, 49(2-3):161–178, 2002.

S. Osentoski and S. Mahadevan. Learning State Action Basis Functions for Hierarchical Markov Decison Processes. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 705–712, 2007.

R. Parr, C. Painter-Wakefiled, L. Li, and M. Littman. Analyzing feature generation for value function approximation. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 737–744, 2007.

R. Patrascu, P. Poupart, D. Schuurmans, C. Boutilier, and C. Guestrin. Greedy Linear Value Function Approximation for Factored Markov Decision Processes. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pages 285–291, 2002.

J. Peters, S. Vijaykumar, and S. Schaal. Reinforcement learning for humanoid robots. In *Proceedings of the Third IEEE-RAS International Conference on Humanoid Robots*, 2003.

M. Petrik. An analysis of Laplacian methods for value function approximation in MDPs. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 2574–2579, 2007.

P. Poupart and C. Boutilier. Value directed compression of POMDPs. In *Proceedings of the International Conference on Neural Information Processing Systems (NIPS)*, 2003.

M. L. Puterman. *Markov Decision Processes*. Wiley Interscience, New York, USA, 1994.

C. Rasmussen and M. Kuss. Gaussian Processes in Reinforcement Learning. In *Proceedings of the International Conference on Neural Information Processing Systems*, pages 751–759. MIT Press, 2004.

C. Rasmussen and C. Williams. *Gaussian Processes for Machine Learning*. MIT Press, 2006.

B. Ravindran and A. Barto. SMDP homomorphisms: An algebraic approach to abstraction in Semi-Markov Decision Processes. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence*, 2003.

S Rosenberg. *The Laplacian on a Riemannian Manifold*. Cambridge University Press, 1997.

S. Roweis and L. Saul. Nonlinear dimensionality reduction by local linear embedding. *Science*, 290:2323–2326, 2000.

B. Sallans and G. Hinton. Reinforcement learning with factored states and actions. *Journal of Machine Learning Research*, 5:1063–1088, 2004.

T. Sato. *Perturbation Theory for Linear Operators*. Springer, 1995.

B. Scholkopf and A. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, 2001.

A. Sherstov and P. Stone. Improving action selection in Markov Decision Processes via knowledge transfer. In *Proceedings of the Twentieth National Conference on Artificial Intelligence*, 2005.

J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE PAMI*, 22:888–905, 2000.

O. Şimşek, A. Wolfe, and A. Barto. Identifying useful subgoals in reinforcement learning by local graph partitioning. In *Proceedings of the Twenty-Second International Conference on Machine Learning*, pages 816–823, 2005.

G. Stewart and J. Sun. *Matrix Perturbation Theory*. Academic Press, 1990.

D. Subramanian. *A Theory of Justified Reformulations*. Ph.D. Thesis, Stanford University, 1989.

M. Sugiyama, H. Hachiya, C. Towell, and S. Vijaykumar. Value function approximation on non-linear manifolds for robot motor control. In *Proceedings of the IEEE Conference on Robots and Automation (ICRA)*, 2007.

R. Sutton and A. G. Barto. *An Introduction to Reinforcement Learning*. MIT Press, 1998.

A. Szlam, M. Maggioni, and R. Coifman. A general framework for adaptive regularization based on diffusion processes on graphs. Technical Report YALE/DCS/TR1365, Yale Univ, July 2006.

J. Tenenbaum, V. de Silva, and J. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290:2319–2323, 2000.

G. Tesauro. Practical issues in temporal difference learning. *Machine Learning*, 8:257–278, 1992.

M. Thornton, R. Drechsler, and D. Miller. *Spectral Methods for VLSI Design*. Kluwer Academic, 2001.

J. Tsitsiklis and B. Van Roy. An analysis of temporal-difference learning with function approximation. *IEEE Transactions on Automatic Control*, 42:674–690, 1997.

P. Utgoff and D. Stracuzzi. Many-layered learning. *Neural Computation*, 14:2497–2529, 2002.

C. Van Loan and N. Pitsianis. Approximation with Kronecker products. In *Linear Algebra for Large Scale and Real Time Applications*, pages 293–314. Kluwer Publications, 1993.

B. Van Roy. *Learning and Value Function Approximation in Complex Decision Processes*. PhD thesis, MIT, 1998.

C. Watkins. *Learning from Delayed Rewards*. PhD thesis, King's College, Cambridge, England, 1989.

C. Williams and M. Seeger. Using the Nyström method to speed up kernel machines. In *Proceedings of the International Conference on Neural Information Processing Systems*, pages 682–688, 2000.

Journal of Machine Learning Research 8 (2007) 2233-2264 Submitted 9/06; Revised 9/07; Published 10/07

Online Learning of Multiple Tasks with a Shared Loss

Ofer Dekel OFERD@CS.HUJI.AC.IL
School of Computer Science and Engineering
The Hebrew University
Jerusalem, 91904, Israel

Philip M. Long PLONG@GOOGLE.COM
Yoram Singer SINGER@GOOGLE.COM
Google Inc.
1600 Amphitheater Parkway
Mountain View, CA 94043, USA

Editor: Peter Bartlett

Abstract

We study the problem of learning multiple tasks in parallel within the online learning framework. On each online round, the algorithm receives an instance for each of the parallel tasks and responds by predicting the label of each instance. We consider the case where the predictions made on each round all contribute toward a common goal. The relationship between the various tasks is defined by a global loss function, which evaluates the overall quality of the multiple predictions made on each round. Specifically, each individual prediction is associated with its own loss value, and then these multiple loss values are combined into a single number using the global loss function. We focus on the case where the global loss function belongs to the family of absolute norms, and present several online learning algorithms for the induced problem. We prove worst-case relative loss bounds for all of our algorithms, and demonstrate the effectiveness of our approach on a large-scale multiclass-multilabel text categorization problem.

Keywords: online learning, multitask learning, multiclass multilabel classiifcation, perceptron

1. Introduction

Multitask learning is the problem of learning several related problems in parallel. In this paper, we discuss the multitask learning problem in the online learning context, and focus on the possibility that the learning tasks contribute toward a common goal. Our hope is that we can benefit from learning the tasks jointly, as opposed to learning each task independently.

For concreteness, we focus on the task of binary classification, and note that our algorithms and analysis can be adapted to regression and multiclass problems using ideas in Crammer et al. (2006). In the online multitask classification setting, we are faced with k separate online binary classification problems, which are presented to us in parallel. The online learning process takes place in a sequence of rounds. At the beginning of round t, the algorithm observes a set of k instances, one for each of the binary classification problems. The algorithm predicts the binary label of each of the k instances, and then receives the k correct labels. At this point, each of the algorithm's predictions is associated with a non-negative loss, and we use $\ell_t = (\ell_{t,1}, \ldots, \ell_{t,k})$ to denote the k-coordinate vector whose elements are the individual loss values associated with the respective tasks. Let $\mathcal{L} : \mathbb{R}^k \to \mathbb{R}_+$ be a predetermined *global loss* function, which is used to combine the individual

loss values into a single number, and define the global loss attained on round t to be $\mathcal{L}(\ell_t)$. At the end of this online round, the algorithm may use the k new labeled examples it has obtained to improve its prediction mechanism for the rounds to come. The goal of the learning algorithm is to suffer the smallest possible cumulative loss over the course of T rounds, $\sum_{t=1}^{T} \mathcal{L}(\ell_t)$.

The choice of the global loss function captures the overall consequences of the individual prediction errors, and therefore how the algorithm should prioritize correcting errors. For example, if $\mathcal{L}(\ell_t)$ is defined to be $\sum_{j=1}^{k} \ell_{t,j}$ then the online algorithm is penalized equally for errors on each of the tasks; this results in effectively treating the tasks independently. On the other hand, if $\mathcal{L}(\ell_t) = \max_j \ell_{t,j}$ then the algorithm is only interested in the worst mistake made on each round. We do not assume that the data sets of the various tasks are similar or otherwise related. Moreover, the examples presented to the algorithm for each of the tasks may come from completely different domains and may possess different characteristics. The multiple tasks are tied together by the way we define the objective of our algorithm.

In this paper, we focus on the case where the global loss function is an *absolute norm*. A norm $\|\cdot\|$ is a function such that $\|\mathbf{v}\| > 0$ for all $\mathbf{v} \neq 0$, $\|0\| = 0$, $\|\lambda \mathbf{v}\| = |\lambda| \|\mathbf{v}\|$ for all $\mathbf{v}$ and all $\lambda \in \mathbb{R}$, and which satisfies the triangle inequality. A norm is said to be absolute if $\|\mathbf{v}\| = \||\mathbf{v}|\|$ for all $\mathbf{v}$, where $|\mathbf{v}|$ is obtained by replacing each component of $\mathbf{v}$ with its absolute value. The most well-known family of absolute norms is the family of p-norms (also called L_p norms), defined for all $p \geq 1$ by

$$\|\mathbf{v}\|_p \;=\; \Big(\sum_{j=1}^{n} |v_j|^p \Big)^{1/p} \;\; .$$

A special member of this family is the L_∞ norm, which is defined to be the limit of the above when p tends to infinity, and can be shown to equal $\max_j |v_j|$. A less known family of absolute norms is the family of r-max norms. For any integer r between 1 and k, the r-max norm of $\mathbf{v} \in \mathbb{R}^k$ is the sum of the absolute values of the r absolutely largest components of $\mathbf{v}$. Formally, the r-max norm is

$$\|\mathbf{v}\|_{r\text{-max}} \;=\; \sum_{j=1}^{r} |v_{\pi(j)}| \;\; \text{where} \;\; |v_{\pi(1)}| \geq |v_{\pi(2)}| \geq \;\ldots\; \geq |v_{\pi(k)}| \;\; . \tag{1}$$

Note that both the L_1 norm and L_∞ norm are special cases of the r-max norm, as well as being p-norms. Actually, the r-max norm can be viewed as a smooth interpolation between the L_1 norm and the L_∞ norm, using Peetre's K-method of norm interpolation (see Appendix A for details).

Since the global loss functions we consider in this paper are norms, the global loss equals zero only if ℓ_t is itself the zero vector. Furthermore, decreasing any individual loss can only decrease the global loss function. Therefore, the simplest solution to our multitask problem is to learn each task individually, and minimize the global loss function implicitly. The natural question which is at the heart of this paper is whether we can do better than this. Our answer to this question is based on the following fundamental view of online learning. On every round, the online learning algorithm balances a trade-off between retaining the information it has acquired on previous rounds and modifying its hypothesis based on the new examples obtained on that round. Instead of balancing this trade-off individually for each of the learning tasks, we can balance it jointly, for all of the tasks. By doing so, we allow ourselves to make a big modification to one of the k hypotheses at the expense of the others. This additional flexibility enables us to directly minimize the specific global loss function we have chosen to use.

To motivate and demonstrate the practicality of our approach, we begin with a handful of concrete examples.

Multiclass Classification using the L_∞ Norm Assume that we are faced with a multiclass classification problem, where the size of the label set is k. One way of solving this problem is by learning k binary classifiers, where each classifier is trained to distinguish between one of the classes and the rest of the classes. This approach is often called the *one-vs-rest* method. If all of the binary classifiers make correct predictions, then one of these predictions should be positive and the rest should be negative. If this is the case, we can correctly predict the corresponding multiclass label. However, if one or more of the binary classifiers makes an incorrect prediction, we can no longer guarantee the correctness of our multiclass prediction. In this sense, a single binary mistake on round t is as bad as many binary mistakes on round t. Therefore, we should only care about the worst binary prediction on round t, and we can do so by choosing the global loss to be $\|\ell_t\|_\infty$.

Another example where the L_∞ norm comes in handy is the case where we are faced with a multiclass problem where the number of labels is huge. Specifically, we would like the running time and the space complexity of our algorithm to scale logarithmically with the number of labels. Assume that the number of different labels is 2^k, enumerate these labels from 0 to 2^k-1, and consider the k-bit binary representation of each label. We can solve the multiclass problem by training k binary classifiers, one for each bit in the binary representation of the label index. If all k classifiers make correct predictions, then we have obtained the binary representation of the correct multiclass label. As before, a single binary mistake is devastating to the multiclass classifier, and the L_∞ norm is the most appropriate means of combining the k individual losses into a global loss.

Vector-Valued Regression using the L_2 Norm Let us deviate momentarily from the binary classification setting, and assume that we are faced with multiple regression problems. Specifically, assume that our task is to predict the three-dimensional position of an object. Each of the three coordinates is predicted using an individual regressor, and the regression loss for each task is simply the absolute difference between the true and the predicted value on the respective axis. In this case, the most appropriate choice of the global loss function is the L_2 norm, which reduces the vector of individual losses to the Euclidean distance between the true and predicted 3-D targets. (Note that we take the actual Euclidean distance and not the squared Euclidean distance often minimized in regression settings).

Error Correcting Output Codes and the r-max Norm Error Correcting Output Codes (ECOC) is a technique for reducing a multiclass classification problem to multiple binary classification problems (Dietterich and Bakiri, 1995). The power of this technique lies in the fact that a correct multiclass prediction can be made even when a few of the binary predictions are wrong. The reduction is represented by a code matrix $M \in \{-1,+1\}^{s\times k}$, where s is the number of multiclass labels and k is the number of binary problems used to encode the original multiclass problem. Each row in M represents one of the s multiclass labels, and each column induces one of the k binary classification problems. Given a multiclass training set $\{(\mathbf{x}_i, y_i)\}_{i=1}^m$, with labels $y_i \in \{1,\ldots,s\}$, the binary problem induced by column j is to distinguish between the positive examples $\{(\mathbf{x}_i, y_i : M_{y_i,j} = +1\}$ and negative examples $\{(\mathbf{x}_i, y_i : M_{y_i,j} = -1\}$. When a new instance is observed, applying the k binary classifiers to it gives a vector of binary predictions, $\hat{\mathbf{y}} = (\hat{y}_1,\ldots,\hat{y}_k) \in \{-1,+1\}^k$. We then predict the multiclass label of this instance to be the index of the row in M which is closest to $\hat{\mathbf{y}}$ in Hamming distance.

Define the *code distance* of M, denoted by $d(M)$, to be the minimal Hamming distance between any two rows in M. It is straightforward to show that a correct multiclass prediction can be guaranteed as long as the number of binary mistakes made on this instance is less than $d(M)/2$. In other

words, making $d(M)/2$ binary mistakes is as bad as making more binary mistakes. Let $r = d(M)/2$. If the binary classifiers are trained in the online multitask setting, we should only be interested in whether the r'th largest loss is less than 1, which would imply that a correct multiclass prediction can be guaranteed. Regretfully, taking the r'th largest element of a vector (in absolute value) does not constitute a norm and thus does not fit in our setting. However, the r-max norm, defined in Equation (1), can serve as a good proxy.

In this paper, we present three families of online multitask algorithms. Each family includes algorithms for every absolute norm. All of the algorithms presented in this paper follow the general skeleton outlined in Figure 1. Specifically, all of our algorithms use linear threshold functions as hypotheses and an additive update rule. The first two families are multitask extensions of the Perceptron algorithm (Rosenblatt, 1958; Novikoff, 1962), while the third family is closely related to the Passive-Aggressive classification algorithm (Crammer et al., 2006). Incidentally, all of the algorithms presented in this paper can be easily transformed into kernel methods. For each algorithm, we prove a relative loss bound, namely, we show that the cumulative global loss attained by the algorithm is comparable to the cumulative loss attained by any fixed set of k linear hypotheses, even defined in hindsight.

Much previous work on theoretical and applied multitask learning has focused on how to take advantage of similarities between the various tasks (Caruana, 1997; Heskes, 1998; Evgeniou et al., 2005; Baxter, 2000; Ben-David and Schuller, 2003; Tsochantaridis et al., 2004); in contrast, we do not assume that the tasks are in any way related. Instead, we consider how to take account of shared consequences of errors. Kivinen and Warmuth (2001) generalized the notion of matching loss (Helmbold et al., 1999) to multi-dimensional outputs. Their construction enables analysis of algorithms that perform multi-dimensional regression by composing linear functions with a variety of transfer functions. It is not obvious how to directly use their work to address the problems that fall into our setting. An analysis of the L_∞ norm of prediction errors is implicit in some past work of Crammer and Singer (2001, 2003). The algorithms presented in Crammer and Singer (2001, 2003) were devised for multiclass categorization with multiple predictors (one per class) and a single instance. The present paper extends the multiclass prediction setting to a broader framework, and tightens the analysis. In contrast to the multiclass prediction setting, the prediction tasks in our setting are tied solely through a globally shared loss. When k, the number of multiple tasks, is set to 1, two of the algorithms presented in this paper as well as the multiclass algorithms in Crammer and Singer (2001, 2003) reduce to the PA-I algorithm, presented in Crammer et al. (2006). Last, we would like to mention in passing that a few learning algorithms for ranking problems decompose the ranking problem into a preference learning task over pairs of instances (see for instance Herbrich et al., 2000; Chapelle and Harchaoui, 2005). The ranking losses employed by such algorithms are typically defined as the sum over pair-based losses. Our setting generalizes such approaches for ranking learning by employing a shared loss which is defined through a norm over the individual pair-based losses.

This paper is organized as follows. In Section 2 we present our problem more formally and prove a key lemma which facilitates the analysis of our algorithms. In Section 3 we present our first family of algorithms, which works in the finite-horizon online setting. In Section 4 we extend the first family of algorithms to the infinite-horizon online setting. Then, in Section 5 we present our third family of algorithms, and show that it shares the analyses of both previous families. The third family of algorithms requires solving a small optimization problem on each online round, and is therefore called the *implicit update* family of algorithms. In Section 6 and Section 7 we describe

input: norm $\|\cdot\|$

initialize: $\mathbf{w}_{1,1} = \ldots = \mathbf{w}_{1,k} = (0,\ldots,0)$

for $t = 1,2,\ldots$

- receive $\mathbf{x}_{t,1},\ldots,\mathbf{x}_{t,k}$
- predict $\text{sign}(\mathbf{w}_{t,j}\cdot\mathbf{x}_{t,j})$ $[1 \le j \le k]$
- receive $y_{t,1},\ldots,y_{t,k}$
- calculate $\ell_{t,j} = \left[1 - y_{t,j}\mathbf{w}_{t,j}\cdot\mathbf{x}_{t,j}\right]_+$ $[1 \le j \le k]$
- suffer loss $\ell_t = \|(\ell_{t,1},\ldots,\ell_{t,n})\|$
- update $\mathbf{w}_{t+1,j} = \mathbf{w}_{t,j} + \tau_{t,j} y_{t,j}\mathbf{x}_{t,j}$ $[1 \le j \le k]$

Figure 1: A general skeleton for an online multitask classification algorithm. A concrete algorithm is obtained by specifying the values of $\tau_{t,j}$.

efficient algorithms for solving the implicit update in the case where the global loss is defined by the L_2 norm or the r-max norm. Experimental results are provided in Section 8 and we conclude the paper in Section 9 with a short discussion.

2. Online Multitask Learning with Additive Updates

We begin by presenting the online multitask classification setting more formally. We are presented with k online binary classification problems in parallel. The instances of each task are drawn from separate instance domains, and for concreteness we assume that the instances of task j are all vectors in $\mathbb{R}^{n_j}$. As stated in the previous section, online learning is performed in a sequence of rounds. On round t, the algorithm observes k instances, $(\mathbf{x}_{t,1},\ldots,\mathbf{x}_{t,k}) \in \mathbb{R}^{n_1}\times\ldots\times\mathbb{R}^{n_k}$. The algorithm maintains k separate classifiers in its internal memory, one for each of the multiple tasks, which are updated from round to round. Each of these classifiers is a margin-based linear predictor, defined by a weight vector. We denote the weight vector used on round t to define the j'th predictor by $\mathbf{w}_{t,j}$ and note that $\mathbf{w}_{t,j} \in \mathbb{R}^{n_j}$. The algorithm uses its classifiers to make k binary predictions, $\hat{y}_{t,1},\ldots,\hat{y}_{t,k}$, where $\hat{y}_{t,j} = \text{sign}(\mathbf{w}_{t,j}\cdot\mathbf{x}_{t,j})$. After making these predictions, the correct labels of the respective tasks, $y_{t,1},\ldots,y_{t,k}$, are revealed and each one of the predictions is evaluated. In this paper we focus on the hinge-loss function as the means of penalizing incorrect predictions. Formally, the loss associated with the j'th task is defined to be

$$\ell_{t,j} \;=\; \left[1 - y_{t,j}\mathbf{w}_{t,j}\cdot\mathbf{x}_{t,j}\right]_+ \;,$$

where $[a]_+ = \max\{0,a\}$. As previously stated, the global loss is then defined to be $\|\ell_t\|$, where $\|\cdot\|$ is a predefined absolute norm. Finally, the algorithm applies an update to each of the online hypotheses, and defines the vectors $\mathbf{w}_{t+1,1},\ldots,\mathbf{w}_{t+1,k}$. All of the algorithms presented in this paper use an additive update rule, and define $\mathbf{w}_{t+1,j}$ to be $\mathbf{w}_{t,j} + \tau_{t,j} y_{t,j}\mathbf{x}_{t,j}$, where $\tau_{t,j}$ is a scalar. The algorithms only differ from one another in the specific way in which $\tau_{t,j}$ is set. For convenience, we

denote $\boldsymbol{\tau}_t = (\tau_{t,1}, \ldots, \tau_{t,k})$. The general skeleton followed by all of our online algorithms is given in Figure 1.

A concept of key importance in this paper is the notion of *dual norms* (Horn and Johnson, 1985). Any norm $\|\cdot\|$ defined on $\mathbb{R}^n$, has a dual norm, also defined on $\mathbb{R}^n$, denoted by $\|\cdot\|^*$ and given by

$$\|\mathbf{u}\|^* \;=\; \max_{\mathbf{v}\in\mathbb{R}^n} \frac{\mathbf{u}\cdot\mathbf{v}}{\|\mathbf{v}\|} \;=\; \max_{\mathbf{v}\in\mathbb{R}^n:\,\|\mathbf{v}\|=1} \mathbf{u}\cdot\mathbf{v} \;\;. \tag{2}$$

The dual of a p-norm is itself a p-norm, and specifically, the dual of $\|\cdot\|_p$ is $\|\cdot\|_q$, where $\frac{1}{q}+\frac{1}{p}=1$. The dual of $\|\cdot\|_\infty$ is $\|\cdot\|_1$ and vice versa. In Appendix A we prove that the dual of $\|\mathbf{v}\|_{r\text{-max}}$ is

$$\|\mathbf{u}\|^*_{r\text{-max}} \;=\; \max\left\{\|\mathbf{u}\|_\infty, \frac{\|\mathbf{u}\|_1}{r}\right\} \;\;. \tag{3}$$

An important property of dual norms, which is an immediate consequence of Equation (2), is that for any $\mathbf{u},\mathbf{v}\in\mathbb{R}^n$ it holds that

$$\mathbf{u}\cdot\mathbf{v} \;\le\; \|\mathbf{u}\|^*\,\|\mathbf{v}\| \;\;. \tag{4}$$

If $\|\cdot\|$ is a p-norm then the above is known as Hölder's inequality, and specifically, if $p=2$ it is called the Cauchy-Schwartz inequality. Two additional properties which we rely on are that the dual of the dual norm is the original norm (see for instance Horn and Johnson, 1985), and that the dual of an absolute norm is also an absolute norm. As previously mentioned, to obtain concrete online algorithms, all that remains is to define the update weights $\tau_{t,j}$ for each task on each round. The different ways of setting $\tau_{t,j}$ discussed in this paper all share the following properties:

- *boundedness:* $\forall\, 1\le t\le T \quad \|\boldsymbol{\tau}_t\|^* \le C$ for some predefined parameter C
- *non-negativity:* $\forall\, 1\le t\le T,\ 1\le j\le k \quad \tau_{t,j}\ge 0$
- *conservativeness:* $\forall\, 1\le t\le T,\ 1\le j\le k \quad (\ell_{t,j}=0)\ \Rightarrow\ (\tau_{t,j}=0)$

Even before specifying the exact value of $\tau_{t,j}$, we can state and prove a powerful lemma which is the crux of our analysis. This lemma will motivate and justify our specific choices of $\tau_{t,j}$ throughout this paper.

Lemma 1 *Let $\{(\mathbf{x}_{t,j}, y_{t,j})\}_{1\le t\le T}^{1\le j\le k}$ be a sequence of T k-tuples of examples, where each $\mathbf{x}_{t,j}\in\mathbb{R}^{n_j}$, and each $y_{t,j}\in\{-1,+1\}$. Let $\mathbf{w}^\star_1,\ldots,\mathbf{w}^\star_k$ be arbitrary vectors where $\mathbf{w}^\star_j\in\mathbb{R}^{n_j}$, and define the hinge loss attained by $\mathbf{w}^\star_j$ on example $(\mathbf{x}_{t,j}, y_{t,j})$ to be $\ell^\star_{t,j} = \left[1 - y_{t,j}\mathbf{w}^\star_j\cdot\mathbf{x}_{t,j}\right]_+$. Let $\|\cdot\|$ be an arbitrary norm and let $\|\cdot\|^*$ denote its dual. Assume we apply an algorithm of the form outlined in Figure 1 to this sequence of examples, where the update weights satisfy the boundedness, non-negativity and conservativeness requirements. Then, for any $C>0$ it holds that*

$$\sum_{t=1}^{T}\sum_{j=1}^{k}\left(2\tau_{t,j}\ell_{t,j} - \tau^2_{t,j}\|\mathbf{x}_{t,j}\|_2^2\right) \;\le\; \sum_{j=1}^{k}\|\mathbf{w}^\star_j\|_2^2 \;+\; 2C\sum_{t=1}^{T}\|\ell^\star_t\| \;\;.$$

Under the assumptions of this lemma, our algorithm competes with a set of fixed linear classifiers, $\mathbf{w}^\star_1,\ldots,\mathbf{w}^\star_k$, which may even be defined in hindsight, after observing all of the inputs and their labels. The right-hand side of the bound is the sum of two terms, a complexity term $\sum_{j=1}^k\|\mathbf{w}^\star_j\|_2^2$ and a term

which is proportional to the cumulative loss of our competitor, $\sum_{t=1}^{T} \|\ell_t^\star\|$. The left hand side of the bound is the term

$$\sum_{t=1}^{T} \sum_{j=1}^{k} \left(2\tau_{t,j}\ell_{t,j} - \tau_{t,j}^2 \|\mathbf{x}_{t,j}\|_2^2\right) \ . \tag{5}$$

This term plays a key role in the derivation of all three families of algorithms presented in the sequel. Each choice of the update weights $\tau_{t,j}$ enables us to prove a different lower bound on Equation (5). Comparing this lower bound with the upper bound in Lemma 1 gives us a loss bound for the respective algorithm. The proof of Lemma 1 is given below.

Proof Define $\Delta_{t,j} = \|\mathbf{w}_{t,j} - \mathbf{w}_j^\star\|_2^2 - \|\mathbf{w}_{t+1,j} - \mathbf{w}_j^\star\|_2^2$. We prove the lemma by bounding $\sum_{t=1}^{T}\sum_{j=1}^{k}\Delta_{t,j}$ from above and from below. Beginning with the upper bound, we note that for each $1 \le j \le k$, $\sum_{t=1}^{T}\Delta_{t,j}$ is a telescopic sum which collapses to

$$\sum_{t=1}^{T} \Delta_{t,j} \ = \ \|\mathbf{w}_{1,j} - \mathbf{w}^\star\|_2^2 - \|\mathbf{w}_{T+1,j} - \mathbf{w}^\star\|_2^2 \ .$$

Using the facts that $\mathbf{w}_{1,j} = (0,\ldots,0)$ and $\|\mathbf{w}_{T+1,j} - \mathbf{w}^\star\|_2^2 \ge 0$ for all $1 \le j \le k$, we conclude that

$$\sum_{t=1}^{T} \sum_{j=1}^{k} \Delta_{t,j} \ \le \ \sum_{j=1}^{k} \|\mathbf{w}_j^\star\|_2^2 \ . \tag{6}$$

Turning to the lower bound, we note that we can consider only non-zero summands which actually contribute to the sum, namely $\Delta_{t,j} \neq 0$. Plugging the definition of $\mathbf{w}_{t+1,j}$ into $\Delta_{t,j}$, we get

$$\begin{aligned}
\Delta_{t,j} &= \|\mathbf{w}_{t,j} - \mathbf{w}_j^\star\|_2^2 - \|\mathbf{w}_{t,j} + \tau_{t,j} y_{t,j}\mathbf{x}_{t,j} - \mathbf{w}_j^\star\|_2^2 \\
&= \tau_{t,j}\left(-2y_{t,j}\mathbf{w}_{t,j}\cdot\mathbf{x}_{t,j} - \tau_{t,j}\|\mathbf{x}_{t,j}\|_2^2 + 2y_{t,j}\mathbf{w}_j^\star\cdot\mathbf{x}_{t,j}\right) \\
&= \tau_{t,j}\left(2(1 - y_{t,j}\mathbf{w}_{t,j}\cdot\mathbf{x}_{t,j}) - \tau_{t,j}\|\mathbf{x}_{t,j}\|_2^2 - 2(1 - y_{t,j}\mathbf{w}_j^\star\cdot\mathbf{x}_{t,j})\right) \ .
\end{aligned} \tag{7}$$

Since our update is conservative, $\Delta_{t,j} \neq 0$ implies that $\ell_{t,j} = 1 - y_{t,j}\mathbf{w}_{t,j}\cdot\mathbf{x}_{t,j}$. By definition, it also holds that $\ell_{t,j}^\star \ge 1 - y_{t,j}\mathbf{w}_j^\star\cdot\mathbf{x}_{t,j}$. Plugging these two facts into Equation (7) and using the fact that $\tau_{t,j}$ is non-negative gives

$$\Delta_{t,j} \ \ge \ \tau_{t,j}\left(2\ell_{t,j} - \tau_{t,j}\|\mathbf{x}_{t,j}\|_2^2 - 2\ell_{t,j}^\star\right) \ .$$

Summing the above over $1 \le j \le k$ gives

$$\sum_{j=1}^{k} \Delta_{t,j} \ \ge \ \sum_{j=1}^{k}\left(2\tau_{t,j}\ell_{t,j} - \tau_{t,j}^2\|\mathbf{x}_{t,j}\|_2^2\right) - 2\sum_{j=1}^{k}\tau_{t,j}\ell_{t,j}^\star \ . \tag{8}$$

Using Equation (4) we know that $\sum_{j=1}^{k}\tau_{t,j}\ell_{t,j}^\star \le \|\boldsymbol{\tau}_t\|^*\|\ell_t^\star\|$. From our assumption that $\|\boldsymbol{\tau}_t\|^* \le C$, we have that $\sum_{j=1}^{k}\tau_{t,j}\ell_{t,j}^\star \le C\|\ell_t^\star\|$. Plugging this inequality into Equation (8) gives

$$\sum_{j=1}^{k} \Delta_{t,j} \ \ge \ \sum_{j=1}^{k}\left(2\tau_{t,j}\ell_{t,j} - \tau_{t,j}^2\|\mathbf{x}_{t,j}\|_2^2\right) - 2C\|\ell_t^\star\| \ .$$

We conclude the proof by summing the above over $1 \le t \le T$ and comparing the result to the upper bound in Equation (6). ■

3. The Finite-Horizon Multitask Perceptron

In this section, we present our first family of online multitask classification algorithms, and prove a relative loss bound for the members of this family. This family includes algorithms for any global loss function defined through an absolute norm. These algorithms are finite-horizon online algorithms, meaning that the number of online rounds, T, is known in advance and is given as a parameter to the algorithm. An analogous family of infinite-horizon algorithms is the topic of the next section.

As previously noted, the Finite-Horizon Multitask Perceptron follows the general skeleton outlined in Figure 1. Given an absolute norm $\|\cdot\|$ and its dual $\|\cdot\|^*$, the multitask Perceptron sets $\tau_{t,j}$ in Figure 1 to

$$\tau_t \;=\; \operatorname*{argmax}_{\tau:\,\|\tau\|^*\le C} \tau\cdot\ell_t \;\;, \tag{9}$$

where $C>0$ is a constant which is specified later in this section. There may exist multiple solutions to the maximization problem above and at least one of these solutions induces a conservative update. In other words, we may assume that the solution to Equation (9) is such that $\tau_{t,j}=0$ at every coordinate j where $\ell_{t,j}=0$. To see that such a solution exists, take an arbitrary optimal solution τ and let $\hat{\tau}$ be defined by

$$\hat{\tau}_j \;=\; \begin{cases} \tau_j & \text{if } \ell_{t,j}\neq 0 \\ 0 & \text{if } \ell_{t,j}=0. \end{cases}$$

Clearly, $\tau\cdot\ell_t=\hat{\tau}\cdot\ell_t$, whereas $\|\hat{\tau}\|^*\le\|\tau\|^*\le C$. If the optimization problem in Equation (9) has multiple solutions that induce conservative updates, assume that one is chosen arbitrarily.

An equivalent way of defining the solution to Equation (9) is by satisfying the equality $\tau_t\cdot\ell_t = C\|\ell_t\|$. To see this equivalence, note that the dual of $\|\cdot\|^*$ is defined by Equation (2) to be

$$\|\ell\|^{**} \;=\; \max_{\tau:\,\|\tau\|^*\le 1} \tau\cdot\ell \;\;.$$

However, since $\|\cdot\|^{**}$ is equivalent to $\|\cdot\|$ (see for instance Theorem 5.5.14 in Horn and Johnson, 1985), we get

$$\|\ell\| \;=\; \max_{\tau:\,\|\tau\|^*\le 1} \tau\cdot\ell \;\;.$$

Using the linearity of $\|\cdot\|^*$, we conclude that $\|\tau/C\|^*=\|\tau\|^*/C$ for any $C>0$, and therefore the above becomes

$$C\|\ell\| \;=\; \max_{\tau:\,\|\tau\|^*\le C} \tau\cdot\ell \;\;.$$

We conclude that

$$\tau_t\cdot\ell_t \;=\; C\|\ell_t\| \tag{10}$$

holds if and only if τ_t is a maximizer of Equation (9).

When the global loss function is a p-norm, the following definition of τ_t solves Equation (9):

$$\tau_{t,j} \;=\; \frac{C\ell_{t,j}^{p-1}}{\|\ell_t\|_p^{p-1}} \;\;. \tag{11}$$

When the global loss function is an r-max norm and π is a permutation such that $\ell_{t,\pi(1)}\ge\ldots\ge\ell_{t,\pi(k)}$, the following definition of τ_t is a solution to Equation (9):

$$\tau_{t,j} \;=\; \begin{cases} C & \text{if } \ell_{t,j}>0 \text{ and } j\in\{\pi(1),\ldots,\pi(r)\} \\ 0 & \text{otherwise.} \end{cases} \tag{12}$$

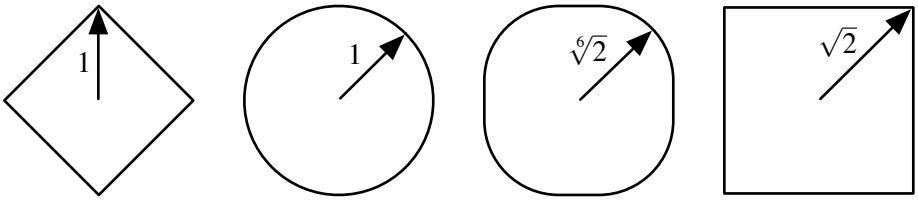


Figure 2: The *remoteness* of a norm is the longest Euclidean length of any vector contained in the norm's unit ball. The longest vector in each of the two-dimensional unit balls above is depicted with an arrow.

Note that when $r = k$, the r-max norm reduces to the L_1 norm and the above becomes the well-known update rule of the Perceptron algorithm (Rosenblatt, 1958; Novikoff, 1962). The correctness of the definitions in Equation (11) and Equation (12) can be easily verified by observing that $\|\boldsymbol{\tau}_t\|^* \leq C$ and that $\boldsymbol{\tau}_t \cdot \ell_t = C\|\ell_t\|$ in both cases.

Before proving a loss bound for the multitask Perceptron, we must introduce another important quantity. This quantity is the *remoteness* of a norm $\|\cdot\|$ defined on $\mathbb{R}^k$, and is defined to be

$$\rho(\|\cdot\|, k) \;=\; \max_{\mathbf{u}\in\mathbb{R}^k} \frac{\|\mathbf{u}\|_2}{\|\mathbf{u}\|} \;=\; \max_{\mathbf{u}\in\mathbb{R}^k : \|\mathbf{u}\|\leq 1} \|\mathbf{u}\|_2 \;. \tag{13}$$

Geometrically, the remoteness of $\|\cdot\|$ is simply the Euclidean length of the longest vector (again, in the Euclidean sense) which is contained in the unit ball of $\|\cdot\|$. This definition is visually depicted in Figure 2. As we show below, the remoteness of the dual norm, $\rho(\|\cdot\|^*, k)$, plays an important role in determining the difficulty of using $\|\cdot\|$ as the global loss function.

For concreteness, we now calculate the remoteness of the duals of p-norms and of r-max norms.

Lemma 2 *The remoteness of a p-norm $\|\cdot\|_q$ equals*

$$\rho(\|\cdot\|_q, k) \;=\; \begin{cases} 1 & \text{if } 1 \leq q \leq 2 \\ k^{(\frac{1}{2}-\frac{1}{q})} & \text{if } 2 < q \end{cases} \;.$$

Before proving the lemma, we note that if $\|\cdot\|_p$ is a p-norm and $\|\cdot\|_q$ is its dual, then we can combine Lemma 2 with the equality $q = \frac{p}{p-1}$ to obtain

$$\rho(\|\cdot\|_q, k) \;=\; \begin{cases} 1 & \text{if } 2 \leq p \\ k^{(\frac{1}{p}-\frac{1}{2})} & \text{if } 1 \leq p < 2 \end{cases} \;.$$

This equivalent form is better suited to our needs. The proof of Lemma 2 is given below.

Proof If $2 \leq p$ then $1 \leq q \leq 2$, and the monotonicity of the p-norms implies that $\|\mathbf{v}\|_q \geq \|\mathbf{v}\|_2$ for all $\mathbf{v} \in \mathbb{R}^k$. Therefore $\|\mathbf{v}\|_2/\|\mathbf{v}\|_q \leq 1$ for all $\mathbf{v} \in \mathbb{R}^k$ and thus $\rho(\|\cdot\|_q, k) \leq 1$. On the other hand,

setting $\mathbf{v} = (1,0,\ldots,0)$, we get $\|\mathbf{v}\|_q = \|\mathbf{v}\|_2$ and therefore $\rho(\|\cdot\|_q,k) \geq 1$. Overall, we have shown that $\rho(\|\cdot\|_q,k) = 1$.

Turning to the case where $1 \leq p < 2$, we note that $q > 2$. Let $\mathbf{v}$ be an arbitrary vector in $\mathbb{R}^k$, and define $\mathbf{u} = (v_1^2,\ldots,v_k^2)$ and $\mathbf{w} = (1,\ldots,1)$. Noting that $\|\cdot\|_{\frac{q}{2}}$ and $\|\cdot\|_{\frac{q}{q-2}}$ are dual norms, we use Hölder's inequality to obtain

$$\mathbf{u}\cdot\mathbf{w} \;\leq\; \|\mathbf{u}\|_{\frac{q}{2}}\,\|\mathbf{w}\|_{\frac{q}{q-2}} \quad .$$

The left-hand side above equals $\|\mathbf{v}\|_2^2$, while the right-hand side above equals $\|\mathbf{v}\|_q^2 k^{1-\frac{2}{q}}$. Therefore, $\|\mathbf{v}\|_2^2/\|\mathbf{v}\|_q^2 \leq k^{1-\frac{2}{q}}$ and taking square-roots on both sides yields $\|\mathbf{v}\|_2/\|\mathbf{v}\|_q \leq k^{\frac{1}{2}-\frac{1}{q}}$. Since this inequality holds for all $\mathbf{v} \in \mathbb{R}^k$, we have shown that $\rho(\|\cdot\|_q,k) \leq k^{\frac{1}{2}-\frac{1}{q}}$. On the other hand, setting $\mathbf{v} = (1,\ldots,1)$, we get $\|\mathbf{v}\|_2 = k^{\frac{1}{2}-\frac{1}{q}}\|\mathbf{v}\|_q$. This proves that $\rho(\|\cdot\|_q,k) \geq k^{\frac{1}{2}-\frac{1}{q}}$, and therefore $\rho(\|\cdot\|_q,k) = k^{\frac{1}{2}-\frac{1}{q}}$. ■

Lemma 3 *Let $\|\cdot\|_{r\text{-max}}$ be a r-max norm and let $\|\cdot\|^*_{r\text{-max}}$ be its dual. The remoteness of $\|\cdot\|^*_{r\text{-max}}$ equals $\sqrt{r}$.*

Proof Using Equation (13), the remoteness of $\|\cdot\|^*_{r\text{-max}}$ is defined to be the maximum value of $\|\mathbf{u}\|_2$ subject to $\|\mathbf{u}\|^*_{r\text{-max}} \leq 1$. Recalling the definition of $\|\cdot\|^*_{r\text{-max}}$ from Equation (3), we can replace this constraint with two constraints $\|\mathbf{u}\|_1 \leq r$ and $\|\mathbf{u}\|_\infty \leq 1$. Moreover, since both the L_1 norm and the L_∞ norm are absolute norms, we can also assume that $\mathbf{u}$ resides in the non-negative orthant. Therefore, we have that $0 \leq u_j \leq 1$ for all $1 \leq j \leq k$. From this we conclude that $u_j^2 \leq u_j$ for all $1 \leq j \leq k$, and thus $\|\mathbf{u}\|_2^2 \leq \|\mathbf{u}\|_1 \leq r$. Hence, $\|\mathbf{u}\|_2 \leq \sqrt{r}$ and $\rho(\|\cdot\|^*_{r\text{-max}},k) \leq \sqrt{r}$. On the other hand, the vector

$$\mathbf{u} \;=\; (\overbrace{1,\ldots,1}^{r},\overbrace{0,\ldots,0}^{k-r})$$

is contained in the unit ball of $\|\cdot\|^*_{r\text{-max}}$, and its Euclidean length is $\sqrt{r}$. Therefore, we also have that $\rho(\|\cdot\|^*_{r\text{-max}},k) \geq \sqrt{r}$, and overall we get $\rho(\|\cdot\|^*_{r\text{-max}},k) = \sqrt{r}$. ■

We are now ready to prove a loss bound for the Finite-Horizon Multitask Perceptron.

Theorem 4 *Let $\{(\mathbf{x}_{t,j},y_{t,j})\}_{1\leq t\leq T}^{1\leq j\leq k}$ be a sequence of T k-tuples of examples, where each $\mathbf{x}_{t,j} \in \mathbb{R}^{n_j}$, $\|\mathbf{x}_{t,j}\|_2 \leq R$ and each $y_{t,j} \in \{-1,+1\}$. Let C be a positive constant and let $\|\cdot\|$ be an absolute norm. Let $\mathbf{w}_1^\star,\ldots,\mathbf{w}_k^\star$ be arbitrary vectors where $\mathbf{w}_j^\star \in \mathbb{R}^{n_j}$, and define the hinge loss incurred by $\mathbf{w}_j^\star$ on example $(\mathbf{x}_{t,j},y_{t,j})$ to be $\ell^\star_{t,j} = \left[1 - y_{t,j}\mathbf{w}_j^\star\cdot\mathbf{x}_{t,j}\right]_+$. If we present this sequence to the finite-horizon multitask Perceptron with the norm $\|\cdot\|$ and the aggressiveness parameter C, then,*

$$\sum_{t=1}^{T}\|\ell_t\| \;\leq\; \frac{1}{2C}\sum_{j=1}^{k}\|\mathbf{w}_j^\star\|_2^2 \;+\; \sum_{t=1}^{T}\|\ell_t^\star\| \;+\; \frac{TR^2C\,\rho^2(\|\cdot\|^*,k)}{2} \quad .$$

Proof The starting point of our analysis is Lemma 1. The choice of $\tau_{t,j}$ in Equation (9) is clearly bounded by $\|\tau_t\|^* \leq C$ and conservative. It is also non-negative, due to the fact that $\|\cdot\|^*$ is an absolute norm and that $\ell_{t,j} \geq 0$. Therefore, the definition of $\tau_{t,j}$ in Equation (9) meets the requirements

of the lemma, and we have

$$\sum_{t=1}^{T}\sum_{j=1}^{k}\left(2\tau_{t,j}\ell_{t,j}-\tau_{t,j}^2\|\mathbf{x}_{t,j}\|_2^2\right) \;\leq\; \sum_{j=1}^{k}\|\mathbf{w}_j^\star\|_2^2 \;+\; 2C\sum_{t=1}^{T}\|\ell_t^\star\| \;\;.$$

Using Equation (10), we rewrite the left-hand side of the above as

$$2C\sum_{t=1}^{T}\|\ell_t\| - \sum_{t=1}^{T}\sum_{j=1}^{k}\tau_{t,j}^2\|\mathbf{x}_{t,j}\|_2^2 \;\;. \tag{14}$$

Using our assumption that $\|\mathbf{x}_{t,j}\|_2^2 \leq R^2$, we know that $\sum_{j=1}^{k}\tau_{t,j}^2\|\mathbf{x}_{t,j}\|_2^2 \leq (R\|\boldsymbol{\tau}_t\|_2)^2$. Using the definition of remoteness, we can upper bound this term by $(R\|\boldsymbol{\tau}_t\|^*\rho(\|\cdot\|^*,k))^2$. Finally, using our upper bound on $\|\boldsymbol{\tau}_t\|^*$ we can further bound this term by $R^2C^2\rho^2(\|\cdot\|^*,k)$. Plugging this bound back into Equation (14) gives

$$2C\sum_{t=1}^{T}\|\ell_t\| \;-\; TR^2C^2\rho^2(\|\cdot\|^*,k) \;\;.$$

Overall, we have shown that

$$2C\sum_{t=1}^{T}\|\ell_t\| \;-\; TR^2C^2\rho^2(\|\cdot\|^*,k) \;\leq\; \sum_{j=1}^{k}\|\mathbf{w}_j^\star\|_2^2 \;+\; 2C\sum_{t=1}^{T}\|\ell_t^\star\| \;\;.$$

Dividing both sides of the above by $2C$ and rearranging terms gives the desired bound. ■

In its current form, the bound in Theorem 4 may seem insignificant, since its right-most term grows linearly with the length of the input sequence, T. This term can be easily controlled by setting C to a value on the order of $1/\sqrt{T}$.

Corollary 5 *Under the assumptions of Theorem 4, if* $C = 1/(\sqrt{T}R^2)$*, then*

$$\sum_{t=1}^{T}\|\ell_t\| \;\leq\; \sum_{t=1}^{T}\|\ell_t^\star\| \;+\; \frac{\sqrt{T}}{2}\left(R^2\sum_{j=1}^{k}\|\mathbf{w}_j^\star\|_2^2 \;+\; \rho^2(\|\cdot\|^*,k)\right) \;\;.$$

This corollary bounds the global loss cumulated by our algorithm with the global loss obtained by any fixed set of hypotheses, plus a term which grows sub-linearly in T. The significance of this term depends on the magnitude of the constant

$$\frac{1}{2}\left(R^2\sum_{j=1}^{k}\|\mathbf{w}_j^\star\|_2^2 \;+\; \rho^2(\|\cdot\|^*,k)\right) \;\;.$$

Our algorithm uses C in its update procedure, and the value of C depends on $\sqrt{T}$. Therefore, the algorithm is a finite horizon algorithm.

Dividing both sides of the inequality in Corollary 5 by T, we see that the average global loss suffered by the multitask Perceptron is upper bounded by the average global loss of the best fixed hypothesis ensemble plus a term that diminishes with T. Using game-theoretic terminology, we can now say that the multitask Perceptron exhibits *no-regret* with respect to any global loss function defined by an absolute norm. The same cannot be said for the naive alternative of learning each

task independently using a separate single-task Perceptron. We show this by presenting a simple counter-example. Specifically, we construct a concrete k-task problem with a specific global loss, an arbitrarily long input sequence $\{(\mathbf{x}_{t,j}, y_{t,j})\}_{1\le t\le T}^{1\le j\le k}$, and fixed weight vectors $\mathbf{u}_1, \ldots, \mathbf{u}_k$ to use for comparison. We then prove that

$$\frac{k+1}{2} \sum_{t=1}^{T} \|\ell_t^\star\|_\infty \;\le\; \sum_{t=1}^{T} \|\hat{\ell}_t\|_\infty \;, \tag{15}$$

where $\hat{\ell}_t$ is the vector of individual losses of the k independent single-task Perceptrons, and, as before, $\ell_t^\star$ is the vector of individual losses of $\mathbf{u}_1, \ldots, \mathbf{u}_k$ respectively. This example demonstrates that a claim along the lines of Corollary 5 cannot be proven for the set of independent single-task Perceptrons.

First, we would like to emphasize that we are considering a version of the single-task Perceptron that updates its hypothesis whenever it suffers a positive hinge-loss, and not only when it makes a prediction mistake. Moreover, when an update is performed, the algorithm defines $\mathbf{w}_{t+1} = \mathbf{w}_t + C y_t \mathbf{x}_t$, where C is a predefined constant. This version of the Perceptron is sometimes called the *aggressive Perceptron*. If we were to use the simplest version of the Perceptron, which updates its hypothesis only when a prediction mistake occurs, then finding a counter-example that achieves Equation (15) would be trivial, without even using the distinction between single-task and multitask Perceptron learning.

Also, we can assume without loss of generality that $1/C = o(T)$, since otherwise, even in the case $k = 1$, simply repeating the same example over and over provides a counterexample.

Moving on to the counter-example itself, assume that our global loss is defined by the L_∞ norm. Let k be at least 2, assume that the instances of all k problems are two dimensional vectors, and set $\mathbf{u}_1 = \ldots = \mathbf{u}_k = (1,1)$. Each of the single-task Perceptrons initializes its hypothesis to $(0,0)$. Assume that all of the labels in the input sequence are positive labels. For $t = 0$, we set $\mathbf{x}_{1,1} = \ldots = \mathbf{x}_{1,k} = (1,0)$. Each one of the independent Perceptrons suffers a positive individual loss and updates its weight vector to $(C,0)$. We continue presenting the same example for $\lceil 1/C \rceil - 1$ additional rounds, which is precisely when all k weight vectors of the Perceptrons become equal to $(\alpha,0)$, with $\alpha \ge 1$. For instance, if $C = O(1/\sqrt{T})$ then the vector $(1,0)$ is presented $O(\sqrt{T})$ times. Meanwhile, the fixed weight vectors $\mathbf{u}_1, \ldots, \mathbf{u}_k$ suffer no loss at all.

Define $t_0 = \lceil 1/C \rceil$, and note that the index of the next online round is $t_0 + 1$. For each t in $t_0 + 1, \ldots, t_0 + k$, we set $\mathbf{x}_{t,t-t_0}$ to $(0,1)$ and $\mathbf{x}_{t,j}$ to $(1,0)$ for all $j \ne t - t_0$. On round t, the $(t - t_0)$'th Perceptron, whose weight vector is $(\alpha,0)$, suffers an individual loss of 1 and updates its weight vector to (α,C). The remaining $k-1$ Perceptrons suffer no individual loss and do not modify their weight vectors. Consequently, $\|\hat{\ell}_t\|_\infty = 1$ on each of these rounds. Once again, the fixed vectors $\mathbf{u}_1, \ldots, \mathbf{u}_k$ suffer no loss at all. On round $t = t_0 + k + 1$, we set $\mathbf{x}_{t,1} = \ldots = \mathbf{x}_{t,k} = (0,-1)$. As a result, each of the Perceptrons suffers a hinge loss of $1 + C$ and updates its weight vector back to $(\alpha,0)$. Since C is positive, we get $\|\hat{\ell}_t\|_\infty \ge 1$. Meanwhile, $\|\ell_t^\star\|_\infty = 2$. We now have that

$$\sum_{t=t_0+1}^{t_0+k+1} \|\hat{\ell}_t\|_\infty \;\ge\; k+1 \quad \text{and} \quad \sum_{t=t_0+1}^{t_0+k+1} \|\ell_t^\star\|_\infty \;=\; 2 \;.$$

Furthermore, the weight vectors of the k single-task Perceptrons have returned to their values at the end of round t_0. Therefore, by repeating the input sequence from round $t_0 + 1$ to round $t_0 + k + 1$ over and over again, we obtain Equation (15).

This concludes the presentation of the counter-example thus showing that a set of independent single-task Perceptrons does not attain no-regret with respect to the L_∞ norm global loss. Similar constructions can be given for other global loss functions. The exception is the L_1 norm, which naturally reduces the multitask Perceptron to k independent single-task Perceptrons.

4. An Extension to the Infinite Horizon Setting

In the previous section, we devised an algorithm which relied on prior knowledge of T, the input sequence length. In this section, we adapt the update procedure from the previous section to the infinite horizon setting, where T is not known in advance. Moreover, the bound we prove in this section holds simultaneously for every prefix of the input sequence. This generalization comes at a price; we can only prove an upper bound on $\sum_t \min\{\ell_t, \ell_t^2\}$, a quantity similar to the cumulative global loss, but not the global loss per se.

To motivate our infinite-horizon algorithm, we take a closer look at the analysis of the finite-horizon algorithm. In the proof of Theorem 4, we lower-bounded the term $\sum_{j=1}^k 2\tau_{t,j}\ell_{t,j} - \tau_{t,j}^2\|\mathbf{x}_{t,j}\|_2^2$ by $2C\|\ell_t\| - R^2C^2\rho^2(\|\cdot\|^*, k)$. The first term in this lower bound is proportional to the global loss suffered on round t, and the second term is a constant. When $\|\ell_t\|$ is smaller than this constant, our lower bound becomes negative. This suggests that the update step-size applied by the finite-horizon Perceptron may have been too large, and that the update step may have overshot its target. As a result, the new hypothesis may be inferior to the previous one. Nevertheless, over the course of T rounds, our positive progress is guaranteed to overshadow our negative progress, and thus we are able to prove Theorem 4. However, if we are interested in a bound which holds for every prefix of the input sequence, we must ensure that every individual update makes positive progress. Concretely, we derive an update for which $\sum_{j=1}^k 2\tau_{t,j}\ell_{t,j} - \tau_{t,j}^2\|\mathbf{x}_{t,j}\|_2^2$ is guaranteed to be non-negative. The vector $\boldsymbol{\tau}_t$ remains in the same direction as before, but by setting its length more carefully, we enforce an update step-size which is never excessively large.

We use ρ to abbreviate $\rho(\|\cdot\|^*, k)$ throughout this section. We replace the definition of $\boldsymbol{\tau}_t$ in Equation (9) with the following definition,

$$\boldsymbol{\tau}_t \;=\; \operatorname*{argmax}_{\boldsymbol{\tau}:\, \|\boldsymbol{\tau}\|^* \le \min\left\{C, \frac{\|\ell_t\|}{R^2\rho^2}\right\}} \boldsymbol{\tau}\cdot\ell_t \;\;, \tag{16}$$

where $C > 0$ is a user defined parameter and $R > 0$ is an upper bound on $\|\mathbf{x}_{t,j}\|_2$ for all $1 \le t \le T$ and all $1 \le j \le k$. As in the previous section, we assume that $\tau_{t,j} = 0$ whenever $\ell_{t,j} = 0$. As in Equation (10), the solution to Equation (16) can be equivalently defined by the equation

$$\boldsymbol{\tau}_t \cdot \ell_t \;=\; \min\left\{C, \frac{\|\ell_t\|}{R^2\rho^2}\right\}\|\ell_t\| \;\;. \tag{17}$$

When the global loss function is a p-norm, the following definition of $\boldsymbol{\tau}_t$ solves Equation (16):

$$\tau_{t,j} \;=\; \begin{cases} \dfrac{\ell_{t,j}^{p-1}}{R^2\rho^2\|\ell_t\|_p^{p-2}} & \text{if } \|\ell_t\|_p \le R^2C\rho^2 \\ \dfrac{C\ell_{t,j}^{p-1}}{\|\ell_t\|_p^{p-1}} & \text{if } \|\ell_t\|_p > R^2C\rho^2. \end{cases}$$

When the global loss function is an r-max norm and π is a permutation such that $\ell_{t,\pi(1)} \geq \ldots \geq \ell_{t,\pi(k)}$, then the following definition of τ_t is a solution to Equation (16):

$$\tau_{t,j} \;=\; \begin{cases} \frac{\|\ell_t\|_{r\text{-max}}}{rR^2} & \text{if } \ell_{t,j} > 0 \text{ and } \|\ell_t\|_{r\text{-max}} \leq R^2 C\rho^2 \text{ and } j \in \{\pi(1),\ldots,\pi(r)\} \\ C & \text{if } \ell_{t,j} > 0 \text{ and } \|\ell_t\|_{r\text{-max}} > R^2 C\rho^2 \text{ and } j \in \{\pi(1),\ldots,\pi(r)\} \\ 0 & \text{otherwise.} \end{cases}$$

The correctness of both definitions of $\tau_{t,j}$ given above can be verified by observing that $\|\tau_t\|^* \leq \min\{C, \frac{\|\ell_t\|}{R^2\rho^2}\}$ and that $\tau_t \cdot \ell_t = \min\{C, \frac{\|\ell_t\|}{R^2\rho^2}\}\|\ell_t\|$ in both cases. We now turn to proving an infinite-horizon cumulative loss bound for our algorithm.

Theorem 6 *Let $\{(\mathbf{x}_{t,j}, y_{t,j})\}_{t=1,2,\ldots}^{1\leq j\leq k}$ be a sequence of k-tuples of examples, where each $\mathbf{x}_{t,j} \in \mathbb{R}^{n_j}$, $\|\mathbf{x}_{t,j}\|_2 \leq R$ and each $y_{t,j} \in \{-1,+1\}$. Let C be a positive constant, let $\|\cdot\|$ be an absolute norm, and let ρ be an abbreviation for $\rho(\|\cdot\|^*, k)$. Let $\mathbf{w}_1^\star, \ldots, \mathbf{w}_k^\star$ be arbitrary vectors where $\mathbf{w}_j^\star \in \mathbb{R}^{n_j}$, and define the hinge loss attained by $\mathbf{w}_j^\star$ on example $(\mathbf{x}_{t,j}, y_{t,j})$ to be $\ell_{t,j}^\star = \left[1 - y_{t,j}\mathbf{w}_j^\star \cdot \mathbf{x}_{t,j}\right]_+$. If we present this sequence to the explicit multitask algorithm with the norm $\|\cdot\|$ and the aggressiveness parameter C, then for every T*

$$1/(R^2\rho^2) \sum_{t\leq T:\|\ell_t\|\leq R^2C\rho^2} \|\ell_t\|^2 \;+\; C \sum_{t\leq T:\|\ell_t\|>R^2C\rho^2} \|\ell_t\| \;\leq\; 2C \sum_{t=1}^{T} \|\ell_t^\star\| \;+\; \sum_{j=1}^{k} \|\mathbf{w}_j^\star\|_2^2 \;.$$

Proof The starting point of our analysis is again Lemma 1. The choice of $\tau_{t,j}$ in Equation (16) is clearly bounded by $\|\tau_t\|^* \leq C$ and conservative. It is also non-negative, due to the fact that $\|\cdot\|^*$ is absolute and that $\ell_{t,j} \geq 0$. Therefore, $\tau_{t,j}$ meets the requirements of Lemma 1, and we have

$$\sum_{t=1}^{T}\sum_{j=1}^{k} \left(2\tau_{t,j}\ell_{t,j} - \tau_{t,j}^2\|\mathbf{x}_{t,j}\|_2^2\right) \;\leq\; \sum_{j=1}^{k} \|\mathbf{w}_j^\star\|_2^2 \;+\; 2C \sum_{t=1}^{T} \|\ell_t^\star\| \;. \tag{18}$$

We now prove our theorem by lower-bounding the left hand side of Equation (18) above. We analyze two different cases. First, if $\|\ell_t\| \leq R^2C\rho^2$ then $\min\{C, \|\ell_t\|/(R^2\rho^2)\} = \|\ell_t\|/(R^2\rho^2)$. Together with Equation (17), this gives

$$2\sum_{j=1}^{k} \tau_{t,j}\ell_{t,j} \;=\; 2\|\tau_t\|^*\,\|\ell_t\| \;=\; 2\frac{\|\ell_t\|^2}{R^2\rho^2} \;. \tag{19}$$

On the other hand, $\sum_{j=1}^{k} \tau_{t,j}^2\|\mathbf{x}_{t,j}\|_2^2$ can be bounded by $\|\tau_t\|_2^2 R^2$. Using the definition of remoteness, we bound this term by $(\|\tau_t\|^*)^2 R^2\rho^2$. Using the fact that, $\|\tau_t\|^* \leq \|\ell_t\|/(R^2\rho^2)$, we bound this term by $\|\ell_t\|^2/(R^2\rho^2)$. Overall, we have shown that

$$\sum_{j=1}^{k} \tau_{t,j}^2\|\mathbf{x}_{t,j}\|_2^2 \;\leq\; \frac{\|\ell_t\|^2}{R^2\rho^2} \;.$$

Subtracting both sides of the above inequality from the respective sides of Equation (19) gives

$$\frac{\|\ell_t\|^2}{R^2\rho^2} \;\leq\; \sum_{j=1}^{k} \left(2\tau_{t,j}\ell_{t,j} - \tau_{t,j}^2\|\mathbf{x}_{t,j}\|_2^2\right) \;. \tag{20}$$

Moving on to the second case, if $\|\ell_t\| > R^2 C\rho^2$ then $\min\{C, \|\ell_t\|/(R^2\rho^2)\} = C$. Using Equation (17), we have that

$$2\sum_{j=1}^{k} \tau_{t,j}\ell_{t,j} \;=\; 2\|\tau_t\|^* \|\ell_t\| \;=\; 2C\|\ell_t\| \;. \tag{21}$$

As before, we can upper bound $\sum_{j=1}^{k} \tau_{t,j}^2 \|\mathbf{x}_{t,j}\|_2^2$ by $(\|\tau_t\|^*)^2 R^2\rho^2$. Using the fact that, $\|\tau_t\|^* \le C$ we can bound this term by $C^2R^2\rho^2$. Finally, using our assumption that $\|\ell_t\| > R^2C\rho^2$, we conclude that

$$\sum_{j=1}^{k} \tau_{t,j}^2 \|\mathbf{x}_{t,j}\|_2^2 \;<\; C\|\ell_t\| \;.$$

Subtracting both sides of the above inequality from the respective sides of Equation (21) gives

$$C\|\ell_t\| \;\le\; \sum_{j=1}^{k} \left(2\tau_{t,j}\ell_{t,j} - \tau_{t,j}^2\|\mathbf{x}_{t,j}\|_2^2\right) \;. \tag{22}$$

Comparing the upper bound in Equation (18) with the lower bounds in Equation (20) and Equation (22) proves the theorem. ■

Corollary 7 *Under the assumptions of Theorem 6, if C is set to be $1/(R^2\rho^2)$ then for every $T' \le T$ it holds that,*

$$\sum_{t=1}^{T'} \min\left\{\|\ell_t\|^2, \|\ell_t\|\right\} \;\le\; 2\sum_{t=1}^{T'} \|\ell_t^\star\| \;+\; R^2\rho^2 \sum_{j=1}^{k} \|\mathbf{w}_j^\star\|_2^2 \;.$$

As noted at the beginning of this section, we do not obtain a cumulative loss bound per se, but rather at a bound on $\sum_t \min\{\ell_t, \ell_t^2\}$. However, this bound holds simultaneously for every prefix of the input sequence, and the algorithm does not rely on knowledge of the input sequence length.

5. The Implicit Online Multitask Update

We now discuss a third family of online multitask algorithms, which leads to the strongest loss bounds of the three families of algorithms presented in this paper. In contrast to the closed form updates of the previous algorithms, the algorithms in this family require solving an optimization problem on every round, and are therefore called *implicit update* algorithms. Although the implementation of specific members of this family may be more involved than the implementation of the multitask Perceptron, we recommend using this family of algorithms in practice. On every round, the set of hypotheses is updated according to the update rule:

$$\begin{aligned}\{\mathbf{w}_{t+1,1}, \ldots, \mathbf{w}_{t+1,k}\} \;=\; & \operatorname*{argmin}_{\mathbf{w}_1,\ldots,\mathbf{w}_k} \; \frac{1}{2}\sum_{j=1}^{k} \|\mathbf{w}_j - \mathbf{w}_{t,j}\|_2^2 + C\|\xi\| \\ & \text{s.t. } \forall j \quad \mathbf{w}_j \cdot \mathbf{x}_{t,j} \ge 1 - \xi_j \quad \text{and} \quad \xi_j \ge 0.\end{aligned} \tag{23}$$

This optimization problem captures the fundamental tradeoff inherent to online learning. On one hand, the term $\sum_{j=1}^{k} \|\mathbf{w}_j - \mathbf{w}_{t,j}\|_2^2$ in the objective function above keeps the new set of hypotheses close to the current set of hypotheses, so as to retain the information learned on previous rounds. On

input: aggressiveness parameter $C > 0$, norm $\|\cdot\|$

initialize $\mathbf{w}_{1,1} = \ldots = \mathbf{w}_{1,k} = (0,\ldots,0)$

for $t = 1,2,\ldots$

- receive $\mathbf{x}_{t,1},\ldots,\mathbf{x}_{t,k}$
- predict $\text{sign}(\mathbf{w}_{t,j}\cdot\mathbf{x}_{t,j})$ $\quad [1 \le j \le k]$
- receive $y_{t,1},\ldots,y_{t,k}$
- suffer loss $\ell_{t,j} = \left[1 - y_{t,j}\mathbf{w}_{t,j}\cdot\mathbf{x}_{t,j}\right]_+$ $\quad [1 \le j \le k]$
- update:

$$\{\mathbf{w}_{t+1,1},\ldots,\mathbf{w}_{t+1,k}\} = \operatorname*{argmin}_{\mathbf{w}_1,\ldots,\mathbf{w}_k} \frac{1}{2}\sum_{j=1}^{k}\|\mathbf{w}_j - \mathbf{w}_{t,j}\|_2^2 + C\|\xi\|$$
$$\text{s.t. } \forall j \quad \mathbf{w}_j\cdot\mathbf{x}_{t,j} \ge 1 - \xi_j \text{ and } \xi_j \ge 0$$

Figure 3: The implicit update algorithm

the other hand, the term $\|\xi\|$ in the objective function, together with the constraints on ξ_j, forces the algorithm to make progress using the new examples obtained on this round. Different choices of the global loss function lead to different definitions of this progress. The pseudo-code of the implicit update algorithm is presented in Figure 3.

Our first task is to show that this update procedure follows the skeleton outlined in Figure 1, and satisfies the requirements of Lemma 1. We do so by finding the dual of the optimization problem given in Equation (23).

Lemma 8 *Let $\|\cdot\|$ be a norm and let $\|\cdot\|^*$ be its dual. Then the online update defined in Equation (23) is equivalent to setting $\mathbf{w}_{t+1,j} = \mathbf{w}_{t,j} + \tau_{t,j}y_{t,j}\mathbf{x}_{t,j}$ for all $1 \le j \le k$, where*

$$\tau_t = \operatorname*{argmax}_{\tau} \sum_{j=1}^{k}\left(2\tau_j\ell_{t,j} - \tau_j^2\|\mathbf{x}_{t,j}\|_2^2\right)$$
$$\text{s.t. } \|\tau\|^* \le C \text{ and } \forall j \ \tau_j \ge 0 \ .$$

Moreover, this update is conservative.

Proof The update step in Equation (23) sets the vectors $\mathbf{w}_{t+1,1},\ldots,\mathbf{w}_{t+1,k}$ to be the solution to the following constrained minimization problem:

$$\min_{\mathbf{w}_1,\ldots,\mathbf{w}_k,\xi\ge 0} \frac{1}{2}\sum_{j=1}^{k}\|\mathbf{w}_j - \mathbf{w}_{t,j}\|_2^2 + C\|\xi\|$$
$$\text{s.t. } \forall j \ y_{t,j}\mathbf{w}_j\cdot\mathbf{x}_{t,j} \ge 1 - \xi_j \ .$$

We begin by using the notion of strong duality to restate this optimization problem in an equivalent form. The objective function above is convex and the constraints are both linear and feasible, therefore Slater's condition (Boyd and Vandenberghe, 2004) holds, and the above problem is equivalent

to

$$\max_{\tau \geq 0} \min_{\mathbf{w}_1,\ldots,\mathbf{w}_k,\xi \geq 0} \mathcal{L}(\tau, \mathbf{w}_1, \ldots, \mathbf{w}_k, \xi) \ ,$$

where $\mathcal{L}(\tau, \mathbf{w}_1, \ldots, \mathbf{w}_k, \xi)$ is defined as follows:

$$\frac{1}{2}\sum_{j=1}^{k} \|\mathbf{w}_j - \mathbf{w}_{t,j}\|_2^2 + C\|\xi\| + \sum_{j=1}^{k} \tau_j (1 - y_{t,j}\mathbf{w}_j \cdot \mathbf{x}_{t,j} - \xi_j) \ .$$

We can rewrite $\mathcal{L}$ as the sum of two terms, the first a function of τ and $\mathbf{w}_1, \ldots, \mathbf{w}_k$ (denoted $\mathcal{L}_1$) and the second a function of τ and $\xi_1, \ldots, \xi_k$ (denoted $\mathcal{L}_2$),

$$\underbrace{\frac{1}{2}\sum_{j=1}^{k} \|\mathbf{w}_j - \mathbf{w}_{t,j}\|_2^2 + \sum_{j=1}^{k} \tau_j (1 - y_{t,j}\mathbf{w}_j \cdot \mathbf{x}_{t,j})}_{\mathcal{L}_1(\tau, \mathbf{w}_1, \ldots, \mathbf{w}_k)} + \underbrace{C\|\xi\| - \sum_{j=1}^{k} \tau_j \xi_j}_{\mathcal{L}_2(\tau, \xi)} \ .$$

Using the notation defined above, our optimization problem becomes,

$$\max_{\tau \geq 0} \left(\min_{\mathbf{w}_1,\ldots,\mathbf{w}_k} \mathcal{L}_1(\tau, \mathbf{w}_1, \ldots, \mathbf{w}_k) + \min_{\xi \geq 0} \mathcal{L}_2(\tau, \xi) \right) \ .$$

For any choice of τ, $\mathcal{L}_1$ is a convex function and we can find $\mathbf{w}_1, \ldots, \mathbf{w}_k$ which minimize it by setting all of its partial derivatives with respect to the elements of $\mathbf{w}_1, \ldots, \mathbf{w}_k$ to zero. Namely,

$$\forall j, l \quad 0 = \frac{\partial \mathcal{L}_1}{\partial w_{j,l}} = w_{j,l} - w_{t,j,l} - \tau_j y_{t,j} x_{t,j,l} \ .$$

from the above we conclude that $\mathbf{w}_j = \mathbf{w}_{t,j} + \tau_j y_{t,j} \mathbf{x}_{t,j}$ for all $1 \leq j \leq k$.

The next step is to show that the update is conservative. If $\ell_{t,j} = 0$ then setting $\mathbf{w}_j = \mathbf{w}_{t,j}$ satisfies the constraint $y_{t,j}\mathbf{w}_j \cdot \mathbf{x}_{t,j} \geq 1 - \xi_j$ with any choice of $\xi_j \geq 0$. Since choosing $\mathbf{w}_j = \mathbf{w}_{t,j}$ minimizes $\|\mathbf{w}_t - \mathbf{w}_{t,j}\|_2^2$ and does not restrict our choice of any other variable, then it is optimal. The relation between $\mathbf{w}_j$ and τ_j now implies that $\tau_j = 0$ whenever $\ell_{t,j} = 0$).

Plugging our expression for $\mathbf{w}_j$ into $\mathcal{L}_1$, we have that

$$\min_{\mathbf{w}_1,\ldots,\mathbf{w}_k} \mathcal{L}_1(\tau, \mathbf{w}_1, \ldots, \mathbf{w}_k) = \sum_{j=1}^{k} \tau_j (1 - y_{t,j}\mathbf{w}_{t,j} \cdot \mathbf{x}_{t,j}) - \frac{1}{2}\sum_{j=1}^{k} \tau_j^2 \|\mathbf{x}_{t,j}\| \ .$$

Since the update is conservative, it holds that $\tau_j(1 - y_{t,j}\mathbf{w}_{t,j} \cdot \mathbf{x}_{t,j}) = \tau_j \ell_{t,j}$. Overall, we have reduced our optimization problem to

$$\tau_t = \operatorname*{argmax}_{\tau \geq 0} \left(\sum_{j=1}^{k} \left(\tau_j \ell_{t,j} - \frac{1}{2}\tau_j^2 \|\mathbf{x}_{t,j}\| \right) + \min_{\xi \geq 0} \mathcal{L}_2(\tau, \xi) \right) \ .$$

We finally turn our attention to $\mathcal{L}_2$ and abbreviate $B(\tau) = \min_{\xi \geq 0} \mathcal{L}_2(\tau, \xi)$. We now claim that B is a barrier function for the constraint $\|\tau\|^* \leq C$, namely

$$B(\tau) = \begin{cases} 0 & \text{if } \|\tau\|^* \leq C \\ -\infty & \text{if } \|\tau\|^* > C \end{cases} \ .$$

To see why this is true, recall that $\|\tau\|^*$ is defined to be

$$\|\tau\|^* \;=\; \max_{\varepsilon \in \mathbb{R}^k} \frac{\sum_{j=1}^k \tau_j \varepsilon_j}{\|\varepsilon\|} \;\; .$$

First, let us consider the case where $\|\tau\|^* > C$. In this case there exists a vector $\bar{\varepsilon}$ for which

$$\sum_{j=1}^k \tau_j \bar{\varepsilon}_j - C\|\bar{\varepsilon}\| \;>\; 0 \;\; .$$

Denote the left hand side of the above by δ. We can assume w.l.o.g. that all the components of $\bar{\varepsilon}$ are non-negative since $\tau \geq 0$. For any $c \geq 0$, we now have that

$$B(\tau) \;=\; \min_{\xi \geq 0} \mathcal{L}_2(\tau, \xi) \;\leq\; \mathcal{L}_2(\tau, c\bar{\varepsilon}) \;=\; -c\delta \;\; .$$

Therefore, by taking c to infinity we get that $B(\tau) = -\infty$.

Turning to the case $\|\tau\|^* \leq C$, we have that $\sum_{j=1}^k \tau_j \xi_j \leq C\|\xi\|$ for any choice of ξ, or in other words, $\min_{\xi \geq 0} \mathcal{L}_2(\tau, \xi) \geq 0$. However, this lower bound is attainable by setting $\xi = 0$. We conclude that if $\|\tau\|^* \leq C$ then $B(\tau) = 0$. The original optimization problem has reduced to the form

$$\tau_t \;=\; \operatorname*{argmax}_{\tau \geq 0} \left(\sum_{j=1}^k \left(\tau_j \ell_{t,j} - \frac{1}{2}\tau_j^2 \|\mathbf{x}_{t,j}\| \right) \;+\; B(\tau) \right) \;\; .$$

Clearly, the above is maximized in the domain where $B(\tau) = 0$. Therefore, we replace the function B with the constraint $\|\tau\|^* \leq C$, and get

$$\tau_t \;=\; \operatorname*{argmax}_{\tau \geq 0 \,:\, \|\tau\|^* \leq C} \;\; \sum_{j=1}^k \left(\tau_j \ell_{t,j} - \frac{1}{2}\tau_j^2 \|\mathbf{x}_{t,j}\| \right) \;\; .$$

■

Lemma 5 proves that the implicit update essentially finds the value of τ_t that maximizes the left-hand side of the bound in Lemma 1. This choice of τ_t produces the tightest loss bounds that can be derived from Lemma 1. In this sense, the implicit update algorithm takes full advantage of our proof technique. An immediate consequence of this observation is that the loss bounds of the multitask Perceptron also hold for the implicit algorithm. More precisely, the bound in Theorem 4 (and Corollary 5) holds not only for the multitask Perceptron, but also for the implicit update algorithm. Equivalently, it can be shown that the bound in Theorem 6 (and Corollary 7) also holds for the implicit update algorithm. We prove this formally below.

Theorem 9 *The bound in Theorem 4 also holds for the implicit update algorithm.*

Proof Let $\tau'_{t,j}$ denote the weights defined by the multitask Perceptron in Equation (9) and let $\tau_{t,j}$ denote the weights assigned by the implicit update algorithm. In the proof of Theorem 4, we showed that,

$$2C\|\ell_t\| - R^2C^2\rho^2 \;\leq\; \sum_{j=1}^k \left(2\tau'_{t,j}\ell_{t,j} - {\tau'_{t,j}}^2 \|\mathbf{x}_{t,j}\|_2^2 \right) \;\; .$$

According to Lemma 8, the weights $\tau_{t,j}$ maximize,

$$\sum_{j=1}^{k} \left(2\tau_{t,j}\ell_{t,j} - \tau_{t,j}^2 \|\mathbf{x}_{t,j}\|_2^2\right) \ ,$$

subject to the constraints $\|\tau_t\|^* \leq C$ and $\tau_{t,j} \geq 0$. Since the weights $\tau'_{t,j}$ also satisfy these constraints, it holds that,

$$\sum_{j=1}^{k} \left(2\tau'_{t,j}\ell_{t,j} - \tau'^2_{t,j} \|\mathbf{x}_{t,j}\|_2^2\right) \ \leq \ \sum_{j=1}^{k} \left(2\tau_{t,j}\ell_{t,j} - \tau_{t,j}^2 \|\mathbf{x}_{t,j}\|_2^2\right) \ .$$

Therefore, we conclude that

$$2C\|\ell_t\| - R^2C^2\rho^2 \ \leq \ \sum_{j=1}^{k} \left(2\tau_{t,j}\ell_{t,j} - \tau_{t,j}^2 \|\mathbf{x}_{t,j}\|_2^2\right) \ . \tag{24}$$

Since $\tau_{t,j}$ is bounded, non-negative, and conservative (due to Lemma 8), the right-hand side of the above inequality is upper-bounded by Lemma 1. Comparing the bound in Equation (24) with the bound in Lemma 1 proves the theorem. ■

In the remainder of this paper, we present efficient algorithms which solve the optimization problem in Equation (23) for different choices of the global loss function.

6. Solving the Implicit Update for the L_2 Norm

Consider the implicit update with the L_2 norm, namely we are trying to solve

$$\tau_t \ = \ \operatorname*{argmax}_{\tau \geq 0 \,:\, \|\tau\|_2 \leq C} \ \sum_{j=1}^{k} \left(\tau_j \ell_{t,j} - \frac{1}{2}\tau_j^2 \|\mathbf{x}_{t,j}\|\right) \ .$$

The Lagrangian of this optimization problem is

$$\mathcal{L} \ = \ \sum_{j=1}^{k} \left(2\tau_{t,j}\ell_{t,j} - \tau_{t,j}^2 \|\mathbf{x}_{t,j}\|_2^2\right) - \theta \left(\sum_{j=1}^{k} \tau_{t,j}^2 - C^2\right) \ ,$$

where θ is a non-negative Lagrange multiplier. The derivative of $\mathcal{L}$ with respect to each $\tau_{t,j}$ is, $2\ell_{t,j} - 2\tau_{t,j}\|\mathbf{x}_{t,j}\|_2^2 - 2\theta\tau_{t,j}$. Setting this derivative to zero, we get

$$\tau_{t,j} \ = \ \frac{\ell_{t,j}}{\|\mathbf{x}_{t,j}\|_2^2 + \theta} \ .$$

The optimum of the *unconstrained* problem is attained by choosing $\tau_{t,j} = \frac{\ell_{t,j}}{\|\mathbf{x}_{t,j}\|_2^2}$ for each j. If, for this choice of τ_t, the constraint $\sum_{j=1}^{k} \tau_{t,j}^2 \leq C^2$ does not hold, then θ must be greater than zero. The KKT complementarity condition implies that in this case the constraint is binding, namely $\sum_{j=1}^{k} \tau_{t,j}^2 = C^2$. In order to find θ, we must now solve the following equation:

$$\sum_{j=1}^{k} \left(\frac{\ell_{t,j}}{\|\mathbf{x}_{t,j}\|_2^2 + \theta}\right)^2 = C^2 \ . \tag{25}$$

The left hand side of the above is monotonically decreasing in θ. We also know that $\theta > 0$. Moreover, setting

$$\theta = \frac{\sqrt{k}\|\ell_t\|_\infty}{C}$$

in the left-hand side of Equation (25) yields a value which is at least C^2, and therefore we conclude that $\theta \leq \frac{\sqrt{k}\|\ell_t\|_\infty}{C}$. These properties enable us to easily find θ using binary search.

In the special case where the norms of all the instances are equal, namely $\|\mathbf{x}_{t,1}\|_2^2 = \ldots = \|\mathbf{x}_{t,k}\|_2^2 = R^2$, Equation (25) gives $\theta = \frac{\|\ell_t\|_2}{C} - R^2$, and therefore $\tau_{t,j} = C\ell_{t,j}/\|\ell_t\|_2$. The general expression for $\tau_{t,j}$ in this case becomes

$$\tau_{t,j} = \begin{cases} \frac{\ell_{t,j}}{R^2} & \text{if } \|\ell_t\|_2 \leq R^2 C \\ \frac{C\ell_{t,j}}{\|\ell_t\|_2} & \text{otherwise} \end{cases} .$$

Note that the above coincides with the definition of τ_t given by the Infinite Horizon Multitask Perceptron for the L_2 norm, as defined in Section 4.

7. Solving the Implicit Update for r-max Norms

We now present an efficient procedure for calculating the update in Equation (23), in the case where the norm being used is the r-max norm. Lemma 8, together with (3), tells us that the update can be calculated by solving the following constrained optimization problem:

$$\begin{aligned} \tau_t &= \operatorname*{argmax}_{\tau} \sum_{j=1}^{k} \left(2\tau_j \ell_{t,j} - \tau_j^2 \|\mathbf{x}_{t,j}\|_2^2\right) \\ &\text{s.t. } \sum_{j=1}^{k} \tau_j \leq Cr \ , \ \forall j \ \ \tau_j \leq C \ , \ \forall j \ \ \tau_j \geq 0 \ . \end{aligned} \tag{26}$$

After dividing the objective function by 2, the Lagrangian of this optimization problem is

$$\sum_{j=1}^{k} \left(\tau_j \ell_{t,j} - \frac{1}{2}\tau_j^2 \|\mathbf{x}_{t,j}\|_2^2\right) + \theta\left(Cr - \sum_{j=1}^{k} \tau_j\right) + \sum_{j=1}^{k} \lambda_j (C - \tau_j) + \sum_{j=1}^{k} \beta_j \tau_j \ ,$$

where θ, the β_j's and the λ_j's are non-negative Lagrange multipliers. The derivative of $\mathcal{L}$ with respect to each τ_j is, $\ell_{t,j} - \tau_j \|\mathbf{x}_{t,j}\|_2^2 - \theta - \lambda_j + \beta_j$. All of these partial derivatives must equal zero at the optimum, and therefore

$$\forall\ 1 \leq j \leq k \quad \tau_j = \frac{\ell_{t,j} - \theta - \lambda_j + \beta_j}{\|\mathbf{x}_{t,j}\|_2^2} \ . \tag{27}$$

The KKT complementarity condition states that the following equalities hold at the optimum:

$$\forall\ 1 \leq j \leq k \quad \lambda_j (C - \tau_j) = 0 \quad \text{and} \quad \beta_j \tau_j = 0 \ . \tag{28}$$

We consider three different cases:

1. Assume that $\ell_{t,j} - \theta < 0$. Since both τ_j and λ_j must be non-negative, then from the definition of τ_j in Equation (27) we learn that β_j must be at least $\theta - \ell_{t,j}$. In other words, β_j is positive. Referring to the right-hand side of Equation (28), we conclude that $\tau_j = 0$.

2. Assume that $0 \leq \ell_{t,j} - \theta \leq C\|\mathbf{x}_{t,j}\|_2^2$. Summing the two equalities in Equation (28) and plugging in the definition of τ_j from Equation (27) results in,

$$\lambda_j \left(C - \frac{\ell_{t,j} - \theta}{\|\mathbf{x}_{t,j}\|_2^2} \right) + \beta_j \left(\frac{\ell_{t,j} - \theta}{\|\mathbf{x}_{t,j}\|_2^2} \right) + \frac{(\beta_j - \lambda_j)^2}{\|\mathbf{x}_{t,j}\|_2^2} \;=\; 0 \;\; . \tag{29}$$

   Using our assumption that $\ell_{t,j} - \theta \geq 0$, along with the requirement that $\beta_j \geq 0$, gives us that $\beta(\ell_{t,j} - \theta)/\|\mathbf{x}_{t,j}\|_2^2 \geq 0$. Equivalently, using our assumption that $\ell_{t,j} - \theta \leq C\|\mathbf{x}_{t,j}\|_2^2$ along with the requirement that $\lambda_j \geq 0$ results in $\lambda\big(C - (\ell_{t,j} + \theta)/\|\mathbf{x}_{t,j}\|_2^2\big) \geq 0$. Plugging the last two inequalities back into Equation (29) gives, $(\beta_j - \lambda_j)^2/\|\mathbf{x}_{t,j}\|_2^2 \leq 0$. The only way that this inequality can hold is if $(\beta_j - \lambda_j) = 0$. Thus, the definition of τ_j in Equation (27) reduces to $\tau_j \;=\; \frac{\ell_{t,j} - \theta}{\|\mathbf{x}_{t,j}\|_2^2}$.

3. Finally, assume that $\ell_{t,j} - \theta > C\|\mathbf{x}_{t,j}\|_2^2$. Since $\tau_j \leq \|\tau\|_\infty \leq C$ and $\beta_j \geq 0$, then from Equation (27) we conclude that λ_j is at least $\ell_{t,j} - \theta - C\|\mathbf{x}_{t,j}\|_2^2$. In other words, λ_j is positive. Referring to the left-hand side of Equation (28), we conclude that $(C - \tau_j) = 0$, and $\tau_j = C$.

Overall, we have shown that there exists some $\theta \geq 0$ such that the optimal update weights take the form

$$\tau_{t,j} \;=\; \begin{cases} 0 & \text{if } \ell_{t,j} - \theta < 0 \\ \frac{\ell_{t,j} - \theta}{\|\mathbf{x}_{t,j}\|_2^2} & \text{if } 0 \leq \ell_{t,j} - \theta \leq C\|\mathbf{x}_{t,j}\|_2^2 \\ C & \text{if } C\|\mathbf{x}_{t,j}\|_2^2 < \ell_{t,j} - \theta \end{cases} \;\; . \tag{30}$$

That is, if the individual loss of task j is smaller than θ then no update is applied to the respective classifier. If the loss is moderate then the size of the update step is proportional to the loss attained, and inverse proportional to the squared norm of the respective instance. In any case, the size of the update step cannot exceed the fixed upper limit C.

We are thus left with the problem of finding the value of θ in Equation (30) which yields the update weights that maximize Equation (26). We denote this value by $\theta^\star$. First note that if we lift the constraint $\sum_{j=1}^k \tau_{t,j} \leq rC$ then the maximum of Equation (26) is obtained by setting $\tau_{t,j} = \min\{\ell_{t,j}/\|\mathbf{x}_{t,j}\|_2^2,\, C\}$ for all j, which is equivalent to setting $\theta = 0$ in Equation (30). Therefore, if

$$\sum_{j=1}^{k} \min \left\{ \frac{\ell_{t,j}}{\|\mathbf{x}_{t,j}\|_2^2},\, C \right\} \;\leq\; rC \;\; ,$$

the solution to Equation (26) is $\tau_{t,j} = \min\{\ell_{t,j}/\|\mathbf{x}_{t,j}\|_2^2,\, C\}$ for all j. Thus, we can focus our attention on the case where

$$\sum_{j=1}^{k} \min \left\{ \frac{\ell_{t,j}}{\|\mathbf{x}_{t,j}\|_2^2},\, C \right\} \;>\; rC \;\; .$$

In this case, $\theta^\star$ must be non-zero in order for the constraint $\sum_{j=1}^k \tau_j \leq rC$ to hold. Once again using the KKT complementarity condition, it follows that $\sum_{j=1}^k \tau_{t,j} = rC$. Now, for every value of θ, define the following two sets of indices:

$$\Psi(\theta) \;=\; \{1 \leq j \leq k \;:\; 0 < \ell_{t,j} - \theta\} \;\; ,$$

and

$$\Phi(\theta) \;=\; \{1 \le j \le k \;:\; C\|\mathbf{x}_{t,j}\|_2^2 < \ell_{t,j} - \theta\} \;\;.$$

Let Ψ and Φ denote the sets $\Psi(\theta^\star)$ and $\Phi(\theta^\star)$ respectively. The semantics of Ψ and Φ are readily available from Equation (30): the set Ψ includes all indices j for which $\tau_j > 0$ in the optimal solution, while Φ includes all indices j for which τ_j is clipped at C in the optimal solution. If we know the value of $\theta^\star$, we can easily obtain the sets Ψ and Φ from their definitions above. However, the converse is also true: if we are able to find the sets Ψ and Φ directly then we can use them to calculate the exact value of $\theta^\star$. Assuming we know Ψ and Φ, and using the fact that $\sum_{j=1}^k \tau_j = rC$, we get

$$\sum_{j \in \Psi \setminus \Phi} \frac{\ell_{t,j} - \theta^\star}{\|\mathbf{x}_{t,j}\|_2^2} \;+\; \sum_{j \in \Phi} C \;=\; rC \;\;.$$

Solving the above for $\theta^\star$ gives

$$\theta^\star \;=\; \frac{\sum_{j \in \Psi \setminus \Phi} \frac{\ell_{t,j}}{\|\mathbf{x}_{t,j}\|_2^2} - rC + \sum_{j \in \Phi} C}{\sum_{j \in \Psi \setminus \Phi} \frac{1}{\|\mathbf{x}_{t,j}\|_2^2}} \;\;. \tag{31}$$

We have thus reduced the optimization problem in Equation (26) to the problem of finding the sets Ψ and Φ. Once we find Ψ and Φ, we can easily calculate $\theta^\star$ using Equation (31) and then obtain $\boldsymbol{\tau}_t$ using Equation (30). Luckily, Ψ and Φ are subsets of $\{1,\ldots,k\}$ and can only be defined in a finite number of ways. A straightforward and excessively inefficient solution is to enumerate over all possible subsets of $\{1,\ldots,k\}$ as candidates for Ψ and Φ, for each pair of candidate sets to compute the corresponding values of θ and $\boldsymbol{\tau}$ using Equation (31) and Equation (30) respectively and then check if the obtained solution is consistent with our constraints ($\theta \ge 0$, $\sum_j \tau_j = rC$ and $0 \le \tau_j \le C$). Of the candidates that turn out to be consistent, we choose the one which maximizes the objective function in Equation (26). This approach is clearly infeasible even for reasonably small values of k. We therefore describe a more efficient procedure for finding Ψ and Φ, whose computational cost is only $O(k\log(k))$.

Let us examine two losses $\ell_{t,r}$ and $\ell_{t,s}$ such that $\ell_{t,r} \le \ell_{t,s}$ and there is no index j for which $\ell_{t,r} < \ell_{t,j} < \ell_{t,s}$. Then, all the sets $\Psi(\theta)$ for $\theta \in [\ell_{t,r}, \ell_{t,s})$ are identical, and equal $\{j \;:\; \ell_{t,j} \ge \ell_{t,r}\}$. Therefore, there are at most k different choices for $\Psi(\theta)$, which can be easily computed by sorting the losses. An analogous argument holds for the set Φ with respect to the values $\ell_{t,j} - C\|\mathbf{x}_{t,j}\|_2^2$. Furthermore, to enumerate all admissible sets $\Psi(\theta)$ and $\Phi(\theta)$ we need not examine their product space. Instead, let $\mathbf{q}$ denote the vector obtained by sorting the union of the sets $\{\ell_{t,j}\}_{j=1}^k$, $\{\ell_{t,j} - C\|\mathbf{x}_{t,j}\|_2^2\}_{j=1}^k$, and $\{0\}$ in ascending order. Extending the above rationale, the sets $\Psi(\theta)$ and $\Phi(\theta)$ are fixed for every $\theta \in [q_i, q_{i+1})$. We can examine every possible pair of candidates $\Psi(\theta), \Phi(\theta)$ by traversing the sorted vector $\mathbf{q}$ of critical values.

Concretely, define $\Psi(q_1) = \{1,\ldots,k\}$ and $\Phi(q_1) = \{1,\ldots,k\}$, and keep them sorted in memory. Use these sets to define θ and $\boldsymbol{\tau}$ as described above, and check if the solution satisfies our constraints. If so, return this value of $\boldsymbol{\tau}$ as the update step for the r-max loss. Otherwise, move on to the next value in $\mathbf{q}$ and evaluate the next pair of candidates. This procedure for choosing θ and $\boldsymbol{\tau}$ implies that if more than one solution satisfies the constraints, we will choose the one encountered first, namely the one for which θ is the smallest. Indeed it can be verified that the smaller θ, the greater the value of the objective function in Equation (26). Given the sets $\Psi(q_i)$ and $\Phi(q_i)$, we can obtain the sets $\Psi(q_{i+1})$ and $\Phi(q_{i+1})$, and recalculate θ, by simply removing from $\Psi(q_i)$ every j for which

$\ell_{t,j} < q_{i+1}$ and removing from $\Phi(q_i)$ every j for which $\ell_{t,j} - C\|\mathbf{x}_{t,j}\|_2^2 < q_{i+1}$. This operation can be done efficiently since the sets $\Psi(q_i)$ and $\Phi(q_i)$ are sorted in memory.

8. Experiments with Text Classification

In this section, we demonstrate the effectiveness of the implicit multitask algorithm on large-scale text categorization problems. Throughout this paper, we have argued that when faced with multiple tasks in parallel, we can often do better than to learn each task individually. The goal of the first two experiments is to demonstrate that this is indeed the case. The third experiment demonstrates that the superiority of the implicit update algorithm, presented in Section 5, over the multitask Perceptron, presented in Sections 3 and 4.

We used the *Reuters Corpus Vol. 1*, which is a collection of over 800K news articles collected from the Reuters newswire over a period of 12 months, in 1996-1997. An average article contains approximately 240 words, and the entire corpus contains over half a million distinct tokens (not including numbers and dates). Each article is associated with *one or more* of 104 possible low-level categories.[1] On average, each article is associated with 1.5 low-level categories. The categorization problem induced by this corpus is referred to as a *multiclass-multilabel* (MCML) problem, since there are multiple possible classes (the 104 categories) and each article may assigned multiple labels. Examples of categories that appear in the corpus are: WEATHER, MONEY MARKETS, and UNEMPLOYMENT. The articles in the corpus are given in their original chronological order, and our goal is to predict the label, or labels, associated with each newly presented article. Our first experiment addresses this problem.

The Reuters corpus also defines 5 high-level meta-categories: CORPORATE/INDUSTRIAL, ECONOMICS, GOVERNMENT/SOCIAL, MARKETS, and OTHER. About 20% of the articles in the corpus are associated with more than one of the five meta-categories. After discarding this 20%, we are left with over 600K documents, each with a single high-level label. This induces a 5-class single-label classification problem. Our second experiment addresses this multiclass single-label problem.

We began by applying some mild preprocessing to the articles in the corpus, which included removal of punctuation, numbers, dates, and stop-words, and a global conversion of the entire corpus to lower-case. Then, each article was mapped to a real vector using a logarithmic bag-of-words representation. Namely, the length of each vector equals the number of distinct tokens in the corpus, and each coordinate represents one of these tokens. If a token appears s times in a given article, then the respective coordinate in the vector is set to $\log_2(1+s)$.

8.1 Multiclass Multilabel Categorization

We trained a separate binary classifier for each of the 104 low-level classes, using the implicit update algorithm presented in Section 5. Given an unseen article, each classifier predicts whether its respective category applies to that article or not. We ran our algorithm using both the L_1 norm and the L_∞ norm as the global loss function. In both cases, the user-defined parameter C was set to 10^{-3}.

The performance of the entire classifier ensemble on each article was evaluated in two ways. First, we examined whether the 104-classifier ensemble predicted the *entire* set of categories per-

1. The original corpus specifies 126 labels which are organized in a hierarchical tree-structure. Of these labels, 104 are low-level categories, which correspond to leaves in the tree. The remaining labels are meta-categories which correspond to inner nodes in the tree.

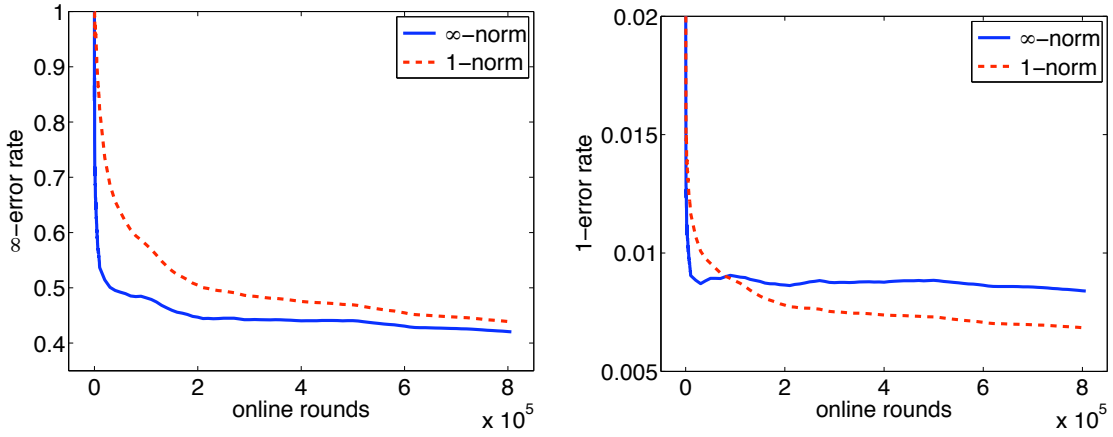


Figure 4: The ∞-error (left) and 1-error (right) error-rates attained by the implicit multitask algorithm using the L_∞ norm (solid) and the L_1 norm (dashed) global loss functions. Note that the two plots are on a very different scale: the two lines on the left-hand plot differ by approximately 3%, whereas the lines on the right-hand plot differ by approximately 0.05%.

fectly. An affirmative answer to this test implies that all 104 classifiers made correct predictions simultaneously. Formally, let $\mathbf{e}_t$ be the vector in $\{0,1\}^{104}$ such that $e_{t,j} = 1$ if and only if $y_{t,j}\mathbf{w}_{t,j} \cdot \mathbf{x}_{t,j} \leq 0$. In other words, $\mathbf{e}_t$ indicates which of the 104 binary classifiers made prediction mistakes on round t. Now define the *∞-error* suffered on round t as $\|\mathbf{e}_t\|_\infty$. Second, we assessed the fraction of categories for which incorrect binary predictions were made. Formally, define the *1-error* suffered on round t as $\|\mathbf{e}_t\|_1/104$. Both measures of error are reasonable, and one should be preferred over the other based on the specific requirements of the underlying application. Since each coordinate of ℓ_t upper-bounds the respective coordinate in $\mathbf{e}_t$, it holds that $\|\mathbf{e}_t\|_\infty \leq \|\ell_t\|_\infty$ and that $\|\mathbf{e}_t\|_1 \leq \|\ell_t\|_1$. Therefore, the L_∞ norm update seems to be a more appropriate choice for minimizing the ∞-error, while the L_1 norm update is the more appropriate choice for minimizing the 1-error. Our experiments confirm this intuitive argument.

The results of our experiments are summarized in Figure 4. The left-hand plot in the figure shows the ∞-error-rate of the L_∞ norm and L_1 norm multitask updates, as the number of examples grows from zero to 800K. The figure clearly shows that the L_∞ norm algorithm does a better job throughout the entire online learning process. The advantage of the L_∞ norm algorithm diminishes as more examples are observed.

The right-hand plot in Figure 4 compares the 1-error-rate of the two updates. In this case, the L_∞ norm update initially takes the lead, but is quickly surpassed by the L_1 norm update. The fact that the L_1 norm update ultimately gains the advantage coincides with our initial intuition. The reason why the L_∞ norm update outperforms the L_1 norm update at first can also be easily explained. The L_1 norm update is quite aggressive, as it modifies every binary classifier that suffers a positive individual loss on every round. Moreover, the L_1 norm update enforces the constraint $\|\boldsymbol{\tau}_t\|_\infty \leq C$. On the other hand, the L_∞ norm update is more cautious, since it enforces the stricter constraint $\|\boldsymbol{\tau}_t\|_1 \leq C$. The aggression of the L_1 norm update causes its initial behavior to be somewhat erratic.

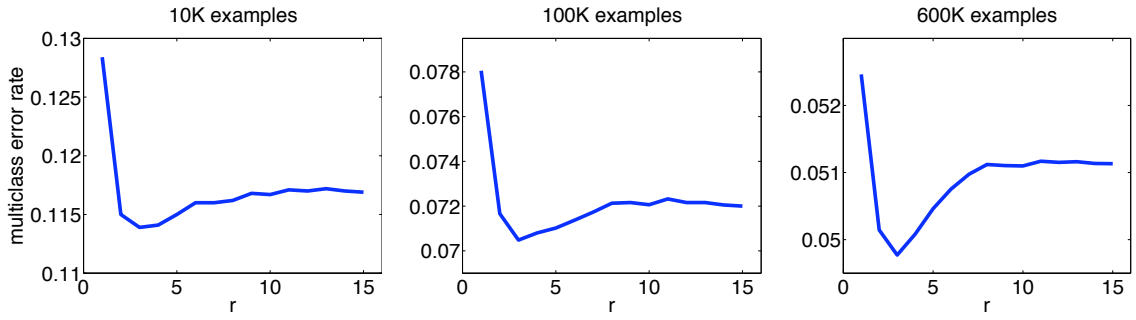


Figure 5: The multiclass error rate of the online ECOC-based classifier, using a 15 column code matrix, with various r-max norms, after observing 10K, 100K, and 600K examples.

At first, many of the L_1 norm updates actually move the classifier ensemble away from its target. Inevitably, it takes the L_1 norm classifier slightly longer to find its path.

8.2 Multiclass Meta-Categorization with ECOC and r-max Norms

Following one of the motivating examples given in the introduction, we used the ECOC method (Dietterich and Bakiri, 1995) to reduce the 5 high-level meta-categories classification task from the Reuters corpus to multiple binary classification tasks. We used the 5×15 Hadamard code matrix, defined as follows:

$$M = \begin{pmatrix} + & + & + & + & + & + & + & + & + & + & + & + & + & + & + \\ + & + & + & + & + & + & + & - & - & - & - & - & - & - & - \\ + & + & + & - & - & - & - & + & + & + & + & - & - & - & - \\ + & - & - & + & + & - & - & + & + & - & - & + & + & - & - \\ - & + & - & + & - & + & - & + & - & + & - & + & - & + & - \end{pmatrix}.$$

This code matrix is derived by taking all 2^4 possible 5-coordinate columns with $+$ in the first position, except for the all-plus column. This is the largest 5-row code matrix that does not induce redundant or trivial binary classification problems. The distance between any two rows of the matrix is 8, therefore this code is guaranteed to correct 4 binary prediction mistakes. We can determine if more than 4 binary mistakes are made on round t by comparing the fifth largest element of ℓ_t with 1. As mentioned in the introduction, taking the fifth largest loss does not constitute a norm, and cannot be used as a global loss within our setting. However, a norm with a similar flavor is the r-max norm, with $r = 5$. Our experiments show that it is actually advantageous to be slightly over-cautious, by setting r to 3 or 4.

The results of our experiments are summarized in Figure 5. We trained 15 binary classifiers, one per each column of M, using the implicit update algorithm presented in Section 5. We used the r-max norm as the algorithm's global loss function, with r set to every integer value between 1 and 15. For each example, all 15 binary classifiers made predictions, and M was used to decode a multiclass prediction, as described in Dietterich and Bakiri (1995). A multiclass error occurs if the predicted label differs from the true label. In Figure 5 we depict the average number of errors that occurred after observing 10K, 100K, and 600K examples, for each value of r. We can see that

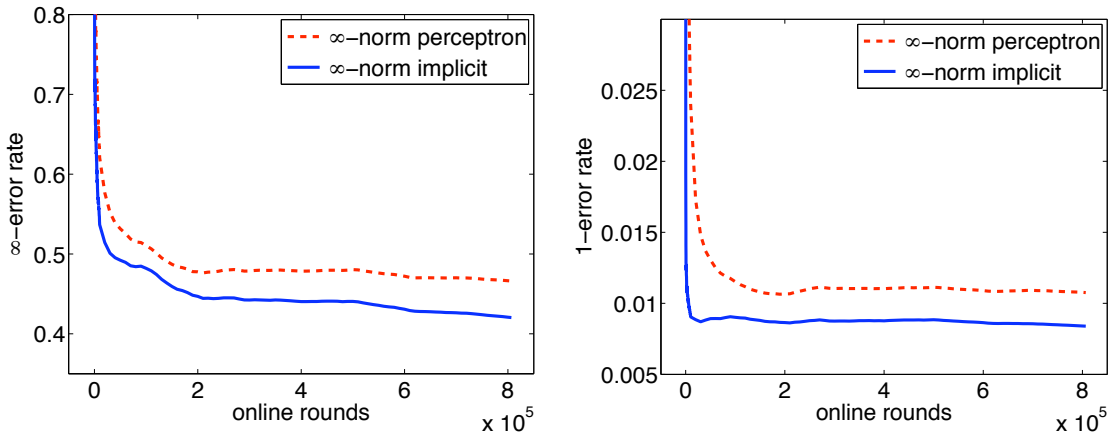


Figure 6: The ∞-error (left) and 1-error (right) attained by the multitask Perceptron (dashed) and the implicit update algorithm (solid) when using the L_∞ norm as a global loss function.

using either the L_1 norm ($r = 15$) or the L_∞ norm ($r = 1$) is suboptimal, and the best performance is consistently reached by setting r to be slightly smaller than half the code distance. Although the theoretically motivated choice of $r = 5$ is not the best, it still yields better results than the two extreme choices, $r = 1$ and $r = 15$.

When we replaced the Hadamard code matrix with the One-vs-Rest code matrix, defined by $2I - 1$ (where I is the 5×5 identity matrix and 1 is the 5×5 all-ones matrix) then the multiclass error after observing 600K examples increases from 5% to around 8%. This justifies using the ECOC method in the first place.

We conclude this experiment by noting that although setting $r = 1$ produces the largest number of multiclass prediction mistakes, it still delivers the best performance if we evaluate the 15 classifier ensemble using the ∞-error defined above.

8.3 The Implicit Update vs. the Multitask Perceptron

From a loss minimization standpoint, Theorem 9 proves that the implicit update, presented in Section 5, is at least as good as the multitask Perceptron variants, presented in Secs. 3 and 4. The following experiment demonstrates that the implicit update is also superior in practice.

We repeated the multitask multi-label experiment described in Section 8.1, using the multitask Perceptron in place of the implicit update algorithm. The infinite horizon extension discussed in Section 4 does not have a significant effect on empirical performance, so we consider only the finite horizon version of the multitask Perceptron, described in Section 3.

When the global loss function is defined using the L_1 norm, both the implicit update and the multitask Perceptron update decouple to independent updates for each individual task. In this case, both algorithms are very similar, their empirical performance is almost identical, and the comparison between them is not very interesting. Therefore, we focus on a global loss defined by the L_∞ norm.

A comparison between the performance of the implicit update and the multitask Perceptron update, both using the L_∞-norm loss, is given in Figure 6. The plot on the left-hand side of the figure compares the two algorithms' ∞-error-rate, and the plot on the right-hand side of the figure

compares their 1-error-rate. The implicit algorithm holds a clear lead over the multitask Perceptron with respect to both error measures, throughout the learning process. These results give empirical validation to the formal comparison of the two algorithms.

9. Discussion

When faced with several online tasks in parallel, it is not always best to distribute the learning effort evenly. In many cases, it may be beneficial to allocate more effort to tasks when they are seen to play "key" roles. In this paper, we presented an online algorithmic framework that does precisely that. The priority given to each task is governed by its relative performance and by the choice of a global loss function.

We presented three families of algorithms, each of which includes an algorithm for every global loss defined by an absolute norm. The first two families are illustrative and theoretically appealing. The third family of algorithms uses the most sophisticated update of the three, and is the one recommended for practical use. We demonstrated the superior performance of the third family of algorithms empirically.

We showed that, in the worst case, the finite horizon multitask Perceptron of Section 3 and the implicit update algorithm of Section 5 both perform asymptotically as well as the best fixed hypothesis ensemble. In other words, these algorithms are no-regret algorithms with respect to any global loss function defined by an absolute norm. The same cannot be said for the naive alternative, where we use multiple independent single-task learning algorithms to solve the multitask problem. We also demonstrated the benefit of the multitask approach over the naive alternative on two large-scale text categorization problems.

Throughout the paper, we assumed that the multiple online tasks are perfectly synchronized, and that a complete k-tuple of examples is observed on every round. This is indeed the case in each of the concrete examples described in the introduction and empirically tested in our experiments. However, in other real-world situations, this may not be the case. Namely, there could occur situations where not all of the tasks are active on every single round. In other words, there may be a subset of "dormant tasks" on each round. For example, say that we are operating an online store and that we have multiple registered customers. Each product in our store is represented by a feature vector, and we train an individual binary classifier for each of our customers. When costumer j visits a product-page on our website, the respective classifier is used to predict whether that customer intends to purchase the product or not. The prediction is then used to decide whether or not to lure the customer away from that page. This setting induces a natural online multitask learning problem. Moreover, only a fraction of the customers is online at any given moment. We consider the tasks of those customers that are not online to be dormant or inactive tasks. At a first glance, the inactive tasks setting may seem to be more complicated than the fully synchronized setting discussed throughout the paper. However, our algorithms and analysis accommodate this extension quite naturally. We simply need to define $\ell_{t,j} = 0$ for every inactive task and apply the multitask update verbatim. Due to the conservativeness assumption, the hypotheses of the inactive tasks will be left intact. Additionally, note that all of the norms discussed in this paper have the property that $\|\mathbf{v}\| = \|\mathbf{v}'\|$, where $\mathbf{v}'$ is the vector obtained by removing all of the zero entries from $\mathbf{v}$. Therefore, we can imagine that the length of the vector ℓ_t changes from round to round, and that the update on each round is applied as if the tasks that are sleeping on that round never existed in the first place. We would also like to note that, although our presentation focuses on multiple binary classifica-

tion tasks, our algorithms and techniques can be adapted to other online learning problems as well. Specifically, a multitask implicit update can be derived for regression and uniclass problems using ideas from Crammer et al. (2006).

The next-step would be to extend our framework from absolute norms to general norms. For example, the family of Mahalanobis norms, defined by $\|\mathbf{z}\|^2 = \mathbf{z}^\top P\mathbf{z}$ (where P is a positive definite matrix) includes norms that are not absolute but which could have interesting applications in our setting. More generally, there exist meaningful global loss functions which are not norms at all.

Another interesting research direction would be to return to the roots of statistical multitask learning, and to try to model generative similarities between the multiple tasks within the online framework. In our work, we completely disregarded any relatedness between the multiple tasks, and only considered the shared consequences of errors. In the game-theoretic spirit of online learning, modeling these similarities would have to be done without making statistical assumptions on the data source.

Appendix A. The K-Method of Norm Interpolation

In this section, we briefly survey Peetre's K-method of norm interpolation. This method takes a pair of norms and smoothly interpolates between them, producing a new family of norms which can be used in our setting. An example of such an interpolation is the family of r-max norms, previously mentioned in this paper. The main practical purpose of this section is to prove that the dual of the r-max norm takes the form given in Equation (3). We do not present the K-method in all its generality, but rather focus only on topics which are relevant to the online multitask learning setting. The interested reader is referred to Bennett and Sharpley (1998) for a more detailed account of interpolation theory.

We begin by presenting Peetre's K-functional and J-functional, and proving that they induce dual norms. Let $\|\cdot\|_{p_1} : \mathbb{R}^k \to \mathbb{R}_+$ and $\|\cdot\|_{p_2} : \mathbb{R}^k \to \mathbb{R}_+$ be two p-norms, and let $\|\cdot\|_{q_1}$ and $\|\cdot\|_{q_2}$ be their respective duals. The K-functional with respect to p_1 and p_2, and with respect to the constant $\alpha > 0$, is defined as

$$\|\mathbf{v}\|_{K(p_1,p_2,\alpha)} \;=\; \min_{\mathbf{w}+\mathbf{z}=\mathbf{v}} \left(\|\mathbf{w}\|_{p_1} + \alpha \|\mathbf{z}\|_{p_2} \right) \;.$$

The J-functional with respect to q_1, q_2, and with respect to the constant $\beta > 0$, is defined as

$$\|\mathbf{u}\|_{J(q_1,q_2,\beta)} \;=\; \max \left\{ \|\mathbf{u}\|_{q_1},\; \beta \|\mathbf{u}\|_{q_2} \right\} \;.$$

The J-functional is obviously a norm: positivity and linearity follow immediately from the fact that $\|\cdot\|_{q_1}$ and $\|\cdot\|_{q_2}$ posses these properties. The triangle inequality follows from

$$\begin{aligned}
\|\mathbf{v}+\mathbf{u}\|_{J(q_1,q_2,\beta)} &= \max \left\{ \|\mathbf{v}+\mathbf{u}\|_{q_1},\; \beta \|\mathbf{v}+\mathbf{u}\|_{q_2} \right\} \\
&\leq \max \left\{ \|\mathbf{v}\|_{q_1} + \|\mathbf{u}\|_{q_1},\; \beta\|\mathbf{v}\|_{q_2} + \beta\|\mathbf{u}\|_{q_2} \right\} \\
&\leq \max \left\{ \|\mathbf{v}\|_{q_1},\; \beta\|\mathbf{v}\|_{q_2} \right\} \;+\; \max \left\{ \|\mathbf{u}\|_{q_1},\; \beta\|\mathbf{u}\|_{q_2} \right\} \\
&= \|\mathbf{v}\|_{J(q_1,q_2,\beta)} \;+\; \|\mathbf{u}\|_{J(q_1,q_2,\beta)} \;.
\end{aligned}$$

Since the J-functional is defined with respect to two absolute norms, it too is an absolute norm.

Instead of explicitly proving that $\|\cdot\|_{K(p_1,p_2,\alpha)}$ is also a norm, we prove that it is the dual of $\|\cdot\|_{J(q_1,q_2,\beta)}$ when $\alpha = 1/\beta$. Since the dual of an absolute norm is itself an absolute norm, and since

the dual of the dual norm is the original norm (Horn and Johnson, 1985), our proof implies that $\|\cdot\|_{K(p_1,p_2,\alpha)}$ is indeed a norm, that it is absolute, and that its dual is $\|\cdot\|_{J(q_1,q_2,1/\alpha)}$.

Theorem 10 *Using the notation defined above,*

$$\|\cdot\|^*_{J(q_1,q_2,\beta)} \equiv \|\cdot\|_{K(p_1,p_2,1/\beta)} .$$

Proof We abbreviate $\|\mathbf{v}\|_J = \|\mathbf{v}\|_{J(q_1,q_2,\beta)}$ and $\|\mathbf{v}\|_K = \|\mathbf{v}\|_{K(p_1,p_2,1/\beta)}$ throughout the proof. First, we show that $\|\mathbf{v}\|^*_J \le \|\mathbf{v}\|_K$ for all $\mathbf{v} \in \mathbb{R}^k$. Let $\mathbf{v}, \mathbf{w}$ and $\mathbf{z}$ be vectors in $\mathbb{R}^k$ such that $\mathbf{v} = \mathbf{w} + \mathbf{z}$. Then for any $\mathbf{u} \in \mathbb{R}^k$, we can use Hölder's inequality to obtain

$$\begin{aligned} \mathbf{u}\cdot\mathbf{v} &= \mathbf{u}\cdot\mathbf{w} + \mathbf{u}\cdot\mathbf{z} \\ &\le \|\mathbf{u}\|_{q_1}\|\mathbf{w}\|_{p_1} + \|\mathbf{u}\|_{q_2}\|\mathbf{z}\|_{p_2} . \end{aligned}$$

By definition, it holds that

$$\|\mathbf{u}\|_{q_1} \le \|\mathbf{u}\|_J \quad \text{and} \quad \|\mathbf{u}\|_{q_2} \le \frac{1}{\beta}\|\mathbf{u}\|_J ,$$

and so

$$\mathbf{u}\cdot\mathbf{v} \le \left(\|\mathbf{w}\|_{p_1} + \frac{1}{\beta}\|\mathbf{z}\|_{p_2}\right)\|\mathbf{u}\|_J .$$

Since the only restriction on $\mathbf{u}, \mathbf{v}, \mathbf{w}$ and $\mathbf{z}$ is that $\mathbf{v} = \mathbf{w} + \mathbf{z}$, we can fix $\mathbf{v}$, choose $\mathbf{u}$ to be the vector which maximizes the left-hand side above subject to $\|\mathbf{u}\|_J \le 1$, and choose $\mathbf{w}$ and $\mathbf{z}$ which minimize the right-hand side above subject to $\mathbf{v} = \mathbf{w} + \mathbf{z}$. This results in

$$\max_{\mathbf{u}\in\mathbb{R}^k : \|\mathbf{u}\|_J \le 1} \mathbf{u}\cdot\mathbf{v} \le \min_{\mathbf{w}+\mathbf{z}=\mathbf{v}} \left(\|\mathbf{w}\|_{p_1} + \frac{1}{\beta}\|\mathbf{z}\|_{p_2}\right) .$$

The left-hand side above is the formal definition of $\|\mathbf{v}\|^*_J$, the right-hand side is the definition of $\|\mathbf{v}\|_K$, and we have proven that $\|\mathbf{v}\|^*_J \le \|\mathbf{v}\|_K$.

To prove the opposite direction, fix $\mathbf{v}$ and let $\mathbf{u}$ be the vector with $\|\mathbf{u}\|_J \le 1$ which maximizes $\mathbf{u}\cdot\mathbf{v}$. We now consider two cases. If $\|\mathbf{u}\|_{q_1} \ge \beta\|\mathbf{u}\|_{q_2}$ then

$$\|\mathbf{v}\|^*_J = \max_{\mathbf{u} : \|\mathbf{u}\|_{q_1} \le 1} \mathbf{u}\cdot\mathbf{v} .$$

Using the duality of $\|\cdot\|_{q_1}$ and $\|\cdot\|_{p_1}$, the right hand-side above equals $\|\mathbf{v}\|_{p_1}$. Since we can choose $\mathbf{w} = \mathbf{v}$ and $\mathbf{z} = 0$, it certainly holds that

$$\|\mathbf{v}\|_{p_1} \ge \min_{\mathbf{w}+\mathbf{z}=\mathbf{v}} \left(\|\mathbf{w}\|_{p_1} + \frac{1}{\beta}\|\mathbf{z}\|_{p_2}\right) = \|\mathbf{v}\|_K .$$

On the other hand, if $\|\mathbf{u}\|_{q_1} \le \beta\|\mathbf{u}\|_{q_2}$ then

$$\|\mathbf{v}\|^*_J = \frac{1}{\beta} \max_{\mathbf{u} : \|\mathbf{u}\|_{p_2} \le 1} \mathbf{u}\cdot\mathbf{v} .$$

Using the duality of $\|\cdot\|_{q_2}$ and $\|\cdot\|_{p_2}$, the right hand-side above equals $\frac{1}{\beta}\|\mathbf{v}\|_{p_2}$. Since we can choose $\mathbf{w}=0$ and $\mathbf{z}=\mathbf{v}$, it holds that

$$\frac{1}{\beta}\|\mathbf{v}\|_{p_2} \;\geq\; \min_{\mathbf{w}+\mathbf{z}=\mathbf{v}} \left(\|\mathbf{w}\|_{p_2}+\frac{1}{\beta}\|\mathbf{z}\|_{p_2}\right) \;=\; \|\mathbf{v}\|_K \;\;.$$

Overall, we have shown that $\|\mathbf{v}\|_J^* \geq \|\mathbf{v}\|_K$. ■

The r-max norm discussed in the paper is an instance of the K-functional, and can be defined as

$$\|\mathbf{v}\|_{r\text{-max}} \;=\; \|\mathbf{v}\|_{K(1,\infty,r)} \;\;.$$

To see why this is true, let ϕ be the absolute value of the r'th absolutely largest coordinate in $\mathbf{v}$. Now define for each $1 \leq j \leq k$

$$w_j = \text{sign}(v_j)\max\{0,|v_j|-\phi\} \quad \text{and} \quad z_j \;=\; \text{sign}(v_j)\min\{|v_j|,\phi\} \;\;.$$

Note that $\mathbf{w}+\mathbf{z}=\mathbf{v}$, and that

$$\|\mathbf{v}\|_{r\text{-max}} \;=\; \|\mathbf{w}\|_1 + r\|\mathbf{z}\|_\infty \;\;.$$

This proves that $\|\mathbf{v}\|_{r\text{-max}} \geq \|\mathbf{v}\|_{K(1,\infty,r)}$.

Turning to the opposite inequality, let $\pi(1),\ldots,\pi(r)$ be the indices of the r absolutely largest elements of $\mathbf{v}$, and let $\mathbf{w}$ and $\mathbf{z}$ be vectors such that $\mathbf{w}+\mathbf{z}=\mathbf{v}$. We now have that

$$\begin{aligned}
\|\mathbf{v}\|_{r\text{-max}} &= \sum_{j=1}^{r} |v_{\pi(j)}| \\
&= \sum_{j=1}^{r} |w_{\pi(j)}+z_{\pi(j)}| \\
&\leq \sum_{j=1}^{r} |w_{\pi(j)}| \;+\; \sum_{j=1}^{r} |z_{\pi(j)}| \\
&\leq \sum_{j=1}^{r} |w_{\pi(j)}| \;+\; r \max_{j=1,\ldots,r} |z_{\pi(j)}| \\
&\leq \sum_{j=1}^{k} |w_j| \;+\; r \max_{j=1,\ldots,k} |z_j| \;=\; \|\mathbf{w}\|_1 \;+\; r\|\mathbf{z}\|_\infty \;\;.
\end{aligned}$$

The above holds for any $\mathbf{w}$ and $\mathbf{z}$ which sum to $\mathbf{v}$, and specifically to those which minimize $\|\mathbf{w}\|_1 + r\|\mathbf{z}\|_\infty$. We conclude that $\|\mathbf{v}\|_{r\text{-max}} \leq \|\mathbf{v}\|_{K(1,\infty,r)}$, and therefore $\|\mathbf{v}\|_{r\text{-max}} = \|\mathbf{v}\|_{K(1,\infty,r)}$.

Finally, we calculate an upper bound on the remoteness of $\|\cdot\|_{J(q_1,q_2,\beta)}$. This enables us to obtain concrete loss bounds for interpolation norms from the theorems proven in this paper. Recall that

$$\rho(\|\cdot\|_{J(q_1,q_2,\beta)},k) \;=\; \max_{\mathbf{u}\in\mathbb{R}^k} \frac{\|\mathbf{u}\|_2}{\|\mathbf{u}\|_{J(q_1,q_2,\beta)}} \;\;.$$

Using the definition of the J-functional, the above becomes

$$\max_{\mathbf{u}\in\mathbb{R}^k} \min\left\{\frac{\|\mathbf{u}\|_2}{\|\mathbf{u}\|_{q_1}}, \frac{\|\mathbf{u}\|_2}{\beta\|\mathbf{u}\|_{q_2}}\right\} \;\;.$$

Using the weak minimax theorem, we can upper-bound the above by

$$\min\left\{\max_{\mathbf{u}\in\mathbb{R}^k}\frac{\|\mathbf{u}\|_2}{\|\mathbf{u}\|_{q_1}},\max_{\mathbf{u}\in\mathbb{R}^k}\frac{\|\mathbf{u}\|_2}{\beta\|\mathbf{u}\|_{q_2}}\right\}\ .$$

Once again using the definition of remoteness, the above can be rewritten as

$$\min\left\{\rho(\|\cdot\|_{q_1},k),\ \frac{\rho(\|\cdot\|_{q_2},k)}{\beta}\right\}\ .$$

Using Lemma 2, we can obtain an explicit upper bound on the remoteness of any interpolation of p-norms.

References

J. Baxter. A model of inductive bias learning. *Journal of Artificial Intelligence Research*, 12: 149–198, 2000.

S. Ben-David and R. Schuller. Exploiting task relatedness for multiple task learning. In *Proceedings of the Sixteenth Annual Conference on Computational Learning Theory*, 2003.

C. Bennett and R. Sharpley. *Interpolation of Operators*. Academic Press, 1998.

S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

R. Caruana. Multitask learning. *Machine Learning*, 28(1):41–75, 1997.

O. Chapelle and Z. Harchaoui. A machine learning approach to conjoint analysis. In *Advances in Neural Information Processing Systems*, volume 17, 2005.

K. Crammer and Y. Singer. On the algorithmic implementation of multiclass kernel-based vector machines. *Journal of Machine Learning Research*, 2:265–292, 2001.

K. Crammer and Y. Singer. Ultraconservative online algorithms for multiclass problems. *Journal of Machine Learning Research*, 3:951–991, 2003.

K. Crammer, O. Dekel, J. Keshet, S. Shalev-Shwartz, and Y. Singer. Online passive aggressive algorithms. *Journal of Machine Learning Research*, 7:551–585, Mar 2006.

T. G. Dietterich and G. Bakiri. Solving multiclass learning problems via error-correcting output codes. *Journal of Artificial Intelligence Research*, 2:263–286, January 1995.

T. Evgeniou, C.Micchelli, and M. Pontil. Learning multiple tasks with kernel methods. *Journal of Machine Learning Research*, 6:615–637, 2005.

D. P. Helmbold, J. Kivinen, and M. Warmuth. Relative loss bounds for single neurons. *IEEE Transactions on Neural Networks*, 10(6):1291–1304, 1999.

R. Herbrich, T. Graepel, and K. Obermayer. Large marging rank boundaries for ordinal regression. In A. Smola, B. Schölkopf, and D. Schuurmans, editors, *Advances in Large Margin Classifiers*. MIT Press, 2000.

T. Heskes. Solving a huge number of silmilar tasks: A combination of multitask learning and a hierarchical bayesian approach. In *Proceedings of the Fifteenth International Conference on Machine Learning*, pages 233–241, 1998.

R. A. Horn and C. R. Johnson. *Matrix Analysis*. Cambridge University Press, 1985.

J. Kivinen and M. Warmuth. Relative loss bounds for multidimensional regression problems. *Journal of Machine Learning*, 45(3):301–329, July 2001.

A. B. J. Novikoff. On convergence proofs on perceptrons. In *Proceedings of the Symposium on the Mathematical Theory of Automata*, volume XII, pages 615–622, 1962.

F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65:386–407, 1958. (Reprinted in *Neurocomputing* (MIT Press, 1988).).

I. Tsochantaridis, T. Hofmann, T. Joachims, and Y. Altun. Support vector machine learning for interdependent and structured output spaces. In *Proceedings of the Twenty-First International Conference on Machine Learning*, 2004.

Journal of Machine Learning Research 8 (2007) 2265-2295 Submitted 6/06; Revised 5/07; Published 10/07

Euclidean Embedding of Co-occurrence Data

Amir Globerson* GAMIR@CSAIL.MIT.EDU
Computer Science and Artificial Intelligence Laboratory
Massachusetts Institute of Technology
Cambridge, MA 02139

Gal Chechik* GAL@AI.STANFORD.EDU
Department of Computer Science
Stanford University
Stanford CA, 94306

Fernando Pereira PEREIRA@CIS.UPENN.EDU
Department of Computer and Information Science
University of Pennsylvania
Philadelphia PA, 19104

Naftali Tishby TISHBY@CS.HUJI.AC.IL
School of Computer Science and Engineering and
The Interdisciplinary Center for Neural Computation
The Hebrew University of Jerusalem
Givat Ram, Jerusalem 91904, Israel

Editor: John Lafferty

Abstract

Embedding algorithms search for a low dimensional continuous representation of data, but most algorithms only handle objects of a single type for which pairwise distances are specified. This paper describes a method for embedding objects of different types, such as images and text, into a single common Euclidean space, based on their co-occurrence statistics. The joint distributions are modeled as exponentials of Euclidean distances in the low-dimensional embedding space, which links the problem to convex optimization over positive semidefinite matrices. The local structure of the embedding corresponds to the statistical correlations via random walks in the Euclidean space. We quantify the performance of our method on two text data sets, and show that it consistently and significantly outperforms standard methods of statistical correspondence modeling, such as multidimensional scaling, IsoMap and correspondence analysis.

Keywords: embedding algorithms, manifold learning, exponential families, multidimensional scaling, matrix factorization, semidefinite programming

1. Introduction

Embeddings of objects in a low-dimensional space are an important tool in unsupervised learning and in preprocessing data for supervised learning algorithms. They are especially valuable for exploratory data analysis and visualization by providing easily interpretable representations of the

*. Both authors contributed equally. Gal Chechik's current address is Google Inc., 1600 Amphitheatre Parkway, Mountain View, CA, 94043.

relationships among objects. Most current embedding techniques build low dimensional mappings that preserve certain relationships among objects. The methods differ in the relationships they choose to preserve, which range from pairwise distances in multidimensional scaling (MDS) (Cox and Cox, 1984) to neighborhood structure in locally linear embedding (Roweis and Saul, 2000) and geodesic structure in IsoMap (Tenenbaum et al., 2000). All these methods operate on objects of a single type endowed with a measure of similarity or dissimilarity.

However, embedding should not be confined to objects of a single type. Instead, it may involve different types of objects provided that those types share semantic attributes. For instance, images and words are syntactically very different, but they can be associated through their meanings. A joint embedding of different object types could therefore be useful when instances are mapped based on their semantic similarity. Once a joint embedding is achieved, it also naturally defines a measure of similarity between objects of the same type. For instance, joint embedding of images and words induces a distance measure between images that captures their semantic similarity.

Heterogeneous objects with a common similarity measure arise in many fields. For example, modern Web pages contain varied data types including text, diagrams and images, and links to other complex objects and multimedia. The objects of different types on a given page have often related meanings, which is the reason they can be found together in the first place. In biology, genes and their protein products are often characterized at multiple levels including mRNA expression levels, structural protein domains, phylogenetic profiles and cellular location. All these can often be related through common functional processes. These processes could be localized to a specific cellular compartment, activate a given subset of genes, or use a subset of protein domains. In this case the specific biological process provides a common "meaning" for several different types of data.

A key difficulty in constructing joint embeddings of heterogeneous objects is to obtain a good similarity measure. Embedding algorithms often use Euclidean distances in some feature space as a measure of similarity. However, with heterogeneous object types, objects of different types may have very different representations (such as categorical variables for some and continuous vectors for others), making this approach infeasible.

The current paper addresses these problems by using object co-occurrence statistics as a source of information about similarity. We name our method *Co-occurrence Data Embedding*, or **CODE**. The key idea is that objects which co-occur frequently are likely to have related semantics. For example, images of dogs are likely to be found in pages that contain words like $\{$*dog, canine, bark*$\}$, reflecting a common underlying semantic class. Co-occurrence data may be related to the geometry of an underlying map in several ways. First, one can simply regard co-occurrence rates as approximating pairwise distances, since rates are non-negative and can be used as input to standard metric-based embedding algorithms. However, since co-occurrence rates do not satisfy metric constraints, interpreting them as distances is quite unnatural, leading to relatively poor results as shown in our experiments.

Here we take a different approach that is more directly related to the statistical nature of the co-occurrence data. We treat the observed object pairs as drawn from a joint distribution that is determined by the underlying low-dimensional map. The distribution is constructed such that a pair of objects that are embedded as two nearby points in the map have a higher statistical interaction than a pair that is embedded as two distant points. Specifically, we transform distances into probabilities in a way that decays exponentially with distance. This exponential form maps sums of distances

into products of probabilities, supporting a generative interpretation of the model as a random walk in the low-dimensional space.

Given empirical co-occurrence counts, we seek embeddings that maximize the likelihood of the observed data. The log-likelihood in this case is a non-concave function, and we describe and evaluate two approaches for maximizing it. One approach is to use a standard conjugate gradient ascent algorithm to find a local optimum. Another approach is to approximate the likelihood maximization using a convex optimization problem, where a convex non-linear function is minimized over the cone of semidefinite matrices. This relaxation is shown to yield similar empirical results to the gradient based method.

We apply CODE to several heterogeneous embedding problems. First, we consider joint embeddings of two object types, namely *words-documents* and *words-authors* in data sets of documents. We next show how CODE can be extended to jointly embed more than two objects, as demonstrated by jointly embedding words, documents, and authors into a single map. We also obtain quantitative measures of performance by testing the degree to which the embedding captures *ground-truth* structures in the data. We use these measures to compare CODE to other embedding algorithms, and find that it consistently and significantly outperforms other methods.

An earlier version of this work was described by Globerson et al. (2005).

2. Problem Formulation

Let X and Y be two categorical variables with finite cardinalities $|X|$ and $|Y|$. We observe a set of pairs $\{x_i, y_i\}_{i=1}^n$ drawn IID from the joint distribution of X and Y. The sample is summarized via its empirical distribution[1] $\overline{p}(x,y)$, which we wish to use for learning about the underlying unknown joint distribution of X and Y. In this paper, we consider models of the unknown distribution that rely on a joint embedding of the two variables. Formally, this embedding is specified by two functions $\phi : X \to \mathbb{R}^q$ and $\psi : Y \to \mathbb{R}^q$ that map both categorical variables into the common low dimensional space $\mathbb{R}^q$, as illustrated in Figure 1.

The goal of a joint embedding is to find a geometry that reflects well the statistical relationship between the variables. To do this, we model the observed pairs as a sample from the parametric distribution $p(x,y;\phi,\psi)$, abbreviated $p(x,y)$ when the parameters are clear from the context. Thus, our models relate the probability $p(x,y)$ of a pair (x,y) to the embedding locations $\phi(x)$ and $\psi(y)$.

In this work, we focus on the special case in which the model distribution depends on the squared Euclidean distance $d^2_{x,y}$ between the embedding points $\phi(x)$ and $\psi(y)$:

$$d^2_{x,y} \quad = \quad \|\phi(x) - \psi(y)\|^2 = \sum_{k=1}^{q} (\phi_k(x) - \psi_k(y))^2 \ .$$

Specifically, we consider models where the probability $p(x,y)$ is proportional to $e^{-d^2_{x,y}}$, up to additional factors described in detail below. This reflects the intuition that closer objects should co-occur more frequently than distant objects. However, a major complication of embedding models is that the embedding locations $\phi(x)$ and $\psi(y)$ should be insensitive to the marginals $p(x) = \sum_y p(x,y)$ and $p(y) = \sum_x p(x,y)$. To see why, consider a value $x \in X$ with a low marginal probability $p(x) \ll 1$, which implies a low $p(x,y)$ for all y. In a model where $p(x,y)$ is proportional to $e^{-d^2_{x,y}}$ this will force

1. The empirical distribution $\overline{p}(x,y)$ is proportional to the number of times the pair (x,y) was observed. The representations $\{(x_i,y_i)\}_{i=1}^n$ and $\overline{p}(x,y)$ are equivalent up to a multiplicative factor.

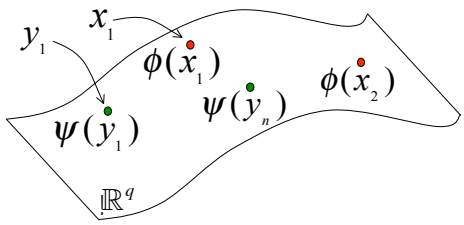


Figure 1: Embedding of X and Y into the same q-dimensional space. The embeddings functions $\phi : X \to \mathbb{R}^q$ and $\psi : Y \to \mathbb{R}^q$ determine the position of each instance in the low-dimensional space.

$\phi(x)$ to be far away from all $\psi(y)$. Such an embedding would reflect the marginal of x rather than its statistical relationship with all the other y values.

In what follows, we describe several methods to address this issue. Section 2.1 discusses symmetric models, and Section 2.2 conditional ones.

2.1 Symmetric Interaction Models

The goal of joint embedding is to have the geometry in the embedded space reflect the statistical relationships between variables, rather than just their joint probability. Specifically, the location $\phi(x)$ should be insensitive to the marginal $p(x)$, which just reflects the chance of observing x rather than the statistical relationship between x and different y values. To achieve this, we start by considering the ratio

$$r_p(x,y) = \frac{p(x,y)}{p(x)p(y)} \ , \ p(x) = \sum_y p(x,y) \ , \ p(y) = \sum_x p(x,y)$$

between the joint probability of x and y and the probability of observing that pair if the occurrences of x and y were independent. This ratio is widely used in statistics and information theory, for instance in the mutual information (Cover and Thomas, 1991), which is the expected value of the log of this ratio: $I(X;Y) = \sum_{x,y} p(x,y) \log \frac{p(x,y)}{p(x)p(y)} = \sum_{x,y} p(x,y) \log r_p(x,y)$. When X and Y are statistically independent, we have $r_p(x,y) = 1$ for all (x,y), and for any marginal distributions $p(x)$ and $p(y)$. Otherwise, high (low) values of $r_p(x,y)$ imply that the probability of $p(x,y)$ is larger (smaller) than the probability assuming independent variables.

Since $r_p(x,y)$ models statistical dependency, it is a natural choice to construct a model where $r_p(x,y)$ is proportional to $e^{-d^2_{x,y}}$. A first attempt at such a model is

$$p(x,y) = \frac{1}{Z} p(x)p(y)e^{-d^2_{x,y}} \ \ , \ \ p(x) = \sum_y p(x,y) \ \ , \ \ p(y) = \sum_x p(x,y) \ , \tag{1}$$

where Z is a normalization term (partition function). The key difficulty with this model is that $p(x)$ and $p(y)$, which appear in the model, are dependent on $p(x,y)$. Hence, some choices of $d^2_{x,y}$ lead to invalid models. As a result, one has to choose $p(x)$, $p(y)$ and $d^2_{x,y}$ jointly such that the

$p(x,y)$ obtained is consistent with the given marginals $p(x)$ and $p(y)$. This significantly complicates parameter estimation in such models, and we do not pursue them further here.[2]

To avoid the above difficulty, we use instead the ratio to the empirical marginals $\overline{p}(x)$ and $\overline{p}(y)$

$$\overline{r}_p(x,y) = \frac{p(x,y)}{\overline{p}(x)\overline{p}(y)} \ , \ \overline{p}(x) = \sum_y \overline{p}(x,y) \ , \ \overline{p}(y) = \sum_x \overline{p}(x,y) \ .$$

This is a good approximation of r_p when $\overline{p}(x), \overline{p}(y)$ are close to $p(x), p(y)$, which is a reasonable assumption for the applications that we consider. Requiring $\overline{r}_p(x,y)$ to be proportional to $e^{-d^2_{x,y}}$, we obtain the following model

$$p_{MM}(x,y) \equiv \frac{1}{Z}\overline{p}(x)\overline{p}(y)e^{-d^2_{x,y}} \quad \forall x \in X, \forall y \in Y \ , \tag{2}$$

where $Z = \sum_{x,y} \overline{p}(x)\overline{p}(y)e^{-d^2_{x,y}}$ is a normalization term. The subscript MM reflects the fact that the model contains the marginal factors $\overline{p}(x)$ and $\overline{p}(y)$. We use different subscripts to distinguish between the models that we consider (see Section 2.3). The distribution $p_{MM}(x,y)$ satisfies $\overline{r}_{p_{MM}}(x,y) \propto e^{-d^2_{x,y}}$, providing a direct relation between statistical dependencies and embedding distances. Note that $p_{MM}(x,y)$ has zero probability for any x or y that are not in the support of $\overline{p}(x)$ or $\overline{p}(y)$ (that is, $\overline{p}(x) = 0$ or $\overline{p}(y) = 0$). This does not pose a problem because such values of X or Y will not be included in the model to begin with, since we essentially cannot learn anything about them when variables are purely categorical. When the X or Y objects have additional structure, it may be possible to infer embeddings of unobserved values. This is discussed further in Section 9.

The model $p_{MM}(x)$ is symmetric with respect to the variables X and Y. The next section describes a model that breaks this symmetry by conditioning on one of the variables.

2.2 Conditional Models

A standard approach to avoid modeling marginal distributions is to use conditional distributions instead of the joint distribution. In some cases, conditional models are a more plausible generating mechanism for the data. For instance, a distribution of authors and words in their works is more naturally modeled as first choosing an author according to some prior and then generating words according to the author's vocabulary preferences. In this case, we can use the embedding distances $d^2_{x,y}$ to model the conditional word generation process rather than the joint distribution of authors and words.

The following equation defines a distance-based model for conditional co-occurrence probabilities:

$$p_{CM}(y|x) \equiv \frac{1}{Z(x)}\overline{p}(y)e^{-d^2_{x,y}} \quad \forall x \in X, \forall y \in Y \ . \tag{3}$$

$Z(x) = \sum_y \overline{p}(y)e^{-d^2_{x,y}}$ is a partition function for the given value x, and the subscript CM reflects the fact that we are conditioning on X and multiplying by the marginal of Y. We can use $p_{CM}(y|x)$ and the empirical marginal $\overline{p}(x)$ to define a joint model $p_{CM}(x,y) \equiv p_{CM}(y|x)\overline{p}(x)$ so that

$$p_{CM}(x,y) = \frac{1}{Z(x)}\overline{p}(x)\overline{p}(y)e^{-d^2_{x,y}} \ . \tag{4}$$

2. Equation 1 bears some resemblance to copula based models (Nelsen, 1999) where joint distributions are modeled as a product of marginals and interaction terms. However, copula models are typically based on continuous variables with specific interaction terms, and thus do not resolve the difficulty mentioned above.

This model satisfies the relation $\overline{r}_{p_{CM}}(x,y) \propto \frac{1}{Z(x)} e^{-d^2_{x,y}}$ between statistical dependency and distance. This implies that for a given x, the nearest neighbor $\psi(y)$ of $\phi(x)$ corresponds to the y with the largest dependency ratio $\overline{r}_{p_{CM}}(x,y)$.

A GENERATIVE PROCESS FOR CONDITIONAL MODELS

One advantage of using a probabilistic model to describe complex data is that the model may reflect a mechanism for generating the data. To study such a mechanism here, we consider a simplified conditional model

$$p_{CU}(y|x) \equiv \frac{1}{Z(x)} e^{-d^2_{x,y}} = \frac{1}{Z(x)} e^{-\|\phi(x)-\psi(y)\|^2} . \tag{5}$$

We also define the corresponding joint model $p_{CU}(x,y) = p_{CU}(y|x)\overline{p}(x)$, as in Equation 4.

The model in Equation 5 states that for a given x, the probability of generating a given y is proportional to $e^{-d^2_{x,y}}$. To obtain a generative interpretation of this model, consider the case where every point in the space $\mathbb{R}^q$ corresponds to $\psi(y)$ for some y (that is, ψ is a surjective map). This will only be possible if there is a one to one mapping between the variable Y and $\mathbb{R}^q$, so for the purpose of this section we assume that Y is not discrete. Sampling a pair (x,y) from $p_{CU}(x,y)$ then corresponds to the following generative procedure:

- Sample a value of x from $\overline{p}(x)$.
- For this x, perform a random walk in the space $\mathbb{R}^q$ starting at the point $\phi(x)$ and terminating after a fixed time T.[3]
- Denote the termination point of the random walk by $\mathbf{z} \in \mathbb{R}^q$.
- Return the value of y for which $\mathbf{z} = \psi(y)$.

The termination point of a random walk has a Gaussian distribution, with a mean given by the starting point $\phi(x)$. The conditional distribution $p_{CU}(y|x)$ has exactly this Gaussian form, and therefore the above process generates pairs according to the distribution $p_{CU}(x,y)$. This process only describes the generation of a single pair (x_i,y_i), and distinct pairs are assumed to be generated IID. It will be interesting to consider models for generating sequences of pairs via one random walk.

A generative process for the model p_{CM} in Equation 3 is less straightforward to obtain. Intuitively, it should correspond to a random walk that is weighted by some prior over Y. Thus, the random walk should be less likely to terminate at points $\psi(y)$ that correspond to low $\overline{p}(y)$. The multiplicative interaction between the exponentiated distance and the prior makes it harder to define a generative process in this case.

2.3 Alternative Models

The previous sections considered several models relating distributions to embeddings. The notation we used for naming the models above is of the form p_{AB} where A and B specify the treatment of the X and Y marginals, respectively. The following values are possible for A and B:

- C : The variable is conditioned on.

3. Different choices of T will correspond to different constants multiplying $d^2_{x,y}$ in Equation 5. We assume here that T is chosen such that this constant is one.

- M : The variable is not conditioned on, and its observed marginal appears in the distribution.
- U : The variable is not conditioned on, and its observed marginal does not appear in the distribution.

This notation can be used to define models not considered above. Some examples, which we also evaluate empirically in Section 8.5 are

$$\begin{aligned} p_{UU}(x,y) &\equiv \frac{1}{Z}e^{-d^2_{x,y}}, \\ p_{MC}(x|y) &\equiv \frac{1}{Z(y)}\overline{p}(x)e^{-d^2_{x,y}}, \\ p_{UC}(x|y) &\equiv \frac{1}{Z(y)}e^{-d^2_{x,y}}. \end{aligned} \tag{6}$$

2.4 Choosing the "Right" Model

The models discussed in Sections 2.1-2.3 present different approaches to relating probabilities to distances. They differ in their treatment of marginals, and in using distances to model either joint or conditional distributions. They thus correspond to different assumptions about the data. For example, conditional models assume an asymmetric generative model, where distances are related only to conditional distributions. Symmetric models may be more appropriate when no such conditional assumption is valid. We performed a quantitative comparison of all the above models on a task of word-document embedding, as described in Section 8.5. Our results indicate that, as expected, models that address both marginals (such as p_{CM} or p_{MM}), and that are therefore directly related to the ratio $\overline{r}_p(x,y)$, outperform models which do not address marginals. Although there are two possible conditional models, conditioning on X or on Y, for the specific task studied in Section 8.5 one of the conditional models is more sensible as a generating mechanism, and indeed yielded better results.

3. Learning the Model Parameters

We now turn to the task of learning the model parameters $\{\phi(x),\psi(y)\}$ from empirical data. In what follows, we focus on the model $p_{CM}(x,y)$ in Equation 4. However, all our derivations are easily applied to the other models in Section 2. Since we have a parametric model of a distribution, it is natural to look for the parameters that maximize the log-likelihood of the observed pairs $\{(x_i,y_i)\}_{i=1}^n$. For a given set of observed pairs, the average log-likelihood is[4]

$$\ell(\phi,\psi) = \frac{1}{n}\sum_{i=1}^{n}\log p_{CM}(x_i,y_i)\ .$$

The log-likelihood may equivalently be expressed in terms of the distribution $\overline{p}(x,y)$ since

$$\ell(\phi,\psi) = \sum_{x,y}\overline{p}(x,y)\log p_{CM}(x,y)\ .$$

4. For conditional models we can consider maximizing only the conditional log-likelihood $\frac{1}{n}\sum_{i=1}^n \log p(y_i|x_i)$. This is equivalent to maximizing the joint log-likelihood for the model $p(y|x)\overline{p}(x)$, and we prefer to focus on joint likelihood maximization so that a unified formulation is used for both joint and conditional models.

As in other cases, maximizing the log-likelihood is also equivalent to minimizing the KL divergence D_{KL} between the empirical and the model distributions, since $\ell(\phi,\psi)$ equals $D_{\mathrm{KL}}\left[\overline{p}(x,y)|p_{CM}(x,y)\right]$ up to an additive constant.

The log-likelihood in our case is given by

$$\begin{aligned}\ell(\phi,\psi) &= \sum_{x,y}\overline{p}(x,y)\log p_{CM}(x,y)\\ &= \sum_{x,y}\overline{p}(x,y)\Big(-d^2_{x,y}-\log Z(x)+\log\overline{p}(x)+\log\overline{p}(y))\Big)\\ &= -\sum_{x,y}\overline{p}(x,y)d^2_{x,y}-\sum_{x}\overline{p}(x)\log Z(x)+const\ ,\end{aligned} \tag{7}$$

where $const=\sum_y\overline{p}(y)\log\overline{p}(y)+\sum_x\overline{p}(x)\log\overline{p}(x)$ is a constant term that does not depend on the parameters $\phi(x)$ and $\psi(y)$.

Finding the optimal parameters now corresponds to solving the following optimization problem

$$(\phi^*,\psi^*)=\arg\max_{\phi,\psi}\ell(\phi,\psi)\ . \tag{8}$$

The log-likelihood is composed of two terms. The first is (minus) the mean distance between x and y. This will be maximized when all distances are zero. This trivial solution is avoided because of the *regularization* term $\sum_x\overline{p}(x)\log Z(x)$, which acts to increase distances between x and y points.

To characterize the maxima of the log-likelihood we differentiate it with respect to the embeddings of individual objects $(\phi(x),\psi(y))$, and obtain the following gradients

$$\begin{aligned}\frac{\partial\ell(\phi,\psi)}{\partial\phi(x)} &= 2\overline{p}(x)\Big(\langle\psi(y)\rangle_{\overline{p}(y|x)}-\langle\psi(y)\rangle_{p_{CM}(y|x)}\Big)\ ,\ ,\\ \frac{\partial\ell(\phi,\psi)}{\partial\psi(y)} &= 2p_{CM}(y)\Big(\psi(y)-\langle\phi(x)\rangle_{p_{CM}(x|y)}\Big)-2\overline{p}(y)\Big(\psi(y)-\langle\phi(x)\rangle_{\overline{p}(x|y)}\Big)\ ,\end{aligned}$$

where $p_{CM}(y)=\sum_x p_{CM}(y|x)\overline{p}(x)$ and $p_{CM}(x|y)=\frac{p_{CM}(x,y)}{p_{CM}(y)}$.

Equating the $\phi(x)$ gradient to zero yields:

$$\langle\psi(y)\rangle_{p_{CM}(y|x)}=\langle\psi(y)\rangle_{\overline{p}(y|x)}\ . \tag{9}$$

If we fix ψ, this equation is formally similar to the one that arises in the solution of conditional maximum entropy models (Berger et al., 1996). However, there is a crucial difference in that the exponent of $p_{CM}(y|x)$ in conditional maximum entropy is linear in the parameters (ϕ in our notation), while in our model it also includes quadratic (norm) terms in the parameters. The effect of Equation 9 can then be described informally as that of choosing $\phi(x)$ so that the expected value of ψ under $p_{CM}(y|x)$ is the same as its empirical average, that is, placing the embedding of x closer to the embeddings of those y values that have stronger statistical dependence with x.

The maximization problem of Equation 8 is not jointly convex in $\phi(x)$ and $\psi(y)$ due to the quadratic terms in d^2_{xy}.[5] To find the local maximum of the log-likelihood with respect to both $\phi(x)$ and $\psi(y)$ for a given embedding dimension q, we use a conjugate gradient ascent algorithm with random restarts.[6] In Section 5 we describe a different approach to this optimization problem.

5. The log-likelihood *is* a convex function of $\phi(x)$ for a constant $\psi(y)$, as noted in Iwata et al. (2005), but is not convex in $\psi(y)$ for a constant $\phi(x)$.

6. The code is provided online at `http://ai.stanford.edu/~gal/`.

4. Relation to Other Methods

In this section we discuss other methods for representing co-occurrence data via low dimensional vectors, and study the relation between these methods and the CODE models.

4.1 Maximizing Correlations and Related Methods

Embedding the rows and columns of a contingency table into a low dimensional Euclidean space was previously studied in the statistics literature. Fisher (1940) described a method for mapping X and Y into scalars $\phi(x)$ and $\psi(y)$ such that the correlation coefficient between $\phi(x)$ and $\psi(y)$ is maximized. The method of Correspondence Analysis (CA) generalizes Fisher's method to non-scalar mappings. More details about CA are given in Appendix A. Similar ideas have been applied to more than two variables in the Gifi system (Michailidis and de Leeuw, 1998). All these methods can be shown to be equivalent to the more widely known *canonical correlation analysis* (CCA) procedure (Hotelling, 1935). In CCA one is given two continuous multivariate random variables X and Y, and aims to find two sets of vectors, one for X and the other for Y, such that the correlations between the projections of the variables onto these vectors are maximized. The optimal projections for X and Y can be found by solving an eigenvalue problem. It can be shown (Hill, 1974) that if one represents X and Y via indicator vectors, the CCA of these vectors (when replicated according to their empirical frequencies) results in Fisher's mapping and CA.

The objective of these correlation based methods is to maximize the correlation coefficient between the embeddings of X and Y. We now discuss their relation to our distance-based method. First, the correlation coefficient is invariant under affine transformations and we can thus focus on centered solutions with a unity covariance matrix solutions: $\langle \phi(X) \rangle = 0, \langle \psi(Y) \rangle = 0$ and $\mathrm{Cov}(\phi(X)) = \mathrm{Cov}(\psi(Y)) = I$. In this case, the correlation coefficient is given by the following expression (we focus on $q = 1$ for simplicity)

$$\rho(\phi(x), \psi(y)) = \sum_{x,y} \overline{p}(x,y)\phi(x)\psi(y) = -\frac{1}{2}\sum_{x,y} \overline{p}(x,y) d^2_{x,y} + 1 \ .$$

Maximizing the correlation is therefore equivalent to minimizing the mean distance across all pairs. This clarifies the relation between CCA and our method: Both methods aim to minimize the average distance between X and Y embeddings. However, CCA forces embeddings to be centered and scaled, whereas our method introduces a global regularization term related to the partition function.

A kernel variant of CCA has been described in Lai and Fyfe (2000) and Bach and Jordan (2002), where the input vectors X and Y are first mapped to a high dimensional space, where linear projection is carried out. This idea could possibly be used to obtain a kernel version of correspondence analysis, although we are not aware of existing work in that direction.

Recently, Zhong et al. (2004) presented a co-embedding approach for detecting unusual activity in video sequences. Their method also minimizes an averaged distance measure, but normalizes it by the variance of the embedding to avoid trivial solutions.

4.2 Distance-Based Embeddings

Multidimensional scaling (MDS) is a well-known geometric embedding method (Cox and Cox, 1984), whose standard version applies to same-type objects with predefined distances. MDS embedding of heterogeneous entities was studied in the context of modeling ranking data (Cox and

Cox, 1984, Section 7.3). These models, however, focus on specific properties of ordinal data and therefore result in optimization principles and algorithms different from our probabilistic interpretation.

Relating Euclidean structure to probability distributions was previously discussed by Hinton and Roweis (2003). They assume that distances between points in some X space are given, and the exponent of these distances induces a distribution $p(x=i|x=j)$ which is proportional to the exponent of the distance between $\phi(i)$ and $\phi(j)$. This distribution is then approximated via an exponent of distances in a low dimensional space. Our approach differs from theirs in that we treat the joint embedding of two different spaces. Therefore, we do not assume a metric structure between X and Y, but instead use co-occurrence data to learn such a structure. The two approaches become similar when $X=Y$ and the empirical data *exactly* obeys an exponential law as in Equation 3.

Iwata et al. (2005) recently introduced the Parametric Embedding (PE) method for visualizing the output of supervised classifiers. They use the model of Equation 3 where Y is taken to be the class label, and X is the input features. Their embedding thus illustrates which X values are close to which classes, and how the different classes are inter-related. The approach presented here can be viewed as a generalization of their approach to the unsupervised case, where X and Y are arbitrary objects.

An interesting extension of locally linear embedding (Roweis and Saul, 2000) to heterogeneous embedding was presented by Ham et al. (2003). Their method essentially forces the outputs of two locally linear embeddings to be aligned such that corresponding pairs of objects are mapped to similar points.

A Bayesian network approach to joint embedding was recently studied in Mei and Shelton (2006) in the context of collaborative filtering.

4.3 Matrix Factorization Methods

The empirical joint distribution $\overline{p}(x,y)$ can be viewed as a matrix $\overline{P}$ of size $|X|\times|Y|$. There is much literature on finding low rank approximations of matrices, and specifically matrices that represent distributions (Hofmann, 2001; Lee and Seung, 1999). Low rank approximations are often expressed as a product UV^T where U and V are two matrices of size $|X|\times q$ and $|Y|\times q$ respectively.

In this context CODE can be viewed as a special type of low rank approximation of the matrix $\overline{P}$. Consider the symmetric model p_{UU} in Equation 6, and the following matrix and vector definitions:[7]

- Let Φ be a matrix of size $|X|\times q$ where the i^{th} row is $\phi(i)$. Let Ψ be a matrix of size $|Y|\times q$ where the i^{th} row is $\psi(i)$.
- Define the column vector $\mathbf{u}(\Phi)\in\mathbb{R}^{|X|}$ as the set of squared Euclidean norms of $\phi(i)$, so that $u_i(\phi)=\|\phi(i)\|^2$. Similarly define $\mathbf{v}(\psi)\in\mathbb{R}^{|Y|}$ as $v_i(\psi)=\|\psi(i)\|^2$.
- Denote the k-dimensional column vector of all ones by $\mathbf{1}_k$.

Using these definitions, the model p_{UU} can then be written in matrix form as

$$\log P_{UU} = -\log Z + 2\Phi\Psi^T - \mathbf{u}(\Phi)\mathbf{1}_{|Y|}^T - \mathbf{1}_{|X|}\mathbf{v}(\Psi)^T$$

where the optimal Φ and Ψ are found by minimizing the KL divergence between $\overline{P}$ and P_{UU}.

7. We consider p_{UU} for simplicity. Other models, such as p_{MM}, have similar interpretations.

The model for $\log P_{UU}$ is in fact low-rank, since the rank of $\log P_{UU}$ is at most $q+2$. However, note that P_{UU} itself will not necessarily have a low rank. Thus, CODE can be viewed as a low-rank matrix factorization method, where the structure of the factorization is motivated by distances between rows of Φ and Ψ, and the quality of the approximation is measured via the KL divergence.

Many matrix factorization algorithms (such as Lee and Seung, 1999) use the term $\Phi\Psi^T$ above, but not the terms $\mathbf{u}(\Phi)$ and $\mathbf{v}(\Psi)$. Another algorithm that uses only the $\Phi\Psi^T$ term, but is more closely related to CODE is the sufficient dimensionality reduction (SDR) method of Globerson and Tishby (2003). SDR seeks a model

$$\log P_{SDR} = -\log Z + \Phi\Psi^T + \mathbf{a}\mathbf{1}_{|Y|}^T + \mathbf{1}_{|X|}\mathbf{b}^T$$

where $\mathbf{a}, \mathbf{b}$ are vectors of dimension $|X|, |Y|$ respectively. As in CODE, the parameters $\Phi, \Psi, \mathbf{a}$ and $\mathbf{b}$ are chosen to maximize the likelihood of the observed data.

The key difference between CODE and SDR lies in the terms $\mathbf{u}(\Phi)$ and $\mathbf{v}(\Psi)$ which are *non-linear* in Φ and Ψ. These arise from the geometric interpretation of CODE that relates distances between embeddings to probabilities. SDR does not have such an interpretation. In fact, the SDR model is invariant to the translation of either of the embedding maps (for instance, $\phi(x)$), while fixing the other map $\psi(y)$. Such a transformation would completely change the distances $d_{x,y}^2$ and is clearly not an invariant property in the CODE models.

5. Semidefinite Representation

The CODE learning problem in Equation 8 is not jointly convex in the parameters ϕ and ψ. In this section we present a convex relaxation of the learning problem. For a sufficiently high embedding dimension this approximation is in fact exact, as we show next. For simplicity, we focus on the p_{CM} model, although similar derivations may be applied to the other models.

5.1 The Full Rank Case

Locally optimal CODE embeddings $\phi(x)$ and $\psi(y)$ may be found using standard unconstrained optimization techniques. However, the Euclidean distances used in the embedding space also allow us to reformulate the problem as constrained convex optimization over the cone of positive semidefinite (PSD) matrices (Boyd and Vandenberghe, 2004).

We begin by showing that for embeddings with dimension $q = |X| + |Y|$, maximizing the CODE likelihood (see Equation 8) is equivalent to minimizing a certain convex non-linear function over PSD matrices. Consider the matrix A whose columns are all the embedded vectors $\phi(x)$ and $\psi(y)$

$$A \equiv [\phi(1), \ldots, \phi(|X|), \psi(1), \ldots, \psi(|Y|)] \ .$$

Define the Gram matrix G as

$$G \equiv A^T A \ .$$

G is a matrix of the dot products between the coordinate vectors of the embedding, and is therefore a symmetric PSD matrix of rank $\leq q$. Conversely, any PSD matrix of rank $\leq q$ can be factorized as $A^T A$, where A is some embedding matrix of dimension q. Thus we can replace optimization over matrices A with optimization over PSD matrices of rank $\leq q$. Note also that the distance between two columns in A is *linearly* related to the Gram matrix via $d_{xy}^2 = g_{xx} + g_{yy} - 2g_{xy}$, and thus the embedding distances are linear functions of the elements of G.

Since the log-likelihood function in Equation 7 depends only on the distances between points in X and in Y, we can write it as a function of G only.[8] In what follows, we focus on the negative log-likelihood $f(G) = -\ell(G)$

$$f(G) = \sum_{x,y} \overline{p}(x,y)(g_{xx} + g_{yy} - 2g_{xy}) + \sum_{x} \overline{p}(x) \log \sum_{y} \overline{p}(y) e^{-(g_{xx}+g_{yy}-2g_{xy})} .$$

The likelihood maximization problem can then be written in terms of constrained minimization over the set of rank q positive semidefinite matrices[9]

$$\begin{array}{ll} \min_G & f(G) \\ \text{s.t.} & G \succeq 0 \\ & \text{rank}(G) \leq q \; . \end{array} \tag{10}$$

Thus, the CODE log-likelihood maximization problem in Equation 8 is equivalent to minimizing a nonlinear objective over the set of PSD matrices of a constrained rank.

When the embedding dimension is $q = |X| + |Y|$ the rank constraint is always satisfied and the problem reduces to

$$\begin{array}{ll} \min_G & f(G) \\ \text{s.t.} & G \succeq 0 \; . \end{array} \tag{11}$$

The minimized function $f(G)$ consists of two convex terms: The first term is a linear function of G; the second term is a sum of $\log \sum \exp$ terms of an affine expression in G. The $\log \sum \exp$ function is convex (Boyd and Vandenberghe, 2004, Section 4.5), and therefore the function $f(G)$ is convex. Moreover, the set of constraints is also convex since the set of PSD matrices is a convex cone (Boyd and Vandenberghe, 2004). We conclude that when the embedding dimension is of size $q = |X| + |Y|$ the optimization problem of Equation 11 is convex, and thus has no local minima.

5.1.1 ALGORITHMS

The convex optimization problem in Equation 11 can be viewed as a PSD constrained geometric program.[10] This is not a semidefinite program (SDP, see Vandenberghe and Boyd, 1996), since the objective function in our case is non-linear and SDPs are defined as having both a linear objective and linear constraints. As a result we cannot use standard SDP tools in the optimization. It seems like such Geometric Program/PSD problems have not been dealt with in the optimization literature, and it will be interesting to develop specialized algorithms for these cases.

The optimization problem in Equation 11 can however be solved using any general purpose convex optimization method. Here we use the *projected gradient* algorithm (Bertsekas, 1976), a simple method for constrained convex minimization. The algorithm takes small steps in the direction of the negative objective gradient, followed by a Euclidean projection on the set of PSD matrices. This projection is calculated by eliminating the contribution of all eigenvectors with negative eigenvalues to the current matrix, similarly to the PSD projection algorithm of Xing et al. (2002). Pseudo-code for this procedure is given in Figure 2.

In terms of complexity, the most time consuming part of the algorithm is the eigenvector calculation which is $O((|X| + |Y|)^3)$ (Pan and Chen, 1999). This is reasonable when $|X| + |Y|$ is a few thousands but becomes infeasible for much larger values of $|X|$ and $|Y|$.

8. We ignore the constant additive terms.

9. The objective $f(G)$ is *minus* the log-likelihood, which is why minimization is used.

10. A geometric program is a convex optimization problem where the objective and the constraints are $\log \sum \exp$ functions of an affine function of the variables (Chiang, 2005).

Input: Empirical distribution $\overline{p}(x,y)$. A step size ε.

Output: PSD matrix of size $|X|+|Y|$ that solves the optimization problem in Equation 11.

Initialize: Set G_0 to the identity matrix of size $|X|+|Y|$.

Iterate:

- Set $\hat{G}_{t+1} = G_t - \varepsilon \bigtriangledown f(G_t)$.
- Calculate the eigen-decomposition of $\hat{G}_{t+1}$: $\hat{G}_{t+1} = \sum_k \lambda_k \mathbf{u}_k \mathbf{u}_k^T$.
- Set $G_{t+1} = \sum_k \max(\lambda_k, 0) \mathbf{u}_k \mathbf{u}_k^T$.

Figure 2: A projected gradient algorithm for solving the optimization problem in Equation 11. To speed up convergence we also use an Armijo rule (Bertsekas, 1976) to select the step size ε at every iteration.

5.2 The Low-Dimensional Case

Embedding into a low dimension requires constraining the rank, but this is difficult since the problem in Equation 10 is not convex in the general case. One approach to obtaining low rank solutions is to optimize over a full rank G and then project it into a lower dimension via spectral decomposition as in Weinberger and Saul (2006) or classical MDS. However, in the current problem, this was found to be ineffective.

A more effective approach in our case is to regularize the objective by adding a term $\lambda \mathrm{Tr}(G)$, for some constant $\lambda > 0$. This keeps the problem convex, since the trace is a linear function of G. Furthermore, since the eigenvalues of G are non-negative, this term corresponds to ℓ_1 regularization on the eigenvalues. Such regularization is likely to result in a sparse set of eigenvalues, and thus in a low dimensional solution, and is indeed a commonly used trick in obtaining such solutions (Fazel et al., 2001). This results in the following regularized problem

$$\begin{array}{ll} \min_G & f(G) + \lambda \mathrm{Tr}(G) \\ \text{s.t.} & G \succeq 0 \,. \end{array} \tag{12}$$

Since the problem is still convex, we can again use a projected gradient algorithm as in Figure 2 for the optimization. We only need to replace $\nabla f(G_t)$ with $\lambda I + \nabla f(G_t)$ where I is an identity matrix of the same size as G.

Now suppose we are seeking a q dimensional embedding, where $q < |X|+|Y|$. We would like to use λ to obtain low dimensional solutions, but to choose the q dimensional solution with maximum log-likelihood. This results in the PSD-CODE procedure described in Figure 3. This approach is illustrated in Figure 4 for $q = 2$. The figure shows log-likelihood values of regularized PSD solutions projected to two dimensions. The values of λ which achieve the optimal likelihood also result in only two significant eigenvalues, showing that the regularization and projection procedure indeed produces low dimensional solutions.

PSD-CODE

Input: A set of regularization parameters $\{\lambda_i\}_{i=1}^n$, an embedding dimension q, and empirical distribution $\overline{p}(x,y)$.

Output: A q dimensional embedding of X and Y

Algorithm

- For each value of λ_i:
  - Use the projected gradient algorithm to solve the optimization problem in Equation 12 with regularization parameter λ_i. Denote the solution by G.
  - Transform G into a rank q matrix G_q by keeping only the q eigenvectors with the largest eigenvalues.
  - Calculate the likelihood of the data under the model given by the matrix G_q. Denote this likelihood by ℓ_i.
- Find the λ_i which maximizes ℓ_i, and return its corresponding embedding.

Figure 3: The PSD-CODE algorithm for finding a low dimensional embedding using PSD optimization.

The PSD-CODE algorithm was applied to subsets of the databases described in Section 7 and yielded similar results to those of the conjugate-gradient based algorithm. We believe that PSD algorithms may turn out to be more efficient in cases where relatively high dimensional embeddings are sought. Furthermore, with the PSD formulation it is easy to introduce additional constraints, for example on distances between subsets of points (Weinberger and Saul, 2006). Section 6.1 considers a model extension that could benefit from such a formulation.

6. Using Additional Co-occurrence Data

The methods described so far use a single co-occurrence table of two objects. However, in some cases we may have access to additional information about (X,Y) and possibly other variables. Below we describe extensions of CODE to these settings.

6.1 Within-Variable Similarity Measures

The CODE models in Section 2 rely only on the co-occurrence of X and Y but assume nothing about similarity between two objects of the same type. Such a similarity measure may often be available and could take several forms. One is a distance measure between objects in X. For example, if $\mathbf{x} \in \mathbb{R}^p$ we may take the Euclidean distance $\|\mathbf{x}_i - \mathbf{x}_j\|_2$ between two vectors $\mathbf{x}_i, \mathbf{x}_j \in \mathbb{R}^p$ as a measure of similarity. This information may be combined with co-occurrence data either by requiring the

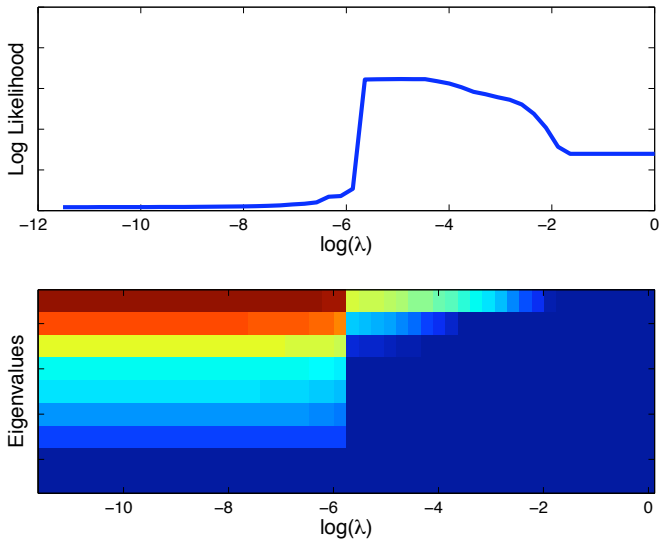


Figure 4: Results for the PSD-CODE algorithm. Data is the 5×4 contingency table in Greenacre (1984) page 55. **Top**: The log-likelihood of the solution projected to two dimensions, as a function of the regularization parameter λ. **Bottom**: The eigenvalues of the Gram matrix obtained using the PSD algorithm for the corresponding λ values. It can be seen that solutions with two dominant eigenvalues have higher likelihoods.

CODE map to agree with the given distances, or by adding a term which penalizes deviations from them.

Similarities between two objects in X may also be given in the form of co-occurrence data. For example, if X corresponds to words and Y corresponds to authors (see Section 7.1), we may have access to joint statistics of words, such as bigram statistics, which give additional information about which words should be mapped together. Alternatively, we may have access to data about collaboration between authors, for example, what is the probability of two authors writing a paper together. This in turn should affect the mapping of authors.

The above example can be formalized by considering two distributions $\overline{p}(x^{(1)}, x^{(2)})$ and $\overline{p}(y^{(1)}, y^{(2)})$ which describe the *within-type* object co-occurrence rates. One can then construct a CODE model as in Equation 3 for $p(x^{(1)}|x^{(2)})$

$$p(x^{(1)}|x^{(2)}) = \frac{\overline{p}(x^{(1)})}{Z(x^{(1)})} e^{-\|\phi(x^{(1)}) - \phi(x^{(2)})\|^2} .$$

Denote the log-likelihood for the above model by $l_x(\phi)$, and the corresponding log-likelihood for $p(y^{(1)}|y^{(2)})$ by $\ell_y(\psi)$. Then we can combine several likelihood terms by maximizing some weighted combination $\ell(\phi, \psi) + \lambda_x \ell_x(\phi) + \lambda_y \ell_y(\psi)$, where $\lambda_x, \lambda_y \geq 0$ reflect the relative weight of each information source.

6.2 Embedding More than Two Variables

The notion of a common underlying semantic space is clearly not limited to two objects. For example, texts, images and audio files may all have a similar meaning and we may therefore wish to embed all three in a single space. One approach in this case could be to use joint co-occurrence statistics $\overline{p}(x,y,z)$ for all three object types, and construct a geometric-probabilistic model for the distribution $p(x,y,z)$ using three embeddings $\phi(x), \psi(y)$ and $\xi(z)$ (see Section 9 for further discussion of this approach). However, in some cases obtaining joint counts over multiple objects may not be easy. Here we describe a simple extension of CODE to the case where more than two variables are considered, but empirical distributions are available only for *pairs* of variables.

To illustrate the approach, consider a case with k different variables $X^{(1)}, \ldots, X^{(k)}$ and an additional variable Y. Assume that we are given empirical joint distributions of Y with each of the X variables $\overline{p}(x^{(1)},y), \ldots, \overline{p}(x^{(k)},y)$. It is now possible to consider a set of k CODE models $p(x^{(i)},y)$ for $i = 1,\ldots,k$,[11] where each $X^{(i)}$ will have an embedding $\phi^{(i)}(x^{(i)})$ but all models will share the same $\psi(y)$ embedding. Given k non-negative weights $w_1, \ldots, w_k$ that reflect the "relative importance" of each $X^{(i)}$ we can consider the total weighted log-likelihood of the k models given by

$$\ell(\phi^{(1)}, \ldots, \phi^{(k)}, \psi) = \sum_i w_i \sum_{x^{(i)},y} \overline{p}(x^{(i)},y) \log p(x^{(i)},y) \ .$$

Maximizing the above log-likelihood will effectively combine structures in all the input distributions $\overline{p}(x^{(i)},y)$. For example if $Y = y$ often co-occurs with $X^{(1)} = x^{(1)}$ *and* $X^{(2)} = x^{(2)}$, likelihood will be increased by setting $\psi(y)$ to be close to both $\phi^{(1)}(x^{(1)})$ and $\phi^{(2)}(x^{(2)})$.

In the example above, it was assumed that only a single variable, Y, was shared between different pairwise distributions. It is straightforward to apply the same approach when more variables are shared: simply construct CODE models for all available pairwise distributions, and maximize their weighted log-likelihood.

Section 7.2 shows how this approach is used to successfully embed three different objects, namely authors, words, and documents in a database of scientific papers.

7. Applications

We demonstrate the performance of co-occurrence embedding on two real-world types of data. First, we use documents from NIPS conferences to obtain documents-word and author-word embeddings. These embeddings are used to visualize various structures in this complex corpus. We also use the multiple co-occurrence approach in Section 6.2 to embed authors, words, and documents into a single map. To provide quantitative assessment of the performance of our method, we apply it to embed the document-word 20 Usenet newsgroups data set, and we use the embedding to predict the class (newsgroup) for each document, which was not available when creating the embedding. Our method consistently outperforms previous unsupervised methods evaluated on this task.

In most of the experiments we use the conditional based model of Equation 4, except in Section 8.5 where the different models of Section 2 are compared.

11. This approach applies to all CODE models, such as p_{MM} or p_{CM}.

7.1 Visualizing a Document Database: The NIPS Database

Embedding algorithms are often used to visualize structures in document databases (Hinton and Roweis, 2003; Lin, 1997; Chalmers and Chitson, 1992). A common approach in these applications is to obtain some measure of similarity between objects of the same type such as words, and approximate it with distances in the embedding space.

Here we used the database of all papers from the NIPS conference until 2003. The database was based on an earlier database created by Roweis (2000), that included volumes 0-12 (until 1999).[12] The most recent three volumes also contain an indicator of the document's topic, for instance, AA for *Algorithms and Architectures*, LT for *Learning Theory*, and NS for *Neuroscience*, as shown in Figure 5.

We first used CODE to embed documents and words into $\mathbb{R}^2$. The results are shown in Figures 5 and 6. The empirical joint distribution was created as follows: for each document, the empirical distribution $\overline{p}(word|doc)$ was the number of times a word appeared in the document, normalized to one; this was then multiplied by a uniform prior $\overline{p}(doc)$ to obtain $\overline{p}(doc, word)$. The CODE model we used was the conditional word-given-document model $p_{CM}(doc, word)$. As Figure 5 illustrates, documents with similar topics tend to be mapped next to each other (for instance, AA near LT and NS near VB), even though the topic labels were not available to the algorithm when learning the embeddings. This shows that words in documents are good indicators of the topics, and that CODE reveals these relations. Figure 6 shows the joint embedding of documents and words. It can be seen that words indeed characterize the topics of their neighboring documents, so that the joint embedding reflects the underlying structure of the data.

Next, we used the data to generate an authors-words matrix $\overline{p}(author, word)$ obtained from counting the frequency with which a given author uses a given word. We could now embed authors and words into $\mathbb{R}^2$, by using CODE to model words given authors $p_{CM}(author, word)$. Figure 7 demonstrates that authors are indeed mapped next to terms relevant to their work, and that authors working on similar topics are mapped to nearby points. This illustrates how co-occurrence of words and authors can be used to induce a metric on authors alone.

These examples show how CODE can be used to visualize the complex relations between documents, their authors, topics and keywords.

7.2 Embedding Multiple Objects: Words, Authors and Documents

Section 6.2 presented an extension of CODE to multiple variables. Here we demonstrate that extension in embedding three object types from the NIPS database: words, authors, and documents. Section 7.1 showed embeddings of $(author, word)$ and $(doc, word)$. However, we may also consider a joint embedding for the objects $(author, word, doc)$, since there is a common semantic space underlying all three. To generate such an embedding, we apply the scheme of Section 6.2 with $Y \equiv word, X^{(1)} \equiv doc$ and $X^{(2)} \equiv author$. We use the two models $p_{CM}(author, word)$ and $p_{CM}(doc, word)$, that is, two conditional models where the *word* variable is conditioned on the *doc* or on the *author* variables. Recall that the embedding of the words is assumed to be the same in both models. We seek an embedding of all three objects that maximizes the weighted sum of the log-likelihood of these two models.

Different strategies may be used to weight the two log-likelihoods. One approach is to assign them equal weight by normalizing each by the total number of joint assignments. This corresponds

12. The data is available online at `http://ai.stanford.edu/~gal/`.

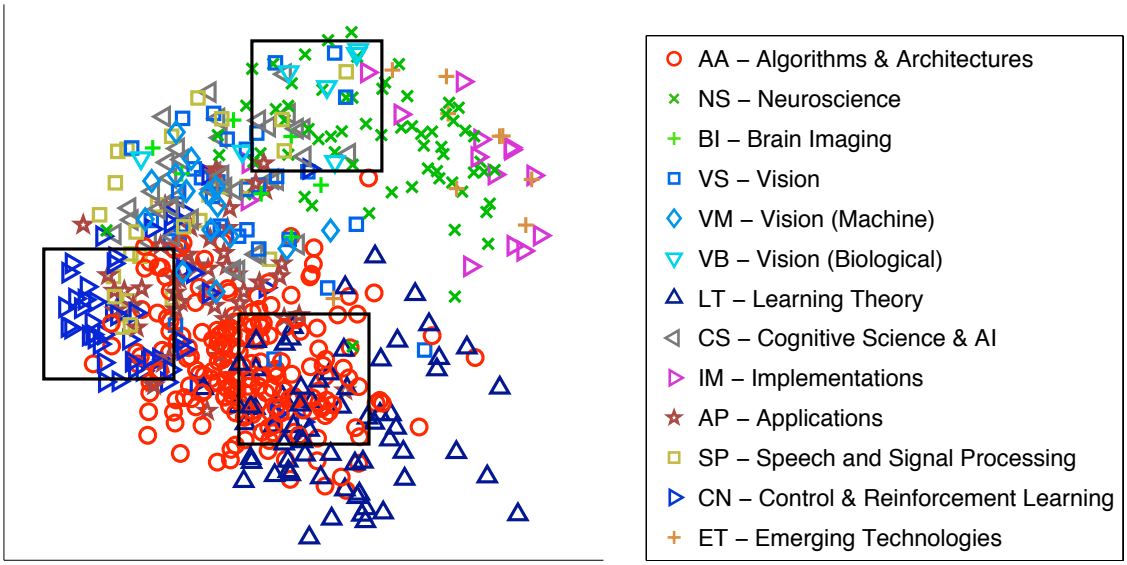


Figure 5: CODE embedding of 2483 documents and 2000 words from the NIPS database (the 2000 most frequent words, excluding the first 100, were used). Embedded documents from NIPS 15-17 are shown, with colors indicating the topic of each document. The word embeddings are not shown.

to choosing $w_i = \frac{1}{|X||Y^{(i)}|}$. For example, in this case the log-likelihood of $p_{CM}(author, word)$ will be weighted by $\frac{1}{|word||author|}$.

Figure 8 shows three insets of an embedding that uses the above weighting scheme.[13] The insets roughly correspond to those in Figure 6. However, here we have all three objects shown on the same map. It can be seen that both authors and words that correspond to a given topic are mapped together with documents about this topic.

It is interesting to study the sensitivity of the result to the choice of weights w_i. To evaluate this sensitivity, we introduce a quantitative measure of embedding quality: the *authorship* measure. The database we generated also includes the Boolean variable $isauthor(doc, author)$ that encodes whether a given author wrote a given document. This information is not available to the CODE algorithm and can be used to evaluate the documents-authors part of the authors-words-documents embedding. Given an embedding, we find the k nearest authors to a given document and calculate what fraction of the document's authors is in this set. We then average this across all k and all documents. Thus, for a document with three authors, this measure will be one if the three nearest authors to the document are its actual authors.

We evaluate the above authorship measure for different values of w_i to study the sensitivity of the embedding quality to changing the weights. Figure 9 shows that for a very large range of w_i values the measure is roughly constant, and it degrades quickly only when close to zero weight is

13. The overall document embedding was similar to Figure 5 and is not shown here.

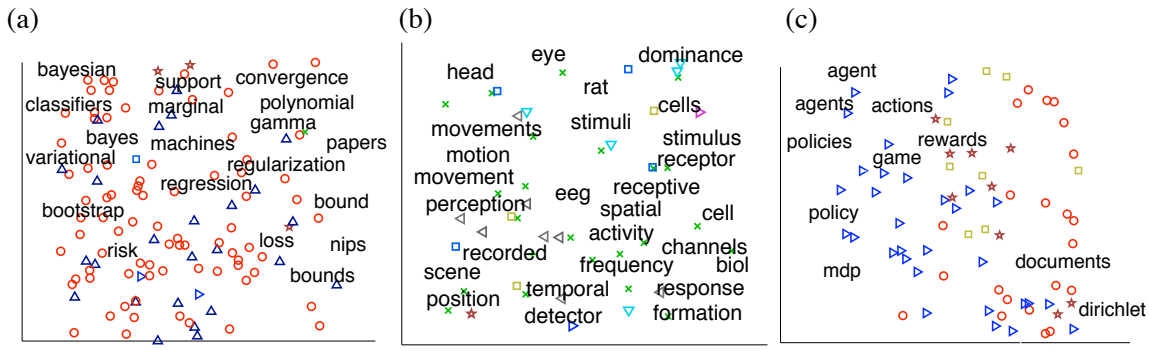


Figure 6: Each panel shows in detail one of the rectangles in Figure 5, and includes both the embedded documents and embedded words. **(a)** The border region between Algorithms and Architectures (AA) and Learning Theory (LT), corresponding to the bottom rectangle in Figure 5. **(b)** The border region between Neuroscience NS and Biological Vision (VB), corresponding to the upper rectangle in Figure 5. **(c)** Control and Reinforcement Learning (CN) region (left rectangle in Figure 5).

assigned to either of the two models. The stability with respect to w_i was also verified visually; embeddings were qualitatively similar for a wide range of weight values.

8. Quantitative Evaluation: The 20 Newsgroups Database

To obtain a quantitative evaluation of the effectiveness of our method, we apply it to a well controlled information retrieval task. The task contains known classes which are not used during learning, but are later used to evaluate the quality of the embedding.

8.1 The Data

We applied CODE to the widely studied 20 newsgroups corpus, consisting of 20 classes of 1000 documents each.[14] This corpus was further pre-processed as described by Chechik and Tishby (2003).[15] We first removed the 100 most frequent words, and then selected the next k most frequent words for different values of k (see below). The data was summarized as a count matrix $n(doc, word)$, which gives the count of each word in a document. To obtain an equal weight for all documents, we normalized the sum of each row in $n(doc, word)$ to one, and multiplied by $\frac{1}{|doc|}$. The resulting matrix is a joint distribution over the document and word variables, and is denoted by $\overline{p}(doc, word)$.

8.2 Methods Compared

Several methods were compared with respect to both homogeneous and heterogeneous embeddings of words and documents.

14. Available from `http://kdd.ics.uci.edu`.
15. Data set available from `http://ai.stanford.edu/~gal/`.

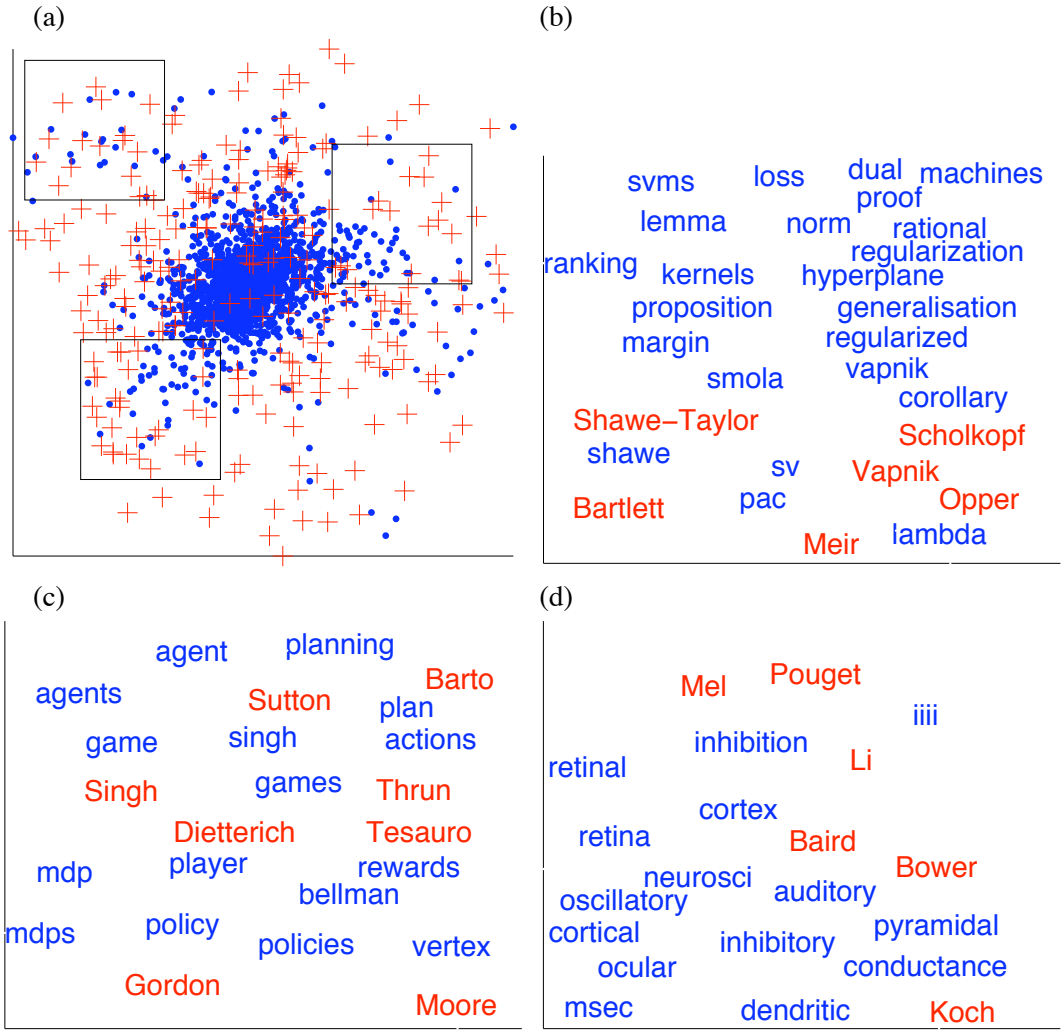


Figure 7: CODE embedding of 2000 words and 250 authors from the NIPS database (the 250 authors with highest word counts were chosen; words were selected as in Figure 5). The top left panel shows embeddings for authors (red crosses) and words (blue dots). Other panels show embedded authors (only first 100 shown) and words for the areas specified by rectangles (words in blue font, authors in red). They can be seen to correspond to learning theory (b), control and reinforcement learning (c) and neuroscience (d).

- **Co-Occurrence Data Embedding (CODE)**. Modeled the distribution of words and documents using the conditional word-given-document model $p_{CM}(doc, word)$ of Equation 4. Models other than p_{CM} are compared in Section 8.5.

- **Correspondence Analysis (CA)**. Applied the CA method to the matrix $\overline{p}(doc, word)$. Appendix A gives a brief review of CA.

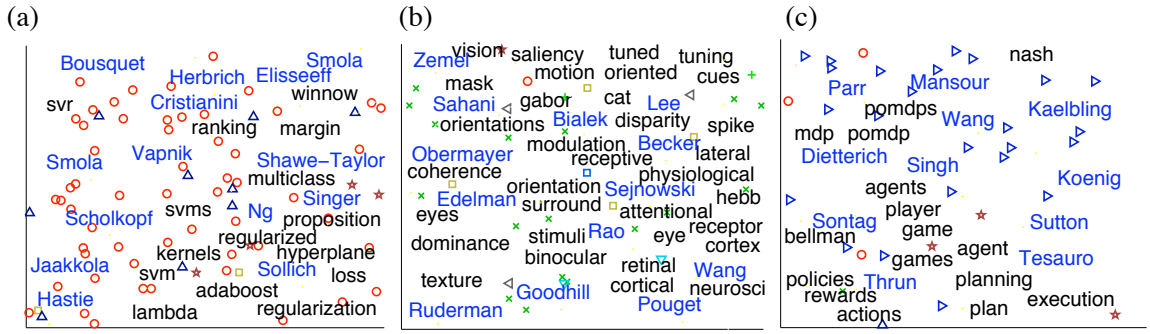


Figure 8: Embeddings of authors, words, and documents as described in Section 7.2. Words are shown in black and authors in blue (author names are capitalized). Only documents with known topics are shown. The representation of topics is as in Figure 5. We used 250 authors and 2000 words, chosen as in Figures 5 and 7. The three figures show insets of the complete embedding, which roughly correspond to the insets in Figure 6. **(a)** The border region between Algorithms and Architectures (AA) and Learning Theory (LT). **(b)** The border region between Neuroscience NS and Biological Vision (VB). **(c)** Control and Reinforcement Learning (CN) region.

- **Singular value decomposition (SVD)**. Applied SVD to two count-based matrices: $\overline{p}(doc, word)$ and $\log(\overline{p}(doc, word)+1)$. Assume the SVD of a matrix P is given by $P = USV^T$ (where S is diagonal with eigenvalues sorted in a decreasing order). Then the document embedding was taken to be $U\sqrt{S}$. Embeddings of dimension q were given by the first q columns of $U\sqrt{S}$. An embedding for words can be obtained in a similar manner, but was not used in the current evaluation.

- **Multidimensional scaling (MDS)**. MDS searches for an embedding of objects in a low dimensional space, based on a predefined set of pairwise distances (Cox and Cox, 1984). One heuristic approach that is sometimes used for embedding co-occurrence data using standard MDS is to calculate distances between row vectors of the co-occurrence matrix, which is given by $\overline{p}(doc, word)$ here. This results in an embedding of the row objects (documents). Column objects (words) can be embedded similarly, but there is no straightforward way of embedding both simultaneously. Here we tested two similarity measures between row vectors: The Euclidean distance, and the cosine of the angle between the vectors. MDS was applied using the implementation in the MATLAB Statistical Toolbox.

- **Isomap**. Isomap first creates a graph by connecting each object to m of its neighbors, and then uses distances of paths in the graph for embedding using MDS. We used the MATLAB implementation provided by the Isomap authors (Tenenbaum et al., 2000), with $m = 10$, which was the smallest value for which graphs were fully connected.

Of the above methods, only CA and CODE were used for joint embedding of words and documents. The other methods are not designed for joint embedding and were only used for embedding documents alone.

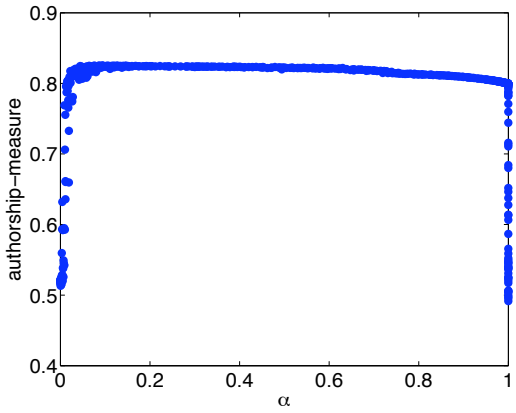


Figure 9: Evaluation of the authors-words-documents embedding for different likelihood weights. The X axis is a number α such that the weight on the $p_{CM}(doc, word)$ log-likelihood is $\frac{\alpha}{|word||doc|}$ and the weight on $p_{CM}(author, word)$ is $\frac{1-\alpha}{|author||doc|}$. The value $\alpha = 0.5$ results in equal weighting of the models after normalizing for size, and corresponds to the embedding shown in Figure 8. The Y axis is the authorship measure reflecting the quality of the joint document-author embedding.

All methods were also tested under several different normalization schemes, including TF/IDF weighting, and no document normalization. Results were consistent across all normalization schemes.

8.3 Quality Measures for Homogeneous and Heterogeneous Embeddings

Quantitative evaluation of embedding algorithms is not straightforward, since a ground-truth embedding is usually not well defined. Here we use the fact that documents are associated with class labels to obtain quantitative measures.

For the *homogeneous* embedding of the document objects, we define a measure denoted by *doc-doc*, which is designed to measure how well documents with identical labels are mapped together. For each embedded document, we measure the fraction of its neighbors that are from the same newsgroup. This is repeated for all neighborhood sizes,[16] and averaged over all documents and sizes, resulting in the *doc-doc* measure. The measure will have the value one for *perfect* embeddings where same topic documents are always closer than different topic documents. For a random embedding, the measure has a value of $1/(\sharp\text{newsgroups})$.

For the *heterogeneous* embedding of documents *and* words into a joint map, we defined a measure denoted by *doc-word*. For each document we look at its k nearest words and calculate their probability under the document's newsgroup.[17] We then average this over all neighborhood sizes of up to 100 words, and over all documents. It can be seen that the *doc-word* measure will be high if documents are embedded near words that are common in their class. This implies that by looking at the words close to a given document, one can infer the document's topic. The *doc-word* measure

16. The maximum neighborhood size is the number of documents per topic.
17. This measure was normalized by the maximum probability of any k words under the given newsgroup, so that it equals one in the optimal case.

could only be evaluated for CODE and CA since these are the only methods that provided joint embeddings.

8.4 Results

Figure 10 (top) illustrates the joint embedding obtained for the CODE model $p_{CM}(doc, word)$ when embedding documents from three different newsgroups. It can be seen that documents in different newsgroups are embedded in different regions. Furthermore, words that are indicative of a newsgroup topic are mapped to the region corresponding to that newsgroup.

To obtain a quantitative estimate of homogeneous document embedding, we evaluated the *doc-doc* measure for different embedding methods. Figure 11 shows the dependence of this measure on neighborhood size and embedding dimensionality, for the different methods. It can be seen that CODE is superior to the other methods across parameter values.

Table 1 summarizes the *doc-doc* measure results for all competing methods for seven different subsets.

Newsgroup Sets	CODE	Isomap	CA	MDS-e	MDS-c	SVD	SVD-l
`comp.os.ms-windows.misc`, `comp.sys.ibm.pc.hardware`	68*	65	56	54	53	51	51
`talk.politics.mideast`, `talk.politics.misc`	85*	83	66	45	73	52	52
`alt.atheism`, `comp.graphics`, `sci.crypt`	66*	58	52	53	62	51	51
`comp.graphics`, `comp.os.ms-windows.misc`	76	77*	55	55	53	56	56
`sci.crypt`, `sci.electronics`	84*	83	83	65	58	56	56
`sci.crypt`, `sci.electronics`, `sci.med`	82*	77	76	51	53	40	50
`sci.crypt`, `sci.electronics`, `sci.med`, `sci.space`	73*	65	58	29	50	31	44

Table 1: *doc-doc* measure values (times 100) for embedding of seven newsgroups subsets. Average over neighborhood sizes $1, \ldots, 1000$. Embedding dimension is $q = 2$. "MDS-e" stands for Euclidean distance, "MDS-c" for cosine distance, "SVD-l" preprocesses the data with $\log(count + 1)$. The best method for each set is marked with an asterisk (*).

To compare performance across several subsets, and since different subsets have different inherent "hardness", we define a normalized measure of purity that rescales the *doc-doc* measure performance for each of the 7 tasks. Results are scaled such that the best performing measure in a task has a normalized value of 1, and the one performing most poorly has a value of 0. As a result, any method that achieves the best performance consistently would achieve a normalized score of one. The normalized results are summarized in Figure 12a. CODE significantly outperforms other methods and IsoMap comes second.

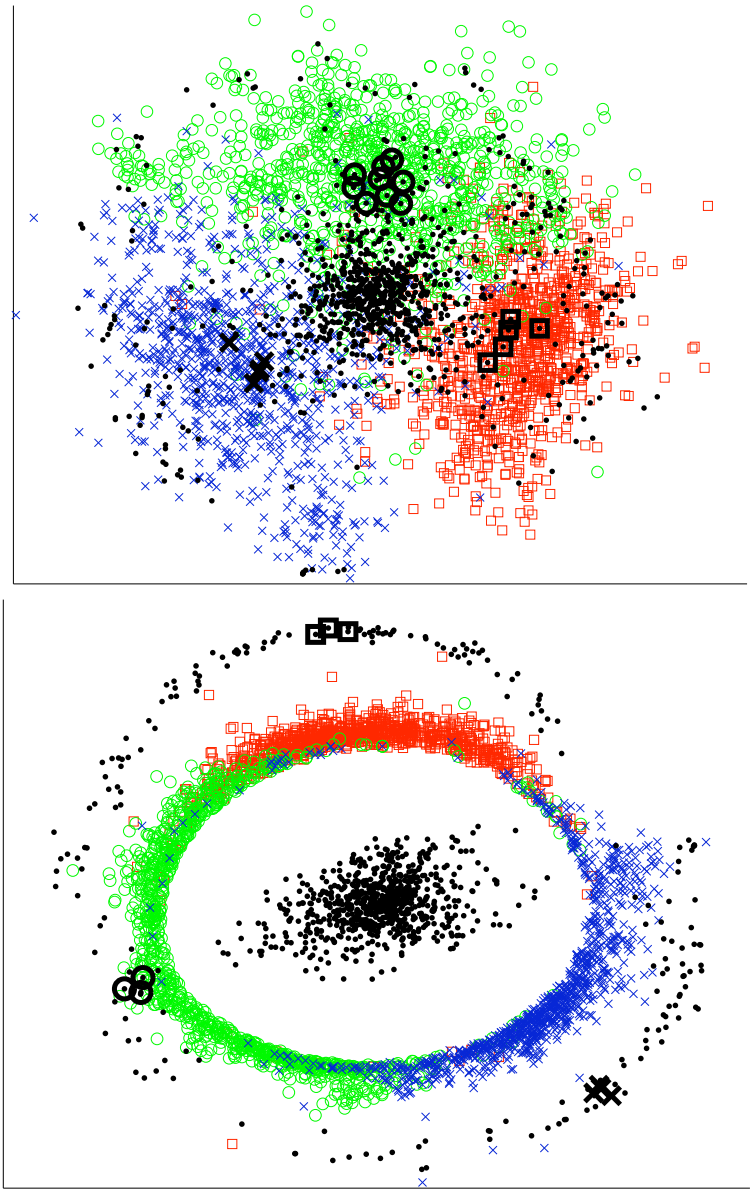


Figure 10: Visualization of two dimensional embeddings of the 20 newsgroups data under two different models. Three newsgroups are embedded: `sci.crypt` (red squares), `sci.electronics` (green circles) and `sci.med` (blue xs). **Top**: The embedding of documents and words using the conditional word-given-document model $p_{CM}(doc, word)$. Words are shown in black dots. Representative words around the median of each class are shown in black, with the marker shape corresponding to the class. They are $\{sick, hospital, study, clinical, diseases\}$ for `med`, $\{signal, filter, circuits, distance, remote, logic, frequency, video\}$ for `electronics`, and $\{legitimate, license, federal, court\}$ for `crypt`. **Bottom**: Embedding under the joint model $p_{MM}(doc, word)$. Representative words were chosen visually to be near the center of the arc corresponding to each class. Words are: $\{eat, AIDS, breast\}$ for `med`, $\{audio, noise, distance\}$ for `electronics`, and $\{classified, secure, scicrypt\}$ for `crypt`.

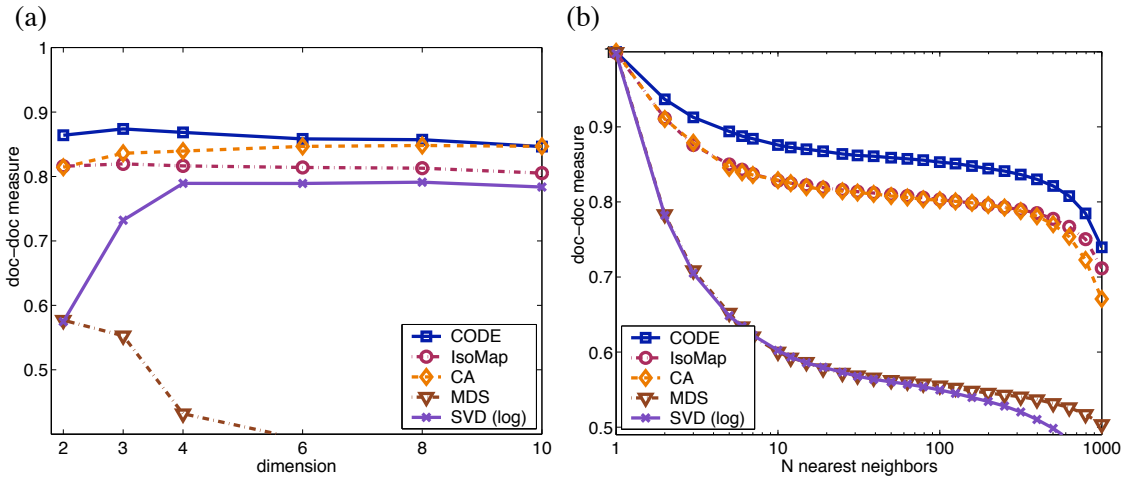


Figure 11: Parametric dependence of the *doc-doc* measure for different algorithms. Embeddings were obtained for the three newsgroups described in Figure 10. **(a)** *doc-doc* as a function of embedding dimensions. Average over neighborhood sizes $1,\ldots,100$. **(b)** *doc-doc* as a function of neighborhood size. Embedding dimension is $q = 2$

The performance of the heterogeneous embedding of words and documents was evaluated using the *doc-word* measure for the CA and CODE algorithms. Results for seven newsgroups are shown in Figure 12b, and CODE is seen to significantly outperform CA.

Finally, we compared the performance of the gradient optimization algorithm to the PSD-CODE method described in Section 5. Here we used a smaller data set because the number of the parameters in the PSD algorithm is quadratic in $|X| + |Y|$. Results for both the *doc-doc* and *doc-word* measures are shown in Figure 13, illustrating the effectiveness of the PSD algorithm, whose performance is similar the to non-convex gradient optimization scheme, and is sometimes even better.

8.5 Comparison Between Different Distribution Models

Section 2 introduced a class of possible probabilistic models for heterogeneous embedding. Here we compare the performance of these models on the 20 Newsgroup data set.

Figure 10 shows an embedding for the conditional model p_{CM} in Equation 3 and for the symmetric model p_{MM}. It can be seen that both models achieve a good embedding of both the relation between documents (different classes mapped to different regions) and document-word relation (words mapped near documents with relevant subjects). However, the p_{MM} model tends to map the documents to a circle. This can be explained by the fact that it also partially models the marginal distribution of documents, which is uniform in this case.

A more quantitative evaluation is shown in Figure 14. The figure compares various CODE models with respect to the *doc-doc* and *doc-word* measures. While all models perform similarly on the *doc-doc* measure, the *doc-word* measure is significantly higher for the two models $p_{MM}(doc, word)$ and $p_{CM}(doc, word)$. These models incorporate the marginals over words, and directly model the statistical dependence ratio $\bar{r}_p(x,y)$, as explained in Section 2. The model $p_{MC}(doc, word)$ does

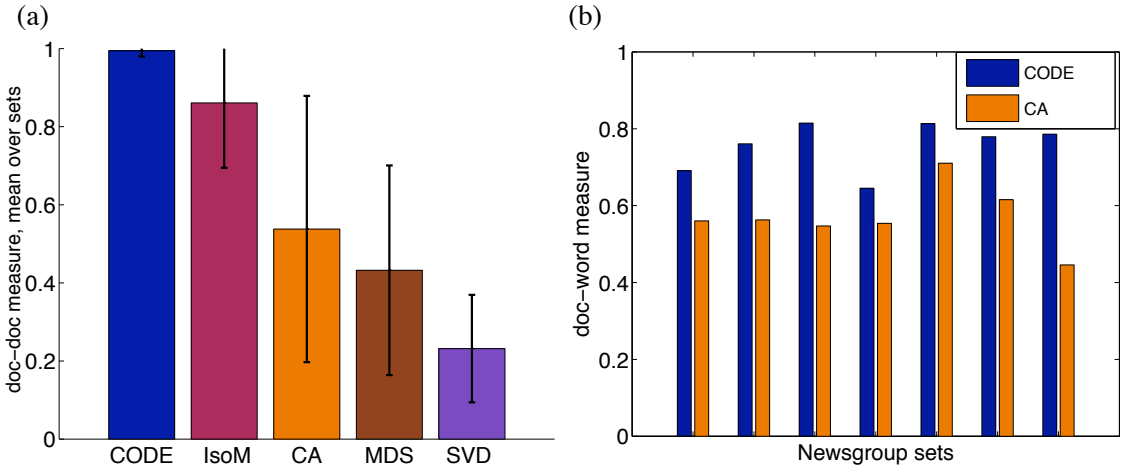


Figure 12: **(a)** *Normalized* doc-doc *measure* (see text) averaged over 7 newsgroup sets. Embedding dimension is $q = 2$. Sets are detailed in Table 1. Normalized *doc-doc* measure was calculated by rescaling at each data set, such that the poorest algorithm has score 0 and the best a score of 1. **b** The *doc-word* measure for the CODE and CA algorithms for the seven newsgroup sets. Embedding dimension is $q = 2$.

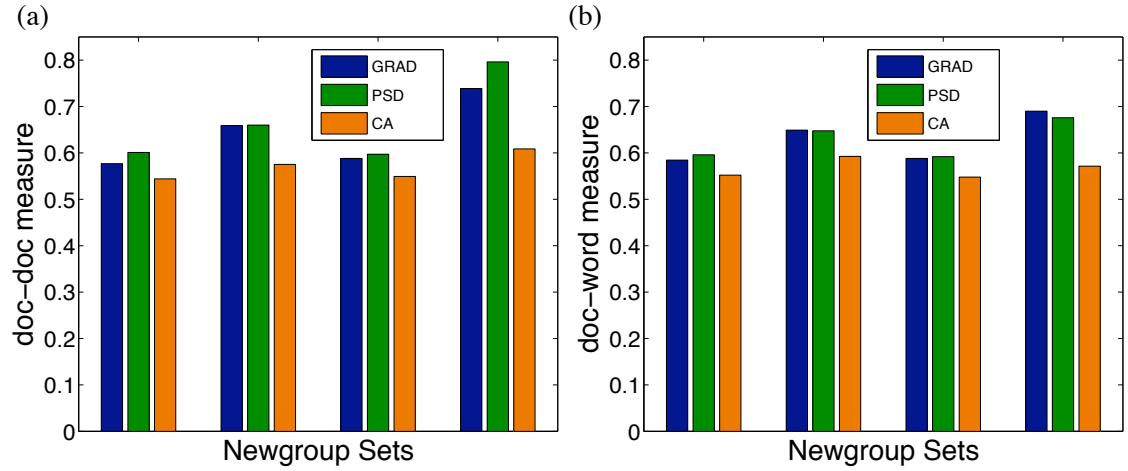


Figure 13: Comparison of the PSD-CODE algorithm with a gradient based maximization of the CODE likelihood (denoted by GRAD) and the correspondence analysis (CA) method. Both CODE methods used the $p_{CM}(doc, word)$ model. Results for $q = 2$ are shown for five newsgroup pairs (given by rows 1,2,4,5 in Table 1). Here 500 words were chosen, and 250 documents taken from each newsgroup. **a.** The *doc-doc* measure. **b.** The *doc-word* measure.

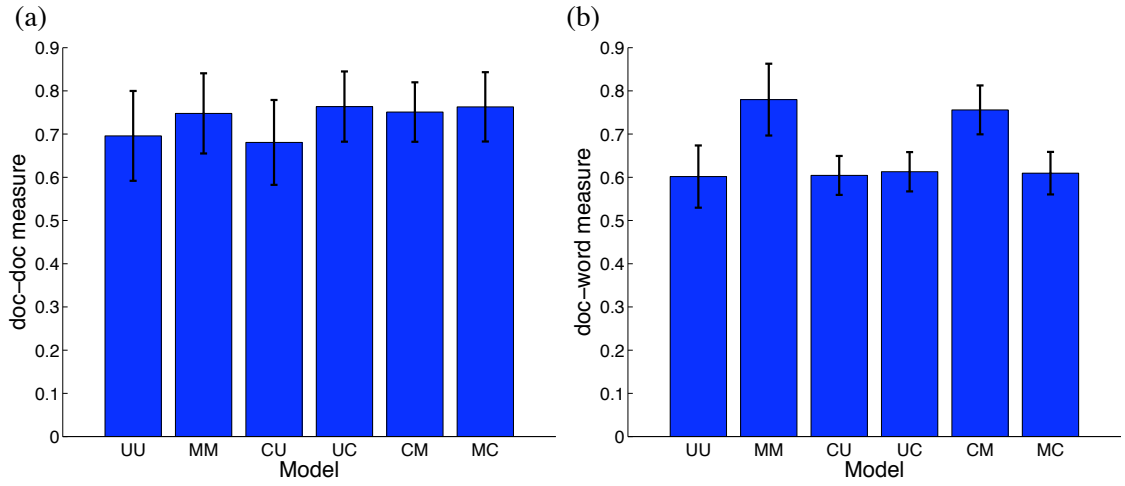


Figure 14: Comparison of different embedding models. Averaged results for the seven newsgroup subsets are shown for the *doc-doc* (left figure) and *doc-word* (right figure) measures. Model names are denoted by two letters (see Section 2.3), which reflect the treatment of the *document* variable (first letter) and *word* variable (second letter). Thus, for example *CM* indicates conditioning on the document variable, whereas *MC* indicates conditioning on the word variable.

not perform as well, presumably because it makes more sense to assume that the document is first chosen, and then a word is chosen given the document, as in the $p_{CM}(doc, word)$ model.

9. Discussion

We presented a method for embedding objects of different types into the same low dimension Euclidean space. This embedding can be used to reveal low dimensional structures when distance measures between objects are unknown or cannot be defined. Furthermore, once the embedding is performed, it induces a meaningful metric between objects of the same type. Such an approach may be used, for example, for embedding images based on accompanying text, and derive the semantic distance between images.

We showed that co-occurrence embedding relates statistical correlation to the local geometric structure of one object type with respect to the other. Thus the local geometry may be used for inferring properties of the data. An interesting open issue is the sensitivity of the solution to sample-to-sample fluctuation in the empirical counts. One approach to the analysis of this problem could be via the Fisher information matrix of the model.

The experimental results shown here focused mainly on the conditional based model of Equation 4. However, different models may be more suitable for data types that have no clear asymmetry.

An important question in embedding objects is whether the embedding is unique, namely, can there be two *different* optimal configurations of points. This question is related to the rigidity and uniqueness of graph embeddings, and in our particular case, complete bipartite graphs. A theorem of Bolker and Roth (1980) asserts that embeddings of complete bipartite graphs with at least 5

vertices on each side, are guaranteed to be rigid, that is they cannot be continuously transformed. This suggests that the CODE embeddings for $|X|, |Y| \geq 5$ are locally unique. However, a formal proof is still needed.

Co-occurrence embedding does not have to be restricted to distributions over pairs of variables, but can be extended to multivariate joint distributions. One such extension of CODE would be to replace the dependence on the pairwise distance $\|\phi(x) - \psi(x)\|$ with a measure of average pairwise distances between multiple objects. For example, given three variables X, Y, Z one can relate $p(x,y,z)$ to the average distance of $\phi(x), \psi(y), \xi(z)$ from their centroid $\frac{1}{3}(\phi(x) + \psi(y) + \xi(z))$. The method can also be augmented to use statistics of same-type objects when these are known, as discussed in Section 6.1.

An interesting problem in many embedding algorithms is generalization to new values. Here this would correspond to obtaining embeddings for values of X or Y such that $\overline{p}(x) = 0$ or $\overline{p}(y) = 0$, for instance because a word did not appear in the sample documents. When variables are purely categorical and there is no intrinsic similarity measure in either the X or Y domains, there is little hope for generalizing to new values. However, in some cases the X or Y variables may have such structure. For example, objects in X may be represented as vectors in $\mathbb{R}^p$. This information can help in generalizing embeddings, since if x_1 is close to x_2 in $\mathbb{R}^p$ it may be reasonable to assume that $\phi(x_1)$ should be close to $\phi(x_2)$. One strategy for applying this intuition is to model $\phi(x)$ as a continuous function of x, for instance a linear map Ax or a kernel-based map. Such an approach has been previously used to extend embedding methods such as LLE to unseen points (Zhang, 2007). This approach can also be used to extend CODE and it will be interesting to study it further. It is however important to stress that in many cases no good metric is known for the input objects, and it is a key advantage of CODE that it can produce meaningful embeddings in this setting.

These extensions and the results presented in this paper suggest that probability-based continuous embeddings of categorical objects could be applied efficiently and provide accurate models for complex high dimensional data.

Appendix A. A Short Review of Correspondence Analysis

Correspondence analysis (CA) is an exploratory data analysis method that embeds two variables X and Y into a low dimensional space such that the embedding reflects their statistical dependence (Greenacre, 1984). Statistical dependence is modeled by the ratio

$$q(x,y) = \frac{\overline{p}(x,y) - \overline{p}(x)\overline{p}(y)}{\sqrt{\overline{p}(x)\overline{p}(y)}}$$

Define the matrix Q such that $Q_{xy} = q(x,y)$. The CA algorithm computes an SVD of Q such that $Q = USV$ where S is diagonal and U, V are rectangular orthogonal matrices. We assume that the diagonal of S is sorted in descending order. To obtain the low dimensional embeddings, one takes the first q columns and rows of $P_x^{-0.5} S\sqrt{U}$ and $P_y^{-0.5} S\sqrt{U}$ respectively, where P_x, P_y are diagonal matrices with $\overline{p}(x), \overline{p}(y)$ on the diagonal. It can be seen that this procedure corresponds to a least squares approximation of the matrix Q via a low dimensional decomposition. Thus, CA cannot be viewed as a statistical model of $\overline{p}(x,y)$, but is rather an L_2 approximation of empirically observed correlation values.

The ratio $q(x,y)$ is closely related to the chi-squared distance between distributions, and there indeed exist interpretations (Greenacre, 1984) of CA which relate it to approximating this distance.

Also, as mentioned in Section 4, it can be shown (Hill, 1974) that CA corresponds to Canonical Correlation Analysis when X and Y are represented via indicator vectors. For example, $X = 3$ is represented as a vector $\mathbf{e} \in \mathbb{R}^{|X|}$ such that $\mathbf{e}(3) = 1$ and all other elements are zero.

References

F. R. Bach and M. I. Jordan. Kernel independent component analysis. *Journal of Machine Learning Research*, 3:1–48, 2002.

A. L. Berger, S.A. Della Pietra, and V.J. Della Pietra. A maximum entropy approach to natural language processing. *Computational Linguistics*, 22(1):39–71, 1996.

D. P. Bertsekas. On the Goldstein-Levitin-Polyak gradient projection method. *IEEE Transactions on Automatic Control*, 21:174–184, 1976.

E.D. Bolker and B. Roth. When is a bipartite graph a rigid framework? *Pacific Journal of Mathematics*, 90:27–44, 1980.

S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge Univ. Press, 2004.

M. Chalmers and P. Chitson. Bead: explorations in information visualization. In *Proceedings of the 15th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 330–337. ACM Press, New York, NY, 1992.

G. Chechik and N. Tishby. Extracting relevant structures with side information. In S. Becker, S. Thrun, and K. Obermayer, editors, *Advances in Neural Information Processing Systems 15*, pages 857–864. MIT Press, Cambridge, MA, 2003.

M. Chiang. Geometric programming for communication systems. *Foundations and Trends in Communications and Information Theory*, 2(1):1–154, 2005.

T.M. Cover and J.A Thomas. *Elements of Information Theory*. Wiley-Interscience, New York, 1991.

T. Cox and M. Cox. *Multidimensional Scaling*. Chapman and Hall, London, 1984.

M. Fazel, H. Hindi, and S. P. Boyd. A rank minimization heuristic with application to minimum order system approximation. In *Proceedings of the American Control Conference*, volume 6, pages 4734–4739. American Automatic Control Council, New York, 2001.

R.A. Fisher. The precision of discriminant functions. *Annals of Eugenics, London*, 10:422–429, 1940.

A. Globerson and N. Tishby. Sufficient dimensionality reduction. *Journal of Machine Learning Research*, 3:1307–1331, 2003.

A. Globerson, G. Chechik, F. Pereira, and N.Tishby. Euclidean embedding of co-occurrence data. In L. K. Saul, Y. Weiss, and L. Bottou, editors, *Advances in Neural Information Processing Systems 17*, pages 497–504. MIT Press, Cambridge, MA, 2005.

M.J. Greenacre. *Theory and Applications of Correspondence Analysis*. Academic Press, London, 1984.

J.H. Ham, D.D. Lee, and L.K. Saul. Learning high dimensional correspondences with low dimensional manifolds. In *Proceedings of the 20th International Conference on Machine Learning. Workshop on The Continuum from Labeled to Unlabeled Data in Machine Learning and Data Mining*, pages 34–41, 2003.

M.O. Hill. Correspondence analysis: A neglected multivariate method. *Applied Statistics*, 23(3): 340–354, 1974.

G. Hinton and S.T. Roweis. Stochastic neighbor embedding. In S. Becker, S. Thrun, and K. Obermayer, editors, *Advances in Neural Information Processing Systems 15*, pages 833–840. MIT Press, Cambridge, MA, 2003.

T. Hofmann. Unsupervised learning by probabilistic latent semantic analysis. *Machine Learning*, 42(1):177–196, 2001.

H. Hotelling. The most predictable criterion. *Journal of Educational Psychology*, 26:139–142, 1935.

T. Iwata, K. Saito, N. Ueda, S. Stromsten, T. Griffiths, and J. Tenenbaum. Parametric embedding for class visualization. In L. K. Saul, Y. Weiss, and L. Bottou, editors, *Advances in Neural Information Processing Systems 17*. MIT Press, Cambridge, MA, 2005.

P.L. Lai and C. Fyfe. Kernel and nonlinear canonical correlation analysis. In *International Joint Conference on Neural Networks*, pages 365–378. IEEE Computer Society, Los Alamitos, CA, 2000.

D. Lee and H. Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755):788–791, 1999.

X. Lin. Map displays for information retrieval. *Journal of the American Society for Information Science*, 48(1):40–54, 1997.

G. Mei and C. R. Shelton. Visualization of collaborative data. In R. Dechter and T. Richardson, editors, *Proceedings of the Twenty-Second International Conference on Uncertainty in Artificial Intelligence*, pages 341–348. AUAI Press, Arlington, VA, 2006.

G. Michailidis and J. de Leeuw. The Gifi system of descriptive multivariate analysis. *Statistical Science*, 13(4):307–336, 1998.

R. B. Nelsen. *An Introduction to Copulas*. Springer, New York, 1999.

V.Y. Pan and Z.Q. Chen. The complexity of the matrix eigenproblem. In *Proceedings of the Thirty-First Annual ACM Symposium on Theory of Computing*, pages 507–516. ACM Press, New York, NY, 1999.

S. T. Roweis and L. K. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290:2323–2326, 2000.

S.T. Roweis. NIPS 0-12 data. http://www.cs.toronto.edu/∼roweis/data.html, 2000.

J.B. Tenenbaum, V. de Silva, and J. C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290:2319–2323, 2000.

L. Vandenberghe and S. Boyd. Semidefinite programming. *SIAM Review*, 38(1):49–95, 1996.

K. Q. Weinberger and L. K. Saul. Unsupervised learning of image manifolds by semidefinite programming. *International Journal of Computer Vision*, 70(1):77–90, 2006.

E. Xing, A. Ng, M. Jordan, and S. Russell. Distance metric learning, with application to clustering with side-information. In S. Becker, S. Thrun, and K. Obermayer, editors, *Advances in Neural Information Processing Systems 15*, pages 505–512. MIT Press, Cambridge, MA, 2002.

S. Yan D. Xu B. Zhang H.J. Zhang. Graph embedding: a general framework for dimensionality reduction. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 40–51, 2007.

H. Zhong, J. Shi, and M. Visontai. Detecting unusual activity in video. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 819–826. IEEE Computer Society, Los-Alamitos, CA, 2004.

Journal of Machine Learning Research 8 (2007) 2297-2345 Submitted 2/06; Revised 8/06; Published 10/07

Harnessing the Expertise of 70,000 Human Editors: Knowledge-Based Feature Generation for Text Categorization*

Evgeniy Gabrilovich† GABR@YAHOO-INC.COM
Shaul Markovitch SHAULM@CS.TECHNION.AC.IL
Department of Computer Science
Technion—Israel Institute of Technology
32000 Haifa, Israel

Editor: Andrew McCallum

Abstract

Most existing methods for text categorization employ induction algorithms that use the words appearing in the training documents as features. While they perform well in many categorization tasks, these methods are inherently limited when faced with more complicated tasks where external knowledge is essential. Recently, there have been efforts to augment these basic features with external knowledge, including semi-supervised learning and transfer learning. In this work, we present a new framework for automatic acquisition of world knowledge and methods for incorporating it into the text categorization process. Our approach enhances machine learning algorithms with features generated from domain-specific and common-sense knowledge. This knowledge is represented by ontologies that contain hundreds of thousands of concepts, further enriched through controlled Web crawling. Prior to text categorization, a feature generator analyzes the documents and maps them onto appropriate ontology concepts that augment the bag of words used in simple supervised learning. Feature generation is accomplished through contextual analysis of document text, thus implicitly performing word sense disambiguation. Coupled with the ability to generalize concepts using the ontology, this approach addresses two significant problems in natural language processing—synonymy and polysemy. Categorizing documents with the aid of knowledge-based features leverages information that cannot be deduced from the training documents alone. We applied our methodology using the Open Directory Project, the largest existing Web directory built by over 70,000 human editors. Experimental results over a range of data sets confirm improved performance compared to the bag of words document representation.

Keywords: feature generation, text classification, background knowledge

1. Introduction

Text categorization deals with assigning category labels to textual documents. Categories come from a fixed set of labels (possibly organized in a hierarchy) and each document may be assigned one or more categories. Text categorization systems are useful in a wide variety of tasks, such as routing news and e-mail to appropriate corporate desks, identifying junk email, or correctly handling intelligence reports.

*. A preliminary version of this paper appeared in the Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence (IJCAI), Edinburgh, UK, August 2005 (Gabrilovich and Markovitch, 2005).

†. Current address: Yahoo! Research, 2821 Mission College Blvd, Santa Clara, CA 95054, USA.

The majority of existing text classification systems use various induction techniques, such as support vector machines, *k*-nearest neighbor algorithm, and neural networks. The features commonly used are the individual words appearing in the training documents (while their order within the document is ignored). The value of a feature for a particular document is usually its occurrence frequency normalized by its occurrence frequency within the whole collection of documents. This representation scheme treats each document as a bag of the words it contains, and is therefore known as the *bag of words* (BOW) approach (Salton and McGill, 1983).

The bag of words method is very effective in easy to medium difficulty categorization tasks where the category of a document can be identified by several easily distinguishable keywords. There are, however, two major weaknesses to the BOW representation scheme that limit its usefulness for more demanding categorization tasks. The first one stems from representing a document as a *bag*, thus ignoring the order of words appearance. This limits the possibility of handling structures that are based on more than one word, and also limits the possibility of disambiguating words based on their context.

The second weakness is the usage of only words that are explicitly mentioned in the training documents, without any knowledge about them. Because this approach cannot generalize over words, words in the testing document that never appeared in the training set are necessarily ignored. Nor can synonymous words that appear infrequently in training documents be used to infer a more general principle that covers several cases.

There have been a number of efforts to extend the basic BOW approach. Several studies augmented the bag of words with n-grams (Caropreso et al., 2001; Peng and Shuurmans, 2003; Mladenic, 1998b; Raskutti et al., 2001) or statistical language models (Peng et al., 2004). Others used linguistically motivated features based on syntactic information, such as that available from part-of-speech tagging or shallow parsing (Sable et al., 2002; Basili et al., 2000). Additional studies researched the use of word clustering (Baker and McCallum, 1998; Bekkerman, 2003; Dhillon et al., 2003), as well as dimensionality reduction techniques such as LSA (Deerwester et al., 1990; Hull, 1994; Zelikovitz and Hirsh, 2001; Cai and Hofmann, 2003).

More recently, there have been a number of efforts to add outside knowledge to supervised machine learning techniques. Transfer learning approaches (Bennett et al., 2003; Do and Ng, 2005; Sutton and McCallum, 1998; Raina et al., 2006) leverage information from different but related learning tasks. Pseudo-relevance feedback (Ruthven and Lalmas, 2003) uses information from the top-ranked documents, which are assumed to be relevant to the query; for example, characteristic terms from such documents may be used for query expansion (Xu and Croft, 1996). Recent studies on semi-supervised methods (Goldberg and Zhu, 2006; Ando and Zhang, 2005a,b; Blei et al., 2003; Nigam et al., 2000; Joachims, 1999b) infer information from unlabeled data, which is often available in much larger amounts than labeled data.

We argue that in order to perform text categorization well, the computer needs access to much more extensive and deep knowledge. Over a decade ago, Lenat and Feigenbaum (1990) formulated the *knowledge principle*, which postulated that "If a program is to perform a complex task well, it must know a great deal about the world it operates in." Text categorization is certainly a complex task. While the basic approaches are able to identify commonalities between documents based on word identity, and more advanced approaches can recognize synonyms, there are cases where identifying commonality between documents requires recognition of more elaborated semantic relations between terms.

For illustration, consider document #15264 in Reuters-21578, which is one of the most frequently used data sets in text categorization research. This document discusses a joint mining venture by a consortium of companies, and belongs to the category "copper." However, this fairly long document mentions only briefly that the aim of this venture is mining copper; rather, its main focus is on the mutual share holdings of the companies involved (Teck Corporation, Cominco, and Lornex Mining), as well as other mining activities of the consortium. Consequently, the three very different text classifiers that we used (SVM, KNN and C4.5) failed to classify the document correctly. This comes as no surprise—"copper" is a fairly small category, and none of these companies, nor the location of the venture (Highland Valley in British Columbia, Canada) is ever mentioned in the training set for this category.

We argue that this need not be the case. When a Reuters editor originally handled this document, she most likely knew quite a lot about the business of these companies, and easily assigned the document to the category "copper." It is this kind of knowledge that we would like machine learning algorithms to have access to.

In this paper we introduce a method for enhancing machine learning algorithms with a large volume of knowledge extracted from available human-generated repositories. Our method capitalizes on the power of existing induction techniques while enriching the language of representation, namely, exploring new feature spaces. Prior to text categorization, we employ a *feature generator* that uses common-sense and domain-specific knowledge to enrich the bag of words with new, more informative and discriminating features. Feature generation is performed automatically, using machine-readable hierarchical repositories of knowledge. Many sources of world knowledge have become available in recent years, thanks to rapid advances in information processing, and Internet proliferation in particular. Examples of general purpose knowledge bases include the Open Directory Project (ODP), Yahoo! Web Directory, and the Wikipedia encyclopedia.

It is interesting to juxtapose our method with above-mentioned alternative approaches that augment the training set of documents with external knowledge. Semi-supervised learning uses unlabeled data to gather additional features beyond those originally available in the input. Transfer learning involves pairs of related learning tasks, so that features constructed while solving one problem can then also be used for solving another problem. On the other hand, the methods we propose in this paper build new features using knowledge explicitly cataloged by humans, which comes in the form of concepts that correspond to the nodes of the Open Directory.

In this paper we use the ODP as a source of background knowledge. The Open Directory catalogs millions of Web sites in a rich hierarchy of 600,000 categories, and represents the collective knowledge of over 70,000 volunteer editors. Thus, in the above example, the feature generator "knows" that the companies mentioned are in the mining business, and that Highland Valley happens to host a copper mine. This information is available in Web pages that discuss the companies and their operations, and are cataloged in corresponding ODP categories such as MINING_AND_DRILLING and METALS. Similarly, Web pages about Highland Valley are cataloged under REGIONAL/NORTH_AMERICA/CANADA/BRITISH_COLUMBIA. To amass this information, we crawl the URLs cataloged in the ODP, thus effectively multiplying the amount of knowledge available many times over. Armed with this knowledge, the feature generator constructs new features that denote these ODP categories, and adds them to the bag of words. The augmented feature space provides text classifiers with a cornucopia of additional information. Indeed, our implementation of the proposed methodology classifies this document correctly. It is essential to mention that this entire scheme works automatically. Given an existing knowledge hierarchy (ODP in this case), the

feature generator examines documents and enriches their representation in a completely mechanical way.

The contributions of this paper are threefold. First, we propose a framework and a collection of algorithms that perform feature generation using very large-scale repositories of human knowledge. Second, we propose a novel kind of contextual analysis performed during feature generation, which views the document text as a sequence of local contexts, and performs implicit word sense disambiguation. Finally, we describe a way to further enhance existing knowledge bases by several orders of magnitude by crawling the World Wide Web. Performing feature generation using external knowledge effectively capitalizes on human knowledge (as encoded by the editors of the Open Directory), leveraging information that cannot be deduced solely from the texts being classified. As we show in Section 5, our approach performs markedly better than the bag of words method.

We believe that this research is only one step towards computerized use of large-scale structured repositories of human knowledge. In our future work, we plan to study possible uses of other knowledge repositories in addition to the Open Directory. We also intend to apply the feature generation methodology to additional natural language processing tasks, as well as to study its applicability beyond text processing. It would also be very interesting to compare the results of the feature generation methodology presented in this paper to other techniques that use unlabeled data, such as semi-supervised and transfer learning; this comparison is also left to future work.

The rest of the paper is organized as follows. In Section 2 we analyze the limitations of the BOW approach. Section 3 describes how our feature generation methodology uses repositories of human knowledge to overcome these limitations. Section 4 instantiates this methodology with a particular knowledge resource, the Open Directory Project. In Section 5 we report the results of evaluating the proposed methodology empirically on a variety of test collections, and outline the implementation details of our system. In Section 6 we discuss our methodology in the context of prior work and related literature. Section 7 concludes the paper and outlines directions for future research.

2. Problems in the Bag of Words Approach

Since the majority of existing text categorization systems employ the bag of words approach to represent documents, we begin by analyzing typical problems and limitations of this method.

1. Words that appear in *testing* documents but not in *training* documents are completely ignored by the basic BOW approach that does not use external data to compensate for such vocabulary mismatch. Since the classification model is built with a subset of words that appear in the training documents, words that do not appear there are excluded by definition. Lacking the ability to analyze such words, the system may overlook important parts of the document being classified.

   Example: Document #15264 from Reuters-21578 described in the Introduction presents a perfect example of this limitation. This document describes a copper-mining venture formed by a group of companies, whose names are not mentioned even once in the training set, and are thus ignored by the classification model.

2. Words that appear infrequently in the training set, or appear just once, are mostly ignored even if they are essential for proper classification. It often happens that human annotators

assign a document to a certain category based on some notion briefly mentioned in the document. If the words that describe this notion do not appear with sufficient frequency elsewhere in the training set, then the system will overlook the real reason for this document's annotation. Consequently, it will either come up with some spurious association between the actual category and unrelated words or ignore this document as a training example altogether.

Example: Suppose we have a collection of pharmaceutical documents and are trying to learn the concept of antibiotics. If a particular training document describes the results of a clinical trial for a new antibiotic drug, and mentions it only by a brand name that does not appear elsewhere in the training set, the system will likely miss an important piece of evidence.

3. The problem described in the previous item can manifest itself in a more extreme way. Suppose we have a group of related words, where each word appears only a few times in the collection, and few documents contain more than one word of the group. As a result, the connection between these words remains implicit and cannot be learned without resorting to external knowledge. External knowledge, however, allows us to determine that certain words are related. Furthermore, we can use the generalization ability of hierarchical knowledge organization to establish that the words correspond to specific instances of the same general notion.

   Example: Consider a collection of clinical narrative reports on administering various antibiotic drugs. Since such reports are circulated among medical professionals, they are likely to refer to specific drugs by name, while omitting the knowledge already shared by the target audience. Hence, the reports will likely not explain that each drug is actually an antibiotic. In the absence of this vital piece of knowledge, the BOW approach can easily fail to learn the notion shared by the reports.

   Speaking more generally, we observe that a critical limitation of the BOW approach lies in its ignorance of the connections between the words. Thus, even more difficult than the problem described in the previous item, is the one where we have several related phrases or longer contexts, while the connection between them is not stated in any single document.

   Example: Consider again a collection of clinical reports, which are inherently rich in diverse medical terminology. Often, each report describes the case of a single patient. Thus, without extensive medical knowledge it would be nearly impossible to learn that Lown-Ganong-Levine Syndrome and Wolff-Parkinson-White Syndrome are different kinds of arrhythmia, while Crigler-Najjar Syndrome and Gilbert Syndrome are two kinds of liver diseases.

4. Because contextual adjacency of words is not taken into account by the BOW approach, word sense disambiguation can only be performed at the level of entire documents, rather than at much more linguistically plausible levels of a single sentence or paragraph.

   Example: As an extreme example of this limitation, consider a document about the Jaguar company establishing a conservation trust to protect its namesake animal (`http://www.jaguarusa.com/us/en/company/news_events/archive/Jaguar_Conservation_trust_longcopy.htm`). This fairly long document is devoted mainly to the preservation of wildlife, while briefly covering the history of the car manufacturer in its last paragraph. Taken as a single bag of words, the document will likely be classified as strongly related to jaguar the animal, while the cursory mention of Jaguar the company will likely be ignored.

Some of these limitations are due to data sparsity—after all, if we had infinite amounts of text on every imaginable topic, the bag of words would perform much better. Many studies in machine learning and natural language processing addressed the sparsity problem. Approaches like smoothing (Chen and Goodman, 1996) allocate some probability mass for unseen events and thus eliminate zero probabilities. These approaches facilitate methods that are sensitive to zero probabilities (e.g., Naive Bayes), but essentially do not use any external knowledge. More elaborate techniques such as transfer learning (Bennett et al., 2003; Do and Ng, 2005; Sutton and McCallum, 1998; Raina et al., 2006) and semi-supervised learning (Goldberg and Zhu, 2006; Ando and Zhang, 2005a,b; Blei et al., 2003; Nigam et al., 2000; Joachims, 1999b), leverage cooccurrence information from similar learning tasks or from unlabeled data. Other studies that addressed the sparsity problem include using the EM algorithm with unlabeled data (Nigam et al., 2006, 2000), latent semantic kernels (Cristianini et al., 2002), transductive inference (Joachims, 1999b), and generalized vector space model (Wong et al., 1985).

Humans avoid these limitations due to their extensive world knowledge, as well as their ability to understand the words in context rather than just view them as an unordered bag. Our approach that uses structured background knowledge is somewhat reminiscent of explanation-based learning (Mitchell et al., 1986; Dejong and Mooney, 1986), where generalizations of previously seen examples are reused in future problem solving tasks, thus mimicking humans' ability to learn from a single example. Later in the paper we show how the above problems and limitations can be resolved through the use of knowledge-based feature generation.

3. Feature Generation Methodology

Having presented the problems with the BOW approach in the previous section, we continue by defining the guidelines for building a feature generation framework that will address and alleviate these problems using repositories of human knowledge.

3.1 Overview

The proposed methodology allows principled and uniform integration of one or more sources of external knowledge to construct new features. These knowledge sources define a collection of concepts that are assigned to documents to qualify their text. In the preprocessing step, we build a feature generator capable of representing documents in the space of these concepts. The feature generator is then invoked prior to text categorization to assign each document with a number of relevant concepts. Subsequently, these concepts give rise to a set of constructed features that provide background knowledge about the document's content. The constructed features can then be used either in conjunction with or in place of the original bag of words. The resulting set undergoes feature selection, and the most discriminative features are retained for document representation. Finally, we use traditional text categorization techniques to learn a text categorizer in the new feature space.

3.2 Requirements on Suitable Knowledge Repositories

We impose the following requirements on knowledge repositories for feature generation:

1. The repository contains a collection of *concepts* organized in a hierarchical tree structure, where edges represent the "is-a" relationship. Each hierarchy node is labeled with a concept,

which is more general than those of its children. Although in principle we could perform feature generation with a flat set of concepts, using a hierarchical ontology allows us to perform powerful generalizations. Optionally, a concept may be accompanied by a brief textual description.

Formally, let *KR* be a knowledge repository that contains concepts $C = \{c_0, \ldots, c_n\}$. Let c_0 be the *root node*, which is more general than any other node. Let $Parent(c_i)$ be a function that uniquely associates a node with its parent in the hierarchy, whereas $Parent(c_0)$ is undefined. Let $Children(c_i)$ be a function that associates a node with a set of its children, where for leaf nodes $Children(c_j) = \emptyset$. When concept c_i is more general than another concept c_j, we denote this by $c_i \sqsubseteq c_j$; this happens when $c_j \in Children^*(c_i)$, where $Children^*$ denotes the recursive application of the function (obviously, $\forall j > 0 : c_0 \sqsubseteq c_j$). If additional textual description is available for a concept, it is denoted by $Description(c_i)$; otherwise this function returns an empty set of words.

2. There is a collection of texts associated with each concept. The feature generator uses these texts to learn the definition and scope of the concept, in order to be able to assign it to relevant documents. We refer to these texts as *textual objects*, and denote the set of such objects associated with concept c_i as $T_i = \{t_{i,1}, \ldots, t_{i,m_i}\}$.

Let W be a set of words. Our goal is to build a mapping function $f : W^* \rightarrow 2^C$. We propose building the mapping function using text categorization techniques. This is a very natural thing to do, as text categorization is all about assigning documents or parts thereof to a predefined set of categories (concepts in our case). One way to do so is to use a binary learning algorithm $L(Pos, Neg)$ to build a set of n binary classifiers, $f_1, \ldots, f_n$, such that $f_i : W^* \rightarrow \{0, 1\}$. This way, individual classifiers are built using the chosen learning algorithm: $f_i = L(T_i, \bigcup_{1 \leq j \leq n, j \neq i} T_j)$. Another way to build such a mapping function is to devise a hierarchical text classifier that takes advantage of the hierarchical organization of categories. In this paper, we use a simpler approach of building a single classifier that simultaneously considers all categories for each input sequence of words.

We believe that the above requirements are not overly restrictive. Indeed, there are quite a few sources of common-sense and domain-specific knowledge that satisfy these requirements. We list below several notable examples.

- Internet directories such as the Yahoo Web Directory (`http://dir.yahoo.com`), the Open Directory Project (`http://www.dmoz.org`) and the LookSmart directory (`http://search.looksmart.com/p/browse`) catalog huge numbers of URLs organized in an elaborate hierarchy. The Web sites pointed at by these URLs can be crawled to gather a wealth of information about each directory node. Here each directory node defines a concept, and crawling the Web sites cataloged under the node provides a collection of textual objects for that node.

- The Medical Subject Headings (MeSH) taxonomy (MeSH, 2003), which defines over 18,000 categories and is cross-linked with the MEDLINE database of medical articles, is a notable example of a domain-specific knowledge base. Here the hierarchy nodes again induce a set of concepts. The MEDLINE links mean that MeSH nodes can be easily associated with numerous scientific articles that are highly relevant to the scope of the node, yielding a set of textual objects for that node.

- Other domain-specific hierarchies are also available, notably in the terminology-rich law domain, which includes the KeySearch taxonomy by WestLaw (`http://west.thomson.com/westlaw/keysearch`) and the Web-based FindLaw hierarchy (`http://www.findlaw.com`) (both of them cross-linked with material relevant for each node).

- The US Patent Classification (`http://www.uspto.gov/go/classification`) and the International Patent Classification (`http://www.wipo.int/classifications/ipc/en`) are exceptionally elaborate taxonomies, where each node is linked to relevant patents.

- The online Wikipedia encyclopedia (`http://www.wikipedia.org`) has a fairly shallow hierarchy but its nodes contain very high-quality articles, which are mostly noise-free (except for occasional spamming).

- In the brick-and-mortar world, library classification systems such as the Universal Decimal Classification (UDC) (Mcilwaine, 2000), the Dewey Decimal Classification (Dewey et al., 2003) or the Library of Congress Classification (Chan, 1999) provide hierarchical structuring of human knowledge for classifying books. By the very virtue of their definition, each hierarchy node can be associated with the text of books cataloged under the node.

In this work we use the ODP as our knowledge base, due to the easy accessibility of its structure and linked resources (cataloged Web sites). However, our methodology is general enough to facilitate other knowledge repositories such as those listed above, and in our future work we intend to explore their utility as well, focusing in particular on the MeSH hierarchy for domain-specific feature generation. In a recent study (Gabrilovich and Markovitch, 2006), we used the Wikipedia encyclopedia as a source of knowledge for feature generation.

A note on terminology is in order here. The most commonly used term for nodes of hierarchical directories of knowledge is "category." In text categorization, however, this term normally refers to topical labels assigned to documents. To prevent possible confusion, we use the word "concept" to refer to the former notion. We represent such concepts as vectors in a high-dimensional space of "attributes." Again, we avoid using the term "features," which is reserved for denoting individual entries of document vectors in text categorization per se.

3.3 Building a Feature Generator

The first step in our methodology is preprocessing, performed once for all future text categorization tasks. We induce a hierarchical text classifier that maps pieces of text onto relevant knowledge concepts, which later serve as generated features. The resulting classifier is called a *feature generator* according to its true purpose in our scheme, as opposed to the text categorizer (or classifier) that we build ultimately. The feature generator represents concepts as vectors of their most characteristic words, which we call *attributes* (reserving the term *features* to denote the properties of documents in text categorization).

The feature generator operates similarly to a regular text classifier—it first learns a classification model in the space of concept attributes, and then identifies a set of concepts that are most appropriate to describe the contents of the input document. Observe that the number of concepts to which the feature generator classifies document text is huge, as suitable knowledge repositories may contain tens and even hundreds of thousands of concepts. Few machine learning algorithms can efficiently

handle so many different classes and about an order of magnitude more of training examples. Suitable candidates include the nearest neighbor and the Naive Bayes classifier (Duda and Hart, 1973), as well as prototype formation methods such as Rocchio (Rocchio, 1971) or centroid-based (Han and Karypis, 2000) classifiers. A radically different approach would avoid considering all existing concepts simultaneously, rather, it would work top-down into the hierarchy, identifying several most suitable concepts at each level, as in the hierarchical text classifiers described in the literature (Koller and Sahami, 1997; Dumais and Chen, 2000; Ruiz and Srinivasan, 2002).

3.3.1 ATTRIBUTE SELECTION

Prior to learning a text classifier that will act as a feature generator, we represent each concept as an attribute vector. To this end, we pool together all the textual objects for the concept and all of its descendants, and represent the accumulated description with a vector of words. Using all encountered words as attributes is impractical because it yields a classification model that is too big, and because this would inevitably increase the level of noise. The former consideration is essential to allow fitting the induced model into computer memory. The latter consideration is particularly important for Web-based knowledge repositories, which are inherently plagued with noise ranging from intentional directory spamming to merely irrelevant information. To remedy the situation, we perform *attribute selection* for each concept prior to learning the feature generator.

To this end, we use standard attribute selection techniques (Sebastiani, 2002) such as information gain, and identify words that are most characteristic of a concept versus all other concepts. This approach to attribute selection is reminiscent of the approaches described by Chakrabarti et al. (1997) and by Koller and Sahami (1997). Let us denote by D_i the collection of textual objects of c_i and its descendants, $D_i = \{t_{j,k} | c_i \sqsubseteq c_j\}$, and by $\overline{D_i}$ the collection of textual objects for all other concepts, $\overline{D_i} = \{t_{l,k} | c_i \not\sqsubseteq c_l\}$. Then, we can assess the discriminative capacity of each word $w \in D_i$ with respect to $\overline{D_i}$. It is essential to note that conventional attribute selection techniques select attributes for c_i from the entire lexicon, $L = D_i \cup \overline{D_i}$. In our case, however, we aim at selecting words that are most characteristic for the concept, and therefore we limit the selection only to words that actually appear in the textual objects for that concept, that is, D_i.

Figure 1 shows the algorithm for building a feature generator. The algorithm uses a global structure $Text(c_i)$ that accumulates textual objects for concept c_i and all of its descendants (attributes for the category are then selected from the words occurring in this pool). We manipulate $Text(c_i)$ as an unordered bag of words. Attribute vectors for each category are stored in $Vector(c_i)$.

3.4 Contextual Feature Generation

Feature generation precedes text categorization, that is, before the induction algorithm is invoked to build the text categorizer, the documents are fed to the feature generator.

Traditionally, feature generation uses the basic features supplied with the training instances to construct more sophisticated features. In the case of text processing, however, important information about word ordering will be lost if the traditional approach is applied to the bag of words. Therefore, we argue that feature generation becomes much more powerful when it operates on the raw document text. But should the generator always analyze the whole document as a single unit, as do regular text classifiers?

Algorithm BUILDFEATUREGENERATOR
  # Compute attribute vectors for all concepts
  BUILDVECTORS(c_0)

  # Use an induction algorithm to train a feature generator FG
  # using the attribute vectors $Vector(c_i)$
  $FG \leftarrow InduceClassifier(\{Vector(c_i)\})$
  # For feature generation efficiency, build an inverted index
  $InvIndex : w \mapsto \{c_i\}$, s.t. $w \in Vector(c_i)$

Algorithm BUILDVECTORS(c_i)
  $Text(c_i) = \emptyset$

  # Traverse the hierarchy bottom-up, collecting the textual objects
  # of the descendants of each category
  For each $child \in Children(c_i)$ do
    BUILDVECTORS(*child*)
    $Text(c_i) \leftarrow Text(c_i) \cup Text(child)$

  # Now add the textual objects for the category itself
  # along with the optional description (if available)
  $Text(c_i) \leftarrow Text(c_i) \cup \{t_{i,1}, \ldots, t_{i,m}\} \cup Description(c_i)$

  # Build the attribute vector by performing attribute selection
  # among the words of $Text(c_i)$
  $Vector(c_i) \leftarrow AttributeSelection(Text(c_i))$
  # Assign values to the selected attributes
  $Vector(c_i) \leftarrow tfidf(Vector(c_i))$

Figure 1: Building a feature generator.

3.4.1 ANALYZING LOCAL CONTEXTS

We believe that considering the document as a single unit can often be misleading: its text might be too diverse to be readily mapped to the right set of concepts, while notions mentioned only briefly may be overlooked. Instead, we propose to partition the document into a series of non-overlapping segments (called *contexts*), and then generate features at this finer level. Each context is classified into a number of concepts in the knowledge base, and pooling these concepts together to describe the entire document results in *multi-faceted* classification. This way, the resulting set of concepts represents the various aspects or sub-topics covered by the document.

Potential candidates for such contexts are simple sequences of words, or more linguistically motivated chunks such as sentences or paragraphs. The optimal resolution for document segmentation can be determined automatically using a validation set. We propose a more principled *multi-resolution* approach that simultaneously partitions the document at several levels of linguistic ab-

straction (windows of words, sentences, paragraphs, up to taking the entire document as one big chunk), and performs feature generation at each of these levels. We rely on the subsequent *feature selection* step (Section 3.4.2) to eliminate extraneous features, preserving only those that genuinely characterize the document. Figure 2 presents the feature generation algorithm.

Algorithm FEATUREGENERATION(D)
  Let CT be a series of contexts for D
  $CT \leftarrow words(D) \cup sentences(D) \cup paragraphs(D) \cup \{D\}$
  Let F be a set of features generated for D
  $F \leftarrow \emptyset$
  For each context $ct \in CT$ perform feature generation:
    $F \leftarrow F \cup FG(ct)$
  Represent D as $BagOfWords(D) \cup F$

Figure 2: Performing feature generation for document D

In fact, the proposed approach tackles two important problems in natural language processing, namely, *synonymy* (the ability of natural languages to express many notions in more than one way), and *polysemy* (the property of natural language words to convey more than a single sense, while certain words may have as many as dozens of different, sometimes unrelated senses). When individual contexts are classified, *word sense disambiguation* is implicitly performed, thus resolving word polysemy to some degree. A context that contains one or more polysemous words is mapped to the concepts that correspond to the sense *shared* by the context words. Thus, the correct sense of each word is determined with the help of its neighbors. At the same time, enriching document representation with high-level concepts and their generalizations addresses the problem of synonymy, as the enhanced representation can easily recognize that two (or more) documents actually talk about related issues, albeit using different vocabularies.

For each context, the feature generator yields a list of concepts ordered by their score, which quantifies their appropriateness to the context. A number of top-scoring concepts are used to actually generate features. For each of these concepts we generate one feature that represents the concept itself, as well an additional group of features that represent ancestors of this concept in the hierarchy of the knowledge repository.

3.4.2 FEATURE SELECTION

Using support vector machines in conjunction with bag of words, Joachims (1998) found that SVMs are very robust even in the presence of numerous features. He further observed that the multitude of features are indeed useful for text categorization. These findings were corroborated in more recent studies (Rogati and Yang, 2002; Brank et al., 2002; Bekkerman, 2003) that observed either no improvement or even small degradation of SVM performance after feature selection.[1] Consequently, many later works using SVMs did not apply feature selection at all (Leopold and Kindermann, 2002; Lewis et al., 2004).

1. Gabrilovich and Markovitch (2004) described a class of problems where feature selection from the bag of words actually improves SVM performance.

This situation changes drastically as we augment the bag of words with generated features. First, nearly any technique for automatic feature generation can easily generate huge numbers of features, which will likely aggravate the "curse of dimensionality." Furthermore, it is feature selection that allows the feature generator to be less than a perfect classifier. When some of the concepts assigned to the document are correct, feature selection can identify them and seamlessly eliminate the spurious ones. We further analyze the utility of feature selection in Section 5.7.

Note also that the categories to which the documents are categorized most likely correspond to a mix of knowledge repository concepts rather than a single one. Therefore, as the feature generator maps documents to a large set of related concepts, it is up to feature selection to retain only those that are relevant to the particular categorization task in hand.

3.4.3 FEATURE VALUATION

In regular text categorization, each word occurrence in document text is initially counted as a unit, and then feature valuation is performed, usually by subjecting these counts to TF.IDF weighting (Salton and Buckley, 1988; Debole and Sebastiani, 2003). To augment the bag of words with generated features and to use a single unified feature set, we need to assign weights to generated features in a compatible manner.

Each generated feature is assigned the basic weight of 1, as in the single occurrence of a word in the bag of words. However, this weight is further multiplied by the classification score produced for each classified concept by the feature generator. This score quantifies the degree of affinity between the concept and the context it was assigned to.

3.4.4 REVISITING THE RUNNING EXAMPLE

Let us revisit the example from Section 1, where we considered a document that belongs to the "copper" category of Reuters-21578. Figure 3 illustrates the process of feature generation for this example. While building the feature generator at the preprocessing stage, our system crawls the Web sites cataloged under mining-related ODP concepts such as BUSINESS/MINING_AND_DRILLING, SCIENCE/TECHNOLOGY/MINING and BUSINESS/INDUSTRIAL_GOODS_AND_SERVICES/MATERIALS/METALS. These include `http://www.teckcominco.com` and `http://www.miningsurplus.com`, which belong to the (now merged) Teck Cominco company. The company's prominence gives it frequent mention in the Web sites we have crawled, and consequently the words "Teck" and "Cominco" are included in the set of attributes selected to represent the above concepts.

During feature generation, the document is segmented into a sequence of contexts The feature generator analyzes these contexts and uses their words (e.g., "Teck" and "Cominco") to map the document to a number of mining-related concepts in the ODP (e.g., BUSINESS/MINING_AND_DRILLING). These concepts, as well as their ancestors in the hierarchy, give rise to a set of generated features that augment the bag of words. Observe that the training documents for the category "copper" underwent similar processing when a text classifier was induced. Consequently, features based on these concepts *were selected* during feature selection and retained in document vectors, thanks to their high predictive capacity. It is due to these features that the document is now categorized correctly, while without feature generation it consistently caused BOW classifiers to err.

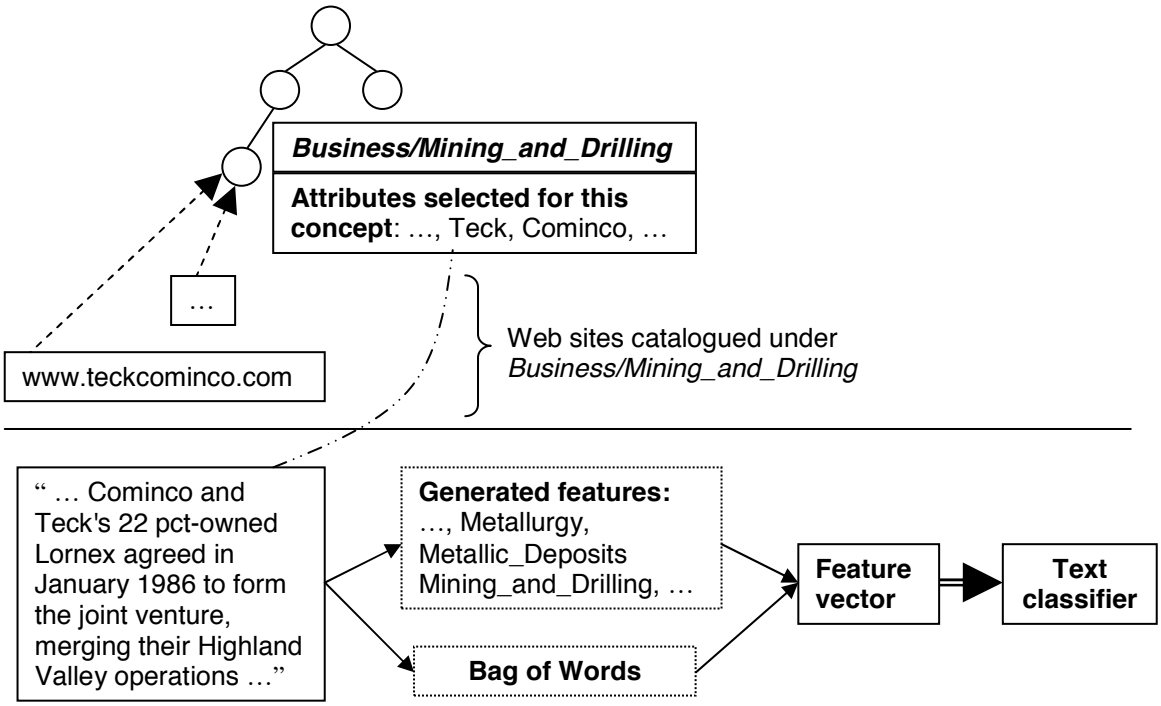


Figure 3: Feature generation example

4. Using the Open Directory for Feature Generation

We now instantiate the general methodology presented in Section 3 to use the Open Directory project as a knowledge repository.

The Open Directory comprises a hierarchy of approximately 600,000 nodes that catalog over 4,000,000 Web sites, each represented by a URL, a title, and a brief summary of its contents. The directory is organized as a tree where each node has a title (defined by its location within the directory, for example, COMPUTERS/ARTIFICIAL_INTELLIGENCE), and about one-third of all nodes have a short textual description. Every ODP node is associated with a collection of URLs to Web sites cataloged under that node, while each URL has a title and a concise summary of the corresponding Web site. The project constitutes an ongoing effort promoted by over 65,000 volunteer editors around the globe, and is arguably the largest publicly available Web directory.[2] Being the result of *pro bono* work, the Open Directory has its share of drawbacks, such as non-uniform coverage, duplicate subtrees in different branches of the hierarchy, and sometimes biased coverage due to peculiar views of the editors in charge. At the same time, however, ODP embeds a colossal amount of human knowledge in a wide variety of areas, covering even very specific scientific and technical concepts.

2. Although the actual size of Yahoo! has not been publicly released in the recent years, it is estimated to be about half the size of the Open Directory. This estimate is based on brute-force exhaustive crawling of the Yahoo! hierarchy. See `http://sewatch.com/reports/directories.html` and `http://www.geniac.net/odp` for more details.

4.1 Multiplying Knowledge Through Web Crawling

We use the textual descriptions of ODP nodes and their URLs as training examples for learning the feature generator. Although these descriptions alone constitute a sizeable amount of information, we devised a way to increase the volume of training data by several orders of magnitude. We do so by crawling the Web sites pointed at by all cataloged URLs, and obtain a small representative sample of each site. Following the scheme introduced by Yang et al. (2002), each link cataloged in the ODP is used to obtain a small representative sample of the target Web site. To this end, we crawl each cataloged site in the BFS order, starting from the URL listed in the directory. A predefined number of Web pages are downloaded, and then concatenated into a synthetic *meta-document*. This meta-document, along with the site description listed in the directory, constitutes the textual object for that site. Pooling together the meta-documents for all sites associated with an ODP node gives us a wealth of additional information about it, which we use to enrich the node summary.

4.2 Noise Reduction and Attribute Selection

Using so much knowledge requires a host of filtering mechanisms that control the quality and utility of the generated features. We now describe these mechanisms in detail. In what follows, we distinguish between *structural noise*, which is inherent to the ODP structure, and *content noise*, which is found in the texts we obtain through crawling the cataloged URLs.

4.2.1 STRUCTURAL NOISE

However elaborate our knowledge repositories are, they necessarily contain concepts that are detrimental to feature generation. These include concepts too specific or situated too deep in the hierarchy, or having too few textual objects to build a representative attribute vector. It is important to observe, however, that whenever we pruned small categories, we assigned all their textual content to their parents. Here again we benefit from the hierarchical organization of the directory, which allows us to aggregate small fragments of specific knowledge at a higher conceptual level, where its accumulated mass becomes sufficient to define a more general concept.

We identified the following potential sources of noise in the Open Directory:

1. The branch TOP/WORLD concentrates material in languages other than English. This entire branch is therefore pruned.

2. Some top-level branches contain concepts that are hardly useful for subsequent text categorization.

   (a) TOP/NEWS is a very elaborate subtree devoted to listing numerous CNN stories on various topics organized by date. The nodes of this subtree represent past dates, and do not correspond to useful knowledge concepts.

   (b) TOP/ADULT lists adult-oriented Web sites, and we believe that the concepts of this subtree are of little use for general purpose text categorization.

   (c) TOP/KIDS_AND_TEENS roughly duplicates the structure of the ODP but only lists resources suitable for children.

   All these branches are pruned as well.

3. Overly small categories (usually situated very deep in the hierarchy) that only contain a handful of URLs, and therefore their scope cannot be learned reliably. We therefore eliminate categories with fewer than 10 URLs or those situated below depth level 7 (the textual content of pruned categories is assigned to their parents).

4. The TOP/REGIONAL branch contains approximately one third of the entire mass of the ODP data, and is devoted to listing English language sites about various geographical regions of the world. This branch is further divided into continents, countries and smaller localities, up to the level of cities, towns and landmarks. However, the hierarchy does not stop at this level, and for most localities it provides much more elaborate classification, similar to that of the higher ODP levels. For example, under the path TOP/REGIONAL/NORTH_AMERICA/UNITED_STATES/NEW_YORK/LOCALITIES/N/NEW_YORK_CITY one finds further subdivisions such as ARTS_AND_ENTERTAINMENT, BUSINESS_AND_ECONOMY, HEALTH, SHOPPING and SOCIETY_AND_CULTURE. A similar set of categories duplicating higher-level distinctions (TOP/ARTS, TOP/BUSINESS etc.) can be also found in the middle of this path at TOP/REGIONAL/NORTH_AMERICA/UNITED_STATES/NEW_YORK.

   ODP classification principles[3] prescribe that businesses that operate in a particular locality (in this example, local to the State of New York or to New York City) should normally be catalogued under the most specific applicable categories, while businesses with global reach should be catalogued somewhere under TOP/BUSINESS; the rationale for choosing other categories (e.g., TOP/SOCIETY/... vs. TOP/REGIONAL/NORTH_AMERICA/UNITED_STATES/NEW_YORK/SOCIETY_AND_CULTURE is similar. However, we believe that when the ODP is used as a knowledge repository to support text categorization, such fine-grained distinctions (e.g., architect offices in Manhattan) are of little use. These categories only pollute the hierarchy with numerous small nodes, each of which only has a small chance of being assigned to any given context.

   Therefore, we eliminate overly specific categories under TOP/REGIONAL by pruning all paths at the level of geographical names. When the feature generator operates on a context describing a particular New York business, it will map the latter to the New York City node, as well as to one or more appropriate nodes under TOP/BUSINESS.

5. Web spam, which comes in the form of URLs that are hardly authoritative or representative of their host category, but are nonetheless included in the directory by a minority of unscrupulous editors. We do not explicitly address the problem of spam here, as it lies beyond the scope of our current study.

4.2.2 CONTENT NOISE

Texts harvested from the WWW are quite different from clean passages in formal written English, and without adequate noise reduction crawled data may do more harm than good. To reduce content noise we perform attribute selection as explained in Section 3.3.1. For example, Table 1 shows the top 10 attributes selected for sample ODP concepts using information gain as the attribute selection criterion. As we can see, the attributes selected for all the sample concepts are very intuitive and plausible.

3. See `http://dmoz.org/guidelines` and `http://dmoz.org/erz/index.html` for general ODP editorial guidelines, and `http://dmoz.org/Regional/faq.html` for Regional-specific issues.

ODP concept	Top 10 selected attributes
Top/Business/Financial_Services	finance, loan, mortgage, equity, insurance, lender, bank, investment, transaction, payment
Top/Computers/Artificial_Intelligence	neural, artificial, algorithm, intelligence, AAAI, Bayesian, probability, IEEE, cognitive, inference
Top/Health/Nutrition	nutrition, diet, nutrient, vitamin, dietary, cholesterol, carbohydrate, intake, protein, fat
Top/Home/Cooking	recipe, sauce, ingredient, soup, salad, casserole, stew, bake, butter, cook
Top/Recreation/Travel	travel, itinerary, trip, destination, cruise, hotel, tour, adventure, travelogue, departure
Top/Regional/Europe/Switzerland[4]	Switzerland, Swiss, Schweiz, und, Suiss, sie, CHF, der, Zurich, Geneva
Top/Science	science, research, scientific, biology, laboratory, analysis, university, theory, study, scientist
Top/Shopping/Gifts	gift, birthday, occasion, basket, card, shipping, baby, keepsake, order, wedding
Top/Society/History	war, history, military, army, civil, historian, soldier, troop, politics, century
Top/Sports/Golf	golf, golfer, tee, hole, fairway, tournament, championship, clubhouse, PGA, par

Table 1: Examples of attribute selection using information gain

4.2.3 LEARNING THE FEATURE GENERATOR

In our current implementation, the feature generator works as a centroid-based classifier (Han and Karypis, 2000), which represents each category as a centroid vector of the pool of textual objects associated with it.[5] Given a fragment of text supplied as input for feature generation, the classifier represents it as an attribute vector in the same space. It then compares this vector to those of all the concepts, and returns the desired number of best-matching ones. Attribute vectors are compared using the cosine metric (Zobel and Moffat, 1998); the value of the metric is treated as the classification score. A number of top-scoring concepts are retained for each input text as generated features. The feature generator also performs *generalization* of these concepts, and constructs features from the classified concepts *per se* as well as their ancestors in the hierarchy.

5. Empirical Evaluation

To evaluate the utility of knowledge-based feature generation, we implemented the proposed methodology using the Open Directory as a source of world knowledge. Throughout the experiments we used an ODP snapshot as of April 2004. Crawling of URLs cataloged in the Open Directory was performed over the period of April–August 2004.

4. Many crawled Web pages under TOP/REGIONAL/EUROPE/SWITZERLAND contain non-English material, hence words like "Schweiz" (German for Switzerland) and "der" (German masculine definite article), which survived stop words removal that is only performed for English.

5. The centroid classifier offers a simple and efficient way to manage the multitude of concepts in the Open Directory; additional machine learning techniques for learning the feature generator are mentioned in Section 3.3.

5.1 Experimental Methodology

We used the following test collections to evaluate our methodology for feature generation:

1. **Reuters-21578** (Reuters, 1997) is historically the most often used data set in text categorization research. Following common practice, we used the ModApte split (9603 training, 3299 testing documents) and two category sets, 10 largest categories and 90 categories with at least one training and testing example.

2. **Reuters Corpus Volume I (RCV1)** (Lewis et al., 2004), with over 800,000 documents and three orthogonal category sets, presents a new challenge for text categorization. Since the original RCV1 data contains a number of errors, we used the corrected version RCV1-v2 (Lewis et al., 2004, Section 4). To speed up experimentation, we used a subset of the corpus with 17,808 training documents (dated August 20–27, 1996) and 5341 testing documents (dated August 28–31, 1996). Following the scheme introduced by Brank et al. (2002), we used 16 Topic and 16 Industry categories, which constitute a representative sample of the full groups of 103 and 354 categories, respectively. We also randomly sampled the Topic and Industry categories into 5 sets of 10 categories each. Table 8 (Appendix A) gives the full definition of the category sets we used.

3. **OHSUMED** (Hersh et al., 1994) is a subset of the MEDLINE database, which contains 348,566 references to documents published in medical journals over the period of 1987–1991. Each reference contains the publication title, and about two-thirds (233,445) also contain an abstract. Each document is labeled with several MeSH (MeSH, 2003) categories. There are over 14,000 distinct categories in the collection, with an average of 13 categories per document. Following Joachims (1998), we used a subset of documents from 1991 that have abstracts, taking the first 10,000 documents for training and the next 10,000 for testing. To limit the number of categories for the experiments, we randomly generated 5 sets of 10 categories each. Table 9 (Appendix A) gives the full definition of the category sets we used.

4. **20 Newsgroups (20NG)** (Lang, 1995) is a well-balanced data set of 20 categories containing 1000 Usenet postings each.

5. **Movie Reviews (Movies)** (Pang et al., 2002) defines a sentiment classification task, where reviews express either positive or negative opinion about the movies. The data set has 1400 documents in two categories (positive/negative).

We used support vector machines[6] as our learning algorithm to build text categorizers, since prior studies found SVMs to have the best performance for text categorization (Sebastiani, 2002; Dumais et al., 1998; Yang and Liu, 1999). Following established practice, we use the precision-recall break-even point (BEP) to measure text categorization performance. For the two Reuters data sets we report both micro- and macro-averaged BEP, since their categories differ in size significantly. Micro-averaged BEP operates at the document level and is primarily affected by categorization performance on larger categories. On the other hand, macro-averaged BEP averages results for individual categories, and thus small categories with few training examples have large impact on the overall performance. For both Reuters data sets we used a fixed data split, and consequently

6. We used the SVM^{light} implementation (Joachims, 1999a).

used macro sign test (S-test) (Yang and Liu, 1999) to assess the statistical significance of differences in classifier performance. For 20NG and Movies we performed 4-fold cross-validation, and used paired t-test to assess the significance.

5.2 Implementation Details

In this section we describe the implementation details and design choices of our system.

5.2.1 CONSTRUCTING THE FEATURE GENERATOR

All ODP data is publicly available in machine-readable RDF format at `http://rdf.dmoz.org`. We used the file `structure.rdf.u8`, which defines the hierarchical structure of the directory, as well as provides category names and descriptions, and the file `content.rdf.u8`, which associates each category with a list of URLs, each having a title and a concise summary of the corresponding Web site. After pruning the TOP/WORLD branch, which contains non-English material, and TOP/ADULT branch, which lists adult-oriented Web sites, we obtained a collection of over 400,000 concepts and 2,800,000 URLs, organized in a very elaborate hierarchy with maximum depth of 13 levels and median depth of 7. Further pruning of too small and deep categories, as well as pruning of the TOP/REGIONAL subtree at the level of geographical names as explained in Section 4.2, reduced the number of concepts to 63,000 (the number of URLs was not reduced, since the entire URL population from pruned nodes is moved to their parents).

Textual descriptions of the concepts and URLs amounted to 436 Mb of text (68 Mb in concept titles and descriptions, 368 Mb in URL titles and summaries). In order to increase available information for training the feature generator, we further populated the ODP hierarchy by crawling all of its URLs, and taking the first 10 pages (in the BFS order) encountered at each site to create a representative meta-document of that site. As an additional noise removal step, we discarded meta-documents containing fewer than 5 distinct terms. This operation yielded 425 Gb worth of HTML files. After eliminating all the markup and truncating overly long files at 50 Kb, we ended up with 70 Gb of additional textual data. Compared to the original 436 Mb of text supplied with the hierarchy, we obtained over a 150-fold increase in the amount of data.

Applying our methodology to a knowledge repository of this scale required an enormous engineering effort. After tokenization and removal of stop words, numbers and mixed alphanumeric strings (e.g., "Win2k" or "4Sale"), we obtained 20,800,000 distinct terms. Further elimination of rare words (occurring in less than 5 documents) and applying the Porter stemming algorithm (Porter, 1980) resulted in a more manageable number of 2,900,000 distinct terms that were used to represent ODP nodes as attribute vectors. Up to 1000 most informative attributes were selected for each ODP node using the Document Frequency criterion (other commonly used feature selection techniques, such as Information Gain, χ^2 and Odds Ratio (Yang and Pedersen, 1997; Rogati and Yang, 2002; Mladenic, 1998a), yielded slightly inferior results in text categorization).

In order to speed up consequent classification of document contexts, we also built an *inverted index* that, given a word, provides a list of concepts that have it in their attribute vector (i.e., the word has been *selected* for this concept).

When assigning weights to individual entries in attribute vectors, we took into consideration the location of original word occurrences. For example, words that occurred in URL titles were assigned higher weight than those in the descriptions. Words originating from the descriptions or meta-

documents corresponding to links prioritized[7] by the ODP editors were also assigned additional weight. We completely ignored node descriptions since these are only available for about 40% of the nodes, and even then the descriptions are rarely used to actually describe the corresponding concept; in many cases they just contain instructions to the editors or explain what kinds of sites should *not* be classified under the node.

The set of attribute vectors underwent TF.IDF weighting, and eventually served to build a centroid-based feature generator.

5.2.2 USING THE FEATURE GENERATOR

We used the *multi-resolution* approach to feature generation, classifying document contexts at the level of individual words, complete sentences, paragraphs, and finally the entire document.[8] For each context, features were generated from the 10 best-matching ODP concepts produced by the feature generator, as well as for all of their ancestors.

5.2.3 TEXT CATEGORIZATION

We conducted the experiments using a text categorization platform of our own design and development named $\mathcal{H}$OGWARTS[9] (Davidov et al., 2004). We opted to build a comprehensive new infrastructure for text categorization, as surprisingly few software tools are publicly available for researchers, while those that are available allow only limited control over their operation. $\mathcal{H}$OGWARTS facilitates full-cycle text categorization including text preprocessing, feature extraction, construction, selection and valuation, followed by actual classification with cross-validation of experiments. The system currently provides part-of-speech tagging (Brill, 1995), sentence boundary detection, stemming (Porter, 1980), WordNet (Fellbaum, 1998) lookup, a variety of feature selection algorithms, and TF.IDF feature weighting schemes. $\mathcal{H}$OGWARTS has over 150 configurable parameters that control its *modus operandi* in minute detail. $\mathcal{H}$OGWARTS interfaces with SVM, KNN and C4.5 text categorization algorithms, and computes all standard measures of categorization performance. $\mathcal{H}$OGWARTS was designed with a particular emphasis on processing efficiency, and portably implemented in the ANSI C++ programming language. The system has built-in loaders for Reuters-21578 (Reuters, 1997), RCV1 (Lewis et al., 2004), 20 Newsgroups (Lang, 1995), Movie Reviews (Pang et al., 2002), and OHSUMED (Hersh et al., 1994), while additional data sets can be easily integrated in a modular way.

In the preprocessing step, each document undergoes the following. Document text is first tokenized, and title words are replicated twice to emphasize their importance. Then, stop words, numbers and mixed alphanumeric strings are removed, and the remaining words are stemmed. The bag of words is next merged with the set of features generated for the document by analyzing its contexts as explained in Section 3.4, and rare features occurring in fewer than 3 documents are removed.

7. ODP editors can highlight especially prominent and important Web sites; sites marked as such appear at the top of category listings and are emphasized with an asterisk (in RDF data files, the corresponding links are marked up with a `<priority>` tag).
8. The 20NG data set is an exception, owing to its high level of intrinsic noise that renders identification of sentence boundaries extremely unreliable, and causes word-level feature generation to produce too many spurious classifications. Consequently, for this data set we restrict the multi-resolution approach to individual paragraphs and the entire document only.
9. *Hogwarts School of Witchcraft and Wizardry* is the educational institution attended by Harry Potter (Rowling, 1997).

Since earlier studies found that most BOW features are indeed useful for SVM text categorization (Section 3.4.2), we take the bag of words in its entirety (with the exception of rare features removed in the previous step). The generated features, however, undergo feature selection using the information gain criterion. Finally, feature valuation is performed using the "ltc" TF.IDF function (logarithmic term frequency and inverse document frequency, followed by cosine normalization) (Salton and Buckley, 1988; Debole and Sebastiani, 2003).

5.3 Qualitative Analysis of Feature Generation

We now study the process of feature generation on a number of actual examples.

5.3.1 FEATURE GENERATION PER SE

In this section we demonstrate ODP-based feature generation for a number of sample sentences taken from CNN and other Web sites. For each example, we discuss a number of highly relevant features found among the top ten generated ones. Online Appendix A (`http://www.cs.technion.ac.il/~gabr/jmlr2006-online-appendix.html`) gives all 10 classifications produced for each context (some of these classifications are less relevant, and are consequently removed during feature selection, as explained in Section 3.4.2 and illustrated in Section 5.3.3.

- **Text:** *"Rumsfeld appeared with Gen. Richard Myers, chairman of the Joint Chiefs of Staff."*

  Sample generated features:

  - SOCIETY/ISSUES/GOVERNMENT_OPERATIONS, SOCIETY/POLITICS—both Donald Rumsfeld and Richard Myers are senior government officers, hence the connection to government operations and politics. Their names have been selected for these ODP concepts, since they appear in many Web sites cataloged under them, such as the National Security Archive at the George Washington University (`http://www.gwu.edu/~nsarchiv`)and the John F. Kennedy School of Government at Harvard University (`http://www.ksg.harvard.edu`).
  - SOCIETY/ISSUES/WARFARE_AND_CONFLICT/SPECIFIC_CONFLICTS/IRAQ, SCIENCE/TECHNOLOGY/MILITARY_SCIENCE, SOCIETY/ISSUES/WARFARE_AND_CONFLICT/WEAPONS—again, both persons mentioned were prominent during the Iraq campaign.
  - SOCIETY/HISTORY/BY_REGION/NORTH_AMERICA/UNITED_STATES/PRESIDENTS/BUSH,_GEORGE_WALKER—Donald Rumsfeld serves as Secretary of Defense under President George W. Bush
  - SOCIETY/POLITICS/CONSERVATISM—Rumsfeld is often seen as holding conservative views on a variety of political issues.

- **Text:** *"The new film follows Anakin's descent into evil and lust for power."*

  Sample generated features:

  - ARTS/MOVIES/TITLES/STAR_WARS_MOVIES is the root of the ODP subtree devoted to the "Star Wars" movie series. The word "Anakin" has been selected as an attribute for this concept due to its numerous occurrences in the cataloged Web sites such as `http://www.theforce.net` and `http://www.starwars.com`.

  - ARTS/PERFORMING_ARTS/ACTING/ACTORS_AND_ACTRESSES/CHRISTENSEN,_HAYDEN is the actor who played Anakin Skywalker; this particular piece of information cannot be inferred from the short input sentence without elaborate background knowledge.

- **Text:** *"On a night when Dirk Nowitzki (34 points), Jerry Stackhouse (29), Josh Howard (19) and Jason Terry (17) all came up big, he couldn't match their offensive contributions."*

  Sample generated features:

  - SPORTS/BASKETBALL/PROFESSIONAL/NBA/DALLAS_MAVERICKS — even though the sentence mentions neither the particular sport nor the name of the team, the power of context is at its best, immediately yielding the correct classification as the best-scoring generated feature. The names of the players mentioned in the context occur often in the Web sites cataloged under this concept, including such resources as `www.nba.com/mavericks`, `http://dallasbasketball.com` and `sports.yahoo.com/nba/teams/dal`.

- **Text:** *"Herceptin is a so-called targeted therapy because of its ability to attack diseased cells and leave healthy ones alone."*

  Sample generated features:

  - HEALTH/CONDITIONS_AND_DISEASES/CANCER/BREAST, SOCIETY/ISSUES/HEALTH/CONDITIONS_AND_DISEASES/CANCER/ALTERNATIVE_TREATMENTS, HEALTH/SUPPORT_GROUPS/CONDITIONS_AND_DISEASES/CANCER provide relevant additional information for Herceptin, a medication for breast cancer. The name of this medicine has been selected for these concepts due to its occurrences in cataloged Web sites such as `www.breastcancer.org`, `www.hopkinsmedicine.org/ breastcenter` and `cancer.gov/cancerinfo/wyntk/breast`.

- Finally, we give an example of how the power of context can be used for word sense disambiguation. The following pair of sentences use the word "tie" in two different meanings — once as a necktie and once as a kind of connection. Even though these sentences contain no distinguishing proper names, the context of the polysemous words allows the feature generator to produce correct suggestions in both cases

  Text: *"Kinship with others is based either on blood ties or on marital ties."*

  Sample generated features:

  - SOCIETY/GENEALOGY
  - HOME/FAMILY
  - SOCIETY/RELATIONSHIPS
  - SCIENCE/SOCIAL_SCIENCES/SOCIOLOGY

  Text: *"Our tie shop includes plain solid colour ties, novelty ties, patterned silk ties, and men's bow ties."*

  Sample generated features:

  - SHOPPING/CLOTHING/MENS/NECKTIES

– SHOPPING/CLOTHING/ACCESSORIES/MENS

– BUSINESS/CONSUMER_GOODS_AND_SERVICES/CLOTHING/ACCESSORIES/TIES_AND_SCARVES

Evidently, many of the generated features could not have been accessed by conventional text classification methods, since heavy use of world knowledge is required to deduce them.

5.3.2 ACTUAL TEXT CATEGORIZATION EXAMPLES UNDER A MAGNIFYING GLASS

Thanks to feature generation, our system correctly classifies the running example document #15264. Let us consider additional testing examples from Reuters-21578 that are incorrectly categorized by the BOW classifier. Document #16143 belongs to the category "money-fx" (money/foreign exchange) and discusses the devaluation of the Kenyan shilling. Even though "money-fx" is one of the 10 largest categories, the word "shilling" does not occur in its training documents even once. However, the feature generator easily recognizes it as a kind of currency, and produces features such as RECREATION/COLLECTING/PAPER_MONEY and RECREATION/COLLECTING/COINS/WORLD_COINS. While analyzing document contexts it also uses other words such as "Central Bank of Kenya" and "devaluation" to correctly map the document to ODP concepts SOCIETY/GOVERNMENT/FINANCE, SCIENCE/SOCIAL_SCIENCES/ECONOMICS and BUSINESS/FINANCIAL_SERVICES/BANKING_SERVICES. Even though the behavior of the Kenyan shilling was never mentioned in the training set, these high-level features were also constructed for many training examples, and consequently the document is now classified correctly.

Similarly, document #18748 discusses Italy's balance of payments and belongs to the category "trade" (interpreted as an economic indicator), while the word "trade" itself does not occur in this short document. However, when the feature generator considers document contexts discussing Italian deficit as reported by the Bank of Italy, it correctly maps them to concepts such as SOCIETY/GOVERNMENT/FINANCE, SOCIETY/ISSUES/ECONOMIC/INTERNATIONAL/TRADE, BUSINESS/INTERNATIONAL_BUSINESS_AND_TRADE. These features, which were also generated for training documents in this category (notably, document #271 on Japanese trade surplus, document #312 on South Korea's account surplus, document #354 on tariff cuts in Taiwan and document #718 on U.S.-Canada trade pact), allow the document to be categorized correctly.

Let us also consider a few documents from the Movie Reviews data set that confuse the BOW classifier (here we consider a training/testing split induced by one particular cross-validation fold). Recall that this data set represents a sentiment classification task, where documents are classified according to the sentiment of the review (positive or negative) rather than its topic. Document #19488 contains a negative review of Star Wars Episode 1, but at the word level it is difficult to judge its true sentiment since positive and negative words are interspersed. For instance, the sentence "Anakin is annoying and unlikeable, instead of cute and huggable as Lucas no doubt intended" contains two words with positive connotation ("cute and huggable") that counterbalance the two words with negative ones ("annoying and unlikeable"). However, given contexts like "The two leads are hideously boring, static characters given little to do and too much time to do it," the feature generator produces features such as ARTS/MOVIES/REVIEWS/TOP_LISTS/BAD_FILMS. This ODP node catalogs Web sites devoted to reviews of bad movies, and the wording of this sample context looks similar to that used in known negative reviews (as cataloged in the ODP). In fact, this particular feature is one of the most informative ones generated for this data set, and it is also produced for contexts like "Next up

#	ODP concept
1	BUSINESS/MINING_AND_DRILLING/MINERAL_EXPLORATION_AND_EXTRACTION
2	BUSINESS/MINING_AND_DRILLING
3	BUSINESS/MINING_AND_DRILLING/MINERAL_EXPLORATION_AND_EXTRACTION/ BASE_METALS
4	SCIENCE/TECHNOLOGY/MINING
5	BUSINESS/MINING_AND_DRILLING/CONSULTING
6	BUSINESS/INVESTING/COMMODITIES,_FUTURES/PRECIOUS_METALS
7	SHOPPING
8	BUSINESS/MINING_AND_DRILLING/MINING_EQUIPMENT
9	BUSINESS/INVESTING/COMMODITIES,_FUTURES/PRECIOUS_METALS/GOLD
10	SCIENCE/TECHNOLOGY/MINING/INVESTMENTS

Table 2: The top ten ODP concepts generated for the sentence "Cominco's share of production was 43,000 short tons of copper, 340,000 ounces of silver and 800 ounces of gold."

we have the dialogue, which is amusingly bad at its best, painful at its worst" and "What ensues is a badly scripted and horribly directed 114 minutes of cinema hell," both found in negative reviews.

As another example, consider document #15111, which contains a positive review of the movie "Soldier." This review, which constantly switches between criticizing and praising the film, easily perplexes the BOW classifier. Interestingly, given the sentence "It is written by David Webb Peoples, who penned the screenplay to the classic Blade Runner and the critically-acclaimed 12 Monkeys," the feature generator constructs the highly informative feature ARTS/MOVIES/REVIEWS/TOP_LISTS/GOOD_FILMS. This is made possible by the references to known good films ("Blade Runner" and "12 Monkeys") that are listed in Web sites devoted to good films (`http://www.filmsite.org` and `http://us.imdb.com/top_250_films`, for example). The same feature was also generated for a number of training documents, and thus helps the classifier to categorize the document correctly.

5.3.3 THE IMPORTANCE OF FEATURE SELECTION

To understand the utility of feature selection, consider a sample sentence from our running example, Reuters document #15264: "Cominco's share of production was 43,000 short tons of copper, 340,000 ounces of silver and 800 ounces of gold." Table 2 gives the top ten ODP concepts generated as features for this context. Most of the assigned concepts deal with mining and drilling, and will eventually be useful features for document classification. However, the concepts BUSINESS/INVESTING/ COMMODITIES,_FUTURES/PRECIOUS_METALS, SHOPPING and BUSINESS/INVESTING/COMMODITIES,_FUTURES/ PRECIOUS_METALS/GOLD have been triggered by the words "gold" and "silver," which are mentioned incidentally and do not describe the gist of the document. Feature selection is therefore needed to eliminate features based on these extraneous concepts.

As another example, consider the following sentence taken from the same document: "'Cominco, 29.5 percent owned by a consortium led by Teck, is optimistic that the talks will soon be concluded,' spokesman Don Townson told Reuters," along with its top ten classifications given in

#	ODP concept
1	BUSINESS/MINING_AND_DRILLING/MINERAL_EXPLORATION_AND_EXTRACTION/ BASE_METALS
2	BUSINESS/MINING_AND_DRILLING/MINERAL_EXPLORATION_AND_EXTRACTION
3	BUSINESS/MINING_AND_DRILLING
4	BUSINESS/MINING_AND_DRILLING/CONSULTING
5	SOCIETY/ISSUES
6	REGIONAL/NORTH_AMERICA/CANADA/BRITISH_COLUMBIA/LOCALITIES/KIMBERLEY
7	SCIENCE/TECHNOLOGY/MINING
8	BUSINESS/MARKETING_AND_ADVERTISING/CONSULTING/SALES
9	REGIONAL/NORTH_AMERICA/CANADA/QUEBEC/REGIONS/NORTHERN_QUEBEC
10	SCIENCE/ENVIRONMENT/MINING

Table 3: The top ten ODP concepts generated for the sentence "'Cominco, 29.5 percent owned by a consortium led by Teck, is optimistic that the talks will soon be concluded,' spokesman Don Townson told Reuters."

Table 3. Here, the concept SOCIETY/ISSUES is generated by the word "Reuters." In turn, the concept BUSINESS/MARKETING_AND_ADVERTISING/CONSULTING/SALES is triggered by the name of the company spokesman, Don Townson. As it happens, a sales consulting company named "Townson & Alexander Consulting Services" is catalogued under this concept. Based on the crawled content of this site, the word "Townson" and other sales-related words in the context (e.g., "percent," "owned," "optimistic," and "consortium") taken together yield this concept in the results. Again, this sales-related concept is hardly useful for categorizing copper-related documents, and features based on it would therefore not be selected.

5.4 The Effect of Feature Generation

We first demonstrate that the performance of basic text categorization in our implementation (column "Baseline" in Table 4) is consistent with the state of the art as reflected in other published studies (all using SVM). On Reuters-21578, Dumais et al. (1998) achieved micro-BEP of 0.920 for 10 categories and 0.870 for all categories. On 20NG, Bekkerman (2003) obtained BEP of 0.856. Pang et al. (2002) obtained accuracy of 0.829 on Movies. The minor variations in performance are due to differences in data preprocessing in the different systems; for example, for the Movies data set we worked with raw HTML files rather than with the official tokenized version, in order to recover sentence and paragraph structure for contextual analysis. For RCV1 and OHSUMED, direct comparison with published results is more difficult because we limited the category sets and the date span of documents to speed up experimentation.

Table 4 shows the results of using feature generation for text categorization, with significant improvements ($p < 0.05$) shown in bold. For both Reuters data sets, we consistently observed larger improvements in macro-averaged BEP, which is dominated by categorization effectiveness on small categories. This goes in line with our expectations that the contribution of external knowledge should be especially prominent for categories with few training examples. As can be readily seen,

Data set	Baseline		Feature generation		Improvement vs. baseline	
	micro BEP	macro BEP	micro BEP	macro BEP	micro BEP	macro BEP
Reuters-21578						
10 categories	0.925	0.874	0.930	0.884	+0.5%	+1.1%
90 categories	0.877	0.602	0.880	0.614	+0.3%	+2.0%
RCV1						
Industry-16	0.642	0.595	0.648	**0.613**	+0.9%	**+3.0%**
Industry-10A	0.421	0.335	**0.457**	**0.420**	**+8.6%**	**+25.4%**
Industry-10B	0.489	0.528	**0.530**	**0.560**	**+8.4%**	**+6.1%**
Industry-10C	0.443	0.414	**0.468**	**0.463**	**+5.6%**	**+11.8%**
Industry-10D	0.587	0.466	0.588	**0.496**	+0.2%	**+6.4%**
Industry-10E	0.648	0.605	0.657	**0.639**	+1.4%	**+5.6%**
Topic-16	0.836	0.591	0.840	**0.660**	+0.5%	**+11.7%**
Topic-10A	0.796	0.587	0.803	**0.692**	+0.9%	**+17.9%**
Topic-10B	0.716	0.618	0.727	**0.655**	+1.5%	**+6.0%**
Topic-10C	0.687	0.604	0.694	**0.618**	+1.0%	**+2.3%**
Topic-10D	0.829	0.673	0.836	0.687	+0.8%	+2.1%
Topic-10E	0.758	0.742	0.762	0.756	+0.5%	+1.9%
OHSUMED						
OHSUMED-10A	0.518	0.417	**0.537**	**0.479**	**+3.7%**	**+14.9%**
OHSUMED-10B	0.656	0.500	0.659	**0.548**	+0.5%	**+9.6%**
OHSUMED-10C	0.539	0.505	0.547	**0.540**	+1.5%	**+6.9%**
OHSUMED-10D	0.683	0.515	0.688	**0.549**	+0.7%	**+6.6%**
OHSUMED-10E	0.442	0.542	0.452	**0.573**	+2.3%	**+5.7%**
20NG	0.854		**0.858**		**+0.5%**	
Movies	0.813		**0.842**		**+3.6%**	

Table 4: Text categorization with and without feature generation

categorization performance was improved for all data sets, with notably high improvements for Reuters RCV1, OHSUMED and Movies. We believe these results clearly demonstrate the advantage of knowledge-based feature generation.

5.5 The Effect of Contextual Analysis

We now explore the various possibilities for defining document contexts for feature generation, that is, chunks of document text that are classified onto the ODP to construct features. Figure 4 shows how text categorization performance on the Movies data set changes for various contexts. The x-axis measures context length in words, and the *FG/words* curve corresponds to applying the feature generator to the context of that size. With these word-level contexts, maximum performance is achieved when using pairs of words (x=2). The *Baseline* line represents text categorization without feature generation. The *FG/doc* line shows what happens when the entire document is used as a single context. In this case, the results are somewhat better than without feature generation (*Baseline*), but are still inferior to the more fine-grained word-level contexts (*FG/words*). However, the best performance by far is achieved with the multi-resolution approach (*FG/multi*), in which we use a

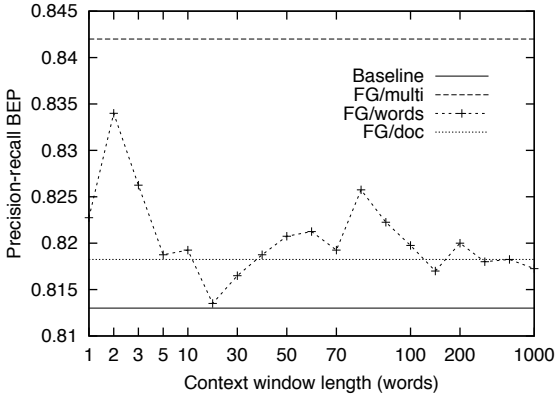


Figure 4: Varying context length (Movies)

series of linguistically motivated chunks of text, starting with individual words, and then generating features from sentences, paragraphs, and finally the entire document.

5.6 The Effect of Knowledge Breadth

In the experiments reported in Section 5.4 we performed feature generation using the entire ODP. It is interesting to observe, however, that four out of the five data sets we used have a fairly narrow scope.[10] Specifically, both Reuters data sets (Reuters-21578 and RCV1) contain predominantly economic news and therefore match the scope of the TOP/BUSINESS branch of the ODP. Similarly, Movie Reviews contains opinions about movies, and therefore fits the scope of TOP/ARTS. OHSUMED contains medical documents, which can be modelled within the scope of TOP/HEALTH and TOP/SCIENCE. In light of this, it could be expected that restricting the feature generator to a particular ODP branch that corresponds to the scope of the test collection would result in much better categorization accuracy due to the elimination of noise in "unused" ODP branches.

Experimental results (Table 5) disprove this hypothesis. As can be seen, in the absolute majority of cases the improvement over the baseline is much smaller than when the entire ODP is used (cf. Table 4). These findings show the superiority of wide general-purpose knowledge over its domain-specific subsets.

5.7 The Utility of Feature Selection

Under the experimental settings defined in Section 5.2, feature generation constructed approximately 4–5 times as many features as are in the bag of words (after rare features that occurred in less than 3 documents were removed). We conducted two experiments to understand the effect of feature selection in conjunction with feature generation.

Since earlier studies found that feature selection from the bag of words impairs SVM performance (Section 3.4.2), we first apply it only to the generated features and use the selected ones to augment the (entire) bag of words. In Figures 5 and 6, the *BOW* line depicts the baseline perfor-

10. The 20 Newsgroups data set consists of 20 diverse categories, each of which corresponds to one or more ODP branches.

Data set	Domain-specific ODP subset			Full ODP	
	Subset description	micro BEP	macro BEP	micro BEP	macro BEP
Reuters-21578	TOP/BUSINESS				
10 categories		+0.4%	+0.6%	+0.5%	+1.1%
90 categories		+0.1%	+1.2%	+0.3%	+2.0%
RCV1	TOP/BUSINESS				
Industry-16		+1.9%	+2.2%	+0.9%	**+3.0%**
Topic-16		+0.5%	+1.4%	+0.5%	**+11.7%**
OHSUMED	TOP/HEALTH				
OHSUMED-10A		+2.1%	+1.7%	**+3.7%**	**+14.9%**
OHSUMED-10B		+0.2%	+1.2%	+0.5%	**+9.6%**
OHSUMED-10C		+1.7%	+2.8%	+1.5%	**+6.9%**
OHSUMED-10D		+0.3%	+1.9%	+0.7%	**+6.6%**
OHSUMED-10E		+2.7%	+1.8%	+2.3%	**+5.7%**
OHSUMED	TOP/HEALTH + TOP/SCIENCE				
OHSUMED-10A		+5.4%	+3.6%	**+3.7%**	**+14.9%**
OHSUMED-10B		+0.3%	+3.4%	+0.5%	**+9.6%**
OHSUMED-10C		+0.6%	+3.8%	+1.5%	**+6.9%**
OHSUMED-10D		+0.9%	+5.8%	+0.7%	**+6.6%**
OHSUMED-10E		+1.6%	+1.8%	+2.3%	**+5.7%**
Movies	TOP/ARTS	**+2.6%**		**+3.6%**	

Table 5: Text categorization with and without feature generation, when only a subset of ODP is used

mance without generated features, while the *BOW+GEN* curve shows the performance of the bag of words augmented with progressively larger fractions of generated features (sorted by information gain). For both data sets, the performance peaks when only a small fraction of the generated features are used, while retaining more generated features has a noticeable detrimental effect.

Our second experiment examined the performance of the generated features alone, without the bag of words (*GEN* curve in Figures 5 and 6). For Movies, discarding the BOW features leads to somewhat worse performance, but the decrease is far less significant than what could be expected—using only the generated features we lose less than 3% in BEP compared with the BOW baseline. For 20NG, a similar experiment sacrifices about 10% of the BOW performance, as this data set is known to have a very diversified vocabulary, for which many studies found feature selection to be particularly harmful. Similarly, for OHSUMED, using only the generated features sacrifices up to 15% in performance, reinforcing the value of precise medical terminology that is discarded in this experiment. However, the situation is reversed for both Reuters data sets. For Reuters-21578, the generated features alone yield a 0.3% improvement in micro- and macro-BEP for 10 categories, while for 90 categories they only lose 0.3% in micro-BEP and 3.5% in macro-BEP compared with the bag of words. For RCV1/Industry-16, disposing of the bag of words reduces BEP performance by 1–3%. Surprisingly, for RCV1/Topic-16 (Figure 6) the generated features *per se* command a 10.8% improvement in macro-BEP, rivaling the performance of *BOW+GEN*, which gains only an-

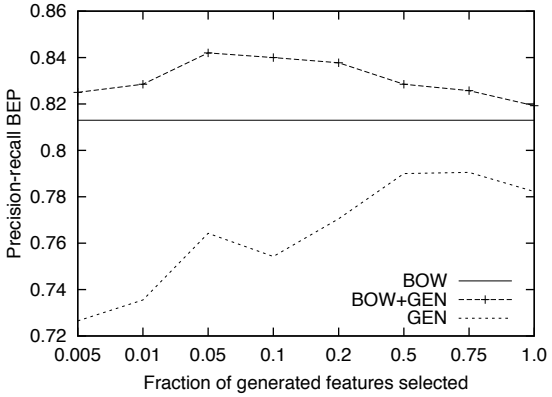


Figure 5: Feature selection (Movies)

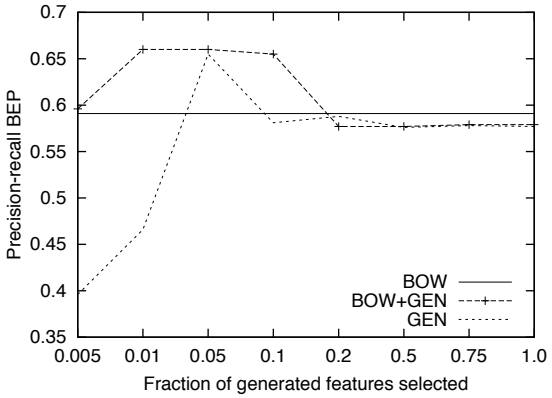


Figure 6: Feature selection (RCV1/Topic-16)

other 1% (Table 4). We interpret these findings as further reinforcement that the generated features improve the quality of the representation.

5.8 The Effect of Category Size

We saw in Section 5.4 that feature generation greatly improves text categorization for smaller categories, as can be evidenced in the greater improvements in macro-BEP. To explore this phenomenon further, we depict in Figures 7 and 8 the relation between the category size and the improvement due to feature generation for RCV1 (the number of categories in each bin appears in parentheses above the bars). To this end, we pooled together the categories that comprised the individual sets (10A–10E) in the Industry and Topic groups, respectively.

As we can readily see, smaller categories tend to benefit more from knowledge-based feature generation. These graphs also explain the more substantial improvements observed for Industry categories compared to Topic categories—as can be seen from the graphs, Topic categories are larger than Industry categories, and the average size of Topic categories (among those we used in this study) is almost 6 times larger than that of Industry categories.

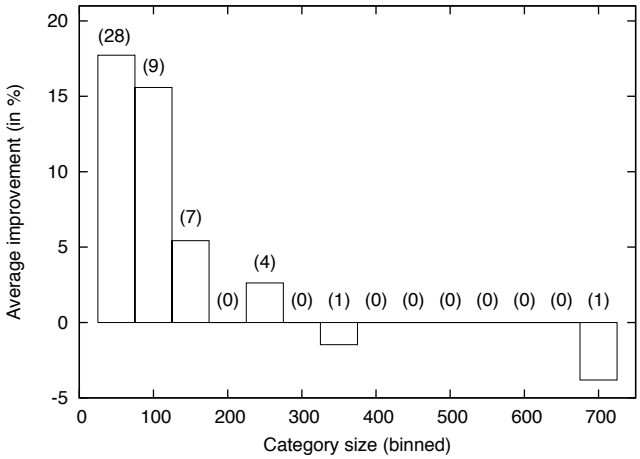


Figure 7: RCV1 (Industry): Average improvement versus category size

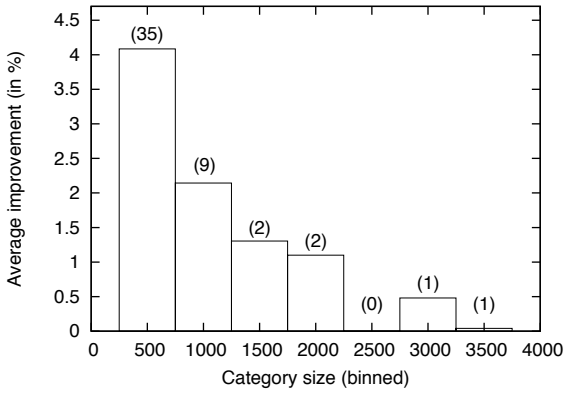


Figure 8: RCV1 (Topic): Average improvement versus category size

5.9 The Effect of Feature Generation for Classifying Short Documents

We conjectured that knowledge-based feature generation might be particularly useful for classifying short documents. To evaluate this hypothesis, we derived several data sets of short documents based on the test collections listed in Section 5.1. Recall that about one-third of the references in OHSUMED have titles but no abstract and can therefore be considered short documents "as-is." We used the same range of documents as in Section 5.1, but considered only those without abstracts. This yielded 4,714 training and 5,404 testing documents. For all other data sets, we created a short document from each original document by taking only the title of the latter (with the exception of Movie Reviews, where documents do not have titles). It should be noted, however, that substituting a title for the full document is a poor man's way to obtain a collection of classified short documents. When documents were first labeled with categories, the human labeller saw the document *in its entirety*. In particular, a category might have been assigned to a document on the basis of some facts mentioned in its body, even though the relevant facts may well be missing from the (short) title. Thus, taking all the categories of the original documents to be "genuine" categories of the title is often misleading. However, because we know of no publicly available test collections of short documents, we decided to use the data sets constructed as explained above. Interestingly, OHSUMED documents without abstracts have been classified as such by humans; working with the OHSUMED-derived data set can thus be considered a "pure" experiment.

Table 6 presents the results of this experiment. As we can see, in the majority of cases (except for RCV1 Topic category sets), feature generation leads to greater improvement on short documents than on regular documents. Notably, the improvements are particularly high for OHSUMED, where "pure" experimentation on short documents is possible (see above).

5.10 Processing Time

Using the ODP as a source of background knowledge requires additional computation. This extra computation includes the (one-time) preprocessing step where the feature generator is built, as well as the actual feature generation performed on documents prior to text categorization. The processing times reported below were measured on a workstation with dual Xeon 2.2 GHz CPU and 2 Gb RAM running the Microsoft Windows XP Professional operating system (Service Pack 1).

Parsing the ODP structure (file `structure.rdf.u8`) took 3 minutes. Parsing the list of ODP URLs (file `content.rdf.u8`) required 3 hours, and parsing the crawled ODP data (meta-documents collected from all cataloged URLs) required 2.6 days. Attribute selection for ODP concepts took 1.5 hours. The cumulative one-time expenditure for building the feature generator was therefore just under 3 days (not counting the actual Web crawling that was performed beforehand).

We benchmarked feature generation in two scenarios—individual words and 10-word windows. In the former case, the feature generator classified approximately 310 words per second, while in the latter case it classified approximately 45 10-word windows per second (i.e., 450 words per second).[11] These times constitute the additional overhead required by feature generation compared with regular text categorization. Table 7 lists the sizes of the test collections we experimented with (see Section 5.1). To speed up experimentation, we used subsets of the entire RCV1 and OHSUMED collections; these subsets comparable in size with 20 Newsgroups and Reuters-21578.

11. Classifying word windows is more efficient due to the sharing of data structures when processing the words in a single context.

DATA SET	SHORT DOCUMENTS						FULL DOCUMENTS	
	Baseline		Feature generation		Improvement vs. baseline		Improvement vs. baseline	
	micro BEP	macro BEP	micro BEP	macro BEP	micro BEP	macro BEP	micro BEP	macro BEP
Reuters-21578								
10 categories	0.868	0.774	0.868	0.777	+0.0%	+0.4%	+0.5%	+1.1%
90 categories	0.793	0.479	0.794	0.498	+0.1%	**+4.0%**	+0.3%	+2.0%
RCV1								
Industry-16	0.454	0.400	0.466	**0.415**	+2.6%	**+3.7%**	+0.9%	**+3.0%**
Industry-10A	0.249	0.199	**0.278**	**0.256**	**+11.6%**	**+28.6%**	**+8.6%**	**+25.4%**
Industry-10B	0.273	0.292	**0.348**	**0.331**	**+27.5%**	**+13.4%**	**+8.4%**	**+6.1%**
Industry-10C	0.209	0.199	**0.295**	**0.308**	**+41.1%**	**+54.8%**	**+5.6%**	**+11.8%**
Industry-10D	0.408	0.361	**0.430**	**0.431**	**+5.4%**	**+19.4%**	+0.2%	**+6.4%**
Industry-10E	0.450	0.410	**0.490**	**0.459**	**+8.9%**	**+12.2%**	+1.4%	**+5.6%**
Topic-16	0.763	0.529	0.763	0.534	+0.0%	+0.9%	+0.5%	**+11.7%**
Topic-10A	0.718	0.507	0.720	0.510	+0.3%	+0.6%	+0.9%	**+17.9%**
Topic-10B	0.647	0.560	0.644	0.560	-0.5%	+0.0%	+1.5%	**+6.0%**
Topic-10C	0.551	0.471	0.561	0.475	+1.8%	+0.8%	+1.0%	**+2.3%**
Topic-10D	0.729	0.535	0.730	**0.553**	+0.1%	**+3.4%**	+0.8%	+2.1%
Topic-10E	0.643	0.636	0.656	0.646	+2.0%	+1.6%	+0.5%	+1.9%
OHSUMED								
OHSUMED-10A	0.302	0.221	**0.357**	**0.253**	**+18.2%**	**+14.5%**	**+3.7%**	**+14.9%**
OHSUMED-10B	0.306	0.187	**0.348**	**0.243**	**+13.7%**	**+29.9%**	+0.5%	**+9.6%**
OHSUMED-10C	0.441	0.296	**0.494**	**0.362**	**+12.0%**	**+22.3%**	+1.5%	**+6.9%**
OHSUMED-10D	0.441	0.356	0.448	**0.419**	+1.6%	**+17.7%**	+0.7%	**+6.6%**
OHSUMED-10E	0.164	0.206	**0.211**	**0.269**	**+28.7%**	**+30.6%**	+2.3%	**+5.7%**
20NG	0.699		**0.740**		**+5.9%**		**+0.5%**	

Table 6: Text categorization of short documents with and without feature generation. (The improvement percentage in the two rightmost columns is computed relative to the baseline shown in Table 4.)

Data set	Number of documents	Number of words[12]
20NG	19,997	5.5 million
Movies	1,400	0.95 million
Reuters-21578	21,902	2.8 milion
RCV1		
- full	804,414	196 million
- used in this study	23,149	5.5 million
OHSUMED		
- full	348,566	57 million
- used in this study	20,000	3.7 million

Table 7: Test collections sizes

In the light of the improvements in categorization accuracy that we report in Section 5.4, we believe that the extra processing time is well compensated for. In operational text categorization systems, documents rarely arrive in huge batches of hundreds of thousands at a time. For example, the RCV1 data set contains all English-language news items published by Reuters over the period of one year. Therefore, in practical settings, once the classification model has been trained, the number of documents it needs to classify per time unit is much more reasonable, and can be easily facilitated by our system.

6. Related Work

To date, quite a few attempts have been made to deviate from the orthodox bag of words paradigm, usually with limited success. In particular, representations based on phrases (Lewis, 1992; Dumais et al., 1998; Fuernkranz et al., 1998), named entities (Kumaran and Allan, 2004), and term clustering (Lewis and Croft, 1990; Bekkerman, 2003) have been explored. However, none of these techniques could possibly overcome the problem underlying the various examples we reviewed in this paper—lack of world knowledge.

In mainstream information retrieval, query expansion techniques are used to augment queries with additional terms. However, this approach does not enhance queries with high-level concepts beyond words or phrases (as this would require indexing the entire document collection accordingly). It occasionally uses WordNet (Fellbaum, 1998) as a source of external knowledge, but queries are more often enriched with individual words, which are chosen either through relevance feedback (Mitra et al., 1998; Xu and Croft, 2000), or by consulting dictionaries and thesauri (Voorhees, 1994, 1998). Ballesteros and Croft (1997) studied query expansion with phrases in the context of cross-lingual information retrieval.

Feature generation techniques were found useful in a variety of machine learning tasks (Markovitch and Rosenstein, 2002; Fawcett, 1993; Matheus, 1991). These techniques search for new features that describe the target concept better than the ones supplied with the training instances. A number of proposed feature generation algorithms (Pagallo and Haussler, 1990; Matheus and Rendell, 1989; Hu and Kibler, 1996; Murphy and Pazzani, 1991) led to significant improvements in performance over a range of classification tasks. However, even though feature generation is an established research area in machine learning, only a few works have applied it to text pro-

12. Measured using the 'wc' utility available on UNIX systems.

cessing (Kudenko and Hirsh, 1998; Mikheev, 1999; Cohen, 2000). It is interesting to observe that traditional machine learning data sets, such as those available from the UCI data repository (Blake and Merz, 1998), are only available as feature vectors, while their feature set is essentially fixed. Textual data, however, is almost always available in raw text format. Thus, in principle, possibilities for feature generation are more plentiful and flexible.

Kudenko and Hirsh (1998) proposed a domain-independent feature generation algorithm that uses Boolean features to test whether certain sub-sequences appear a minimum number of times. They applied the algorithm to three toy problems in topic spotting and book passage categorization. Mikheev (1999) used a feature collocation lattice as a feature generation engine within a maximum entropy framework and applied it to document categorization, sentence boundary detection, and part-of-speech tagging. This work used information about individual words, bigrams and trigrams to prebuild the feature space. A set of feature cliques with the highest log-likelihood estimate was then selected. Cohen (2000) researched the problem of automatically discovering features useful for classification according to the given labels, given a set of labeled instances *not accompanied by a feature set*. Problems of this kind occur, for example, when classifying names of musical artists by music genres, or names of computer games by categories such as quest or action. The paper proposed to collect relevant Web pages, and then define features based on words from HTML headers that co-occur with the names to be classified. The fact that a word appears in an HTML header usually signifies its importance, and hence potential usefulness, for classification. The author also identified another source of features on the basis of their *positions* inside HTML documents, where position is defined as a sequence of tags in the HTML parsing tree, between the root of the tree and the name of interest. For example, if a name appears frequently in tables, this characteristic may be defined as a feature.

Several studies performed feature construction using WordNet and other domain-specific dictionaries (Scott, 1998; Urena-Lopez et al., 2001; Bloehdorn and Hotho, 2004; Wang et al., 2003). Scott (1998) completely replaced a bag of words with a bag of synsets[13]. Urena-Lopez et al. (2001) used WordNet in conjunction with Rocchio (Rocchio, 1971) and Widrow-Hoff (Lewis et al., 1996; Widrow and Stearns, 1985, Ch. 6) linear classifiers to fine-tune the category vectors. Wang et al. (2003) used Medical Subject Headings (MeSH) (MeSH, 2003) to replace the bag of words with canonical medical terms; Bloehdorn and Hotho (2004) used a similar approach to augment Reuters-21578 documents with WordNet synsets and OHSUMED medical documents with MeSH terms.

It should be noted, however, that WordNet was not originally designed to be a powerful knowledge base, but rather a lexical database more suitable for peculiar lexicographers' needs. Specifically, WordNet has the following drawbacks when used as a knowledge base for text categorization:

- WordNet has a fairly small coverage—for the test collections we used in this paper, up to 50% of their unique words are missing from WordNet. In particular, many proper names, slang and domain-specific technical terms are not included in WordNet, which was designed as a general-purpose dictionary.

- Additional information about synsets (beyond their identity) is very limited. This is because WordNet implements a *differential* rather than *constructive* lexical semantics theory, so that glosses that accompany the synsets are mainly designed to distinguish the synsets rather than provide a definition of the sense or concept. Usage examples that occasionally constitute part

13. A *synset* is WordNet notion for a sense shared by a group of synonymous words.

of the gloss serve the same purpose. Without such auxiliary information, reliable word sense disambiguation is almost impossible.

- WordNet was designed by professional linguists who are trained to recognize minute differences in word senses. As a result, common words have far too many distinct senses to be useful in information retrieval (Mihalcea, 2003); for example, the word "make" has as many as 48 senses as a verb alone. Such fine-grained distinctions between synsets present an additional difficulty for word sense disambiguation.

We illustrate these drawbacks on two specific examples in Appendix B, where we juxtapose WordNet-based and ODP-based feature generation.

The methodology we propose in this paper does not suffer from the above shortcomings. Crawling all the Web sites cataloged in the Open Directory results in exceptionally wide word coverage. Furthermore, the crawled texts provide a plethora of information about each ODP concept.

To the best of our knowledge, with the exception of the above WordNet studies, there have been no attempts to date to automatically use large-scale repositories of structured background knowledge for text categorization. An interesting approach to using non-structured background knowledge was proposed by Zelikovitz and Hirsh (2000). This work uses a collection of unlabeled examples as intermediaries in comparing testing examples with the training ones. Specifically, when an unknown test instance does not appear to resemble any labeled training instances, unlabeled examples that are similar to both may be used as "bridges." Using this approach, it is possible to handle the situation where the training and the test document have few or no words in common. The unlabeled documents are used to define a cosine similarity metric, which is then used by the KNN algorithm for actual text categorization. This approach, however, suffers from efficiency problems, as looking for intermediaries to compare every two documents makes it necessary to explore a combinatorial search space. In a subsequent paper, Zelikovitz and Hirsh (2001) proposed an alternative way to use unlabeled documents as background knowledge. In this work, unlabeled texts are pooled together with the training documents to compute a Latent Semantic Analysis (LSA) (Deerwester et al., 1990) model. The resulting LSA metric then facilitates comparison of test documents to training documents. The addition of unlabeled documents significantly increases the amount of data on which word cooccurrence statistics is estimated, thus providing a solution to text categorization problems where training data is particularly scarce.

The methodology described in this paper uses external knowledge explicitly cataloged by humans to enhance machine learning algorithms. There have also been other studies (notably, using semi-supervised learning methodology) that augmented the bag of words approach to text categorization with external knowledge distilled from unlabelled data (Goldberg and Zhu, 2006; Ando and Zhang, 2005a,b; Blei et al., 2003; Nigam et al., 2000; Joachims, 1999b). Consequently, it would be very interesting to compare the performance of these two approaches empirically. Intuitively, some inferences, such as those described in Section 4, would be hard to make by using solely unstructured data. On the other hand, unstructured data is more readily available, so it is possible that semi-supervised methods can compensate for the lack of structure by increasing the volume of the data. There are, however, non-trivial research questions regarding an appropriate setup for such a comparison. For example, assuming our methodology is based on the ODP as described in this paper, what corpus should be used by the semi-supervised learner? And if one of the methods shows better performance, should it be attributed to the method or to the particular knowledge source being used? We plan to investigate these and other related questions in our future work. In any case, it is

most likely that each of the methods has its own strengths, and finding a way to combine them can be a very interesting research direction.

While our approach relies on existing repositories of classified knowledge, there is a large body of research on extracting facts through Web mining (Cafarella et al., 2005; Etzioni et al., 2004), so it would be interesting to consider using such extracted facts to drastically increase the amount of available knowledge, especially when measures are taken to ascertain correctness of the extracted information (Downey et al., 2005).

In a recent study (Gabrilovich and Markovitch, 2007), we applied our methodology to the problem of computing semantic relatedness of words and texts, for which previous state of the art results have been based on LSA. In that work we proposed Explicit Semantic Analysis (ESA), which represents fragments of text in the space of knowledge concepts defined in the Open Directory or in Wikipedia. ESA uses the same basic feature generation methodology that we presented herein, but represents texts in the space of all available concepts (discarding the bag of words altogether), rather then augmenting the bag of words with a few top scoring concepts. Compared with the existing state of the art, using ESA results in substantial improvements in correlation of computed relatedness scores with human judgments: from $r = 0.56$ to 0.75 for individual words and from $r = 0.60$ to 0.72 for longer texts. These findings prove that the benefits of using distilled human knowledge are much greater than merely using cooccurrence statistics gathered from a collection of auxiliary unlabeled texts.

Our use of local contexts to facilitate fine-grained feature generation is reminiscent of the intra-document dynamics analysis proposed by Gabrilovich et al. (2004) for characterization of news article types. The latter work manipulated sliding contextual windows of the same size to make their scores directly comparable. As we showed in Section 5.5, the multi-resolution approach, which operates at several levels of linguistic abstraction, is superior to fixed-size windows for the case of text categorization. Incidentally, the term "Local Context Analysis" is also used in an entirely different branch of information retrieval. Xu and Croft (2000) used this term to refer to a particular kind of query expansion, where a query is expanded *in the context* of top-ranked retrieved documents.

In our methodology, we first learn a text classifier that maps local document contexts onto ODP concepts, and then use this classifier for feature generation in other learning tasks. This framework is clearly related to the area of *transfer learning*, where knowledge learned in one domain is transferred to another domain. Some works in this area assume a set of related classification tasks, and learn shared parameters. For example, Caruana (1997) trained one neural network for several related classification tasks, such that nodes in the hidden level were useful across the tasks. Jebara (2004) presented a method for feature and kernel selection across related tasks. Do and Ng (2005) described a general way of using softmax regression for learning a parameter function from a set of classification problems, so that the learned parameter function can then be used for future learning tasks. Bennett et al. (2005) introduced a method for learning a meta-classifier over several domains. The meta-classifier combines reliability indicators (Toyama and Horvitz, 2000) with the base classifiers to improve their performance. Several studies in NLP (Sutton and McCallum, 1998; Chang et al., 2006; Raina et al., 2007; Ando and Zhang, 2005a) and image classification (Wu and Dietterich, 2004; Raina et al., 2007; Ando and Zhang, 2005a) used a cascade approach, where a classifier trained on one task is used as a feature for another task. This type of transfer is the most similar to ours, as we also use a very large set of such classifiers trained on the ODP as features in other learning tasks.

7. Conclusions and Future Work

In this paper we proposed a feature generation methodology for text categorization. In order to render machine learning algorithms with the common-sense and domain-specific knowledge of humans, we use large hierarchical knowledge repositories to build a *feature generator*. These knowledge repositories, which have been manually crafted by human editors, provide a fully automatic way to tap into the collective knowledge of tens of thousands of people. The feature generator analyzes documents prior to text categorization and augments the conventional bag of words representation with relevant concepts from the knowledge repository. The enriched representation contains information that cannot be deduced from the document text alone.

In Section 2 we listed several limitations of the BOW approach, and in the subsequent sections we showed how they are resolved by our methodology. In particular, external knowledge allows us to reason about words that appear in the testing set but not in the training set. We can also use hierarchically organized knowledge to make powerful generalizations, making it possible to know that certain infrequent words belong to more general classes of words. Externally supplied knowledge can also help in those cases when some information vital for classification is omitted from training texts because it is assumed to be shared by the target readership.

We also described multi-resolution analysis, which examines the document text at several levels of linguistic abstraction and performs feature generation at each level. When polysemous words are considered in their native context, word sense disambiguation is implicit. Implicit disambiguation allows the feature generator to cope with word synonymy and polysemy. Furthermore, when the document text is processed at several levels of granularity, even briefly mentioned aspects can be identified and used. These might easily have been overlooked if the document were processed as one large chunk of text.

Empirical evaluation definitively confirmed that knowledge-based feature generation brings text categorization to a new level of performance. Interestingly, the sheer breadth and depth of the ODP, further boosted by crawling the URLs cataloged in the directory, brought about improvements both in regular text categorization as well as in the (non-topical) sentiment classification task.

Given the domain-specific nature of some test collections, we also compared the utility of narrow domain-specific knowledge with that of larger amounts of information covering all branches of knowledge. Perhaps surprisingly, we found that even for narrow-scope test collections, a wide coverage knowledge base yielded substantially greater improvements than its domain-specific subsets. This observation reinforces the *breadth hypothesis*, formulated by Lenat and Feigenbaum (1990), that "to behave intelligently in unexpected situations, an agent must be capable of falling back on increasingly general knowledge."

We believe that this research only scratches the surface of what can be achieved with knowledge-rich features. In our future work, we plan to investigate new algorithms for mapping document contexts onto hierarchy concepts, as well as new techniques for selecting attributes that are most characteristic of every concept. We intend to apply focused crawling to collect only relevant Web pages when cataloged URLs are crawled; we also plan to apply page segmentation techniques to eliminate noise from crawled pages (Yu et al., 2003). In addition to the ODP, we also plan to make use of domain-specific hierarchical knowledge bases, such as the Medical Subject Headings (MeSH).

In its present form, our method can inherently be applied only for improving representation of textual documents. Indeed, to date we applied our feature generation methodology for improving the

performance of text categorization. However, we believe our approach can also be applied beyond mere text, as long as the objects to be manipulated are accompanied with some textual description. As an example, consider a collection of medical records containing test results paired with narrative reports. Performing feature generation from narrative reports is likely to produce pertinent concepts that can be used for augmenting the original record. Indeed, prior studies (Hripcsak et al., 1995) showed that natural language processing techniques can be used to extract vital information from narrative reports in automated decision-support systems.

Finally, we conjecture that knowledge-based feature generation will also be useful for other information retrieval tasks beyond text categorization, and we intend to investigate this in our future work. Specifically, we intend to apply feature generation to information search and word sense disambiguation. In the search scenario, we are studying ways to augment both the query and documents in the collection with generated features. This way, documents will be indexed in the augmented space of words plus concepts. In this respect, we are exploring possible use of relevance feedback techniques (Ruthven and Lalmas, 2003) in order to augment the query with most useful generated features. Current approaches to word sense disambiguation represent contexts that contain ambiguous words using the bag of words augmented with part-of-speech information. To this end, we believe representation of such contexts can be greatly improved if we use feature generation to map such contexts into relevant knowledge concepts. Anecdotal evidence (such as the last example in Section 5.3.1) implies our method has much promise for improving the state of the art in word sense disambiguation.

Acknowledgments

We thank Lev Finkelstein and Alex Gontmakher for many helpful discussions. This research was partially supported by the Technion's Counter-Terrorism Competition and by the MUSCLE Network of Excellence.

Appendix A. Definitions of Category Sets for RCV1 and OHSUMED

This Appendix gives the full definition of the category sets we used for RCV1 (Table 8) and OHSUMED (Table 9).

Set name	Categories comprising the set
Topic-16	e142, gobit, e132, c313, e121, godd, ghea, e13, c183, m143, gspo, c13, e21, gpol, m14, c15
Topic-10A	e31, c41, c151, c313, c31, m13, ecat, c14, c331, c33
Topic-10B	m132, c173, g157, gwea, grel, c152, e311, c21, e211, c16
Topic-10C	c34, c13, gtour, c311, g155, gdef, e21, genv, e131, c17
Topic-10D	c23, c411, e13, gdis, c12, c181, gpro, c15, g15, c22
Topic-10E	c172, e513, e12, ghea, c183, gdip, m143, gcrim, e11, gvio
Industry-16	i81402, i79020, i75000, i25700, i83100, i16100, i1300003, i14000, i3302021, i8150206, i0100132, i65600, i3302003, i8150103, i3640010, i9741102
Industry-10A	i47500, i5010022, i3302021, i46000, i42400, i45100, i32000, i81401, i24200, i77002
Industry-10B	i25670, i61000, i81403, i34350, i1610109, i65600, i3302020, i25700, i47510, i9741110
Industry-10C	i25800, i41100, i42800, i16000, i24800, i02000, i34430, i36101, i24300, i83100
Industry-10D	i1610107, i97400, i64800, i0100223, i48300, i81502, i34400, i82000, i42700, i81402
Industry-10E	i33020, i82003, i34100, i66500, i1300014, i34531, i16100, i22450, i22100, i42900

Table 8: Definition of RCV1 category sets used in the experiments

Set name	Categories comprising the set (parentheses contain MeSH identifiers)
OHSUMED-10A	B-Lymphocytes (D001402); Metabolism, Inborn Errors (D008661); Creatinine (D003404); Hypersensitivity (D006967); Bone Diseases, Metabolic (D001851); Fungi (D005658); New England (D009511); Biliary Tract (D001659); Forecasting (D005544); Radiation (D011827)
OHSUMED-10B	Thymus Gland (D013950); Insurance (D007341); Historical Geographic Locations (D017516); Leukocytes (D007962); Hemodynamics (D006439); Depression (D003863); Clinical Competence (D002983); Anti-Inflammatory Agents, Non-Steroidal (D000894); Cytophotometry (D003592); Hydroxy Acids (D006880)
OHSUMED-10C	Endothelium, Vascular (D004730); Contraceptives, Oral, Hormonal (D003278); Acquired Immunodeficiency Syndrome (D000163); Gram-Positive Bacteria (D006094); Diarrhea (D003967); Embolism and Thrombosis (D016769); Health Behavior (D015438); Molecular Probes (D015335); Bone Diseases, Developmental (D001848); Referral and Consultation (D012017)
OHSUMED-10D	Antineoplastic and Immunosuppressive Agents (D000973); Receptors, Antigen, T-Cell (D011948); Government (D006076); Arthritis, Rheumatoid (D001172); Animal Structures (D000825); Bandages (D001458); Italy (D007558); Investigative Techniques (D008919); Physical Sciences (D010811); Anthropology (D000883)
OHSUMED-10E	HTLV-BLV Infections (D006800); Hemoglobinopathies (D006453); Vulvar Diseases (D014845); Polycyclic Hydrocarbons, Aromatic (D011084); Age Factors (D000367); Philosophy, Medical (D010686); Antigens, CD4 (D015704); Computing Methodologies (D003205); Islets of Langerhans (D007515); Regeneration (D012038)

Table 9: Definition of OHSUMED category sets used in the experiments

Appendix B. Comparing Knowledge Sources for Feature Generation: ODP versus WordNet

In Section 6 we surveyed the shortcomings of WordNet as a possible source for knowledge-based feature generation. To demonstrate these shortcomings, we juxtapose WordNet-based and ODP-based feature generation for two of sample sentences we examined in Section 5.3.1 (we repeat the ODP context classifications for readers' convenience). We used WordNet version 1.6 to look up the words. In what follows, synsets are denoted with curly braces, and noun and verb synsets are followed by their immediate hypernym (more general synset), if applicable.

- **Text:** *"Rumsfeld appeared with Gen. Richard Myers, chairman of the Joint Chiefs of Staff."*

  ODP classifications:

  - SOCIETY/ISSUES/GOVERNMENT_OPERATIONS
  - SOCIETY/POLITICS
  - SOCIETY/ISSUES/WARFARE_AND_CONFLICT/SPECIFIC_CONFLICTS/IRAQ
  - SCIENCE/TECHNOLOGY/ MILITARY_SCIENCE
  - SOCIETY/ISSUES/WARFARE_AND_CONFLICT/WEAPONS
  - SOCIETY/HISTORY/BY_REGION/NORTH_AMERICA/UNITED_STATES/PRESIDENTS/ BUSH,_GEORGE_WALKER
  - SOCIETY/POLITICS/CONSERVATISM

  WordNet :
  {**Rumsfeld**} → { }; *(word not present in WordNet)*

  {look, **appear**, seem} → {be}; {**appear**}; {**appear**, come out} → {happen, materialize}; {**appear**, seem} → {be}; {**appear**, come along}; {**appear**} → {perform, execute, do}

  {**Gen**} → {information, info}

  {**Richard**} → { }; *(word not present in WordNet)*

  {**Myers**} → { }; *(word not present in WordNet)*

  {president, **chairman**, chairwoman, chair, chairperson} → {presiding officer}; {chair, **chairman**} → {head, lead}

  {**joint**, articulation, articulatio} → {body part}; {**joint**} → {spot}; {articulation, join, **joint**, juncture, junction} → {connection, connexion, link}; {roast, **joint**} → {cut, cut of meat}; {**joint**} → {junction, conjunction}; {**joint**, marijuana cigarette, reefer, stick} → {cigarette, cigaret, coffin nail, butt, fag}

  {**joint**} → {fit, go}; {**joint**, articulate} → {supply, provide, render, furnish}; {**joint**} → {fasten, fix, secure}

  {**joint** (vs. separate)}; {joint}

  {head, **chief**, top dog} → {leader}; {foreman, **chief**, gaffer, honcho, boss} → {supervisor}

  {**staff**} → {force, personnel}; {**staff**} → {stick}; {**staff**, faculty} → {body}; {**staff**} → {symbol}; {**staff**, stave} → {musical notation}

  {**staff**} → {provide, supply, ply, cater}

- **Text:** *"Herceptin is a so-called targeted therapy because of its ability to attack diseased cells and leave healthy ones alone."*

  ODP classifications:

  - HEALTH/CONDITIONS_AND_DISEASES/CANCER/BREAST
  - SOCIETY/ISSUES/HEALTH/CONDITIONS_AND_DISEASES/CANCER/ALTERNATIVE_TREATMENTS
  - HEALTH/SUPPORT_GROUPS/CONDITIONS_AND_DISEASES/CANCER

  WordNet:

  {**Herceptin**} → { }; *(word not present in WordNet)*

  {alleged (prenominal), **so-called**, supposed} → {questionable (vs. unquestionable)}

  {**target**, aim, place, direct, point} → {aim, take, train, take aim, direct}

  {**therapy**} → {medical care, medical aid}

  {**ability**} → {quality}

  {**ability**, power} → {cognition, knowledge}

  {**attack**, onslaught, onset, onrush} → {operation}; {**attack**} → {turn, play}; {fire, **attack**, flak, blast} → {criticism, unfavorable judgment}; {approach, **attack**, plan of attack} → {conceptualization, conceptualisation, formulation, formularizing, formularising}; {**attack**, attempt} → {battery, assault, assault and battery}; {**attack**, tone-beginning} → {beginning, start, commencement}; {**attack**} → {affliction}; {**attack**, assault} → {attention, attending};

  {**attack**, assail} → {fight, struggle}; {**attack**, round, assail, lash out, snipe, assault} → {criticize, criticise, pick apart}; {**attack**, aggress} → {act, move}; {assail, assault, set on, **attack**}; {**attack**} → {begin, get, start out, start, set about, set out, commence}; {**attack**} → {affect}

  {assault (prenominal), **attack** (prenominal)} → {offensive (vs. defensive)};

  {**diseased**, morbid, pathologic, pathological} → {unhealthy (vs. healthy)};

  {**cell**} → {compartment}; {**cell**} → {entity, something}; {**cell**, electric cell} → {electrical device}; {**cell**, cadre} → {political unit}; {**cell**, cubicle} → {room}; {**cell**, jail cell, prison cell} → {room}

  {**leave**, leave of absence} → {time off}; {**leave**} → {permission}; {farewell, **leave**, leave-taking, parting} → {departure, going, going away, leaving};

  {**leave**, go forth, go away}; *(16 more verb senses omitted for brevity)*

  {**healthy** (vs. unhealthy)}; {**healthy**} → {sound (vs. unsound)}; {**healthy**, salubrious, good for you (predicate)} → {wholesome (vs. unwholesome)}; {fit (vs. unfit), **healthy**} → {able, able-bodied}; {**healthy**, intelligent, levelheaded, sound} → {reasonable (vs. unreasonable), sensible};

  {**one**, 1, I, ace, single, unity} → {digit}; {**one**} → {unit}

  {**alone** (predicate)} → {unsocial (vs. social)}; {**alone** (predicate), lone (prenominal), lonely (prenominal), solitary} → {unaccompanied (vs. accompanied)}; {**alone** (predicate), only} →

{exclusive (vs. inclusive)}; {**alone** (predicate), unique, unequaled, unequalled, unparalleled} → {incomparable (vs. comparable), uncomparable}

{entirely, exclusively, solely, **alone**, only}; {**alone**, unaccompanied}

Evidently, WordNet classifications are overly general and diverse because context words cannot be properly disambiguated. Furthermore, owing to lack of proper names, WordNet cannot possibly provide the wealth of information encoded in the Open Directory, which easily overcomes the drawbacks of WordNet.

References

Rie Kubota Ando and Tong Zhang. Framework for learning predictive structures from multiple tasks and unlabeled data. *Journal of Machine Learning Research*, pages 1817–1853, 2005a.

Rie Kubota Ando and Tong Zhang. A high-performance semi-supervised learning method for text chunking. In *Proceedings of the 43rd Annual Meeting of the ACL*, pages 1–9, Ann Arbor, MI, June 2005b.

Douglas Baker and Andrew K. McCallum. Distributional clustering of words for text classification. In Bruce Croft, Alistair Moffat, Cornelis J. Van Rijsbergen, Ross Wilkinson, and Justin Zobel, editors, *Proceedings of the 21st ACM International Conference on Research and Development in Information Retrieval*, pages 96–103, Melbourne, AU, 1998. ACM Press, New York, US. URL `http://www.cs.cmu.edu/ mccallum/papers/clustering-sigir98.ps.gz`.

Lisa Ballesteros and Bruce Croft. Phrasal translation and query expansion techniques for cross-language information retrieval. In *Proceedings of the 20th ACM International Conference on Research and Development in Information Retrieval*, pages 84–91, 1997.

Roberto Basili, Alessandro Moschitti, and Maria T. Pazienza. Language-sensitive text classification. In *Proceedings of RIAO-00, 6th International Conference "Recherche d'Information Assistee par Ordinateur"*, pages 331–343, Paris, France, 2000.

Ron Bekkerman. Distributional clustering of words for text categorization. Master's thesis, Technion, 2003.

Paul N. Bennett, Susan T. Dumais, and Eric Horvitz. Inductive transfer for text classification using generalized reliability indicators. In *Proceedings of the ICML-2003 Workshop on The Continuum from Labeled to Unlabeled Data*, 2003.

Paul N. Bennett, Susan T. Dumais, and Eric Horvitz. The combination of text classifiers using reliability indicators. *Information Retrieval*, 8(1):67–100, 2005.

Cathy Blake and Christopher Merz. UCI Repository of machine learning databases, 1998. `http://www.ics.uci.edu/~mlearn/MLRepository.html`.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent direchlet allocation. *Journal of Machine Learning Research*, 3:993–1022, 2003.

Stephan Bloehdorn and Andreas Hotho. Boosting for text classification with semantic features. In *Proceedings of the MSW 2004 Workshop at the 10th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 70–87, 2004.

Janez Brank, Marko Grobelnik, Natasa Milic-Frayling, and Dunia Mladenic. Interaction of feature selection methods and linear classification models. In *Workshop on Text Learning held at ICML-2002*, 2002.

Eric Brill. Transformation-based error-driven learning and natural language processing: A case study in part of speech tagging. *Computational Linguistics*, 21(4):543–565, 1995.

Michael Cafarella, Doug Downey, Stephen Soderland, and Oren Etzioni. Knowitnow: Fast, scalable information extraction from the web. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, Vancouver, Canada, October 2005.

Lijuan Cai and Thomas Hofmann. Text categorization by boosting automatically extracted concepts. In *Proceedings of the 26th International Conference on Research and Development in Information Retrieval*, pages 182–189, 2003.

Maria Fernanda Caropreso, Stan Matwin, and Fabrizio Sebastiani. A learner-independent evaluation of the usefulness of statistical phrases for automated text categorization. In Amita G. Chin, editor, *Text Databases and Document Management: Theory and Practice*, pages 78–102. Idea Group Publishing, Hershey, US, 2001. URL `http://faure.iei.pi.cnr.it/ fabrizio/Publications/TD01a/TD01a.pdf`.

Rich Caruana. Multitask learning. *Machine Learning*, 28(1):41–75, 1997.

Soumen Chakrabarti, Byron Dom, Rakesh Agrawal, and Prabhakar Raghavan. Using taxonomy, discriminants, and signatures for navigating in text databases. In *Proceedings of the 23rd VLDB Conference*, pages 446–455, 1997.

Lois Mai Chan. *A Guide to the Library of Congress Classification*. Libraries Unlimited, 5th edition, 1999.

Ming-Wei Chang, Quang Do, and Dan Roth. Multilingual dependency parsing: A pipeline approach. In *Recent Advances in Natural Language Processing*, pages 195–204, 2006.

Stanley Chen and Joshua Goodman. A high-performance semi-supervised learning method for text chunking. In *Proceedings of the 34th Annual Meeting of the ACL*, 1996.

William W. Cohen. Automatically extracting features for concept learning from the web. In *Proceedings of the 17th International Conference on Machine Learning*, 2000.

Nello Cristianini, John Shawe-Taylor, and Huma Lodhi. Latent semantic kernels. *Journal of Intelligent Information Systems*, 18(2/3):127–152, 2002.

Dmitry Davidov, Evgeniy Gabrilovich, and Shaul Markovitch. Parameterized generation of labeled datasets for text categorization based on a hierarchical directory. In *Proceedings of the 27th ACM International Conference on Research and Development in Information Retrieval*, pages 250–257, 2004.

Franca Debole and Fabrizio Sebastiani. Supervised term weighting for automated text categorization. In *Proceedings of SAC-03, 18th ACM Symposium on Applied Computing*, pages 784–788, 2003.

Scott Deerwester, Susan Dumais, George Furnas, Thomas Landauer, and Richard Harshman. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41 (6):391–407, 1990.

Gerald Dejong and Raymond Mooney. Explanation-based learning: An alternative view. *Machine Learning*, 1(2):145–176, 1986.

Melvil Dewey, Joan S. Mitchell, Julianne Beall, Giles Martin, Winton E. Matthews, and Gregory R. New, editors. *Dewey Decimal Classification and Relative Index*. Online Computer Library Center (OCLC), 22nd edition, 2003.

Inderjit Dhillon, Subramanyam Mallela, and Rahul Kumar. A divisive information-theoretic feature clustering algorithm for text classification. *Journal of Machine Learning Research*, 3:1265–1287, March 2003. URL `http://www.jmlr.org/papers/volume3/dhillon03a/dhillon03a.pdf`.

Chuong Do and Andrew Ng. Transfer learning for text classification. In *Proceedings of Neural Information Processing Systems (NIPS)*, 2005.

Doug Downey, Oren Etzioni, and Stephen Soderland. A probabilistic model of redundancy in information extraction. In *Proceedings of the 19th International Joint Conference on Artificial Intelligence*, Edinburgh, Scotand, August 2005.

Richard Duda and Peter Hart. *Pattern Classification and Scene Analysis*. John Wiley and Sons, 1973.

Susan Dumais and Hao Chen. Hierarchical classification of web content. In *Proceedings of the 23rd ACM International Conference on Research and Development in Information Retrieval*, pages 256–263, 2000.

Susan Dumais, John Platt, David Heckerman, and Mehran Sahami. Inductive learning algorithms and representations for text categorization. In *Proceedings of the 7th ACM International Conference on Information and Knowledge Management*, pages 148–155, 1998.

Oren Etzioni, Michael Cafarella, Doug Downey, Stanley Kok, Ana-Maria Popescu, Tal Shaked, Stephen Soderland, Daniel Weld, and Alexander Yates. Webscale information extraction in knowitall (preliminary results). In *Proceedings of the 13th International World Wide Web Conference (WWW'04)*, New York, USA, May 2004. ACM Press.

Tom Fawcett. *Feature Discovery for Problem Solving Systems*. PhD thesis, UMass, May 1993.

Christiane Fellbaum, editor. *WordNet: An Electronic Lexical Database*. MIT Press, Cambridge, MA, 1998.

Johannes Fuernkranz, Tom Mitchell, and Ellen Riloff. A case study in using linguistic phrases for text categorization on the WWW. In Mehran Sahami, editor, *Learning for Text Categorization: Proceedings of the 1998 AAAI/ICML Workshop*, pages 5–12. AAAI Press, Madison, Wisconsin, 1998.

Evgeniy Gabrilovich and Shaul Markovitch. Text categorization with many redundant features: Using aggressive feature selection to make SVMs competitive with C4.5. In *Proceedings of the 21st International Conference on Machine Learning*, pages 321–328, 2004.

Evgeniy Gabrilovich and Shaul Markovitch. Feature generation for text categorization using world knowledge. In *Proceedings of the 19th International Joint Conference on Artificial Intelligence*, pages 1048–1053, Edinburgh, Scotand, August 2005.

Evgeniy Gabrilovich and Shaul Markovitch. Overcoming the brittleness bottleneck using Wikipedia: Enhancing text categorization with encyclopedic knowledge. In *Proceedings of the 21st National Conference on Artificial Intelligence*, pages 1301–1306, July 2006.

Evgeniy Gabrilovich and Shaul Markovitch. Computing semantic relatedness using wikipedia-based explicit semantic analysis. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence*, pages 1606–1611, January 2007.

Evgeniy Gabrilovich, Susan Dumais, and Eric Horvitz. Newsjunkie: Providing personalized newsfeeds via analysis of information novelty. In *Proceedings of the Thirteenth International World Wide Web Conference (WWW2004)*, pages 482–490, New York, NY, May 2004. ACM Press.

Andrew Goldberg and Xiaojin Zhu. Seeing stars when there aren't many stars: Graph-based semi-supervised learning for sentiment categorization. In *Workshop on Textgraphs: Graph-based Algorithms for Natural Language Processing, HLT-NAACL 2006*, 2006.

Eui-Hong (Sam) Han and George Karypis. Centroid-based document classification: Analysis and experimental results. In *Proceedings of the Fourth European Conference on Principles and Practice of Knowledge Discovery in Databases*, September 2000.

William Hersh, Chris Buckley, T.J. Leone, and David Hickam. OHSUMED: An interactive retrieval evaluation and new large test collection for research. In *Proceedings of the 17th ACM International Conference on Research and Development in Information Retrieval*, pages 192–201, 1994.

George Hripcsak, Carol Friedman, Philip O. Alderson, William DuMouchel, Stephen B. Johnson, and Paul D. Clayton. Unlocking clinical data from narrative reports: a study of natural language processing. *Annals of Internal Medicine*, 122(9):681–688, 1995.

Yuh-Jyh Hu and Dennis Kibler. A wrapper approach for constructive induction. In *The Thirteenth National Conference on Artificial Intelligence*, pages 47–52, 1996.

David A. Hull. Improving text retrieval for the routing problem using latent semantic indexing. In W. Bruce Croft and Cornelis J. Van Rijsbergen, editors, *Proceedings of the 17th ACM International Conference on Research and Development in Information Retrieval*, pages 282–289, Dublin, Ireland, 1994. Springer Verlag, Heidelberg, Germany. URL `http://www.acm.org/pubs/articles/proceedings/ir/188490/p282-hull/p282-hull.pdf`.

Tony Jebara. Multi-task feature and kernel selection for svms. In *Proceedings of the Twenty-First International Conference on Machine Learning*, pages 55–63, 2004.

Thorsten Joachims. Text categorization with support vector machines: Learning with many relevant features. In *Proceedings of the European Conference on Machine Learning*, pages 137–142, 1998.

Thorsten Joachims. Making large-scale SVM learning practical. In B. Schoelkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods – Support Vector Learning*, pages 169–184. The MIT Press, 1999a.

Thorsten Joachims. Transductive inference for text classification using support vector machines. In *Proceedings of the 13th International Conference on Machine Learning*, 1999b.

Daphne Koller and Mehran Sahami. Hierarchically classifying documents using very few words. In *Proceedings of the 14th International Conference on Machine Learning*, pages 170–178, 1997.

Daniel Kudenko and Haym Hirsh. Feature generation for sequence categorization. In *Proceedings of the 15th Conference of the American Association for Artificial Intelligence*, pages 733–738, 1998.

Giridhar Kumaran and James Allan. Text classification and named entities for new event detection. In *Proceedings of the 27th ACM International Conference on Research and Development in Information Retrieval*, pages 297–304, 2004.

Ken Lang. Newsweeder: Learning to filter netnews. In *Proceedings of the 12th International Conference on Machine Learning*, pages 331–339, 1995.

Douglas Lenat and Edward Feigenbaum. On the thresholds of knowledge. *Artificial Intelligence*, 47:185–250, 1990.

Edda Leopold and Joerg Kindermann. Text categorization with support vector machines: How to represent texts in input space. *Machine Learning*, 46:423–444, 2002.

David D. Lewis. An evaluation of phrasal and clustered representations on a text categorization task. In *Proceedings of the 15th ACM International Conference on Research and Development in Information Retrieval*, pages 37–50, 1992.

David D. Lewis and W. Bruce Croft. Term clustering of syntactic phrases. In *Proceedings of the 13th ACM International Conference on Research and Development in Information Retrieval*, pages 385–404, 1990.

David D. Lewis, Robert E. Schapire, James P. Callan, and Ron Papka. Training algorithms for linear text classifiers. In *Proceedings of the 19th ACM International Conference on Research and Development in Information Retrieval*, pages 298–306, 1996.

David D. Lewis, Yiming Yang, Tony Rose, and Fan Li. RCV1: A new benchmark collection for text categorization research. *Journal of Machine Learning Research*, 5:361–397, 2004.

Shaul Markovitch and Danny Rosenstein. Feature generation using general constructor functions. *Machine Learning*, 49(1):59–98, 2002.

Christopher J. Matheus. The need for constructive induction. In L.A. Birnbaum and G.C. Collins, editors, *Proceedings of the Eighth International Workshop on Machine Learning*, pages 173–177, 1991.

Christopher J. Matheus and Larry A. Rendell. Constructive induction on decision trees. In *Proceedings of the 11th International Conference on Artificial Intelligence*, pages 645–650, 1989.

Ia Mcilwaine. *The Universal Decimal Classification: Guide to its Use*. UDC Consortium, 2000.

MeSH. Medical subject headings (MeSH). National Library of Medicine, 2003. `http://www.nlm.nih.gov/mesh`.

Rada Mihalcea. Turning wordnet into an information retrieval resource: Systematic polysemy and conversion to hierarchical codes. *International Journal of Pattern Recognition and Artificial Intelligence (IJPRAI)*, 17(1):689–704, 2003.

Andrei Mikheev. Feature lattices and maximum entropy models. *Information Retrieval*, 1999.

Tom Mitchell, Richard Keller, and Smadar Kedar-Cabelli. Explanation-based generalization: A unifying view. *Machine Learning*, 1(1):47–80, 1986.

Mandar Mitra, Amit Singhal, and Chris Buckley. Improving automatic query expansion. In *Proceedings of the 21st ACM International Conference on Research and Development in Information Retrieval*, pages 206–214, 1998.

Dunja Mladenic. Feature subset selection in text learning. In *Proceedings of ECML-98, 10th European Conference on Machine Learning*, pages 95–100, 1998a.

Dunja Mladenic. Turning Yahoo into an automatic web-page classifier. In *Proceedings of 13th European Conference on Artificial Intelligence*, pages 473–474, 1998b.

Patrick M. Murphy and Michael J. Pazzani. ID2-of-3: Constructive induction of M-of-N concepts for discriminators in decision trees. In *Proceedings of the 8th International Conference on Machine Learning*, pages 183–188. Morgan Kaufmann, 1991.

Kamal Nigam, Andrew McCallum, Sebastian Thrun, and Tom Mitchell. Text classification from labeled and unlabeled documents using EM. *Machine Learning*, 39(2-3):103–134, 2000.

Kamal Nigam, Andrew McCallum, and Tom Mitchell. Semi-supervised text classification using EM. In Olivier Chapelle, Bernhard Schoelkopf, and Alexander Zien, editors, *Semi-Supervised Learning*. MIT Press, Boston, MA, 2006.

Giulia Pagallo and David Haussler. Boolean feature discovery in empirical learning. *Machine Learning*, 5(1):71–99, 1990. ISSN 0885-6125.

Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. Thumbs up? Sentiment classification using machine learning techniques. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 79–86, 2002.

Fuchun Peng and Dale Shuurmans. Combining naive Bayes and n-gram language models for text classification. In *Proceedings of the 25th European Conference on Information Retrieval Research (ECIR-03)*, pages 335–350, 2003.

Fuchun Peng, Dale Schuurmans, and Shaojun Wang. Augmenting naive Bayes classifiers with statistical language models. *Information Retrieval*, 7(3-4):317–345, 2004.

Martin Porter. An algorithm for suffix stripping. *Program*, 14(3):130–137, 1980.

Rajat Raina, Andrew Ng, and Daphne Koller. Constructing informative priors using transfer learning. In *Proceedings of the 23rd International Conference on Machine Learning (ICML)*, Pittsburgh, PA, 2006.

Rajat Raina, Alexis Battle, Honglak Lee, Benjamin Packer, and Andrew Y. Ng. Self-taught learning: Transfer learning from unlabeled data. In *ICML '07: Proceedings of the 24th International Conference on Machine learning*, 2007.

Bhavani Raskutti, Herman Ferra, and Adam Kowalczyk. Second order features for maximizing text classification performance. In L. De Raedt and P. Flach, editors, *Proceedings of the European Conference on Machine Learning (ECML)*, Lecture notes in Artificial Intelligence (LNAI) 2167, pages 419–430. Springer-Verlag, 2001.

Reuters. *Reuters-21578 text categorization test collection, Distribution 1.0*. Reuters, 1997. `daviddlewis.com/resources/testcollections/reuters21578`.

Joseph John Rocchio. Relevance feedback in information retrieval. In *The SMART Retrieval System: Experiments in Automatic Document Processing*, pages 313–323. Prentice Hall, 1971.

Monica Rogati and Yiming Yang. High-performing feature selection for text classification. In *Proceedings of the International Conference on Information and Knowledge Management (CIKM'02)*, pages 659–661, 2002.

J.K. Rowling. *Harry Potter and the Philosopher's Stone*. Bloomsbury, 1997.

Miguel E. Ruiz and Padmini Srinivasan. Hierarchical text categorization using neural networks. *Information Retrieval*, 5:87–118, 2002.

Ian Ruthven and Mounia Lalmas. A survey on the use of relevance feedback for information access systems. *Knowledge Engineering Review*, 18(2):95–145, 2003.

Carl Sable, Kathleen McKeown, and Kenneth W. Church. NLP found helpful (at least for one text categorization task). In *Conference on Empirical Methods in Natural Language Processing*, pages 172–179, 2002.

Gerard Salton and Chris Buckley. Term weighting approaches in automatic text retrieval. *Information Processing and Management*, 24(5):513–523, 1988.

Gerard Salton and Michael McGill. *An Introduction to Modern Information Retrieval*. McGraw-Hill, 1983.

Sam Scott. Feature engineering for a symbolic approach to text classification. Master's thesis, U. Ottawa, 1998.

Fabrizio Sebastiani. Machine learning in automated text categorization. *ACM Computing Surveys*, 34(1):1–47, 2002. URL `http://faure.iei.pi.cnr.it/ fabrizio/Publications/ACMCS02.pdf`.

Charles Sutton and Andrew McCallum. Composition of conditional random fields for transfer learning. In *Emprical Methods in Natural Language Processing (HLT/EMNLP)*, 1998.

Kentaro Toyama and Eric Horvitz. Bayesian modality fusion: Probabilistic integration of multiple vision algorithms for head tracking. In *Proceedings of the 4th Asian Conference on Computer Vision*, 2000.

Alfonso Urena-Lopez, Manuel Buenaga, and Jose M. Gomez. Integrating linguistic resources in TC through WSD. *Computers and the Humanities*, 35:215–230, 2001.

Ellen M. Voorhees. Query expansion using lexical-semantic relations. In *Proceedings of the 17th International Conference on Research and Development in Information Retrieval*, pages 61–69, 1994.

Ellen M. Voorhees. Using wordnet for text retrieval. In Christiane Fellbaum, editor, *WordNet, an Electronic Lexical Database*. The MIT Press, 1998.

Bill B. Wang, R.I. McKay, Hussein A. Abbass, and Michael Barlow. A comparative study for domain ontology guided feature extraction. In *Proceedings of the 26th Australian Computer Science Conference (ASCS-2003)*, pages 69–78, 2003.

Bernard Widrow and Samuel Stearns. *Adaptive Signal Processing*. Prentice Hall, 1985.

Michael Wong, Wojciech Ziarko, and Patrick C.N. Wong. Generalized vector spaces model in information retrieval. In *Proceedings of the 8th ACM International Conference on Research and Development in Information Retrieval*, pages 18–25, 1985.

Pengcheng Wu and Thomas G. Dietterich. Improving svm accuracy by training on auxiliary data sources. In *Proceedings of the Twenty-First International Conference on Machine Learning*, pages 871–878, New York, NY, USA, 2004. ACM Press.

Jinxi Xu and W. Bruce Croft. Query expansion using local and global document analysis. In *Proceedings of the 19th International Conference on Research and Development in Information Retrieval*, pages 4–11, 1996.

Jinxi Xu and W. Bruce Croft. Improving the effectiveness of information retrieval with local context analysis. *ACM Transactions on Information Systems*, 18(1):79–112, 2000.

Yiming Yang and Xin Liu. A re-examination of text categorization methods. In *Proceedings of the 22nd International Conference on Research and Development in Information Retrieval*, pages 42–49, 1999.

Yiming Yang and Jan Pedersen. A comparative study on feature selection in text categorization. In *Proceedings of the 14th International Conference on Machine Learning*, pages 412–420, 1997.

Yiming Yang, Sean Slattery, and Rayid Ghani. A study of approaches to hypertext categorization. *Journal of Intelligent Information Systems*, 18(2/3):219–241, 2002.

Shipeng Yu, Deng Cai, Ji-Rong Wen, and Wei-Ying Ma. Improving pseudo-relevance feedback in web information retrieval using web page segmentation. In *Proceedings of the 12th International World Wide Web Conference (WWW'03)*, Budapest, Hungary, May 2003. ACM Press.

Sarah Zelikovitz and Haym Hirsh. Improving short-text classification using unlabeled background knowledge to assess document similarity. In *Proceedings of the 17th International Conference on Machine Learning*, pages 1183–1190, 2000.

Sarah Zelikovitz and Haym Hirsh. Using LSI for text classification in the presence of background text. In *Proceedings of the Conference on Information and Knowledge Management*, pages 113–118, 2001.

Justin Zobel and Alistair Moffat. Exploring the similarity space. *ACM SIGIR Forum*, 32(1):18–34, 1998.

Journal of Machine Learning Research 8 (2007) 2347-2368 Submitted 12/06; Revised 7/07; Published 10/07

AdaBoost is Consistent

Peter L. Bartlett BARTLETT@STAT.BERKELEY.EDU
Department of Statistics and Computer Science Division
University of California
Berkeley, CA 94720-3860, USA

Mikhail Traskin MTRASKIN@STAT.BERKELEY.EDU
Department of Statistics
University of California
Berkeley, CA 94720-3860, USA

Editor: Yoav Freund

Abstract

The risk, or probability of error, of the classifier produced by the AdaBoost algorithm is investigated. In particular, we consider the stopping strategy to be used in AdaBoost to achieve universal consistency. We show that provided AdaBoost is stopped after $n^{1-\varepsilon}$ iterations—for sample size n and $\varepsilon \in (0,1)$—the sequence of risks of the classifiers it produces approaches the Bayes risk.

Keywords: boosting, adaboost, consistency

1. Introduction

Boosting algorithms are an important recent development in classification. These algorithms belong to a group of voting methods (see, for example, Schapire, 1990; Freund, 1995; Freund and Schapire, 1996, 1997; Breiman, 1996, 1998), that produce a classifier as a linear combination of *base* or *weak* classifiers. While empirical studies show that boosting is one of the best off the shelf classification algorithms (see Breiman, 1998) theoretical results do not give a complete explanation of their effectiveness.

The first formulations of boosting by Schapire (1990), Freund (1995), and Freund and Schapire (1996, 1997) considered boosting as an iterative algorithm that is run for a fixed number of iterations and at every iteration it chooses one of the base classifiers, assigns a weight to it and eventually outputs the classifier that is the weighted majority vote of the chosen classifiers. Later Breiman (1997, 1998, 2004) pointed out that boosting is a gradient descent type algorithm (see also Friedman et al., 2000; Mason et al., 2000).

Experimental results by Drucker and Cortes (1996), Quinlan (1996), Breiman (1998), Bauer and Kohavi (1999) and Dietterich (2000) showed that boosting is a very effective method, that often leads to a low test error. It was also noted that boosting continues to decrease test error long after the sample error becomes zero: though it keeps adding more weak classifiers to the linear combination of classifiers, the generalization error, perhaps surprisingly, usually does not increase. However some of the experiments suggested that there might be problems, since boosting performed worse than bagging in the presence of noise (Dietterich, 2000), and boosting concentrated not only on the "hard" areas, but also on outliers and noise (Bauer and Kohavi, 1999). And indeed, some more experiments, for example by Friedman et al. (2000), Grove and Schuurmans (1998) and Mason et al.

(2000), see also Bickel et al. (2006), as well as some theoretical results (for example, Jiang, 2002) showed that boosting, ran for an arbitrary large number of steps, overfits, though it takes a very long time to do it.

Upper bounds on the risk of boosted classifiers were obtained, based on the fact that boosting tends to maximize the margin of the training examples (Schapire et al., 1998; Koltchinskii and Panchenko, 2002), but Breiman (1999) pointed out that margin-based bounds do not completely explain the success of boosting methods. In particular, these results do not resolve the issue of consistency: they do not explain under which conditions we may expect the risk to converge to the Bayes risk. A recent work by Reyzin and Schapire (2006) shows that while maximization of the margin is useful, it should not be done at the expense of the classifier complexity.

Breiman (2004) showed that under some assumptions on the underlying distribution "population boosting" converges to the Bayes risk as the number of iterations goes to infinity. Since the population version assumes infinite sample size, this does not imply a similar result for AdaBoost, especially given results of Jiang (2002), that there are examples when AdaBoost has prediction error asymptotically suboptimal at $t = \infty$ (t is the number of iterations).

Several authors have shown that *modified* versions of AdaBoost are consistent. These modifications include restricting the l_1-norm of the combined classifier (Mannor et al., 2003; Blanchard et al., 2003; Lugosi and Vayatis, 2004; Zhang, 2004) , and restricting the step size of the algorithm (Zhang and Yu, 2005). Jiang (2004) analyses the unmodified boosting algorithm and proves a process consistency property, under certain assumptions. Process consistency means that there exists a sequence (t_n) such that if AdaBoost with sample size n is stopped after t_n iterations, its risk approaches the Bayes risk. However Jiang also imposes strong conditions on the underlying distribution: the distribution of X (the predictor) has to be absolutely continuous with respect to Lebesgue measure and the function $F_B(X) = (1/2)\ln(\mathbf{P}(Y=1|X)/\mathbf{P}(Y=-1|X))$ has to be continuous on $\mathcal{X}$. Also Jiang's proof is not constructive and does not give any hint on when the algorithm should be stopped. Bickel et al. (2006) prove a consistency result for AdaBoost, under the assumption that the probability distribution is such that the steps taken by the algorithm are not too large. In this paper, we study stopping rules that guarantee consistency. In particular, we are interested in AdaBoost, not a modified version. Our main result (Corollary 9) demonstrates that a simple stopping rule suffices for consistency: the number of iterations is a fixed function of the sample size. We assume only that the class of base classifiers has finite VC-dimension, and that the span of this class is sufficiently rich. Both assumptions are clearly necessary.

2. Notation

Here we describe the AdaBoost procedure formulated as a coordinate descent algorithm and introduce definitions and notation. We consider a binary classification problem. We are given $\mathcal{X}$, the measurable (feature) space, and $\mathcal{Y} = \{-1, 1\}$, the set of (binary) labels. We are given a sample $S_n = \{(X_i, Y_i)\}_{i=1}^n$ of i.i.d. observations distributed as the random variable $(X, Y) \sim \mathcal{P}$, where $\mathcal{P}$ is an unknown distribution. Our goal is to construct a classifier $g_n : \mathcal{X} \to \mathcal{Y}$ based on this sample. The quality of the classifier g_n is given by the misclassification probability

$$L(g_n) = \mathbf{P}(g_n(X) \neq Y | S_n).$$

Of course we want this probability to be as small as possible and close to the Bayes risk

$$L^\star = \inf_g L(g) = \mathrm{E}(\min\{\eta(X), 1-\eta(X)\}),$$

where the infimum is taken over all possible (measurable) classifiers and $\eta(\cdot)$ is a conditional probability

$$\eta(x) = \mathbf{P}(Y = 1|X = x).$$

The infimum above is achieved by the Bayes classifier $g^\star(x) = g(2\eta(x) - 1)$, where

$$g(x) = \begin{cases} 1 & , \quad x > 0, \\ -1 & , \quad x \leq 0. \end{cases}$$

We are going to produce a classifier as a linear combination of *base* classifiers in $\mathcal{H} = \{h|h : \mathcal{X} \to \mathcal{Y}\}$. We shall assume that class $\mathcal{H}$ has a finite VC (Vapnik-Chervonenkis) dimension $d_{VC}(\mathcal{H}) = \max\left\{|S| : S \subseteq \mathcal{X}, \left|\mathcal{H}_{|S}\right| = 2^{|S|}\right\}$.

AdaBoost works to find a combination f that minimizes the convex criterion

$$\frac{1}{n}\sum_{i=1}^{n} \exp(-Y_i f(X_i)).$$

Many of our results are applicable to a broader family of such algorithms, where the function $\alpha \mapsto \exp(-\alpha)$ is replaced by another function φ. Thus, for a function $\varphi : \mathbb{R} \to \mathbb{R}^+$, we define the empirical φ-risk and the φ-risk,

$$R_{\varphi,n}(f) = \frac{1}{n}\sum_{i=1}^{n} \varphi(Y_i f(X_i)) \qquad \text{and} \qquad R_\varphi(f) = \mathrm{E}\varphi(Yf(X)).$$

Clearly, the function φ needs to be appropriate for classification, in the sense that a measurable f that minimizes $R_\varphi(f)$ should have minimal risk. This is equivalent (see Bartlett et al., 2006) to φ satisfying the following condition ('classification calibration'). For all $0 \leq \eta \leq 1, \eta \neq 1/2$,

$$\inf\{\eta\varphi(\alpha) + (1-\eta)\varphi(-\alpha) : \alpha(2\eta - 1) \leq 0\} > \inf\{\eta\varphi(\alpha) + (1-\eta)\varphi(-\alpha) : \alpha \in \mathbb{R}\}. \qquad (1)$$

We shall assume that φ satisfies (1).

Then the boosting procedure can be described as follows.

1. Set $f_0 \equiv 0$. Choose number of iterations t.

2. For $k = 1, \ldots, t$, set

   $$f_k = f_{k-1} + \alpha_{k-1} h_{k-1},$$

   where the following holds for some fixed $\gamma \in (0, 1]$ independent of k.

   $$R_{\varphi,n}(f_k) \leq \gamma \inf_{h \in \mathcal{H}, \alpha \in \mathbb{R}} R_{\varphi,n}(f_{k-1} + \alpha h) + (1-\gamma) R_{\varphi,n}(f_{k-1}). \qquad (2)$$

   We call α_i the step size of the algorithm at step i.

3. Output $g \circ f_t$ as the final classifier.

The choice of $\gamma < 1$ in the above algorithm allows approximate minimization. Notice that the original formulation of AdaBoost assumed exact minimization in (2), which corresponds to $\gamma = 1$.

We shall also use the convex hull of $\mathcal{H}$ scaled by $\lambda \geq 0$,

$$\mathcal{F}_\lambda = \left\{ f \,\middle|\, f = \sum_{i=1}^{n} \lambda_i h_i, n \in \mathbb{N} \cup \{0\}, \lambda_i \geq 0, \sum_{i=1}^{n} \lambda_i = \lambda, h_i \in \mathcal{H} \right\}$$

as well as the set of k-combinations, $k \in \mathbb{N}$, of functions in $\mathcal{H}$

$$\mathcal{F}^k = \left\{ f \,\middle|\, f = \sum_{i=1}^{k} \lambda_i h_i, \lambda_i \in \mathbb{R}, h_i \in \mathcal{H} \right\}.$$

We also need to define the $l_\star$-norm: for any $f \in \mathcal{F}$

$$\|f\|_\star = \inf\left\{ \sum |\alpha_i|, f = \sum \alpha_i h_i, h_i \in \mathcal{H} \right\}.$$

Define the squashing function $\pi_l(\cdot)$ to be

$$\pi_l(x) = \begin{cases} l & , \quad x > l, \\ x & , \quad x \in [-l, l], \\ -l & , \quad x < -l. \end{cases}$$

Then the set of truncated functions is

$$\pi_l \circ \mathcal{F} = \left\{ \tilde{f} | \tilde{f} = \pi_l(f), f \in \mathcal{F} \right\}.$$

The set of classifiers based on a class $\mathcal{F}$ is denoted by

$$g \circ \mathcal{F} = \{ \tilde{f} | \tilde{f} = g(f), f \in \mathcal{F} \}.$$

Define the derivative of an arbitrary function $Q(\cdot)$ in the direction of h as

$$Q'(f;h) = \left. \frac{\partial Q(f + \lambda h)}{\partial \lambda} \right|_{\lambda=0}.$$

The second derivative $Q''(f;h)$ is defined similarly.

3. Consistency of Boosting Procedure

In this section, we present the proof of the consistency of AdaBoost. We begin with an overview.

The usual approach to proving consistency involves a few key steps (see, for example, Bartlett et al., 2004). The first is a comparison theorem, which shows that as the φ-risk $R_\varphi(f_n)$ approaches $R_\varphi^\star$ (the infimum over measurable functions of R_φ), $L(f_n)$ approaches $L^\star$. The classification calibration condition (1) suffices for this (Bartlett et al., 2006). The second step is to show that the class of functions is suitably rich so that there is some sequence of elements $\bar{f}_n$ for which $\lim_{n\to\infty} R_\varphi(\bar{f}_n) = R_\varphi^\star$. The third step is to show that the φ-risk of the estimate f_n approaches that of the reference sequence $\bar{f}_n$. For instance, for a method of sieves that minimizes the empirical φ-risk over a suitable set $\mathcal{F}_n$ (which increases with the sample size n), one could define the reference sequence $\bar{f}_n$ as the minimizer of the φ-risk in $\mathcal{F}_n$. Then, provided that the sets $\mathcal{F}_n$ grow suitably slowly with n, the maximal deviation over $\mathcal{F}_n$ between empirical φ-risk and φ-risk would converge to zero. Such a uniform convergence result would imply that the sequence f_n has φ-risk converging to $R_\varphi^\star$.

The key difficulty with this approach is that the concentration inequalities behind the uniform convergence results are valid only for a suitably small class of suitably bounded functions. However boosting in general and AdaBoost in particular may produce functions that cannot be appropriately bounded. To circumvent this difficulty, we rely on the observation that, for the purposes of classification, we can replace the function f returned by AdaBoost by any function f' that satisfies $\text{sign}(f') = \text{sign}(f)$. Therefore we consider the clipped version $\pi_\lambda \circ f_t$ of the function returned by AdaBoost after t iterations. This clipping ensures that the functions f_t are suitably bounded. Furthermore, the complexity of the clipped class (as measured by its pseudo-dimension—see Pollard, 1984) grows slowly with the stopping time t, so we can show that the φ-risk of a clipped function is not much larger than its empirical φ-risk. Lemma 4 provides the necessary details. In order to compare the empirical φ-risk of the clipped function to that of a suitable reference sequence $\bar{f}_n$, we first use the fact that the empirical φ-risk of a clipped function $\pi_\lambda \circ f_t$ is not much larger than the empirical φ-risk of f_t.

The next step is to relate $R_{\varphi,n}(f_t)$ to $R_{\varphi,n}(\bar{f}_n)$. The choice of a suitable sieve depends on what can be shown about the progress of the algorithm. We consider an increasing sequence of $l_\star$-balls, and define $\bar{f}_n$ as the (near) minimizer of the φ-risk in the appropriate $l_\star$-ball. Theorems 6 and 8 show that as the stopping time increases, the empirical φ-risk of the function returned by AdaBoost is not much larger than that of $\bar{f}_n$. Finally Hoeffding's inequality shows that the empirical φ-risks of the reference functions $\bar{f}_n$ are close to their φ-risks. Combining all the pieces, the φ-risk of $\pi_\lambda \circ f_t$ approaches $R^\star_\varphi$, provided the stopping time increases suitably slowly with the sample size. The consistency of AdaBoost follows.

We now describe our assumptions. First, we shall impose the following condition.

Condition 1 ***Denseness.*** *Let the distribution $\mathcal{P}$ and class $\mathcal{H}$ be such that*

$$\lim_{\lambda\to\infty} \inf_{f\in\mathcal{F}_\lambda} R_\varphi(f) = R^\star_\varphi,$$

where $R^\star_\varphi = \inf R_\varphi(f)$ over all measurable functions.

For many classes $\mathcal{H}$, the above condition is satisfied for all possible distributions $\mathcal{P}$. Lugosi and Vayatis (2004, Lemma 1) discuss sufficient conditions for Condition 1. As an example of such a class, we can take the class of indicators of all rectangles or the class of indicators of half-spaces defined by hyperplanes or the class of binary trees with the number of terminal nodes equal to $d+1$ (we consider trees with terminal nodes formed by successive univariate splits), where d is the dimensionality of $\mathcal{X}$ (see Breiman, 2004).

The following set of conditions deals with uniform convergence and convergence of the boosting algorithm. The main theorem (Theorem 1) shows that these, together with Condition 1, suffice for consistency of the boosting procedure. Later in this section we show that the conditions are satisfied by AdaBoost.

Condition 2 *Let n be sample size. Let there exist non-negative sequences $t_n \to \infty$, $\zeta_n \to \infty$ and a sequence $\{\bar{f}_n\}_{n=1}^\infty$ of reference functions such that*

$$R_\varphi(\bar{f}_n) \underset{n\to\infty}{\to} R^\star,$$

and suppose that the following conditions are satisfied.

a. ***Uniform convergence of t_n-combinations.***

$$\sup_{f\in\pi_{\zeta_n}\circ\mathcal{F}^{t_n}} |R_\varphi(f) - R_{\varphi,n}(f)| \xrightarrow[n\to\infty]{a.s.} 0. \tag{3}$$

b. ***Convergence of empirical φ-risks for the sequence $\{\bar{f}_n\}_{n=1}^\infty$.***

$$\max\left\{0, R_{\varphi,n}(\bar{f}_n) - R_\varphi(\bar{f}_n)\right\} \xrightarrow[n\to\infty]{a.s.} 0. \tag{4}$$

c. ***Algorithmic convergence of t_n-combinations.***

$$\max\left\{0, R_{\varphi,n}(f_{t_n}) - R_{\varphi,n}(\bar{f}_n)\right\} \xrightarrow[n\to\infty]{a.s.} 0. \tag{5}$$

Now we state the main theorem.

Theorem 1 *Assume φ is classification calibrated and convex. Assume, without loss of generality, that for $\varphi_\lambda = \inf_{x\in[-\lambda,\lambda]}\varphi(x)$,*

$$\lim_{\lambda\to\infty}\varphi_\lambda = \inf_{x\in(-\infty,\infty)}\varphi(x) = 0. \tag{6}$$

Let Condition 2 be satisfied. Then the boosting procedure stopped at step t_n returns a sequence of classifiers f_{t_n} almost surely satisfying $L(g(f_{t_n})) \to L^\star$ as $n\to\infty$.

Remark 2 *Note that Condition (6) could be replaced by the mild condition that the function φ is bounded below.*

Proof For almost every outcome ω on the probability space $(\Omega,\mathcal{S},\mathbf{P})$ we can define sequences $\varepsilon_n^1(\omega)\to 0$, $\varepsilon_n^2(\omega)\to 0$ and $\varepsilon_n^3(\omega)\to 0$, such that for almost all ω the following inequalities are true.

$$\begin{aligned}
R_\varphi(\pi_{\zeta_n}(f_{t_n})) &\le R_{\varphi,n}(\pi_{\zeta_n}(f_{t_n})) + \varepsilon_n^1(\omega) \quad \text{by (3)} \\
&\le R_{\varphi,n}(f_{t_n}) + \varepsilon_n^1(\omega) + \varphi_{\zeta_n} \qquad (7)\\
&\le R_{\varphi,n}(\bar{f}_n) + \varepsilon_n^1(\omega) + \varphi_{\zeta_n} + \varepsilon_n^2(\omega) \quad \text{by (5)} \\
&\le R_\varphi(\bar{f}_n) + \varepsilon_n^1(\omega) + \varphi_{\zeta_n} + \varepsilon_n^2(\omega) + \varepsilon_n^3(\omega) \quad \text{by (4).} \qquad (8)
\end{aligned}$$

Inequality (7) follows from the convexity of $\varphi(\cdot)$ (see Lemma 14 in Appendix E). By (6) and choice of the sequence $\{\bar{f}_n\}_{n=1}^\infty$ we have $R_\varphi(\bar{f}_n)\to R^\star$ and $\varphi_{\zeta_n}\to 0$. And from (8) follows $R_\varphi(\pi_{\zeta_n}(f_{t_n}))\to R^\star$ a.s. Eventually we can use the result by Bartlett et al. (2006, Theorem 3) to conclude that

$$L(g(\pi_{\zeta_n}(f_{t_n})))\xrightarrow{a.s.}L^\star.$$

But for $\zeta_n > 0$ we have $g(\pi_{\zeta_n}(f_{t_n})) = g(f_{t_n})$, therefore

$$L(g(f_{t_n}))\xrightarrow{a.s.}L^\star.$$

Hence, the boosting procedure is consistent if stopped after t_n steps. ■

The almost sure formulation of Condition 2 does not provide explicit rates of convergence of $L(g(f_{t_n}))$ to $L^\star$. However, a slightly stricter form of Condition 2, which allows these rates to be calculated, is considered in Appendix A.

In the following sections, we show that Condition 2 can be satisfied for some choices of φ. We shall treat parts (a)–(c) separately.

3.1 Uniform Convergence of t_n-Combinations

Here we show that Condition 2 (a) is satisfied for a variety of functions φ, and in particular for exponential loss used in AdaBoost. We begin with a simple lemma (see Freund and Schapire, 1997, Theorem 8 or Anthony and Bartlett, 1999, Theorem 6.1):

Lemma 3 *For any $t \in \mathbb{N}$ if $d_{VC}(\mathcal{H}) \geq 2$ the following holds:*

$$d_P(\mathcal{F}^t) \leq 2(t+1)(d_{VC}(\mathcal{H})+1)\log_2[2(t+1)/\ln 2],$$

where $d_P(\mathcal{F}^t)$ is the pseudo-dimension of class $\mathcal{F}^t$.

The proof of consistency is based on the following result, which builds on the result by Koltchinskii and Panchenko (2002) and resembles a lemma due to Lugosi and Vayatis (2004, Lemma 2).

Lemma 4 *For a continuous function φ define the Lipschitz constant*

$$L_{\varphi,\zeta} = \inf\{L | L > 0, |\varphi(x) - \varphi(y)| \leq L|x-y|, -\zeta \leq x, y \leq \zeta\}$$

and maximum absolute value of $\varphi(\cdot)$ when argument is in $[-\zeta,\zeta]$

$$M_{\varphi,\zeta} = \max_{x\in[-\zeta,\zeta]} |\varphi(x)|.$$

Then for $V = d_{VC}(\mathcal{H})$, $c = 24\int_0^1 \sqrt{\ln\frac{8e}{\varepsilon^2}}d\varepsilon$ and any n, $\zeta > 0$ and $t > 0$,

$$\mathrm{E}\sup_{f\in\pi_\zeta\circ\mathcal{F}^t} |R_\varphi(f) - R_{\varphi,n}(f)| \leq c\zeta L_{\varphi,\zeta}\sqrt{\frac{(V+1)(t+1)\log_2[2(t+1)/\ln 2]}{n}}.$$

Also, for any $\delta > 0$, with probability at least $1-\delta$,

$$\begin{aligned}\sup_{f\in\pi_\zeta\circ\mathcal{F}^t} |R_\varphi(f) - R_{\varphi,n}(f)| &\leq c\zeta L_{\varphi,\zeta}\sqrt{\frac{(V+1)(t+1)\log_2[2(t+1)/\ln 2]}{n}} \\ &+ M_{\varphi,\zeta}\sqrt{\frac{\ln(1/\delta)}{2n}}. \qquad (9)\end{aligned}$$

Proof The proof is given in Appendix B. ■

Now, if we choose ζ and δ as functions of n, such that $\sum_{n=1}^\infty \delta^2(n) < \infty$ and right hand side of (9) converges to 0 as $n \to \infty$, we can appeal to Borel-Cantelli lemma and conclude, that for such choice of ζ_n and δ_n Condition 2 (a) holds.

Lemma 4, unlike Lemma 2 of Lugosi and Vayatis (2004), allows us to choose the number of steps t, which describes the complexity of the linear combination of base functions, and this is essential for the proof of the consistency. It is easy to see that for AdaBoost (i.e., $\varphi(x) = e^{-x}$) we can choose $\zeta = \kappa \ln n$ and $t = n^{1-\varepsilon}$ with $\kappa > 0$, $\varepsilon \in (0,1)$ and $2\kappa - \varepsilon < 0$.

3.2 Convergence of Empirical φ-Risks for the Sequence $\{\bar{f}_n\}_{n=1}^{\infty}$.

To show that Condition 2(b) is satisfied for a variety of loss functions we use Hoeffding's inequality.

Theorem 5 *Define the variation of a function φ on the interval $[-a,a]$ (for $a>0$) as*

$$V_{\varphi,a} = \sup_{x\in[-a,a]} \varphi(x) - \inf_{x\in[-a,a]} \varphi(x).$$

If a sequence $\{\bar{f}_n\}_{n=1}^{\infty}$ satisfies the condition $\bar{f}_n(x) \in [-\lambda_n,\lambda_n], \forall x \in \mathcal{X}$, where $\lambda_n > 0$ is chosen so that $V_{\varphi,\lambda_n} = o(\sqrt{n/\ln n})$, then

$$\max\{0, R_{\varphi,n}(\bar{f}_n) - R_\varphi(\bar{f}_n)\} \xrightarrow[n\to\infty]{a.s.} 0. \tag{10}$$

Proof Since we restricted the range of $\bar{f}_n$ to the interval $[-\lambda_n,\lambda_n]$, we have, almost surely, $\varphi(Y\bar{f}_n(X)) \in [a,b]$, where $b-a \le V_{\varphi,\lambda_n}$. Therefore Hoeffding's inequality guarantees that for all ε_n

$$\mathbf{P}\left(R_{\varphi,n}(\bar{f}_n) - R_\varphi(\bar{f}_n) \ge \varepsilon_n\right) \le \exp\left(-2n\varepsilon_n^2/V_{\varphi,\lambda_n}^2\right) = \delta_n.$$

To prove the statement of the theorem we require $\varepsilon_n = o(1)$ and $\sum \delta_n < \infty$. Then we appeal to the Borel-Cantelli lemma to conclude that (10) holds. These restrictions are satisfied if

$$V_{\varphi,\lambda_n}^2 = o\left(\frac{n}{\ln n}\right)$$

and the statement of the theorem follows. ■

The choice of λ_n in the above theorem depends on the loss function φ. In the case of the AdaBoost loss $\varphi(x) = e^{-x}$ we shall choose $\lambda_n = \kappa \ln n$, where $\kappa \in (0,1/2)$. One way to guarantee that the functions $\bar{f}_n$ satisfy condition $\bar{f}_n(x) \in [-\lambda_n,\lambda_n], \forall x \in \mathcal{X}$, is to choose $\bar{f}_n \in \mathcal{F}_{\lambda_n}$.

3.3 Algorithmic Convergence of AdaBoost

So far we dealt with the statistical properties of the function we are minimizing; now we turn to the algorithmic part. Here we show that Condition 2(c) is satisfied for the AdaBoost algorithm. We need the following simple consequence of the proof of Bickel et al. (2006, Theorem 1).

Theorem 6 *Let the function $Q(f)$ be convex in f and twice differentiable in all directions $h \in \mathcal{H}$. Let $Q^\star = \lim_{\lambda\to\infty} \inf_{f\in\mathcal{F}_\lambda} Q(f)$. Assume that $\forall c_1, c_2$, such that $Q^\star < c_1 < c_2 < \infty$,*

$$\begin{aligned} 0 &< \inf\{Q''(f;h) : c_1 < Q(f) < c_2, h \in \mathcal{H}\} \\ &\le \sup\{Q''(f;h) : Q(f) < c_2, h \in \mathcal{H}\} < \infty. \end{aligned}$$

Also assume the following approximate minimization scheme for $\gamma \in (0,1]$. Define $f_{k+1} = f_k + \alpha_{k+1}h_{k+1}$ such that

$$Q(f_{k+1}) \le \gamma \inf_{h\in\mathcal{H},\alpha\in\mathbb{R}} Q(f_k + \alpha h) + (1-\gamma)Q(f_k)$$

and

$$Q(f_{k+1}) = \inf_{\alpha\in\mathbb{R}} Q(f_k + \alpha h_{k+1}).$$

Then for any reference function $\bar{f}$ and the sequence of functions f_m, produced by the boosting algorithm, the following bound holds $\forall m > 0$ such that $Q(f_m) > Q(\bar{f})$.

$$Q(f_m) \le Q(\bar{f}) + \sqrt{\frac{8B^3(Q(f_0)-Q(\bar{f}))^2}{\gamma^2\beta^3}\left(\ln\frac{\ell_0^2+c_3 m}{\ell_0^2}\right)^{-\frac{1}{2}}}, \tag{11}$$

where $\ell_k = \left\|\bar{f}-f_k\right\|_\star$, $c_3 = 2(Q(f_0)-Q(\bar{f}))/\beta$, $\beta = \inf\{Q''(f;h) : Q(\bar{f}) < Q(f) < Q(f_0), h \in \mathcal{H}\}$, $B = \sup\{Q''(f;h) : Q(f) < Q(f_0), h \in \mathcal{H}\}$.

Proof The statement of the theorem is a version of a result implicit in the proof of (Bickel et al., 2006, Theorem 1). The proof is given in Appendix C. ■

Remark 7 *Results in Zhang and Yu (2005, e.g., Lemma 4.1) provide similar bounds under either an assumption of a bounded step size of the boosting algorithm or a positive lower bound on $Q''(f;h)$ for all f,h. Since we consider boosting algorithms with unrestricted step size, the only option would be to assume a positive lower bound on the second derivative. While such an assumption is fine for the quadratic loss $\varphi(x) = x^2$, second derivative $R_n''(f;h)$ of the empirical risk for the exponential loss used by the AdaBoost algorithm can not be bounded from below by a positive constant in a general case. Theorem 6 makes a mild assumption that second derivative is positive for all f such that $R(f) > R^\star$ ($R_n(f) > R_n^\star$) .*

It is easy to see, that the theorem above applies to the AdaBoost algorithm, since there we first choose the direction (base classifier) h_i and then we compute the step size α_i as

$$\alpha_i = \frac{1}{2}\ln\frac{1-\varepsilon_i}{\varepsilon_i} = \frac{1}{2}\ln\frac{R(f_i)-R'(f_i;h_i)}{R(f_i)+R'(f_i;h_i)}.$$

Now we only have to recall that this value of α_i corresponds to exact minimization in the direction h_i.

From now on we are going to specialize to AdaBoost and use $\varphi(x) = e^{-x}$. Hence we drop the subscript φ in $R_{\varphi,n}$ and R_φ and use R_n and R respectively.

Theorem 6 allows us to get an upper bound on the difference between the *exp*-risk of the function output by AdaBoost and the *exp*-risk of the appropriate reference function. For brevity in the next theorem we make an assumption $R^\star > 0$, though a similar result can be stated for $R^\star = 0$. For completeness, the corresponding theorem is given in Appendix D.

Theorem 8 *Assume $R^\star > 0$. Let t_n be the number of steps we run AdaBoost. Let $\lambda_n = \kappa \ln n$, $\kappa \in (0,1/2)$. Let $a > 1$ be an arbitrary fixed number. Let $\{\bar{f}_n\}_{n=1}^\infty$ be a sequence of functions such that $\bar{f}_n \in \mathcal{F}_{\lambda_n}$. Then with probability at least $1-\delta_n$, where $\delta_n = \exp\left(-2(R^\star)^2 n^{1-2\kappa}/a^2\right)$, the following holds*

$$R_n(f_{t_n}) \le R_n(\bar{f}_n) + \frac{2\sqrt{2}(1-R^\star(a-1)/a)}{\gamma\left(\frac{a-1}{a}R^\star\right)^{3/2}}\left(\ln\frac{\lambda_n^2+2t_n(a/(a-1)-R^\star)/R^\star}{\lambda_n^2}\right)^{-1/2}.$$

Proof This theorem follows directly from Theorem 6. Because in AdaBoost

$$R_n''(f;h) = \frac{1}{n}\sum_{i=1}^n (-Y_i h(X_i))^2 e^{-Y_i f(X_i)} = \frac{1}{n}\sum_{i=1}^n e^{-Y_i f(X_i)} = R_n(f),$$

then all the conditions in Theorem 6 are satisfied as long as $R_n(\bar{f}_n) > 0$ (with $Q(f)$ replaced by $R_n(f)$) and in the Equation (11) we have $B = R_n(f_0) = 1$, $\beta \geq R_n(\bar{f}_n)$, $\left\|f_0 - \bar{f}_n\right\|_\star \leq \lambda_n$. Since for t such that $R_n(f_t) \leq R_n(\bar{f}_n)$ the theorem is trivially true, we only have to notice that $\exp(Y_i \bar{f}_n(X_i)) \in [0, n^\kappa]$, hence Hoeffding's inequality guarantees that

$$\mathbf{P}\left(\frac{1}{n}\sum_{i=1}^{n} e^{Y_i \bar{f}_n(X_i)} - \mathrm{E}e^{Y\bar{f}_n(X)} \leq -\frac{R^\star}{a}\right) \leq \exp\left(-2(R^\star)^2 n^{1-2\kappa}/a^2\right) = \delta_n,$$

where we choose and fix the constant $a > 1$ arbitrarily and independently of n and the sequence $\{\bar{f}_n\}_{n=1}^\infty$. Therefore with probability at least $1-\delta_n$ we bound empirical risk from below as $R_n(\bar{f}_n) \geq R(\bar{f}_n) - R^\star/a \geq R^\star - R^\star/a = R^\star(a-1)/a$, since $R(\bar{f}_n) \geq R^\star$. Therefore $\beta \geq R^\star(a-1)/a$ and the result follows immediately from Equation (11) if we use the fact that $R^\star > 0$. ■

It is easy to see that choice of λ_n's in the above theorem ensures that $\sum_{n=1}^\infty \delta_n < \infty$, therefore Borel-Cantelli lemma guarantees that for $t_n \to \infty$ sufficiently fast (for example as $O(n^\alpha)$ for $\alpha \in (0,1)$)

$$\max\{0, R_n(f_{t_n}) - R_n(\bar{f}_n)\} \xrightarrow[n\to\infty]{a.s.} 0.$$

If in addition to the conditions of Theorem 8 we shall require that

$$R(\bar{f}_n) \leq \inf_{f \in \mathcal{F}_{\lambda_n}} R(f) + \varepsilon_n,$$

for some $\varepsilon_n \to 0$, then together with Condition 1 this will imply $R(\bar{f}_n) \to R^*$ as $n \to \infty$ and Condition 2 (c) follows.

3.4 Consistency of AdaBoost

Having all the ingredients at hand, consistency of AdaBoost is a simple corollary of Theorem 1.

Corollary 9 *Assume $V = d_{VC}(\mathcal{H}) < \infty$,*

$$\lim_{\lambda\to\infty} \inf_{f\in\mathcal{F}_\lambda} R(f) = R^\star$$

and $t_n = n^{1-\varepsilon}$ for $\varepsilon \in (0,1)$. Then AdaBoost stopped at step t_n returns a sequence of classifiers almost surely satisfying $L(g(f_{t_n})) \to L^\star$.

Proof First assume $L^\star > 0$. For the exponential loss function this implies $R^\star > 0$. As was suggested after the proof of Lemma 4 we may choose $\zeta_n = \kappa \ln n$ for $2\kappa - \varepsilon < 0$ (which also implies $\kappa < 1/2$) to satisfy Condition 2 (a). Recall that discussion after the proof of the Theorem 8 suggests choice of the sequence $\{\bar{f}_n\}_{n=1}^\infty$ of reference functions such that $\bar{f}_n \in \mathcal{F}_{\lambda_n}$ and

$$R(\bar{f}_n) \leq \inf_{f\in\mathcal{F}_{\lambda_n}} R(f) + \varepsilon_n$$

for $\varepsilon_n \to 0$ and $\lambda_n = \kappa \ln n$ with $\kappa \in (0, 1/2)$ to ensure that Condition 2 (c) holds. Eventually, as it follows from the discussion after the proof of the Theorem 5, choice of the sequence $\{\bar{f}_n\}_{n=1}^\infty$ to satisfy Condition 2(c) also ensures that Condition 2(b) holds. Since function $\varphi(x) = e^{-x}$ is clearly

classification calibrated and conditions of this Corollary assume Condition 1 then all the conditions of Theorem 1 hold and consistency of the AdaBoost algorithm follows.

For $L^\star = 0$ the proof is similar, but we need to use Theorem 13 in Appendix D instead of Theorem 8. ■

4. Discussion

We showed that AdaBoost is consistent if stopped sufficiently early, after t_n iterations, for $t_n = O(n^{1-\varepsilon})$ with $\varepsilon \in (0,1)$. We do not know whether this number can be increased. Results by Jiang (2002) imply that for some $\mathcal{X}$ and function class $\mathcal{H}$ the AdaBoost algorithm will achieve zero training error after t_n steps, where $n^2/t_n = o(1)$ (see also work by Mannor and Meir (2001, Lemma 1) for an example of $\mathcal{X} = \mathbb{R}^d$ and $\mathcal{H} = \{\text{linear classifiers}\}$, for which perfect separation on the training sample is guaranteed after $8n^2 \ln n$ iterations), hence if run for that many iterations, the AdaBoost algorithm does not produce a consistent classifier. We do not know what happens in between $O(n^{1-\varepsilon})$ and $O(n^2 \ln n)$. Lessening this gap is a subject of further research.

The AdaBoost algorithm, as well as other versions of the boosting procedure, replaces the $0-1$ loss with a convex function φ to overcome algorithmic difficulties associated with the non-convex optimization problem. In order to conclude that $R_\varphi(f_n) \to R_\varphi^\star$ implies $L(g(f_n)) \to L^\star$ we want φ to be classification calibrated and this requirement cannot be relaxed, as shown by Bartlett et al. (2006).

The statistical part of the analysis, summarized in Lemma 4 and Theorem 5, works for quite an arbitrary loss function φ. The only restriction imposed by Lemma 4 is that φ must be Lipschitz on any compact set. This requirement is an artifact of our proof and is caused by the use of the "contraction principle". It can be relaxed in some cases: Shen et al. (2003) use the classification calibrated loss function

$$\psi(x) = \begin{cases} 2 & , \quad x < 0, \\ 1-x & , \quad 0 \le x < 1, \\ 0 & , \quad x \ge 1, \end{cases}$$

which is non-Lipschitz on any interval $[-\lambda, \lambda], \lambda > 0$.

The algorithmic part, presented by Theorems 6 and 8, concentrated on the analysis of the exponential (AdaBoost) loss $\varphi(x) = e^{-x}$. This approach also works for the quadratic loss $\varphi(x) = (1-x)^2$. Theorem 6 assumes that the second derivative $R''_\varphi(f;h)$ is bounded from below by a positive constant, possibly dependent on the value of $R_\varphi(f)$, as long as $R_\varphi(f) > R_\varphi^\star$. This condition is clearly satisfied for $\varphi(x) = (1-x)^2$: $R''_\varphi(f;h) \equiv 2$ and we do not need an analog of Theorem 8; Theorem 6 suffices. Lemma 4 can be applied for the quadratic loss with $L_{\varphi,\lambda} = 2(1+\lambda)$ and $M_{\varphi,\lambda} = (1+\lambda)^2$. We may choose t_n, λ_n, ζ_n the same as for the exponential loss or set $\lambda_n = n^{1/4-\vartheta_1}$, $\vartheta_1 \in (0, 1/4)$, $\zeta_n = n^{\rho-\vartheta_2}$, $\vartheta_2 = (0, \rho)$, $\rho = \min(\varepsilon/2, 1/4)$ to get the following analog of Corollary 9.

Corollary 10 *Assume $\varphi(x) = (1-x)^2$. Assume $V = d_{VC}(\mathcal{H}) < \infty$,*

$$\lim_{\lambda \to \infty} \inf_{f \in \mathcal{F}_\lambda} R(f) = R^\star$$

and $t_n = n^{1-\varepsilon}$ for $\varepsilon \in (0,1)$. Then boosting procedure stopped at step t_n returns a sequence of classifiers almost surely satisfying $L(g(f_{t_n})) \to L^\star$.

We cannot make analogous conclusion about other loss functions. For example for logit loss $\varphi(x) = \ln(1+e^{-x})$, Lemma 4 and Theorem 5 work, since $L_{\varphi,\lambda} = 1$ and $M_{\varphi,\lambda} = \ln(1+e^{\lambda})$, hence choosing t_n, λ_n, ζ_n as for either the exponential or quadratic losses will work. The assumption of the Theorem 6 also holds with $R''_{\varphi,n}(f;h) \geq R_{\varphi,n}(f)/n$, though the resulting inequality is trivial: the factor $1/n$ in this bound precludes us from finding an analog of Theorem 8. A similar problem arises in the case of the modified quadratic loss $\varphi(x) = [\max(1-x,0)]^2$, for which $R''_{\varphi,n}(f;h) \geq 2/n$. Generally, any loss function with "really flat" regions may cause trouble. Another issue is the very slow rate of convergence in Theorems 6 and 8. Hence further research intended either to improve convergence rates or extend the applicability of these theorems to loss functions other than exponential and quadratic is desirable.

Acknowledgments

This work was supported by the NSF under award DMS-0434383. The authors would like to thank Peter Bickel for useful discussions, as well as Jean-Philippe Vert and two anonymous referees for their comments and suggestions.

Appendix A. Rate of Convergence of $L(g(f_{t_n}))$ to $L^\star$

Here we formulate Condition 2 in a stricter form and prove consistency along with a rate of convergence of the boosting procedure to the Bayes risk.

Condition 3 *Let n be sample size. Let there exist non-negative sequences $t_n \to \infty$, $\zeta_n \to \infty$, $\delta_n^j \to 0$ such that $\sum_{i=1}^{\infty} \delta_i^j < \infty$, $j = 1,2,3$, $\varepsilon_n^k \to 0$, $k = 1,2,3$, a sequence $\{\bar{f}_n\}_{n=1}^{\infty}$ of reference functions such that*

$$R_\varphi(\bar{f}_n) \underset{n\to\infty}{\to} R^\star,$$

and the following conditions hold.

a. ***Uniform convergence of t_n-combinations.***

$$\mathbf{P}\left(\sup_{f \in \pi_{\zeta_n} \circ \mathcal{F}^{t_n}} |R_\varphi(f) - R_{\varphi,n}(f)| > \varepsilon_n^1\right) < \delta_n^1. \tag{12}$$

b. ***Convergence of empirical φ-risks for the sequence $\{\bar{f}_n\}_{n=1}^{\infty}$.***

$$\mathbf{P}\left(R_{\varphi,n}(\bar{f}_n) - R_\varphi(\bar{f}_n) > \varepsilon_n^2\right) < \delta_n^2. \tag{13}$$

c. ***Algorithmic convergence of t_n-combinations.***

$$\mathbf{P}\left(R_{\varphi,n}(f_{t_n}) - R_{\varphi,n}(\bar{f}_n) > \varepsilon_n^3\right) < \delta_n^3. \tag{14}$$

Now we state the analog of Theorem 1.

Theorem 11 *Assume φ is classification calibrated and convex, and for $\varphi_\lambda = \inf_{x\in[-\lambda,\lambda]} \varphi(x)$ without loss of generality assume*

$$\lim_{\lambda\to\infty} \varphi_\lambda = \inf_{x\in(-\infty,\infty)} \varphi(x) = 0. \tag{15}$$

Let Condition 3 be satisfied. Then the boosting procedure stopped at step t_n returns a sequence of classifiers f_{t_n} almost surely satisfying $L(g(f_{t_n})) \to L^\star$ as $n \to \infty$.

Proof Consider the following sequence of inequalities.

$$\begin{aligned}
R_\varphi(\pi_{\zeta_n}(f_{t_n})) &\leq R_{\varphi,n}(\pi_{\zeta_n}(f_{t_n})) + \varepsilon_n^1 \quad \text{by (12)} && (16)\\
&\leq R_{\varphi,n}(f_{t_n}) + \varepsilon_n^1 + \varphi_{\zeta_n} \\
&\leq R_{\varphi,n}(\bar{f}_n) + \varepsilon_n^1 + \varphi_{\zeta_n} + \varepsilon_n^3 \quad \text{by (14)} && (17)\\
&\leq R_\varphi(\bar{f}_n) + \varepsilon_n^1 + \varphi_{\zeta_n} + \varepsilon_n^3 + \varepsilon_n^2 \quad \text{by (13).} && (18)
\end{aligned}$$

Inequalities (16), (18) and (17) hold with probability at least $1-\delta_n^1$, $1-\delta_n^2$ and $1-\delta_n^3$ respectively. We assumed in Condition 3 that $R_\varphi(\bar{f}_n) \to R^\star$ and (15) implies that $\varphi_{\zeta_n} \to 0$ by the choice of the sequence ζ_n. Now we appeal to the Borel-Cantelli lemma and arrive at $R_\varphi(\pi_{\zeta_n}(f_{t_n})) \to R^\star$ a.s. Eventually we can use Theorem 3 by Bartlett et al. (2006) to conclude that

$$L(g(\pi_{\zeta_n}(f_{t_n}))) \xrightarrow{a.s.} L^\star.$$

But for $\zeta_n > 0$ we have $g(\pi_{\zeta_n}(f_{t_n})) = g(f_{t_n})$, therefore

$$L(g(f_{t_n})) \xrightarrow{a.s.} L^\star.$$

Hence the boosting procedure is consistent if stopped after t_n steps. ■

We could prove Theorem 11 by using the Borel-Cantelli lemma and appealing to Theorem 1, but the above proof allows the following corollary on the rate of convergence.

Corollary 12 *Let the conditions of Theorem 11 be satisfied. Then there exists a non-decreasing function ψ, such that $\psi(0) = 0$, and with probability at least $1-\delta_n^1-\delta_n^2-\delta_n^3$*

$$L(g(f_{t_n})) - L^\star \leq \psi^{-1}\left((\varepsilon_n^1+\varepsilon_n^2+\varepsilon_n^3+\varphi_{\zeta_n}) + \left(\inf_{f\in\mathcal{F}_{\lambda_n}} R_\varphi - R_\varphi^\star\right)\right), \tag{19}$$

where ψ^{-1} is the inverse of ψ.

Proof From Theorem 3 of Bartlett et al. (2006), if ϕ is convex we have that

$$\psi(\theta) = \phi(0) - \inf\left\{\frac{1+\theta}{2}\phi(\alpha) + \frac{1-\theta}{2}\phi(-\alpha) : \alpha\in\mathbb{R}\right\},$$

and for any distribution and any measurable function f

$$L(g(f)) - L^\star \leq \psi^{-1}\left(R_\varphi(f) - R_\varphi^\star\right).$$

On the other hand,

$$R_\varphi(f) - R_\varphi^\star = \left(R_\varphi(f) - \inf_{f\in\mathcal{F}_{\lambda_n}} R_\varphi\right) + \left(\inf_{f\in\mathcal{F}_{\lambda_n}} R_\varphi - R_\varphi^\star\right).$$

The proof of Theorem 11 shows that for function f_{t_n} with probability at least $1-\delta_n^1-\delta_n^2$

$$R_\varphi(f_{t_n}) - \inf_{f\in\mathcal{F}_{\lambda_n}} R_\varphi \le \varepsilon_n^1 + \varepsilon_n^2 + \varepsilon_n^3 + \varphi_{\zeta_n}.$$

Putting all the components together we obtain (19). ■

The second term under ψ^{-1} in (19) is an approximation error and, in a general case, it may decrease arbitrarily slowly. However, if it is known that it decreases sufficiently fast, the first term becomes an issue. For example Corollary 9, even if the approximation error decreases sufficiently fast, will give a convergence rate of the order $O\left((\ln n)^{-\frac{1}{4}}\right)$. This follows from Example 1 by Bartlett et al. (2006), where it is shown that for AdaBoost (exponential loss function) $\psi^{-1}(x) \le \sqrt{2x}$, and the fact that both ε_n^1 and ε_n^2, as well as φ_{ζ_n}, in Corollary 9 decrease at the rate $O(n^{1-\alpha})$ (in fact, α's might be different for all three of them), hence everything is dominated by ε_n^3, which is $O\left((\ln n)^{-\frac{1}{2}}\right)$.

Appendix B. Proof of Lemma 4

For convenience, we state the lemma once again.

Lemma 4 *For a continuous function φ define the Lipschitz constant*

$$L_{\varphi,\zeta} = \inf\{L | L>0, |\varphi(x)-\varphi(y)| \le L|x-y|, -\zeta \le x,y \le \zeta\}$$

and maximum absolute value of $\varphi(\cdot)$ when argument is in $[-\zeta,\zeta]$

$$M_{\varphi,\zeta} = \max_{x\in[-\zeta,\zeta]} |\varphi(x)|.$$

Then for $V = d_{VC}(\mathcal{H})$, $c = 24\int_0^1 \sqrt{\ln\frac{8e}{\varepsilon^2}}d\varepsilon$ and any n, $\zeta>0$ and $t>0$,

$$\mathrm{E}\sup_{f\in\pi_\zeta\circ\mathcal{F}^t} |R_\varphi(f) - R_{\varphi,n}(f)| \le c\zeta L_{\varphi,\zeta}\sqrt{\frac{(V+1)(t+1)\log_2[2(t+1)/\ln 2]}{n}}.$$

Also, for any $\delta>0$, with probability at least $1-\delta$,

$$\begin{aligned}\sup_{f\in\pi_\zeta\circ\mathcal{F}^t} |R_\varphi(f) - R_{\varphi,n}(f)| &\le c\zeta L_{\varphi,\zeta}\sqrt{\frac{(V+1)(t+1)\log_2[2(t+1)/\ln 2]}{n}} \\ &+ M_{\varphi,\zeta}\sqrt{\frac{\ln(1/\delta)}{2n}}.\end{aligned}$$

Proof The proof of this lemma is similar to the proof of Lugosi and Vayatis (2004, Lemma 2) in that we begin with symmetrization followed by the application of the "contraction principle". We use symmetrization to get

$$\mathrm{E}\sup_{f\in\pi_\zeta\circ\mathcal{F}^t} |R_\varphi(f) - R_{\varphi,n}(f)| \le 2\mathrm{E}\sup_{f\in\pi_\zeta\circ\mathcal{F}^t} \left|\frac{1}{n}\sum_{i=1}^n \sigma_i(\varphi(-Y_i f(X_i)) - \varphi(0))\right|,$$

where σ_i are i.i.d. with $\mathbf{P}(\sigma_i = 1) = \mathbf{P}(\sigma_i = -1) = 1/2$. Then we use the "contraction principle" (see Ledoux and Talagrand, 1991, Theorem 4.12, pp. 112–113) with a function $\psi(x) = (\varphi(x) - \varphi(0))/L_{\varphi,\zeta}$ to get

$$\begin{aligned} \mathrm{E} \sup_{f \in \pi_\zeta \circ \mathcal{F}^t} |R_\varphi(f) - R_{\varphi,n}(f)| &\leq 4L_{\varphi,\zeta} \mathrm{E} \sup_{f \in \pi_\zeta \circ \mathcal{F}^t} \left| \frac{1}{n} \sum_{i=1}^n -\sigma_i Y_i f(X_i) \right| \\ &= 4L_{\varphi,\zeta} \mathrm{E} \sup_{f \in \pi_\zeta \circ \mathcal{F}^t} \left| \frac{1}{n} \sum_{i=1}^n \sigma_i f(X_i) \right|. \end{aligned}$$

Next we proceed and find the supremum. Notice, that functions in $\pi_\zeta \circ \mathcal{F}^t$ are bounded and clipped to absolute value equal ζ, therefore we can rescale $\pi_\zeta \circ \mathcal{F}^t$ by $(2\zeta)^{-1}$ and get

$$\mathrm{E} \sup_{f \in \pi_\zeta \circ \mathcal{F}^t} \left| \frac{1}{n} \sum_{i=1}^n \sigma_i f(X_i) \right| = 2\zeta \mathrm{E} \sup_{f \in (2\zeta)^{-1} \circ \pi_\zeta \circ \mathcal{F}^t} \left| \frac{1}{n} \sum_{i=1}^n \sigma_i f(X_i) \right|.$$

Next, we use Dudley's entropy integral (Dudley, 1999) to bound the right hand side above

$$\mathrm{E} \sup_{f \in (2\zeta)^{-1} \circ \pi_\zeta \circ \mathcal{F}^t} \left| \frac{1}{n} \sum_{i=1}^n \sigma_i f(X_i) \right| \leq \frac{12}{\sqrt{n}} \int_0^\infty \sqrt{\ln \mathcal{N}(\varepsilon, (2\zeta)^{-1} \circ \pi_\zeta \circ \mathcal{F}^t, L_2(P_n))} d\varepsilon.$$

Since, for $\varepsilon > 1$, the covering number $\mathcal{N}$ is 1, the upper integration limit can be taken as 1, and we can use Pollard's bound (Pollard, 1990) for $F \subseteq [0,1]^X$,

$$\mathcal{N}(\varepsilon, F, L_2(P)) \leq 2 \left(\frac{4e}{\varepsilon^2} \right)^{d_P(F)},$$

where $d_P(F)$ is a pseudo-dimension, and obtain for $\tilde{c} = 12 \int_0^1 \sqrt{\ln \frac{8e}{\varepsilon^2}} d\varepsilon$,

$$\mathrm{E} \sup_{f \in (2\zeta)^{-1} \circ \pi_\zeta \circ \mathcal{F}^t} \left| \frac{1}{n} \sum_{i=1}^n \sigma_i f(X_i) \right| \leq \tilde{c} \sqrt{\frac{d_P((2\zeta)^{-1} \circ \pi_\zeta \circ \mathcal{F}^t)}{n}}.$$

Also notice that constant $\tilde{c}$ does not depend on $\mathcal{F}^t$ or ζ. Next, since $(2\zeta)^{-1} \circ \pi_\zeta$ is non-decreasing, we use the inequality $d_P((2\zeta)^{-1} \circ \pi_\zeta \circ \mathcal{F}^t) \leq d_P(\mathcal{F}^t)$ (for example, Anthony and Bartlett, 1999, Theorem 11.3) to obtain

$$\mathrm{E} \sup_{f \in (2\zeta)^{-1} \circ \pi_\zeta \circ \mathcal{F}^t} \left| \frac{1}{n} \sum_{i=1}^n \sigma_i f(X_i) \right| \leq c \sqrt{\frac{d_P(\mathcal{F}^t)}{n}}.$$

And then, since Lemma 3 gives an upper-bound on the pseudo-dimension of the class $\mathcal{F}^t$, we have

$$\mathrm{E} \sup_{f \in \pi_\zeta \circ \mathcal{F}^t} \left| \frac{1}{n} \sum_{i=1}^n \sigma_i f(X_i) \right| \leq c\zeta \sqrt{\frac{(V+1)(t+1)\log_2[2(t+1)/\ln 2]}{n}},$$

with the constant c above being independent of $\mathcal{H}$, t and ζ. To prove the second statement we use McDiarmid's bounded difference inequality (Devroye et al., 1996, Theorem 9.2, p. 136), since for all $i \in \{1,\dots,n\}$

$$\sup_{(x_j,y_j)_{j=1}^n,(x_i',y_i')} \left| \sup_{f\in\pi_\zeta\circ\mathcal{F}^t} |R_\varphi(f) - R_{\varphi,n}(f)| - \sup_{f\in\pi_\zeta\circ\mathcal{F}^t} |R_\varphi(f) - R'_{\varphi,n}(f)| \right| \le \frac{M_{\varphi,\zeta}}{n},$$

where $R'_{\varphi,n}(f)$ is obtained from $R_{\varphi,n}(f)$ by changing each pair (x_i,y_i) to an independent pair (x_i',y_i'). This completes the proof of the lemma. ■

Appendix C. Proof of Theorem 6

For convenience, we state the theorem once again.

Theorem 6 *Let the function $Q(f)$ be convex in f and twice differentiable in all directions $h \in \mathcal{H}$. Let $Q^\star = \lim_{\lambda\to\infty} \inf_{f\in\mathcal{F}_\lambda} Q(f)$. Assume that $\forall c_1, c_2$, such that $Q^\star < c_1 < c_2 < \infty$,*

$$\begin{aligned} 0 &< \inf\{Q''(f;h) : c_1 < Q(f) < c_2, h \in \mathcal{H}\} \\ &\le \sup\{Q''(f;h) : Q(f) < c_2, h \in \mathcal{H}\} < \infty. \end{aligned}$$

Also assume the following approximate minimization scheme for $\gamma \in (0,1]$. Define $f_{k+1} = f_k + \alpha_{k+1}h_{k+1}$ such that

$$Q(f_{k+1}) \le \gamma \inf_{h\in\mathcal{H},\alpha\in\mathbb{R}} Q(f_k + \alpha h) + (1-\gamma)Q(f_k)$$

and

$$Q(f_{k+1}) = \inf_{\alpha\in\mathbb{R}} Q(f_k + \alpha h_{k+1}).$$

Then for any reference function $\bar{f}$ and the sequence of functions f_m, produced by the boosting algorithm, the following bound holds $\forall m > 0$ such that $Q(f_m) > Q(\bar{f})$.

$$Q(f_m) \le Q(\bar{f}) + \sqrt{\frac{8B^3(Q(f_0) - Q(\bar{f}))^2}{\gamma^2\beta^3}} \left(\ln \frac{\ell_0^2 + c_3 m}{\ell_0^2} \right)^{-\frac{1}{2}},$$

where $\ell_k = \left\| \bar{f} - f_k \right\|_\star$, $c_3 = 2(Q(f_0) - Q(\bar{f}))/\beta$, $\beta = \inf\{Q''(f;h) : Q(\bar{f}) < Q(f) < Q(f_0), h \in \mathcal{H}\}$, $B = \sup\{Q''(f;h) : Q(f) < Q(f_0), h \in \mathcal{H}\}$.

Proof The statement of the theorem is a version of a result implicit in the proof of Theorem 1 by Bickel et al. (2006). If for some m we have $Q(f_m) \le Q(\bar{f})$, then the theorem is trivially true for all $m' \ge m$. Therefore, we are going to consider only the case when $Q(f_m) > Q(\bar{f})$. We shall also assume $Q(f_{m+1}) \ge Q(\bar{f})$ (the impact of this assumption will be discussed later). Define $\varepsilon_m = Q(f_m) - Q(\bar{f})$. By convexity of $Q(\cdot)$,

$$|Q'(f_m; f_m - \bar{f})| \ge \varepsilon_m. \tag{20}$$

Let $f_m - \bar{f} = \sum \tilde{\alpha}_i \tilde{h}_i$, where $\tilde{\alpha}_i$ and $\tilde{h}_i$ correspond to the best representation (with the l_1-norm of $\tilde{\alpha}$ equal the $l_\star$-norm). Then from (20) and linearity of the derivative we have

$$\varepsilon_m \le \left| \sum \tilde{\alpha}_i Q'(f_m; \tilde{h}_i) \right| \le \sup_{h\in\mathcal{H}} |Q'(f_m;h)| \sum |\tilde{\alpha}_i|,$$

therefore

$$\sup_{h\in\mathcal{H}} Q'(f_m;h) \geq \frac{\varepsilon_m}{\left\|f_m-\bar{f}\right\|_\star} = \frac{\varepsilon_m}{\ell_m}. \tag{21}$$

Next,

$$Q(f_m+\alpha h_m) = Q(f_m)+\alpha Q'(f_m;h_m)+\frac{1}{2}\alpha^2 Q''(\tilde{f}_m;h_m),$$

where $\tilde{f}_m = f_m+\tilde{\alpha}_m h_m$, for $\tilde{\alpha}_m \in [0,\alpha_m]$. By assumption $\tilde{f}_m$ is on the path from f_m to f_{m+1}, and we have assumed exact minimization in the given direction, hence f_{m+1} is the lowest point in the direction h_m starting from f_m, so we have the following bounds

$$Q(\bar{f}) < Q(f_{m+1}) \leq Q(\tilde{f}_m) \leq Q(f_m) \leq Q(f_0).$$

Then by the definition of β, which depends on $Q(\bar{f})$, we have

$$Q(f_{m+1}) \geq Q(f_m)+\inf_{\alpha\in\mathbb{R}}(\alpha Q'(f_m;h_m)+\frac{1}{2}\alpha^2\beta) = Q(f_m)-\frac{|Q'(f_m;h_m)|^2}{2\beta}. \tag{22}$$

On the other hand,

$$\begin{aligned} Q(f_m+\alpha_m h_m) &\leq \gamma \inf_{h\in\mathcal{H},\alpha\in\mathbb{R}} Q(f_m+\alpha h)+(1-\gamma)Q(f_m) \\ &\leq \gamma \inf_{h\in\mathcal{H},\alpha\in\mathbb{R}} \left(Q(f_m)+\alpha Q'(f_m;h)+\frac{1}{2}\alpha^2 B)\right)+(1-\gamma)Q(f_m) \\ &= Q(f_m)-\gamma\frac{\sup_{h\in\mathcal{H}}|Q'(f_m;h)|^2}{2B}. \end{aligned} \tag{23}$$

Therefore, combining (22) and (23), we get

$$|Q'(f_m;h_m)| \geq \sup_{h\in\mathcal{H}}|Q'(f_m;h)|\sqrt{\frac{\gamma\beta}{B}}. \tag{24}$$

Another Taylor expansion, this time around f_{m+1} (and we again use the fact that f_{m+1} is the minimum on the path from f_m), gives us

$$Q(f_m) = Q(f_{m+1})+\frac{1}{2}\alpha_m^2 Q''(\tilde{\tilde{f}}_m;h_m), \tag{25}$$

where $\tilde{\tilde{f}}_m$ is some (other) function on the path from f_m to f_{m+1}. Therefore, if $|\alpha_m| < \sqrt{\gamma}|Q'(f_m;h_m)|/B$, then

$$Q(f_m)-Q(f_{m+1}) < \frac{\gamma|Q'(f_m;h_m)|^2}{2B},$$

but by (23)

$$Q(f_m)-Q(f_{m+1}) \geq \frac{\gamma\sup_{h\in\mathcal{H}}|Q'(f_m;h)|^2}{2B} \geq \frac{\gamma|Q'(f_m;h_m)|^2}{2B},$$

therefore we conclude, by combining (24) and (21), that

$$|\alpha_m| \geq \frac{\sqrt{\gamma}|Q'(f_m;h_m)|}{B} \geq \frac{\gamma\sqrt{\beta}\sup_{h\in\mathcal{H}}|Q'(f_m;h)|}{B^{3/2}} \geq \frac{\gamma\varepsilon_m\sqrt{\beta}}{\ell_m B^{3/2}}. \tag{26}$$

Using (25) we have

$$\sum_{i=0}^{m} \alpha_i^2 \le \frac{2}{\beta} \sum_{i=0}^{m} (Q(f_i) - Q(f_{i+1})) \le \frac{2}{\beta}(Q(f_0) - Q(\bar{f})). \tag{27}$$

Recall that

$$\begin{aligned} \left\|f_m - \bar{f}\right\|_\star &\le \left\|f_{m-1} - \bar{f}\right\|_\star + |\alpha_{m-1}| \le \left\|f_0 - \bar{f}\right\|_\star + \sum_{i=0}^{m-1} |\alpha_i| \\ &\le \left\|f_0 - \bar{f}\right\|_\star + \sqrt{m} \left(\sum_{i=0}^{m-1} \alpha_i^2 \right)^{1/2}, \end{aligned}$$

therefore, combining with (27) and (26), since the sequence ε_i is decreasing,

$$\begin{aligned} \frac{2}{\beta}(Q(f_0) - Q(\bar{f})) &\ge \sum_{i=0}^{m} \alpha_i^2 \\ &\ge \frac{\gamma^2 \beta}{B^3} \sum_{i=0}^{m} \frac{\varepsilon_i^2}{\ell_i^2} \\ &\ge \frac{\gamma^2 \beta}{B^3} \varepsilon_m^2 \sum_{i=0}^{m} \frac{1}{\left(\ell_0 + \sqrt{i} \left(\sum_{j=0}^{i-1} \alpha_j^2 \right)^{1/2} \right)^2} \\ &\ge \frac{\gamma^2 \beta}{B^3} \varepsilon_m^2 \sum_{i=0}^{m} \frac{1}{\left(\ell_0 + \sqrt{i} \left(\frac{2(Q(f_0) - Q(\bar{f}))}{\beta} \right)^{1/2} \right)^2} \\ &\ge \frac{\gamma^2 \beta}{2B^3} \varepsilon_m^2 \sum_{i=0}^{m} \frac{1}{\ell_0^2 + \frac{2(Q(f_0) - Q(\bar{f}))}{\beta} i}. \end{aligned}$$

Since

$$\sum_{i=0}^{m} \frac{1}{a+bi} \ge \int_0^{m+1} \frac{dx}{a+bx} = \frac{1}{b} \ln \frac{a + b(m+1)}{a},$$

then

$$\frac{2}{\beta}(Q(f_0) - Q(\bar{f})) \ge \frac{\gamma^2 \beta^2}{4B^3 (Q(f_0) - Q(\bar{f}))} \varepsilon_m^2 \ln \frac{\ell_0^2 + \frac{2(Q(f_0) - Q(\bar{f}))}{\beta}(m+1)}{\ell_0^2}.$$

Therefore

$$\varepsilon_m \le \sqrt{\frac{8B^3 (Q(f_0) - Q(\bar{f}))^2}{\gamma^2 \beta^3}} \left(\ln \frac{\ell_0^2 + \frac{2(Q(f_0) - Q(\bar{f}))}{\beta}(m+1)}{\ell_0^2} \right)^{-\frac{1}{2}}. \tag{28}$$

The proof of the above inequality for index m works as long as $Q(f_{m+1}) \ge Q(\bar{f})$. If $\bar{f}$ is such that $Q(f_m) \ge Q(\bar{f})$ for all m, then we do not need to do anything else. However, if there exists m' such that $Q(f_{m'}) < Q(\bar{f})$ and $Q(f_{m'-1}) \ge Q(\bar{f})$, then the above proof is not valid for index $m' - 1$. To overcome this difficulty, we notice that $Q(f_{m'-1})$ is bounded from above by $Q(f_{m'-2})$, therefore to get a bound that holds for all m (except for $m = 0$) we may use a bound for ε_{m-1} to bound

$Q(f_m) - Q(\bar{f}) = \varepsilon_m$: shift (decrease) the index m on the right hand side of (28) by one. This completes the proof of the theorem. ■

Appendix D. Zero Bayes Risk

Here we consider a modification of Theorem 8. In this case our assumptions imply that $R^\star = 0$, and the proof presented above does not work. However for AdaBoost we can modify the proof appropriately to show an adequate convergence rate.

Theorem 13 *Assume $R^\star = 0$. Let t_n be a number of steps we run AdaBoost. Let $\lambda_n = \kappa \ln\ln n$ for $\kappa \in (0, 1/6)$. Let $\varepsilon_n = n^{-\nu}$, for $\nu \in (0, 1/2)$. Then with probability at least $1 - \delta_n$, where $\delta_n = \exp\left(-2n^{1-2\nu}/(\ln n)^{2\kappa}\right)$, for some constant C that depends on $\mathcal{H}$ and $\mathcal{P}$ but does not depend on n, for n such that*

$$\frac{C}{(\ln n)^\kappa} > \frac{2}{n^\nu}$$

the following holds

$$\begin{aligned} R_n(f_{t_n}) &\leq R_n(\bar{f}_n) \\ &+ \sqrt{\frac{16 R_n^3(f_0)|R_n(f_0) - R_n(\bar{f}_n)|^2 (\ln n)^{3\kappa}}{C\gamma^2}} \\ &\times \left(\ln \frac{(\kappa \ln\ln n)^2 + 4|R_n(f_0) - R_n(\bar{f}_n)|(\ln n)^\kappa t_n / C}{(\kappa \ln\ln n)^2}\right)^{-1/2}. \end{aligned}$$

Proof For the exponential loss assumption $R^\star = 0$ is equivalent to $L^\star = 0$. It also implies that the fastest decrease rate of the function $\tau : \lambda \to \inf_{f \in \mathcal{F}_\lambda} R(f)$ is $O(e^{-\lambda})$. To see this, assume that for some λ there exists $f \in \mathcal{F}_\lambda$ such that $L(g(f)) = 0$ (i.e., we have achieved perfect classification). Clearly, for any $a > 0$

$$R(af) = \mathrm{E} e^{-Yaf(X)} = \mathrm{E}\left(e^{-Yf(X)}\right)^a \geq (\inf_{x,y} e^{-yf(x)})^a.$$

Therefore, choose $\lambda_n = \kappa \ln\ln n$. Then $\inf_{f \in \mathcal{F}_{\lambda_n}} R(f) \geq C(\ln n)^{-\kappa}$, where C depends on $\mathcal{H}$ and $\mathcal{P}$, but does not depend on n. On the other hand Hoeffding's inequality for $\bar{f}_n \in \mathcal{F}_{\lambda_n}$ guarantees that

$$\mathbf{P}\left(R(\bar{f}_n) - R_n(\bar{f}_n) \geq \varepsilon_n\right) \leq \exp\left(-2n\varepsilon_n^2/(\ln n)^{2\kappa}\right) = \delta_n.$$

Choice of $\varepsilon_n = n^{-\nu}$ for $\nu \in (0, 1/2)$ ensures that $\delta_n \to 0$. This allows to conclude that with probability at least $1 - \delta_n$ empirical risk $R_n(\bar{f}_n)$ can be lower bounded as

$$R_n(\bar{f}_n) \geq R(\bar{f}_n) - \varepsilon_n$$

and for n large enough for

$$\frac{C}{(\ln n)^\kappa} > \frac{2}{n^\nu}$$

to hold we get a lower bound on β in (11) of Theorem 6 as

$$\beta \geq \frac{C}{2(\ln n)^{\kappa}}.$$

Since for $\bar{f}_n$ such that $R_n(\bar{f}_n) > R_n(f_0)$ theorem trivially holds, we only have to plug $R_n(\bar{f}_n) = 0$, $B = R_n(f_0)$ and $\beta = C(\ln n)^{\kappa}/2$ into (11) to get the statement of the theorem. Obviously, this bound holds for $R^{\star} > 0$. ■

Appendix E.

Lemma 14 *Let the function $\varphi : \mathbb{R} \to \mathbb{R}_+ \cup \{0\}$ be convex. Then for any $\lambda > 0$*

$$\varphi(\pi_\lambda(x)) \leq \varphi(x) + \inf_{z \in [-\lambda, \lambda]} \varphi(z). \tag{29}$$

Proof If $x \in [-\lambda, \lambda]$ then the statement of the lemma is clearly true. Without loss of generality assume $x > \lambda$; case $x < -\lambda$ is similar. Then we have two possibilities.

1. $\varphi(x) \geq \varphi(\lambda) = \varphi(\pi_\lambda(x))$ and (29) is obvious.
2. $\varphi(x) < \varphi(\lambda)$. Due to convexity, for any $z < \lambda$ we have $\varphi(z) > \varphi(\lambda)$, therefore

$$\varphi(\pi_\lambda(x)) = \varphi(\lambda) \leq \varphi(\lambda) + \varphi(x) = \inf_{z \in [-\lambda, \lambda]} \varphi(z) + \varphi(x).$$

The statement of the lemma is proven. ■

References

Martin Anthony and Peter L. Bartlett. *Neural Network Learning: Theoretical Foundations*. Cambridge University Press, 1999.

Peter L. Bartlett, Michael I. Jordan, and Jon D. McAuliffe. Discussion of boosting papers. *The Annals of Statistics*, 32(1):85–91, 2004.

Peter L. Bartlett, Michael I. Jordan, and Jon D. McAuliffe. Convexity, classification, and risk bounds. *Journal of the American Statistical Association*, 101(473):138–156, 2006.

Eric Bauer and Ron Kohavi. An empirical comparison of voting classification algorithms: Bagging, boosting and variants. *Machine Learning*, 36:105–139, 1999.

Peter J. Bickel, Ya'acov Ritov, and Alon Zakai. Some theory for generalized boosting algorithms. *Journal of Machine Learning Research*, 7:705–732, May 2006.

Gilles Blanchard, Gábor Lugosi, and Nicolas Vayatis. On the rate of convergence of regularized boosting classifiers. *Journal of Machine Learning Research*, 4:861–894, 2003.

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

Leo Breiman. Arcing the edge. Technical Report 486, Department of Statistics, University of California, Berkeley, 1997.

Leo Breiman. Prediction games and arcing algorithms. *Neural Computation*, 11:1493–1517, 1999. (Was Department of Statistics, U.C. Berkeley Technical Report 504, 1997).

Leo Breiman. Arcing classifiers (with discussion). *The Annals of Statistics*, 26(3):801–849, 1998. (Was Department of Statistics, U.C. Berkeley Technical Report 460, 1996).

Leo Breiman. Population theory for predictor ensembles. *The Annals of Statistics*, 32(1):1–11, 2004. (See also Department of Statistics, U.C. Berkeley Technical Report 579, 2000).

Luc Devroye, László Györfi, and Gábor Lugosi. *A Probabilistic Theory of Pattern Recognition*. Springer, New York, 1996.

Thomas G. Dietterich. An experimental comparison of three methods for constructing ensembles of decision trees: bagging, boosting, and randomization. *Machine Learning*, 40(2):139–158, 2000.

Harris Drucker and Corinna Cortes. Boosting decision trees. In D.S. Touretzky, M.C. Mozer, and M.E. Hasselmo, editors, *Advances in Neural Information Processing Systems 8*, pages 479–485. M.I.T. Press, 1996.

Richard M. Dudley. *Uniform Central Limit Theorems*. Cambridge University Press, Cambridge, MA, 1999.

Yoav Freund. Boosting a weak learning algorithm by majority. *Information and Computation*, 121: 256–285, 1995.

Yoav Freund and Robert E. Schapire. Experiments with a new boosting algorithm. In *13th International Conference on Machine Learning*, pages 148–156, San Francisco, 1996. Morgan Kaufman.

Yoav Freund and Robert E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, 1997.

Jerome Friedman, Trevor Hastie, and Robert Tibshirani. Additive logistic regression: A statistical view of boosting. *The Annals of Statistics*, 28:337–407, 2000.

Adam J. Grove and Dale Schuurmans. Boosting in the limit: Maximizing the margin of learned ensembles. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence*, pages 692–699, Menlo Park, CA, 1998. AAAI Press.

Wenxin Jiang. On weak base hypotheses and their implications for boosting regression and classification. *The Annals of Statistics*, 30:51–73, 2002.

Wenxin Jiang. Process consistency for AdaBoost. *The Annals of Statistics*, 32(1):13–29, 2004.

Vladimir Koltchinskii and Dmitry Panchenko. Empirical margin distributions and bounding the generalization error of combined classifiers. *The Annals of Statistics*, 30:1–50, 2002.

Michel Ledoux and Michel Talagrand. *Probability in Banach Spaces*. Springer-Verlag, New York, 1991.

Gábor Lugosi and Nicolas Vayatis. On the Bayes-risk consistency of regularized boosting methods. *The Annals of Statistics*, 32(1):30–55, 2004.

Shie Mannor and Ron Meir. Weak learners and improved rates of convergence in boosting. In *Advances in Neural Information Processing Systems*, 13, pages 280–286, 2001.

Shie Mannor, Ron Meir, and Tong Zhang. Greedy algorithms for classification – consistency, convergence rates, and adaptivity. *Journal of Machine Learning Research*, 4:713–742, 2003.

Llew Mason, Jonathan Baxter, Peter L. Bartlett, and Marcus Frean. Boosting algorithms as gradient descent. In S.A. Solla, T.K. Leen, and K.-R. Muller, editors, *Advances in Neural Information Processing Systems*, 12, pages 512–518. MIT Press, 2000.

David Pollard. *Convergence of Stochastic Processes*. Springer-Verlag, New York, 1984.

David Pollard. *Empirical Processes: Theory and Applications*. IMS, 1990.

J. Ross Quinlan. Bagging, boosting, and C4.5. In *13 AAAI Conference on Artificial Intelligence*, pages 725–730, Menlo Park, CA, 1996. AAAI Press.

Lev Reyzin and Robert E. Schapire. How boosting the margin can also boost classifier complexity. In *ICML '06: Proceedings of the 23rd international conference on Machine learning*, pages 753–760, New York, NY, USA, 2006. ACM Press. ISBN 1-59593-383-2. doi: http://doi.acm.org/10.1145/1143844.1143939.

Robert E. Schapire. The strength of weak learnability. *Machine Learning*, 5:197–227, 1990.

Robert E. Schapire, Yoav Freund, Peter L. Bartlett, and Wee Sun Lee. Boosting the margin: A new explanation for the effectiveness of voting methods. *The Annals of Statistics*, 26:1651–1686, 1998.

Xiaotong Shen, George C. Tseng, Xuegong Zhang, and Wing H. Wong. On ψ-learning. *Journal of the American Statistical Association*, 98(463):724–734, 2003.

Tong Zhang. Statistical behavior and consistency of classification methods based on convex risk minimization. *The Annals of Statistics*, 32(1):56–85, 2004.

Tong Zhang and Bin Yu. Boosting with early stopping: convergence and consistency. *The Annals of Statistics*, 33:1538–1579, 2005.

Journal of Machine Learning Research 8 (2007) 2369-2403 Submitted 4/07; Revised 8/07; Published 10/07

The On-Line Shortest Path Problem Under Partial Monitoring

András György GYA@SZIT.BME.HU
Machine Learning Research Group
Computer and Automation Research Institute
Hungarian Academy of Sciences
Kende u. 13-17, Budapest, Hungary, H-1111

Tamás Linder LINDER@MAST.QUEENSU.CA
Department of Mathematics and Statistics
Queen's University, Kingston, Ontario
Canada K7L 3N6

Gábor Lugosi GABOR.LUGOSI@GMAIL.COM
ICREA and Department of Economics
Universitat Pompeu Fabra
Ramon Trias Fargas 25-27
08005 Barcelona, Spain

György Ottucsák OTI@SZIT.BME.HU
Department of Computer Science and Information Theory
Budapest University of Technology and Economics
Magyar Tudósok Körútja 2.
Budapest, Hungary, H-1117

Editor: Leslie Pack Kaelbling

Abstract

The on-line shortest path problem is considered under various models of partial monitoring. Given a weighted directed acyclic graph whose edge weights can change in an arbitrary (adversarial) way, a decision maker has to choose in each round of a game a path between two distinguished vertices such that the loss of the chosen path (defined as the sum of the weights of its composing edges) be as small as possible. In a setting generalizing the multi-armed bandit problem, after choosing a path, the decision maker learns only the weights of those edges that belong to the chosen path. For this problem, an algorithm is given whose average cumulative loss in n rounds exceeds that of the best path, matched off-line to the entire sequence of the edge weights, by a quantity that is proportional to $1/\sqrt{n}$ and depends only polynomially on the number of edges of the graph. The algorithm can be implemented with complexity that is linear in the number of rounds n (i.e., the average complexity per round is constant) and in the number of edges. An extension to the so-called label efficient setting is also given, in which the decision maker is informed about the weights of the edges corresponding to the chosen path at a total of $m \ll n$ time instances. Another extension is shown where the decision maker competes against a time-varying path, a generalization of the problem of tracking the best expert. A version of the multi-armed bandit setting for shortest path is also discussed where the decision maker learns only the total weight of the chosen path but not the weights of the individual edges on the path. Applications to routing in packet switched networks along with simulation results are also presented.

Keywords: on-line learning, shortest path problem, multi-armed bandit problem

1. Introduction

In a sequential decision problem, a decision maker (or forecaster) performs a sequence of actions. After each action the decision maker suffers some loss, depending on the response (or state) of the environment, and its goal is to minimize its cumulative loss over a certain period of time. In the setting considered here, no probabilistic assumption is made on how the losses corresponding to different actions are generated. In particular, the losses may depend on the previous actions of the decision maker, whose goal is to perform well relative to a set of reference forecasters (the so-called "experts") for any possible behavior of the environment. More precisely, the aim of the decision maker is to achieve asymptotically the same average (per round) loss as the best expert.

Research into this problem started in the 1950s (see, for example, Blackwell, 1956 and Hannan, 1957 for some of the basic results) and gained new life in the 1990s following the work of Vovk (1990), Littlestone and Warmuth (1994), and Cesa-Bianchi et al. (1997). These results show that for any bounded loss function, if the decision maker has access to the past losses of all experts, then it is possible to construct on-line algorithms that perform, for any possible behavior of the environment, almost as well as the best of N experts. More precisely, the per round cumulative loss of these algorithms is at most as large as that of the best expert plus a quantity proportional to $\sqrt{\ln N/n}$ for any bounded loss function, where n is the number of rounds in the decision game. The logarithmic dependence on the number of experts makes it possible to obtain meaningful bounds even if the pool of experts is very large.

In certain situations the decision maker has only limited knowledge about the losses of all possible actions. For example, it is often natural to assume that the decision maker gets to know only the loss corresponding to the action it has made, and has no information about the loss it would have suffered had it made a different decision. This setup is referred to as the *multi-armed bandit problem*, and was considered, in the adversarial setting, by Auer et al. (2002) who gave an algorithm whose normalized regret (the difference of the algorithm's average loss and that of the best expert) is upper bounded by a quantity which is proportional to $\sqrt{N \ln N/n}$. Note that, compared to the *full information* case described above where the losses of all possible actions are revealed to the decision maker, there is an extra $\sqrt{N}$ factor in the performance bound, which seriously limits the usefulness of the bound if the number of experts is large.

Another interesting example for the limited information case is the so-called *label efficient decision problem* (see Helmbold and Panizza, 1997) in which it is too costly to observe the state of the environment, and so the decision maker can query the losses of all possible actions for only a limited number of times. A recent result of Cesa-Bianchi, Lugosi, and Stoltz (2005) shows that in this case, if the decision maker can query the losses m times during a period of length n, then it can achieve $O(\sqrt{\ln N/m})$ normalized regret relative to the best expert.

In many applications the set of experts has a certain structure that may be exploited to construct efficient on-line decision algorithms. The construction of such algorithms has been of great interest in computational learning theory. A partial list of works dealing with this problem includes Herbster and Warmuth (1998), Vovk (1999), Bousquet and Warmuth (2002), Schapire and Helmbold (1997), Takimoto and Warmuth (2003), Kalai and Vempala (2003) and György et al. (2004a,b, 2005a). For a more complete survey, we refer to Cesa-Bianchi and Lugosi (2006, Chapter 5).

In this paper we study the on-line shortest path problem, a representative example of structured expert classes that has received attention in the literature for its many applications, including, among others, routing in communication networks; see, for example, Takimoto and Warmuth (2003), Awer-

buch et al. (2005), or György and Ottucsák (2006), and adaptive quantizer design in zero-delay lossy source coding; see, György et al. (2004a,b, 2005b). In this problem, a weighted directed (acyclic) graph is given whose edge weights can change in an arbitrary manner, and the decision maker has to pick in each round a path between two given vertices, such that the weight of this path (the sum of the weights of its composing edges) be as small as possible.

Efficient solutions, with time and space complexity proportional to the number of edges rather than to the number of paths (the latter typically being exponential in the number of edges), have been given in the full information case, where in each round the weights of all the edges are revealed after a path has been chosen; see, for example, Mohri (1998), Takimoto and Warmuth (2003), Kalai and Vempala (2003), and György et al. (2005a).

In the bandit setting only the weights of the edges or just the sum of the weights of the edges composing the chosen path are revealed to the decision maker. If one applies the general bandit algorithm of Auer et al. (2002), the resulting bound will be too large to be of practical use because of its square-root-type dependence on the number of paths N. On the other hand, using the special graph structure in the problem, Awerbuch and Kleinberg (2004) and McMahan and Blum (2004) managed to get rid of the exponential dependence on the number of edges in the performance bound. They achieved this by extending the exponentially weighted average predictor and the follow-the-perturbed-leader algorithm of Hannan (1957) to the generalization of the multi-armed bandit setting for shortest paths, when only the sum of the weights of the edges is available for the algorithm. However, the dependence of the bounds obtained in Awerbuch and Kleinberg (2004) and McMahan and Blum (2004) on the number of rounds n is significantly worse than the $O(1/\sqrt{n})$ bound of Auer et al. (2002). Awerbuch and Kleinberg (2004) consider the model of "non-oblivious" adversaries for shortest path (i.e., the losses assigned to the edges can depend on the previous actions of the forecaster) and prove an $O(n^{-1/3})$ bound for the expected normalized regret. McMahan and Blum (2004) give a simpler algorithm than in Awerbuch and Kleinberg (2004) however obtain a bound of the order of $O(n^{-1/4})$ for the expected regret.

In this paper we provide an extension of the bandit algorithm of Auer et al. (2002) unifying the advantages of the above approaches, with a performance bound that is polynomial in the number of edges, and converges to zero at the right $O(1/\sqrt{n})$ rate as the number of rounds increases. We achieve this bound in a model which assumes that the losses of all edges on the path chosen by the forecaster are available separately after making the decision. We also discuss the case (considered by Awerbuch and Kleinberg, 2004 and McMahan and Blum, 2004) in which only the total loss (i.e., the sum of the losses on the chosen path) is known to the decision maker. We exhibit a simple algorithm which achieves an $O(n^{-1/3})$ normalized regret *with high probability* against "non-oblivious" adversary. In this case it remains an open problem to find an algorithm whose cumulative loss is polynomial in the number of edges of the graph and decreases as $O(n^{-1/2})$ with the number of rounds. Throughout the paper we assume that the number of rounds n in the prediction game is known in advance to the decision maker.

In Section 2 we formally define the on-line shortest path problem, which is extended to the multi-armed bandit setting in Section 3. Our new algorithm for the shortest path problem in the bandit setting is given in Section 4 together with its performance analysis. The algorithm is extended to solve the shortest path problem in a combined label efficient multi-armed bandit setting in Section 5. Another extension, when the algorithm competes against a time-varying path is studied in Section 6. An algorithm for the "restricted" multi-armed bandit setting (when only the sums

of the losses of the edges are available) is given in Section 7. Simulation results are presented in Section 8.

2. The Shortest Path Problem

Consider a network represented by a set of vertices connected by edges, and assume that we have to send a stream of packets from a distinguished vertex, called *source*, to another distinguished vertex, called *destination*. At each time slot a packet is sent along a chosen route connecting source and destination. Depending on the traffic, each edge in the network may have a different delay, and the total delay the packet suffers on the chosen route is the sum of delays of the edges composing the route. The delays may change from one time slot to the next one in an arbitrary way, and our goal is to find a way of choosing the route in each time slot such that the sum of the total delays over time is not significantly more than that of the best fixed route in the network. This adversarial version of the routing problem is most useful when the delays on the edges can change dynamically, even depending on our previous routing decisions. This is the situation in the case of ad-hoc networks, where the network topology can change rapidly, or in certain secure networks, where the algorithm has to be prepared to handle denial of service attacks, that is, situations where willingly malfunctioning vertices and links increase the delay; see, for example, Awerbuch et al. (2005).

This problem can be cast naturally as a sequential decision problem in which each possible route is represented by an action. However, the number of routes is typically exponentially large in the number of edges, and therefore computationally efficient algorithms are called for. Two solutions of different flavor have been proposed. One of them is based on a follow-the-perturbed-leader forecaster, see Kalai and Vempala (2003), while the other is based on an efficient computation of the exponentially weighted average forecaster, see, for example, Takimoto and Warmuth (2003). Both solutions have different advantages and may be generalized in different directions.

To formalize the problem, consider a (finite) directed acyclic graph with a set of edges $E = \{e_1, \ldots, e_{|E|}\}$ and a set of vertices V. Thus, each edge $e \in E$ is an ordered pair of vertices (v_1, v_2). Let u and v be two distinguished vertices in V. A *path* from u to v is a sequence of edges $e^{(1)}, \ldots, e^{(k)}$ such that $e^{(1)} = (u, v_1)$, $e^{(j)} = (v_{j-1}, v_j)$ for all $j = 2, \ldots, k-1$, and $e^{(k)} = (v_{k-1}, v)$. Let $\mathcal{P} = \{i_1, \ldots, i_N\}$ denote the set of all such paths. For simplicity, we assume that every edge in E is on some path from u to v and every vertex in V is an endpoint of an edge (see Figure 1 for examples).

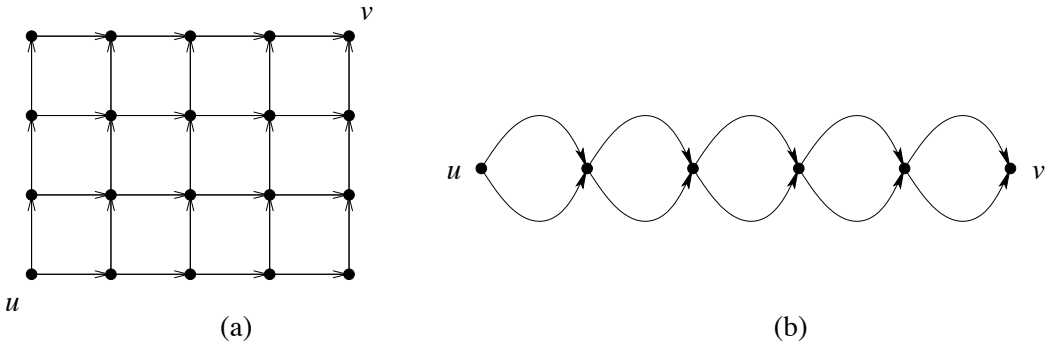


Figure 1: Two examples of directed acyclic graphs for the shortest path problem.

In each round $t = 1, \ldots, n$ of the decision game, the decision maker chooses a path I_t among all paths from u to v. Then a loss $\ell_{e,t} \in [0,1]$ is assigned to each edge $e \in E$. We write $e \in i$ if the edge $e \in E$ belongs to the path $i \in \mathcal{P}$, and with a slight abuse of notation the loss of a path i at time slot t is also represented by $\ell_{i,t}$. Then $\ell_{i,t}$ is given as

$$\ell_{i,t} = \sum_{e \in i} \ell_{e,t}$$

and therefore the cumulative loss up to time t of each path i takes the additive form

$$L_{i,t} = \sum_{s=1}^{t} \ell_{i,s} = \sum_{e \in i} \sum_{s=1}^{t} \ell_{e,s}$$

where the inner sum on the right-hand side is the loss accumulated by edge e during the first t rounds of the game. The cumulative loss of the algorithm is

$$\widehat{L}_t = \sum_{s=1}^{t} \ell_{I_s,s} = \sum_{s=1}^{t} \sum_{e \in I_s} \ell_{e,s} \ .$$

It is well known that for a general loss sequence, the decision maker must be allowed to use randomization to be able to approximate the performance of the best expert (see, e.g., Cesa-Bianchi and Lugosi, 2006). Therefore, the path I_t is chosen randomly according to some distribution p_t over all paths from u to v. We study the normalized regret over n rounds of the game

$$\frac{1}{n}\left(\widehat{L}_n - \min_{i \in \mathcal{P}} L_{i,n}\right)$$

where the minimum is taken over all paths i from u to v.

For example, the exponentially weighted average forecaster (Vovk, 1990; Littlestone and Warmuth, 1994; Cesa-Bianchi et al., 1997), calculated over all possible paths, has regret

$$\frac{1}{n}\left(\widehat{L}_n - \min_{i \in \mathcal{P}} L_{i,n}\right) \leq K\left(\sqrt{\frac{\ln N}{2n}} + \sqrt{\frac{\ln(1/\delta)}{2n}}\right)$$

with probability at least $1-\delta$, where N is the total number of paths from u to v in the graph and K is the length of the longest path.

3. The Multi-Armed Bandit Setting

In this section we discuss the "bandit" version of the shortest path problem. In this setup, which is more realistic in many applications, the decision maker has only access to the losses corresponding to the paths it has chosen. For example, in the routing problem this means that information is available on the delay of the route the packet is sent on, and not on other routes in the network.

We distinguish between two types of bandit problems, both of which are natural generalizations of the simple bandit problem to the shortest path problem. In the first variant, the decision maker has access to the losses of those edges that are on the path it has chosen. That is, after choosing a path I_t at time t, the value of the loss $\ell_{e,t}$ is revealed to the decision maker if and only if $e \in I_t$. We study this case and its extensions in Sections 4, 5, and 6.

The second variant is a more restricted version in which the loss of the chosen path is observed, but no information is available on the individual losses of the edges belonging to the path. That is, after choosing a path I_t at time t, only the value of the loss of the path $\ell_{I_t,t}$ is revealed to the decision maker. Further on we call this setting as the *restricted* bandit problem for shortest path. We consider this restricted problem in Section 7.

Formally, the on-line shortest path problem in the multi-armed bandit setting is described as follows: at each time instance $t = 1,\ldots,n$, the decision maker picks a path $I_t \in \mathcal{P}$ from u to v. Then the environment assigns loss $\ell_{e,t} \in [0,1]$ to each edge $e \in E$, and the decision maker suffers loss $\ell_{I_t,t} = \sum_{e\in I_t} \ell_{e,t}$. In the unrestricted case the losses $\ell_{e,t}$ are revealed for all $e \in I_t$, while in the restricted case only $\ell_{I_t,t}$ is revealed. Note that in both cases $\ell_{e,t}$ may depend on $I_1,\ldots,I_{t-1}$, the earlier choices of the decision maker.

For the basic multi-armed bandit problem, Auer et al. (2002) gave an algorithm, based on exponential weighting with a biased estimate of the gains combined with uniform exploration. Applying their algorithm to the on-line shortest path problem in the bandit setting results in a performance that can be bounded, for any $0 < \delta < 1$ and fixed time horizon n, with probability at least $1-\delta$, by

$$\frac{1}{n}\left(\widehat{L}_n - \min_{i\in\mathcal{P}} L_{i,n}\right) \le \frac{11K}{2}\sqrt{\frac{N\ln(N/\delta)}{n}} + \frac{K\ln N}{2n} \ .$$

(The constants follow from a slightly improved version; see Cesa-Bianchi and Lugosi (2006).)

However, for the shortest path problem this bound is unacceptably large because, unlike in the full information case, here the dependence on the number of all paths N is not merely logarithmic, while N is typically exponentially large in the size of the graph (as in the two simple examples of Figure 1). Note that this bound also holds for the restricted setting as only the total losses on the paths are used. In order to achieve a bound that does not grow exponentially with the number of edges of the graph, it is imperative to make use of the dependence structure of the losses of the different actions (i.e., paths). Awerbuch and Kleinberg (2004) and McMahan and Blum (2004) do this by extending low complexity predictors, such as the follow-the-perturbed-leader forecaster (Hannan, 1957; Kalai and Vempala, 2003) to the restricted bandit setting. However, in both cases the price to pay for the polynomial dependence on the number of edges is a worse dependence on the length n of the game.

4. A Bandit Algorithm for Shortest Paths

In this section we describe a variant of the bandit algorithm of Auer et al. (2002) which achieves the desired performance for the shortest path problem. The new algorithm uses the fact that when the losses of the edges of the chosen path are revealed, then this also provides some information about the losses of each path sharing common edges with the chosen path.

For each edge $e \in E$, and $t = 1,2,\ldots$, introduce the *gain* $g_{e,t} = 1 - \ell_{e,t}$, and for each path $i \in \mathcal{P}$, let the gain be the sum of the gains of the edges on the path, that is,

$$g_{i,t} = \sum_{e\in i} g_{e,t} \ .$$

The conversion from losses to gains is done in order to facilitate the subsequent performance analysis. This has technical reasons. For the ordinary bandit problem the regret bounds of the order of $O(\sqrt{n^{-1}N\log N})$ were proved based on gains by Auer et al. (2002) and it was only recently shown

by Allenberg et al. (2006) and Auer and Ottucsák (2006) that it is possible to achieve the same type of bound for an algorithm based on losses. However, we do not know how to convert the latter algorithm into one that is efficiently computable for the shortest path problem.

To simplify the conversion, we assume that each path $i \in \mathcal{P}$ is of the same length K for some $K > 0$. Note that although this assumption may seem to be restrictive at the first glance, from each acyclic directed graph (V,E) one can construct a new graph by adding at most $(K-2)(|V|-2)+1$ vertices and edges (with constant weight zero) to the graph without modifying the weights of the paths such that each path from u to v will be of length K, where K denotes the length of the longest path of the original graph. If the number of edges is quadratic in the number of vertices, the size of the graph is not increased substantially. We describe a simple algorithm to do this in the Appendix.

A main feature of the algorithm, shown in Figure 2, is that the gains are estimated for each edge and not for each path. This modification results in an improved upper bound on the performance with the number of edges in place of the number of paths. Moreover, using dynamic programming as in Takimoto and Warmuth (2003), the algorithm can be computed efficiently. Another important ingredient of the algorithm is that one needs to make sure that every edge is sampled sufficiently often. To this end, we introduce a set $\mathcal{C}$ of *covering paths* with the property that for each edge $e \in E$ there is a path $i \in \mathcal{C}$ such that $e \in i$. Observe that one can always find such a covering set of cardinality $|\mathcal{C}| \leq |E|$.

We note that the algorithm of Auer et al. (2002) is a special case of the algorithm below: For any multi-armed bandit problem with N experts, one can define a graph with two vertices u and v, and N directed edges from u to v with weights corresponding to the losses of the experts. The solution of the shortest path problem in this case is equivalent to that of the original bandit problem with choosing expert i if the corresponding edge is chosen. For this graph, our algorithm reduces to the original algorithm of Auer et al. (2002).

Note that the algorithm can be efficiently implemented using dynamic programming, similarly to Takimoto and Warmuth [28]. See the upcoming Theorem 2 for the formal statement.

The main result of the paper is the following performance bound for the shortest-path bandit algorithm. It states that the normalized regret of the algorithm, after n rounds of play, is, roughly, of the order of $K\sqrt{|E|\ln N/n}$ where $|E|$ is the number of edges of the graph, K is the length of the paths, and N is the total number of paths.

Theorem 1 *For any $\delta \in (0,1)$ and parameters $0 \leq \gamma < 1/2$, $0 < \beta \leq 1$, and $\eta > 0$ satisfying $2\eta K|\mathcal{C}| \leq \gamma$, the performance of the algorithm defined above can be bounded, with probability at least $1-\delta$, as*

$$\frac{1}{n}\left(\widehat{L}_n - \min_{i\in\mathcal{P}} L_{i,n}\right) \leq K\gamma + 2\eta K^2|\mathcal{C}| + \frac{K}{n\beta}\ln\frac{|E|}{\delta} + \frac{\ln N}{n\eta} + |E|\beta.$$

In particular, choosing $\beta = \sqrt{\frac{K}{n|E|}\ln\frac{|E|}{\delta}}$, $\gamma = 2\eta K|\mathcal{C}|$, and $\eta = \sqrt{\frac{\ln N}{4nK^2|\mathcal{C}|}}$ yields for all $n \geq \max\left\{\frac{K}{|E|}\ln\frac{|E|}{\delta}, 4|\mathcal{C}|\ln N\right\}$,

$$\frac{1}{n}\left(\widehat{L}_n - \min_{i\in\mathcal{P}} L_{i,n}\right) \leq 2\sqrt{\frac{K}{n}}\left(\sqrt{4K|\mathcal{C}|\ln N} + \sqrt{|E|\ln\frac{|E|}{\delta}}\right).$$

Parameters: real numbers $\beta > 0, 0 < \eta, \gamma < 1$.
Initialization: Set $w_{e,0} = 1$ for each $e \in E$, $\overline{w}_{i,0} = 1$ for each $i \in \mathcal{P}$, and $\overline{W}_0 = N$. For each round $t = 1, 2, \ldots$

(a) Choose a path I_t at random according to the distribution p_t on $\mathcal{P}$, defined by

$$p_{i,t} = \begin{cases} (1-\gamma)\frac{\overline{w}_{i,t-1}}{\overline{W}_{t-1}} + \frac{\gamma}{|\mathcal{C}|} & \text{if } i \in \mathcal{C} \\ (1-\gamma)\frac{\overline{w}_{i,t-1}}{\overline{W}_{t-1}} & \text{if } i \notin \mathcal{C}. \end{cases}$$

(b) Compute the probability of choosing each edge e as

$$q_{e,t} = \sum_{i:e\in i} p_{i,t} = (1-\gamma)\frac{\sum_{i:e\in i}\overline{w}_{i,t-1}}{\overline{W}_{t-1}} + \gamma\frac{|\{i \in \mathcal{C} : e \in i\}|}{|\mathcal{C}|}.$$

(c) Calculate the estimated gains

$$g'_{e,t} = \begin{cases} \frac{g_{e,t}+\beta}{q_{e,t}} & \text{if } e \in I_t \\ \frac{\beta}{q_{e,t}} & \text{otherwise.} \end{cases}$$

(d) Compute the updated weights

$$\begin{aligned} w_{e,t} &= w_{e,t-1}e^{\eta g'_{e,t}} \\ \overline{w}_{i,t} &= \prod_{e\in i} w_{e,t} = \overline{w}_{i,t-1}e^{\eta g'_{i,t}} \end{aligned}$$

where $g'_{i,t} = \sum_{e\in i} g'_{e,t}$, and the sum of the total weights of the paths

$$\overline{W}_t = \sum_{i\in\mathcal{P}} \overline{w}_{i,t}.$$

Figure 2: A bandit algorithm for shortest path problems

The proof of the theorem is based on the analysis of the original algorithm of Auer et al. (2002) with necessary modifications required to transform parts of the argument from paths to edges, and to use the connection between the gains of paths sharing common edges.

For the analysis we introduce some notation:

$$G_{i,n} = \sum_{t=1}^{n} g_{i,t} \qquad \text{and} \qquad G'_{i,n} = \sum_{t=1}^{n} g'_{i,t}$$

for each $i \in \mathcal{P}$ and

$$G_{e,n} = \sum_{t=1}^{n} g_{e,t} \qquad \text{and} \qquad G'_{e,n} = \sum_{t=1}^{n} g'_{e,t}$$

for each $e \in E$, and

$$\widehat{G}_n = \sum_{t=1}^{n} g_{I_t,t}.$$

Note that $g'_{e,t}$, $g'_{i,t}$, $G'_{e,n}$, and $G'_{i,n}$ are random variables that depend on I_t.

The following lemma, shows that the deviation of the true cumulative gain from the estimated cumulative gain is of the order of $\sqrt{n}$. The proof is a modification of Cesa-Bianchi and Lugosi (2006, Lemma 6.7).

Lemma 2 *For any* $\delta \in (0,1)$, $0 \leq \beta < 1$ *and* $e \in E$ *we have*

$$\mathbb{P}\left[G_{e,n} > G'_{e,n} + \frac{1}{\beta}\ln\frac{|E|}{\delta}\right] \leq \frac{\delta}{|E|} .$$

Proof Fix $e \in E$. For any $u > 0$ and $c > 0$, by the Chernoff bound we have

$$\mathbb{P}[G_{e,n} > G'_{e,n} + u] \leq e^{-cu}\mathbb{E}e^{c(G_{e,n}-G'_{e,n})} . \tag{1}$$

Letting $u = \ln(|E|/\delta)/\beta$ and $c = \beta$, we get

$$e^{-cu}\mathbb{E}e^{c(G_{e,n}-G'_{e,n})} = e^{-\ln(|E|/\delta)}\mathbb{E}e^{\beta(G_{e,n}-G'_{e,n})} = \frac{\delta}{|E|}\mathbb{E}e^{\beta(G_{e,n}-G'_{e,n})} ,$$

so it suffices to prove that $\mathbb{E}e^{\beta(G_{e,n}-G'_{e,n})} \leq 1$ for all n. To this end, introduce

$$Z_t = e^{\beta(G_{e,t}-G'_{e,t})} .$$

Below we show that $\mathbb{E}_t[Z_t] \leq Z_{t-1}$ for $t \geq 2$ where $\mathbb{E}_t$ denotes the conditional expectation $\mathbb{E}[\cdot|I_1,\ldots,I_{t-1}]$. Clearly,

$$Z_t = Z_{t-1}\exp\left(\beta\left(g_{e,t} - \frac{\mathbb{1}_{\{e\in I_t\}}g_{e,t}+\beta}{q_{e,t}}\right)\right) .$$

Taking conditional expectations, we obtain

$$\begin{aligned}
&\mathbb{E}_t[Z_t] \\
&= Z_{t-1}\mathbb{E}_t\left[\exp\left(\beta\left(g_{e,t} - \frac{\mathbb{1}_{\{e\in I_t\}}g_{e,t}+\beta}{q_{e,t}}\right)\right)\right] \\
&= Z_{t-1}e^{-\frac{\beta^2}{q_{e,t}}}\mathbb{E}_t\left[\exp\left(\beta\left(g_{e,t} - \frac{\mathbb{1}_{\{e\in I_t\}}g_{e,t}}{q_{e,t}}\right)\right)\right] \\
&\leq Z_{t-1}e^{-\frac{\beta^2}{q_{e,t}}}\mathbb{E}_t\left[1+\beta\left(g_{e,t} - \frac{\mathbb{1}_{\{e\in I_t\}}g_{e,t}}{q_{e,t}}\right)+\beta^2\left(g_{e,t} - \frac{\mathbb{1}_{\{e\in I_t\}}g_{e,t}}{q_{e,t}}\right)^2\right] &(2)\\
&= Z_{t-1}e^{-\frac{\beta^2}{q_{e,t}}}\mathbb{E}_t\left[1+\beta^2\left(g_{e,t} - \frac{\mathbb{1}_{\{e\in I_t\}}g_{e,t}}{q_{e,t}}\right)^2\right] &(3)\\
&\leq Z_{t-1}e^{-\frac{\beta^2}{q_{e,t}}}\mathbb{E}_t\left[1+\beta^2\left(\frac{\mathbb{1}_{\{e\in I_t\}}g_{e,t}}{q_{e,t}}\right)^2\right] \\
&\leq Z_{t-1}e^{-\frac{\beta^2}{q_{e,t}}}\left(1+\frac{\beta^2}{q_{e,t}}\right) \\
&\leq Z_{t-1}. &(4)
\end{aligned}$$

Here (2) holds since $\beta \leq 1$, $g_{e,t} - \frac{\mathbb{1}_{\{e\in I_t\}}g_{e,t}}{q_{e,t}} \leq 1$ and $e^x \leq 1+x+x^2$ for $x \leq 1$. (3) follows from $\mathbb{E}_t\left[\frac{\mathbb{1}_{\{e\in I_t\}}g_{e,t}}{q_{e,t}}\right] = g_{e,t}$. Finally, (4) holds by the inequality $1+x \leq e^x$. Taking expectations on both

sides proves $\mathbb{E}[Z_t] \le \mathbb{E}[Z_{t-1}]$. A similar argument shows that $\mathbb{E}[Z_1] \le 1$, implying $\mathbb{E}[Z_n] \le 1$ as desired. □

Proof of Theorem 1. As usual in the analysis of exponentially weighted average forecasters, we start with bounding the quantity $\ln \frac{\overline{W}_n}{\overline{W}_0}$. On the one hand, we have the lower bound

$$\ln \frac{\overline{W}_n}{\overline{W}_0} = \ln \sum_{i \in \mathcal{P}} e^{\eta G'_{i,n}} - \ln N \ge \eta \max_{i \in \mathcal{P}} G'_{i,n} - \ln N \,. \tag{5}$$

To derive a suitable upper bound, first notice that the condition $\eta \le \frac{\gamma}{2K|C|}$ implies $\eta g'_{i,t} \le 1$ for all i and t, since

$$\eta g'_{i,t} = \eta \sum_{e \in i} g'_{e,t} \le \eta \sum_{e \in i} \frac{1+\beta}{q_{e,t}} \le \frac{\eta K(1+\beta)|C|}{\gamma} \le 1$$

where the second inequality follows because $q_{e,t} \ge \gamma/|C|$ for each $e \in E$.
Therefore, using the fact that $e^x \le 1 + x + x^2$ for all $x \le 1$, for all $t = 1, 2, \ldots$ we have

$$\begin{aligned}
\ln \frac{\overline{W}_t}{\overline{W}_{t-1}} &= \ln \sum_{i \in \mathcal{P}} \frac{\overline{w}_{i,t-1}}{\overline{W}_{t-1}} e^{\eta g'_{i,t}} \\
&= \ln \left(\sum_{i \in \mathcal{P}} \frac{p_{i,t} - \frac{\gamma}{|C|} \mathbb{1}_{\{i \in C\}}}{1-\gamma} e^{\eta g'_{i,t}} \right) \qquad (6) \\
&\le \ln \left(\sum_{i \in \mathcal{P}} \frac{p_{i,t} - \frac{\gamma}{|C|} \mathbb{1}_{\{i \in C\}}}{1-\gamma} \left(1 + \eta g'_{i,t} + \eta^2 g'^2_{i,t} \right) \right) \\
&\le \ln \left(1 + \sum_{i \in \mathcal{P}} \frac{p_{i,t}}{1-\gamma} \left(\eta g'_{i,t} + \eta^2 g'^2_{i,t} \right) \right) \\
&\le \frac{\eta}{1-\gamma} \sum_{i \in \mathcal{P}} p_{i,t} g'_{i,t} + \frac{\eta^2}{1-\gamma} \sum_{i \in \mathcal{P}} p_{i,t} g'^2_{i,t} \qquad (7)
\end{aligned}$$

where (6) follows form the definition of $p_{i,t}$, and (7) holds by the inequality $\ln(1+x) \le x$ for all $x > -1$.

Next we bound the sums in (7). On the one hand,

$$\begin{aligned}
\sum_{i \in \mathcal{P}} p_{i,t} g'_{i,t} &= \sum_{i \in \mathcal{P}} p_{i,t} \sum_{e \in i} g'_{e,t} = \sum_{e \in E} g'_{e,t} \sum_{i \in \mathcal{P} : e \in i} p_{i,t} \\
&= \sum_{e \in E} g'_{e,t} q_{e,t} = g_{I_t,t} + |E|\beta.
\end{aligned}$$

On the other hand,

$$\begin{aligned}
\sum_{i\in\mathcal{P}} p_{i,t} g'^2_{i,t} &= \sum_{i\in\mathcal{P}} p_{i,t} \left(\sum_{e\in i} g'_{e,t}\right)^2 \\
&\leq \sum_{i\in\mathcal{P}} p_{i,t} K \sum_{e\in i} g'^2_{e,t} \\
&= K \sum_{e\in E} g'^2_{e,t} \sum_{i\in\mathcal{P}: e\in i} p_{i,t} \\
&= K \sum_{e\in E} g'^2_{e,t} q_{e,t} \\
&= K \sum_{e\in E} q_{e,t} g'_{e,t} \frac{\beta + \mathbb{1}_{\{e\in I_t\}} g_{e,t}}{q_{e,t}} \\
&\leq K(1+\beta) \sum_{e\in E} g'_{e,t}
\end{aligned}$$

where the first inequality is due to the inequality between the arithmetic and quadratic mean, and the second one holds because $g_{e,t} \leq 1$. Therefore,

$$\ln \frac{\overline{W}_t}{\overline{W}_{t-1}} \leq \frac{\eta}{1-\gamma} \left(g_{I_t,t} + |E|\beta\right) + \frac{\eta^2 K(1+\beta)}{1-\gamma} \sum_{e\in E} g'_{e,t} \,.$$

Summing for $t = 1, \ldots, n$, we obtain

$$\begin{aligned}
\ln \frac{\overline{W}_n}{\overline{W}_0} &\leq \frac{\eta}{1-\gamma} \left(\widehat{G}_n + n|E|\beta\right) + \frac{\eta^2 K(1+\beta)}{1-\gamma} \sum_{e\in E} G'_{e,n} \\
&\leq \frac{\eta}{1-\gamma} \left(\widehat{G}_n + n|E|\beta\right) + \frac{\eta^2 K(1+\beta)}{1-\gamma} |\mathcal{C}| \max_{i\in\mathcal{P}} G'_{i,n}
\end{aligned}$$

where the second inequality follows since $\sum_{e\in E} G'_{e,n} \leq \sum_{i\in\mathcal{C}} G'_{i,n}$. Combining the upper bound with the lower bound (5), we obtain

$$\widehat{G}_n \geq (1-\gamma-\eta K(1+\beta)|\mathcal{C}|) \max_{i\in\mathcal{P}} G'_{i,n} - \frac{1-\gamma}{\eta} \ln N - n|E|\beta.$$

Now using Lemma 2 and applying the union bound, for any $\delta \in (0,1)$ we have that, with probability at least $1-\delta$,

$$\widehat{G}_n \geq (1-\gamma-\eta K(1+\beta)|\mathcal{C}|) \left(\max_{i\in\mathcal{P}} G_{i,n} - \frac{K}{\beta} \ln \frac{|E|}{\delta}\right) - \frac{1-\gamma}{\eta} \ln N - n|E|\beta \,,$$

where we used $1-\gamma-\eta K(1+\beta)|\mathcal{C}| \geq 0$ which follows from the assumptions of the theorem. Since $\widehat{G}_n = Kn - \widehat{L}_n$ and $G_{i,n} = Kn - L_{i,n}$ for all $i \in \mathcal{P}$, we have

$$\begin{aligned}
\widehat{L}_n &\leq Kn(\gamma + \eta(1+\beta)K|\mathcal{C}|) + (1-\gamma-\eta(1+\beta)K|\mathcal{C}|) \min_{i\in\mathcal{P}} L_{i,n} \\
&\quad + (1-\gamma-\eta(1+\beta)K|\mathcal{C}|) \frac{K}{\beta} \ln \frac{|E|}{\delta} + \frac{1-\gamma}{\eta} \ln N + n|E|\beta
\end{aligned}$$

with probability at least $1-\delta$. This implies

$$\begin{aligned}\widehat{L}_n - \min_{i\in\mathcal{P}} L_{i,n} &\leq K n\gamma + \eta(1+\beta)nK^2|\mathcal{C}| + \frac{K}{\beta}\ln\frac{|E|}{\delta} + \frac{1-\gamma}{\eta}\ln N + n|E|\beta \\ &\leq K n\gamma + 2\eta nK^2|\mathcal{C}| + \frac{K}{\beta}\ln\frac{|E|}{\delta} + \frac{\ln N}{\eta} + n|E|\beta\end{aligned}$$

with probability at least $1-\delta$, which is the first statement of the theorem. Setting

$$\beta = \sqrt{\frac{K}{n|E|}\ln\frac{|E|}{\delta}} \qquad \text{and} \qquad \gamma = 2\eta K|\mathcal{C}|$$

results in the inequality

$$\widehat{L}_n - \min_{i\in\mathcal{P}} L_{i,n} \leq 4\eta nK^2|\mathcal{C}| + \frac{\ln N}{\eta} + 2\sqrt{nK|E|\ln\frac{|E|}{\delta}}$$

which holds with probability at least $1-\delta$ if $n \geq (K/|E|)\ln(|E|/\delta)$ (to ensure $\beta \leq 1$). Finally, setting

$$\eta = \sqrt{\frac{\ln N}{4nK^2|\mathcal{C}|}}$$

yields the last statement of the theorem ($n \geq 4\ln N|\mathcal{C}|$ is required to ensure $\gamma \leq 1/2$). □

Next we analyze the computational complexity of the algorithm. The next result shows that the algorithm is feasible as its complexity is linear in the size (number of edges) of the graph.

Theorem 3 *The proposed algorithm can be implemented efficiently with time complexity $O(n|E|)$ and space complexity $O(|E|)$.*

Proof The two complex steps of the algorithm are steps (a) and (b), both of which can be computed, similarly to Takimoto and Warmuth (2003), using dynamic programming. To perform these steps efficiently, first we order the vertices of the graph. Since we have an acyclic directed graph, its vertices can be labeled (in $O(|E|)$ time) from 1 to $|V|$ such that $u = 1$, $v = |V|$, and if $(v_1, v_2) \in E$, then $v_1 < v_2$. For any pair of vertices $u_1 < v_1$ let $\mathcal{P}_{u_1,v_1}$ denote the set of paths from u_1 to v_1, and for any vertex $s \in V$, let

$$H_t(s) = \sum_{i\in\mathcal{P}_{s,v}} \prod_{e\in i} w_{e,t}$$

and

$$\widehat{H}_t(s) = \sum_{i\in\mathcal{P}_{u,s}} \prod_{e\in i} w_{e,t} \ .$$

Given the edge weights $\{w_{e,t}\}$, $H_t(s)$ can be computed recursively for $s = |V|-1,\ldots,1$, and $\widehat{H}_t(s)$ can be computed recursively for $s = 2,\ldots,|V|$ in $O(|E|)$ time (letting $H_t(v) = \widehat{H}_t(u) = 1$ by definition). In step (a), first one has to decide with probability γ whether I_t is generated according to the graph weights, or it is chosen uniformly from $\mathcal{C}$. If I_t is to be drawn according to the graph weights, it can be shown that its vertices can be chosen one by one such that if the first k vertices

of I_t are $v_0 = u, v_1, \ldots, v_{k-1}$, then the next vertex of I_t can be chosen to be any $v_k > v_{k-1}$, satisfying $(v_{k-1}, v_k) \in E$, with probability $w_{(v_{k-1},v_k),t-1} H_{t-1}(v_k)/H_{t-1}(v_{k-1})$. The other computationally demanding step, namely step (b), can be performed easily by noting that for any edge (v_1, v_2),

$$\begin{aligned} q_{(v_1,v_2),t} &= (1-\gamma)\frac{\widehat{H}_{t-1}(v_1) w_{(v_1,v_2),t-1} H_{t-1}(v_2)}{H_{t-1}(u)} \\ &\quad + \gamma \frac{|\{i \in \mathcal{C} : (v_1,v_2) \in i\}|}{|\mathcal{C}|} \end{aligned}$$

as desired. □

5. A Combination of the Label Efficient and Bandit Settings

In this section we investigate a combination of the multi-armed bandit and the label efficient problems. This means that the decision maker only has access to the losses of all the edges on the chosen path upon request and the total number of requests must be bounded by a constant m. This combination is motivated by some applications, in which feedback information is costly to obtain.

In the general label efficient decision problem, after taking an action, the decision maker has the option to query the losses of all possible actions. For this problem, Cesa-Bianchi et al. (2005) proved an upper bound on the normalized regret of order $O(K\sqrt{\ln(4N/\delta)/(m)})$ which holds with probability at least $1-\delta$, where K is the length of the longest path in the graph.

Our model of the label-efficient bandit problem for shortest paths is motivated by an application to a particular packet switched network model. This model, called the cognitive packet network, was introduced by Gelenbe et al. (2004, 2001). In these networks a particular type of packets, called smart packets, are used to explore the network (e.g., the delay of the chosen path). These packets do not carry any useful data; they are merely used for exploring the network. The other type of packets are the data packets, which do not collect any information about their paths. The task of the decision maker is to send packets from the source to the destination over routes with minimum average transmission delay (or packet loss). In this scenario, smart packets are used to query the delay (or loss) of the chosen path. However, as these packets do not transport information, there is a tradeoff between the number of queries and the usage of the network. If data packets are on the average α times larger than smart packets (note that typically $\alpha \gg 1$) and ε is the proportion of time instances when smart packets are used to explore the network, then $\varepsilon/(\varepsilon + \alpha(1-\varepsilon))$ is the proportion of the bandwidth sacrificed for well informed routing decisions.

We study a combined algorithm which, at each time slot t, queries the loss of the chosen path with probability ε (as in the solution of the label efficient problem proposed in Cesa-Bianchi et al., 2005), and, similarly to the multi-armed bandit case, computes biased estimates $g'_{i,t}$ of the true gains $g_{i,t}$. Just as in the previous section, it is assumed that each path of the graph is of the same length K.

The algorithm differs from our bandit algorithm of the previous section only in step (c), which is modified in the spirit of Cesa-Bianchi et al. (2005). The modified step is given in Figure 3.

The performance of the algorithm is analyzed in the next theorem, which can be viewed as a combination of Theorem 1 in the preceding section and Theorem 2 of Cesa-Bianchi et al. (2005).

(c') Draw a Bernoulli random variable S_t with $\mathbb{P}(S_t = 1) = \varepsilon$, and compute the estimated gains

$$g'_{e,t} = \begin{cases} \frac{g_{e,t}+\beta}{\varepsilon q_{e,t}} S_t & \text{if } e \in I_t \\ \frac{\beta}{\varepsilon q_{e,t}} S_t & \text{if } e \notin I_t \end{cases} .$$

Figure 3: Modified step for the label efficient bandit algorithm for shortest paths

Theorem 4 *For any* $\delta \in (0,1)$, $\varepsilon \in (0,1]$, *parameters* $\eta = \sqrt{\frac{\varepsilon \ln N}{4nK^2|C|}}$, $\gamma = \frac{2\eta K|C|}{\varepsilon} \le 1/2$, *and* $\beta = \sqrt{\frac{K}{n|E|\varepsilon} \ln \frac{2|E|}{\delta}} \le 1$, *and for all*

$$n \ge \frac{1}{\varepsilon} \max \left\{ \frac{K^2 \ln^2(2|E|/\delta)}{|E| \ln N}, \frac{|E| \ln(2|E|/\delta)}{K}, 4|C| \ln N \right\}$$

the performance of the algorithm defined above can be bounded, with probability at least $1-\delta$, *as*

$$\begin{aligned} &\frac{1}{n}\left(\widehat{L}_n - \min_{i \in \mathcal{P}} L_{i,t}\right) \\ &\quad \le \sqrt{\frac{K}{n\varepsilon}} \left(4\sqrt{K|C| \ln N} + 5\sqrt{|E| \ln \frac{2|E|}{\delta}} + \sqrt{8K \ln \frac{2}{\delta}} \right) + \frac{4K}{3n\varepsilon} \ln \frac{2N}{\delta} \\ &\quad \le \frac{27K}{2} \sqrt{\frac{|E| \ln \frac{2N}{\delta}}{n\varepsilon}} . \end{aligned}$$

If ε is chosen as $(m - \sqrt{2m \ln(1/\delta)})/n$ then, with probability at least $1-\delta$, the total number of queries is bounded by m (Cesa-Bianchi and Lugosi, 2006, Lemma 6.1) and the performance bound above is of the order of $K\sqrt{|E| \ln(N/\delta)/m}$.

Similarly to Theorem 1, we need a lemma which reveals the connection between the true and the estimated cumulative losses. However, here we need a more careful analysis because the "shifting term" $\frac{\beta}{\varepsilon q_{e,t}} S_t$, is a random variable.

Lemma 5 *For any* $0 < \delta < 1$, $0 < \varepsilon \le 1$, *for any*

$$n \ge \frac{1}{\varepsilon} \max \left\{ \frac{K^2 \ln^2(2|E|/\delta)}{|E| \ln N}, \frac{K \ln(2|E|/\delta)}{|E|} \right\} ,$$

parameters

$$\frac{2\eta K|C|}{\varepsilon} \le \gamma, \qquad \eta = \sqrt{\frac{\varepsilon \ln N}{4nK^2|C|}} \quad \textit{and} \quad \beta = \sqrt{\frac{K}{n|E|\varepsilon} \ln \frac{2|E|}{\delta}} \le 1 ,$$

and $e \in E$, *we have*

$$\mathbb{P}\left[G_{e,n} > G'_{e,n} + \frac{4}{\beta\varepsilon} \ln \frac{2|E|}{\delta} \right] \le \frac{\delta}{2|E|} .$$

Proof Fix $e \in E$. Using (1) with $u = \frac{4}{\beta\varepsilon} \ln \frac{2|E|}{\delta}$ and $c = \frac{\beta\varepsilon}{4}$, it suffices to prove for all n that

$$\mathbb{E}\left[e^{c(G_{e,n} - G'_{e,n})}\right] \leq 1 \,.$$

Similarly to Lemma 2 we introduce $Z_t = e^{c(G_{e,t} - G'_{e,t})}$ and we show that $Z_1, \ldots, Z_n$ is a supermartingale, that is $\mathbb{E}_t[Z_t] \leq Z_{t-1}$ for $t \geq 2$ where $\mathbb{E}_t$ denotes $\mathbb{E}[\cdot|(I_1, S_1), \ldots, (I_{t-1}, S_{t-1})]$. Taking conditional expectations, we obtain

$$\begin{aligned}
\mathbb{E}_t[Z_t] &= Z_{t-1}\mathbb{E}_t\left[e^{c\left(g_{e,t} - \frac{\mathbb{1}_{\{e\in I_t\}} S_t g_{e,t} + S_t\beta}{q_{e,t}\varepsilon}\right)}\right] \\
&\leq Z_{t-1}\mathbb{E}_t\left[1 + c\left(g_{e,t} - \frac{\mathbb{1}_{\{e\in I_t\}} S_t g_{e,t} + S_t\beta}{q_{e,t}\varepsilon}\right)\right. \\
&\qquad \left. + c^2\left(g_{e,t} - \frac{\mathbb{1}_{\{e\in I_t\}} S_t g_{e,t} + S_t\beta}{q_{e,t}\varepsilon}\right)^2\right]. \qquad (8)
\end{aligned}$$

Since

$$\mathbb{E}_t\left[g_{e,t} - \frac{\mathbb{1}_{\{e\in I_t\}} S_t g_{e,t} + S_t\beta}{q_{e,t}\varepsilon}\right] = -\frac{\beta}{q_{e,t}}$$

and

$$\mathbb{E}_t\left[\left(g_{e,t} - \frac{\mathbb{1}_{\{e\in I_t\}} S_t g_{e,t}}{q_{e,t}\varepsilon}\right)^2\right] \leq \mathbb{E}_t\left[\left(\frac{\mathbb{1}_{\{e\in I_t\}} S_t g_{e,t}}{q_{e,t}\varepsilon}\right)^2\right] \leq \frac{1}{q_{e,t}\varepsilon}$$

we get from (8) that

$$\begin{aligned}
&\mathbb{E}_t[Z_t] \\
&\leq Z_{t-1}\mathbb{E}_t\left[1 - \frac{c\beta}{q_{e,t}} + \frac{c^2}{q_{e,t}\varepsilon} + c^2\left(\frac{2\mathbb{1}_{\{e\in I_t\}} S_t g_{e,t}\beta}{q_{e,t}^2\varepsilon^2} - \frac{2g_{e,t}S_t\beta}{q_{e,t}\varepsilon} + \frac{S_t\beta^2}{q_{e,t}^2\varepsilon^2}\right)\right] \\
&\leq Z_{t-1}\left(1 + \frac{c}{q_{e,t}}\left(-\beta + \frac{c}{\varepsilon} + c\beta\left(\frac{2}{\varepsilon} + \frac{\beta}{q_{e,t}\varepsilon}\right)\right)\right). \qquad (9)
\end{aligned}$$

Since $c = \beta\varepsilon/4$ we have

$$\begin{aligned}
-\beta + \frac{c}{\varepsilon} + c\beta\left(\frac{2}{\varepsilon} + \frac{\beta}{q_{e,t}\varepsilon}\right) &= -\frac{3\beta}{4} + \frac{\beta^2\varepsilon}{4}\left(\frac{2}{\varepsilon} + \frac{\beta}{q_{e,t}\varepsilon}\right) \\
&= -\frac{3\beta}{4} + \frac{\beta^2}{2} + \frac{\beta^3}{4q_{e,t}} \\
&\leq -\frac{\beta}{4} + \frac{\beta^3}{4q_{e,t}} \\
&\leq -\frac{\beta}{4} + \frac{\beta^3|C|}{4\gamma} \qquad (10) \\
&\leq 0, \qquad (11)
\end{aligned}$$

where (10) follows from $q_{e,t} \geq \frac{\gamma}{|\mathcal{C}|}$ and (11) holds since $\beta \leq 1$ and by

$$\frac{\beta^2 |\mathcal{C}|}{\gamma} \leq \frac{\beta^2 \varepsilon}{2\eta K} \leq 1 \ ,$$

and the last inequality is ensured by $n \geq \frac{K^2 \ln^2(2|E|/\delta)}{\varepsilon |E| \ln N}$, the assumption of the lemma.

Combining (9) and (11) we get that $\mathbb{E}_t[Z_t] \leq Z_{t-1}$. Taking expectations on both sides of the inequality, we get $\mathbb{E}[Z_t] \leq \mathbb{E}[Z_{t-1}]$ and since $\mathbb{E}[Z_1] \leq 1$, we obtain $\mathbb{E}[Z_n] \leq 1$ as desired. □

Proof of Theorem 4. The proof of the theorem is a generalization of that of Theorem 1, and follows the same lines with some extra technicalities to handle the effects of the modified step (c'). Therefore, in the following we emphasize only the differences. First note that (5) and (7) also hold in this case. Bounding the sums in (7), one obtains

$$\sum_{i \in \mathcal{P}} p_{i,t} g'_{i,t} = \frac{S_t}{\varepsilon} \left(g_{I_t,t} + |E|\beta \right)$$

and

$$\sum_{i \in \mathcal{P}} p_{i,t} g'^2_{i,t} \leq \frac{1}{\varepsilon} K(1+\beta) \sum_{e \in E} g'_{e,t} \ .$$

Plugging these bounds into (7) and summing for $t = 1, \ldots, n$, we obtain

$$\ln \frac{\overline{W}_n}{\overline{W}_0} \quad \leq \quad \frac{\eta}{1-\gamma} \sum_{t=1}^{n} \frac{S_t}{\varepsilon} \left(g_{I_t,t} + |E|\beta \right) + \frac{\eta^2 K(1+\beta)}{(1-\gamma)\varepsilon} |\mathcal{C}| \max_{i \in \mathcal{P}} G'_{i,n} \ .$$

Combining the upper bound with the lower bound (5), we obtain

$$\sum_{t=1}^{n} \frac{S_t}{\varepsilon} \left(g_{I_t,t} + |E|\beta \right) \quad \geq \quad \left(1 - \gamma - \frac{\eta K(1+\beta)|\mathcal{C}|}{\varepsilon} \right) \max_{i \in \mathcal{P}} G'_{i,n} - \frac{\ln N}{\eta} \ . \tag{12}$$

To relate the left-hand side of the above inequality to the real gain $\sum_{t=1}^{n} g_{I_t,t}$, notice that

$$X_t = \frac{S_t}{\varepsilon} \left(g_{I_t,t} + |E|\beta \right) - \left(g_{I_t,t} + |E|\beta \right)$$

is a martingale difference sequence with respect to $(I_1, S_1), (I_2, S_2), \ldots$. Now for all $t = 1, \ldots, n$, we have the bound

$$\begin{aligned} \mathbb{E}\left[X_t^2 | (I_1,S_1), \ldots, (I_{t-1}, S_{t-1}) \right] \quad &\leq \quad \mathbb{E}\left[\frac{S_t}{\varepsilon^2} (g_{I_t,t} + |E|\beta)^2 \middle| (I_1,S_1), \ldots, (I_{t-1}, S_{t-1}) \right] \\ &\leq \quad \frac{(K + |E|\beta)^2}{\varepsilon} \\ &\leq \quad \frac{4K^2}{\varepsilon} \stackrel{\text{def}}{=} \sigma^2, \end{aligned} \tag{13}$$

where (13) holds by $n \geq \frac{|E| \ln(2|E|/\delta)}{K\varepsilon}$ (to ensure $\beta|E| \leq K$). We know that

$$X_t \in \left[-2K, \left(\frac{1}{\varepsilon} - 1 \right) 2K \right]$$

for all t. Now apply Bernstein's inequality for martingale differences (see Lemma 14 in the Appendix) to obtain

$$\mathbb{P}\left[\sum_{t=1}^{n} X_t > u\right] \leq \frac{\delta}{2}\,, \tag{14}$$

where

$$u = \sqrt{2n\frac{4K^2}{\varepsilon}\ln\left(\frac{2}{\delta}\right)} + \frac{4K}{3\varepsilon}\ln\left(\frac{2}{\delta}\right).$$

From (14) we get

$$\mathbb{P}\left[\sum_{t=1}^{n} \frac{S_t}{\varepsilon}\left(g_{I_t,t} + |E|\beta\right) \geq \widehat{G}_n + \beta n|E| + u\right] \leq \frac{\delta}{2}\,. \tag{15}$$

Now Lemma 5, the union bound, and (15) combined with (12) yield, with probability at least $1-\delta$,

$$\begin{aligned}\widehat{G}_n \;\geq\; & \left(1-\gamma-\frac{\eta K(1+\beta)|C|}{\varepsilon}\right)\left(\max_{i\in\mathcal{P}} G_{i,n} - \frac{4K}{\beta\varepsilon}\ln\frac{2|E|}{\delta}\right) \\ & -\frac{\ln N}{\eta} - \beta n|E| - u\end{aligned}$$

since the coefficient of $G'_{i,n}$ is greater than zero by the assumptions of the theorem.

Since $\widehat{G}_n = Kn - \widehat{L}_n$ and $G_{i,n} = Kn - L_{i,n}$, we have

$$\begin{aligned}\widehat{L}_n \;\leq\; & \left(1-\gamma-\frac{K(1+\beta)\eta|C|}{\varepsilon}\right)\min_{i\in\mathcal{P}} L_{i,n} + Kn\left(\gamma+\frac{K(1+\beta)\eta|C|}{\varepsilon}\right) \\ & + \left(1-\gamma-\frac{K(1+\beta)\eta|C|}{\varepsilon}\right)\frac{4K}{\beta\varepsilon}\ln\frac{2|E|}{\delta} + \beta n|E| + \frac{\ln N}{\eta} + u \\ \leq\; & \min_{i\in\mathcal{P}} L_{i,n} + Kn\left(\gamma+\frac{K(1+\beta)\eta|C|}{\varepsilon}\right) + 5\beta n|E| + \frac{\ln N}{\eta} + u\,,\end{aligned}$$

where we used the fact that $\frac{K}{\beta\varepsilon}\ln\frac{2|E|}{\delta} = \beta n|E|$.

Substituting the value of β, η and γ, we have

$$\begin{aligned}\widehat{L}_n - \min_{i\in\mathcal{P}} L_{i,n} \leq & Kn\frac{2K\eta|C|}{\varepsilon} + Kn\frac{2K\eta|C|}{\varepsilon} + \frac{\ln N}{\eta} + 5\beta n|E| + u \\ \leq & 4K\sqrt{\frac{n|C|\ln N}{\varepsilon}} + 5\sqrt{\frac{n|E|K\ln(2|E|/\delta)}{\varepsilon}} + u \\ \leq & \sqrt{\frac{nK}{\varepsilon}}\left(4\sqrt{K|C|\ln N} + 5\sqrt{|E|\ln(2|E|/\delta)} + \sqrt{8K\ln(2/\delta)}\right) \\ & + \frac{4K}{3\varepsilon}\ln(2/\delta)\end{aligned}$$

as desired. □

6. A Bandit Algorithm for Tracking the Shortest Path

Our goal in this section is to extend the bandit algorithm so that it is able to compete with time-varying paths under small computational complexity. This is a variant of the problem known as *tracking the best expert*; see, for example, Herbster and Warmuth (1998), Vovk (1999), Auer and Warmuth (1998), Bousquet and Warmuth (2002) and Herbster and Warmuth (2001).

To describe the loss the decision maker is compared to, consider the following "m-partition" prediction scheme: the sequence of paths is partitioned into $m+1$ contiguous segments, and on each segment the scheme assigns exactly one of the N paths. Formally, an m-partition $\texttt{Part}(n,m,\mathbf{t},\underline{i})$ of the n paths is given by an m-tuple $\mathbf{t}=(t_1,\ldots,t_m)$ such that $t_0=1<t_1<\cdots<t_m<n+1=t_{m+1}$, and an $(m+1)$-vector $\underline{i}=(i_0,\ldots,i_m)$ where $i_j\in\mathcal{P}$. At each time instant t, $t_j\le t<t_{j+1}$, path i_j is used to predict the best path. The cumulative loss of a partition $\texttt{Part}(n,m,\mathbf{t},\underline{i})$ is

$$L(\texttt{Part}(n,m,\mathbf{t},\underline{i}))=\sum_{j=0}^{m}\sum_{t=t_j}^{t_{j+1}-1}\ell_{i_j,t}=\sum_{j=0}^{m}\sum_{t=t_j}^{t_{j+1}-1}\sum_{e\in i_j}\ell_{e,t}.$$

The goal of the decision maker is to perform as well as the best time-varying path (partition), that is, to keep the normalized regret

$$\frac{1}{n}\left(\widehat{L}_n-\min_{\mathbf{t},\underline{i}}L(\texttt{Part}(n,m,\mathbf{t},\underline{i}))\right)$$

as small as possible (with high probability) for all possible outcome sequences.

In the "classical" tracking problem there is a relatively small number of "base" experts and the goal of the decision maker is to predict as well as the best "compound" expert (i.e., time-varying expert). However in our case, base experts correspond to all paths of the graph between source and destination whose number is typically exponentially large in the number of edges, and therefore we cannot directly apply the computationally efficient methods for tracking the best expert. György et al. (2005a) develop efficient algorithms for tracking the best expert for certain large and structured classes of base experts, including the shortest path problem. The purpose of the following algorithm, shown in Figure 4, is to extend the methods of György et al. (2005a) to the bandit setting when the forecaster only observes the losses of the edges on the chosen path.

The following performance bounds shows that the normalized regret with respect to the best time-varying path which is allowed to switch paths m times is roughly of the order of $K\sqrt{(m/n)|C|\ln N}$.

Theorem 6 *For any $\delta\in(0,1)$ and parameters $0\le\gamma<1/2$, $\alpha,\beta\in[0,1]$, and $\eta>0$ satisfying $2\eta K|C|\le\gamma$, the performance of the algorithm defined above can be bounded, with probability at least $1-\delta$, as*

$$\begin{aligned}&\frac{1}{n}\left(\widehat{L}_n-\min_{\mathbf{t},\underline{i}}L(\texttt{Part}(n,m,\mathbf{t},\underline{i}))\right)\\&\quad\le\quad K\left(\gamma+\eta(1+\beta)K|C|\right)+\frac{K(m+1)}{n\beta}\ln\frac{|E|(m+1)}{\delta}\\&\qquad+\beta|E|+\frac{1}{n\eta}\ln\left(\frac{N^{m+1}}{\alpha^m(1-\alpha)^{n-m-1}}\right).\end{aligned}$$

Parameters: real numbers $\beta > 0, 0 < \eta, \gamma < 1, 0 \le \alpha \le 1$.
Initialization: Set $w_{e,0} = 1$ for each $e \in E$, $\overline{w}_{i,0} = 1$ for each $i \in \mathcal{P}$, and $\overline{W}_0 = N$. For each round $t = 1, 2, \ldots$

(a) Choose a path I_t according to the distribution p_t defined by

$$p_{i,t} = \begin{cases} (1-\gamma)\frac{\overline{w}_{i,t-1}}{\overline{W}_{t-1}} + \frac{\gamma}{|\mathcal{C}|} & \text{if } i \in \mathcal{C}; \\ (1-\gamma)\frac{\overline{w}_{i,t-1}}{\overline{W}_{t-1}} & \text{if } i \notin \mathcal{C}. \end{cases}$$

(b) Compute the probability of choosing each edge e as

$$q_{e,t} = \sum_{i:e\in i} p_{i,t} = (1-\gamma)\frac{\sum_{i:e\in i}\overline{w}_{i,t-1}}{\overline{W}_{t-1}} + \gamma\frac{|\{i \in \mathcal{C} : e \in i\}|}{|\mathcal{C}|}.$$

(c) Calculate the estimated gains

$$g'_{e,t} = \begin{cases} \frac{g_{e,t}+\beta}{q_{e,t}} & \text{if } e \in I_t; \\ \frac{\beta}{q_{e,t}} & \text{otherwise.} \end{cases}$$

(d) Compute the updated weights

$$\begin{aligned} \overline{v}_{i,t} &= \overline{w}_{i,t-1} e^{\eta g'_{i,t}} \\ \overline{w}_{i,t} &= (1-\alpha)\overline{v}_{i,t} + \frac{\alpha}{N}\overline{W}_t \end{aligned}$$

where $g'_{i,t} = \sum_{e\in i} g'_{e,t}$ and $\overline{W}_t$ is the sum of the total weights of the paths, that is,

$$\overline{W}_t = \sum_{i\in\mathcal{P}} \overline{v}_{i,t} = \sum_{i\in\mathcal{P}} \overline{w}_{i,t}.$$

Figure 4: A bandit algorithm for tracking shortest paths

In particular, choosing

$$\beta = \sqrt{\frac{K(m+1)}{n|E|}\ln\frac{|E|(m+1)}{\delta}}, \qquad \gamma = 2\eta K|\mathcal{C}|, \qquad \alpha = \frac{m}{n-1},$$

and

$$\eta = \sqrt{\frac{(m+1)\ln N + m\ln\frac{e(n-1)}{m}}{4nK^2|\mathcal{C}|}}$$

we have, for all $n \ge \max\left\{\frac{K(m+1)}{|E|}\ln\frac{|E|(m+1)}{\delta}, 4|\mathcal{C}|D\right\}$,

$$\frac{1}{n}\left(\widehat{L}_n - \min_{\mathbf{t},\underline{i}} L(\mathtt{Part}(n,m,\mathbf{t},\underline{i}))\right) \le 2\sqrt{\frac{K}{n}}\left(\sqrt{4K|\mathcal{C}|D} + \sqrt{|E|(m+1)\ln\frac{|E|(m+1)}{\delta}}\right),$$

where

$$D = (m+1)\ln N + m\left(1 + \ln\frac{n-1}{m}\right).$$

The proof of the theorem is a combination of that of our Theorem 1 and Theorem 1 of György et al. (2005a). We will need the following three lemmas.

Lemma 7 *For any $1 \le t \le t' \le n$ and any $i \in \mathcal{P}$,*

$$\frac{\overline{v}_{i,t'}}{\overline{w}_{i,t-1}} \ge e^{\eta G'_{i,[t,t']}}(1-\alpha)^{t'-t}$$

where $G'_{i,[t,t']} = \sum_{\tau=t}^{t'} g'_{i,\tau}$.

Proof The proof is a straightforward modification of the one in Herbster and Warmuth (1998). From the definitions of $v_{i,t}$ and $w_{i,t}$ (see step (d) of the algorithm) it is clear that for any $\tau \ge 1$,

$$\overline{w}_{i,\tau} = (1-\alpha)\overline{v}_{i,\tau} + \frac{\alpha}{N}\overline{W}_\tau \ge (1-\alpha)e^{\eta g'_{i,\tau}}\overline{w}_{i,\tau-1}\ .$$

Applying this equation iteratively for $\tau = t, t+1, \ldots, t'-1$, and the definition of $\overline{v}_{i,t}$ (step (d)) for $\tau = t'$, we obtain

$$\begin{aligned}\overline{v}_{i,t'} = \overline{w}_{i,t'-1}e^{\eta g'_{i,t'}} &\ge e^{\eta g'_{i,t'}}\prod_{\tau=t}^{t'-1}\left((1-\alpha)e^{\eta g'_{i,\tau}}\right)\overline{w}_{i,t-1} \\ &= e^{\eta G'_{i,[t,t']}}(1-\alpha)^{t'-t}\overline{w}_{i,t-1}\end{aligned}$$

which implies the statement of the lemma. □

Lemma 8 *For any $t \ge 1$ and $i, j \in \mathcal{P}$, we have*

$$\frac{\overline{w}_{i,t}}{\overline{v}_{j,t}} \ge \frac{\alpha}{N}$$

Proof By the definition of $\overline{w}_{i,t}$ we have

$$\overline{w}_{i,t} = (1-\alpha)\overline{v}_{i,t} + \frac{\alpha}{N}\overline{W}_t \ge \frac{\alpha}{N}\overline{W}_t \ge \frac{\alpha}{N}\overline{v}_{j,t}\ .$$

This completes the proof of the lemma. □

The next lemma is a simple corollary of Lemma 2.

Lemma 9 *For any $\delta \in (0,1)$, $0 \le \beta \le 1$, $t \ge 1$ and $e \in E$ we have*

$$\mathbb{P}\left[G_{e,t} > G'_{e,t} + \frac{1}{\beta}\ln\frac{|E|(m+1)}{\delta}\right] \le \frac{\delta}{|E|(m+1)}\ .$$

Proof of Theorem 6. Again, we upper bound $\ln \overline{W}_n/\overline{W}_0$ the same way as in Theorem 1. Then we get

$$\ln \frac{\overline{W}_n}{\overline{W}_0} \quad \leq \quad \frac{\eta}{1-\gamma}\left(\widehat{G}_n + n|E|\beta\right) + \frac{\eta^2 K(1+\beta)}{1-\gamma}|\mathcal{C}| \max_{i\in\mathcal{P}} G'_{i,n} \,. \tag{16}$$

Let $\texttt{Part}(n,m,\mathbf{t},\underline{i})$ be an arbitrary partition. Then the lower bound is obtained as

$$\ln \frac{\overline{W}_n}{\overline{W}_0} = \ln \sum_{j\in\mathcal{P}} \frac{\overline{w}_{j,n}}{N} = \ln \sum_{j\in\mathcal{P}} \frac{\overline{v}_{j,n}}{N} \geq \ln \frac{\overline{v}_{i_m,n}}{N}$$

(recall that i_m denotes the path used in the last segment of the partition). Now $v_{i_m,n}$ can be rewritten in the form of the following telescoping product

$$\overline{v}_{i_m,n} = \overline{w}_{i_0,t_0-1} \frac{\overline{v}_{i_0,t_1-1}}{\overline{w}_{i_0,t_0-1}} \prod_{j=1}^{m} \left(\frac{\overline{w}_{i_j,t_j-1}}{\overline{v}_{i_{j-1},t_j-1}} \frac{\overline{v}_{i_j,t_{j+1}-1}}{\overline{w}_{i_j,t_j-1}} \right).$$

Therefore, applying Lemmas 7 and 8, we have

$$\begin{aligned} \overline{v}_{i_m,n} &\geq \overline{w}_{i_0,t_0-1} \left(\frac{\alpha}{N}\right)^m \prod_{j=0}^{m} \left((1-\alpha)^{t_{j+1}-1-t_j} e^{\eta G'_{i_j,[t_j,t_{j+1}-1]}} \right) \\ &= \left(\frac{\alpha}{N}\right)^m e^{\eta G'(\texttt{Part}(n,m,\mathbf{t},\underline{i}))} (1-\alpha)^{n-m-1}. \end{aligned}$$

Combining the lower bound with the upper bound (16), we have

$$\begin{aligned} &\ln\left(\frac{\alpha^m(1-\alpha)^{n-m-1}}{N^{m+1}}\right) + \max_{\mathbf{t},\underline{i}} \eta G'(\texttt{Part}(n,m,\mathbf{t},\underline{i})) \\ &\leq \tfrac{\eta}{1-\gamma}\left(\widehat{G}_n + n|E|\beta\right) + \tfrac{\eta^2 K(1+\beta)}{1-\gamma}|\mathcal{C}| \max_{i\in\mathcal{P}} G'_{i,n} \,, \end{aligned}$$

where we used the fact that $\texttt{Part}(n,m,\mathbf{t},\underline{i})$ is an arbitrary partition. After rearranging and using $\max_{i\in\mathcal{P}} G'_{i,n} \leq \max_{\mathbf{t},\underline{i}} G'(\texttt{Part}(n,m,\mathbf{t},\underline{i}))$ we get

$$\begin{aligned} \widehat{G}_n \quad \geq \quad & (1-\gamma-\eta K(1+\beta)|\mathcal{C}|) \max_{\mathbf{t},\underline{i}} G'(\texttt{Part}(n,m,\mathbf{t},\underline{i})) \\ & -n|E|\beta - \frac{1-\gamma}{\eta} \ln\left(\frac{N^{m+1}}{\alpha^m(1-\alpha)^{n-m-1}}\right). \end{aligned}$$

Now since $1-\gamma-\eta K(1+\beta)|\mathcal{C}| \geq 0$, by the assumptions of the theorem and from Lemma 9 with an application of the union bound we obtain that, with probability at least $1-\delta$,

$$\begin{aligned} \widehat{G}_n \geq\, & (1-\gamma-\eta K(1+\beta)|\mathcal{C}|) \left(\max_{\mathbf{t},\underline{i}} G(\texttt{Part}(n,m,\mathbf{t},\underline{i})) - \frac{K(m+1)}{\beta} \ln \frac{|E|(m+1)}{\delta} \right) \\ & - n|E|\beta - \frac{1-\gamma}{\eta} \ln\left(\frac{N^{m+1}}{\alpha^m(1-\alpha)^{n-m-1}}\right). \end{aligned}$$

Since $\widehat{G}_n = Kn - \widehat{L}_n$ and $G(\texttt{Part}(n,m,\mathbf{t},\underline{i})) = Kn - L(\texttt{Part}(n,m,\mathbf{t},\underline{i}))$, we have

$$\begin{aligned}\widehat{L}_n &\leq (1-\gamma-\eta K(1+\beta)|\mathcal{C}|)\min_{\mathbf{t},\underline{i}} L(\texttt{Part}(n,m,\mathbf{t},\underline{i})) + Kn(\gamma+\eta(1+\beta)K|\mathcal{C}|)\\ &\quad + (1-\gamma-\eta(1+\beta)K|\mathcal{C}|)\,\frac{K(m+1)}{\beta}\ln\frac{|E|(m+1)}{\delta} + n|E|\beta\\ &\quad + \frac{1}{\eta}\ln\left(\frac{N^{m+1}}{\alpha^m(1-\alpha)^{n-m-1}}\right).\end{aligned}$$

This implies that, with probability at least $1-\delta$,

$$\begin{aligned}&\widehat{L}_n - \min_{\mathbf{t},\underline{i}} L(\texttt{Part}(n,m,\mathbf{t},\underline{i}))\\ &\quad\leq Kn(\gamma+\eta(1+\beta)K|\mathcal{C}|) + \frac{K(m+1)}{\beta}\ln\frac{|E|(m+1)}{\delta}\\ &\qquad + n|E|\beta + \frac{1}{\eta}\ln\left(\frac{N^{m+1}}{\alpha^m(1-\alpha)^{n-m-1}}\right).\end{aligned} \tag{17}$$

To prove the second statement, let $H(p) = -p\ln p - (1-p)\ln(1-p)$ and $D(p\,\|\,q) = p\ln\frac{p}{q} + (1-p)\ln\frac{1-p}{1-q}$. Optimizing the value of α in the last term of (17) gives

$$\begin{aligned}&\frac{1}{\eta}\ln\left(\frac{N^{m+1}}{\alpha^m(1-\alpha)^{n-m-1}}\right)\\ &\quad= \frac{1}{\eta}\left((m+1)\ln(N) + m\ln\frac{1}{\alpha} + (n-m-1)\ln\frac{1}{1-\alpha}\right)\\ &\quad= \frac{1}{\eta}\big((m+1)\ln(N) + (n-1)(D_b(\alpha^*\,\|\,\alpha) + H_b(\alpha^*))\big)\end{aligned}$$

where $\alpha^* = \frac{m}{n-1}$. For $\alpha = \alpha^*$ we obtain

$$\begin{aligned}&\frac{1}{\eta}\ln\left(\frac{N^{m+1}}{\alpha^m(1-\alpha)^{n-m-1}}\right)\\ &\quad= \frac{1}{\eta}\left((m+1)\ln(N) + (n-1)(H_b(\alpha^*))\right)\\ &\quad= \frac{1}{\eta}\left((m+1)\ln(N) + m\ln((n-1)/m)\right.\\ &\qquad\left.+(n-m-1)\ln(1+m/(n-m-1))\right)\\ &\quad\leq \frac{1}{\eta}\left((m+1)\ln(N) + m\ln((n-1)/m) + m\right)\\ &\quad= \frac{1}{\eta}\left((m+1)\ln(N) + m\ln\frac{e(n-1)}{m}\right) \stackrel{\text{def}}{=} \frac{1}{\eta}D\end{aligned}$$

where the inequality follows since $\ln(1+x) \leq x$ for $x > 0$. Therefore

$$\begin{aligned}&\widehat{L}_n - \min_{\mathbf{t},\underline{i}} L(\texttt{Part}(n,m,\mathbf{t},\underline{i}))\\ &\quad\leq Kn(\gamma+\eta(1+\beta)K|\mathcal{C}|) + \frac{K(m+1)}{\beta}\ln\frac{|E|(m+1)}{\delta} + n|E|\beta + \frac{1}{\eta}D\,.\end{aligned}$$

which is the first statement of the theorem. Setting

$$\beta = \sqrt{\frac{K(m+1)}{n|E|}\ln\frac{|E|(m+1)}{\delta}}, \gamma = 2\eta K|\mathcal{C}|, \text{ and } \eta = \sqrt{\frac{D}{4nK^2|\mathcal{C}|}}$$

results in the second statement of the theorem, that is,

$$\begin{aligned}\widehat{L}_n - \min_{\mathbf{t},\underline{i}} L(\mathtt{Part}(n,m,\mathbf{t},\underline{i})) \\ \leq \quad 2\sqrt{nK}\left(\sqrt{4K|\mathcal{C}|D} + \sqrt{|E|(m+1)\ln\frac{|E|(m+1)}{\delta}}\right). \quad \square\end{aligned}$$

For $t = 1$, choose I_1 uniformly from the set $\mathcal{P}$. For $t \geq 2$,

(a) Draw a Bernoulli random variable Γ_t with $\mathbb{P}(\Gamma_t = 1) = \gamma$.

(b) If $\Gamma_t = 1$, then choose I_t uniformly from $\mathcal{C}$.

(c) If $\Gamma_t = 0$,

(c1) choose τ_t randomly according to the distribution

$$\mathbb{P}\{\tau_t = t'\} = \begin{cases} \frac{(1-\alpha)^{t-1}Z_{1,t-1}}{\overline{W}_{t-1}} & \text{for } t' = 1 \\ \frac{\alpha(1-\alpha)^{t-t'}\overline{W}_{t'}Z_{t',t-1}}{N\overline{W}_t} & \text{for } t' = 2,\ldots,t \end{cases}$$

where $Z_{t',t-1} = \sum_{i\in\mathcal{P}} e^{\eta G'_{i,[t',t-1]}}$ for $t' = 1,\ldots,t-1$, and $Z_{t,t-1} = N$;

(c2) given $\tau_t = t'$, choose I_t randomly according to the probabilities

$$\mathbb{P}\{I_t = i|\tau_t = t'\} = \begin{cases} \frac{e^{\eta G'_{i,[t',t-1]}}}{Z_{t',t-1}} & \text{for } t' = 1,\ldots,t-1 \\ \frac{1}{N} & \text{for } t' = t. \end{cases}$$

Figure 5: An alternative bandit algorithm for tracking shortest paths

Similarly to György et al. (2005a), the proposed algorithm has an alternative version, shown in Figure 5, which is efficiently computable. With a slight modification of the proof of Theorem 2 in György et al. (2005a), it can be shown that the alternative and the original algorithms are equivalent. Moreover, in a way completely analogous to György et al. (2005a), in this alternative formulation of the algorithm one can compute the probabilities the normalization factors $Z_{t',t-1}$ efficiently, as the baseline bandit algorithm for shortest paths has an $O(n|E|)$ time complexity by Theorem 3. Therefore the factors $\overline{W}_t$ and hence the probabilities $\mathbb{P}\{I_t = i|\tau_t = t'\}$ can also be computed efficiently as in György et al. (2005a). In particular, it follows from Theorem 3 of György et al. (2005a) that the time complexity of the alternative bandit algorithm for tracking the shortest path is $O(n^2|E|)$.

7. An Algorithm for the Restricted Multi-Armed Bandit Problem

In this section we consider the situation where the decision maker receives information only about the performance of the whole chosen path, but the individual edge losses are not available. That is, the forecaster has access to the sum $\ell_{I_t,t}$ of losses over the chosen path I_t but not to the losses $\{\ell_{e,t}\}_{e\in I_t}$ of the edges belonging to I_t.

This is the problem formulation considered by McMahan and Blum (2004) and Awerbuch and Kleinberg (2004). McMahan and Blum provided a relatively simple algorithm whose regret is at most of the order of $n^{-1/4}$, while Awerbuch and Kleinberg gave a more complex algorithm to achieve $O(n^{-1/3})$ regret. In this section we combine the strengths of these papers, and propose a simple algorithm with regret at most of the order of $n^{-1/3}$. Moreover, our bound holds with high probability, while the above-mentioned papers prove bounds for the expected regret only. The proposed algorithm uses ideas very similar to those of McMahan and Blum (2004). The algorithm alternates between choosing a path from a "basis" B to obtain unbiased estimates of the loss (exploration step), and choosing a path according to exponential weighting based on these estimates.

A simple way to describe a path $i \in \mathcal{P}$ is a binary row vector with $|E|$ components which are indexed by the edges of the graph such that, for each $e \in E$, the eth entry of the vector is 1 if $e \in i$ and 0 otherwise. With a slight abuse of notation we will also denote by i the binary row vector representing path i. In the previous sections, where the loss of each edge along the chosen path is available to the decision maker, the complexity stemming from the large number of paths was reduced by representing all information in terms of the edges, as the set of edges spans the set of paths. That is, the vector corresponding to a given path can be expressed as the linear combination of the unit vectors associated with the edges (the eth component of the unit vector representing edge e is 1, while the other components are 0). While the losses corresponding to such a spanning set are not observable in the restricted setting of this section, one can choose a subset of $\mathcal{P}$ that forms a *basis*, that is, a collection of b paths which are linearly independent and each path in $\mathcal{P}$ can be expressed as a linear combination of the paths in the basis. We denote by B the $b \times |E|$ matrix whose rows $b^1,\dots,b^b$ represent the paths in the basis. Note that b is equal to the maximum number of linearly independent vectors in $\{i : i \in \mathcal{P}\}$, so $b \le |E|$.

Let $\ell_t^{(E)}$ denote the (column) vector of the edge losses $\{\ell_{e,t}\}_{e\in E}$ at time t, and let $\ell_t^{(B)} = (\ell_{b^1,t},\dots,\ell_{b^b,t})^T$ be a b-dimensional column vector whose components are the losses of the paths in the basis B at time t. If $\alpha_{b^1}^{(i,B)},\dots,\alpha_{b^b}^{(i,B)}$ are the coefficients in the linear combination of the basis paths expressing path $i \in \mathcal{P}$, that is, $i = \sum_{j=1}^b \alpha_{b^j}^{(i,B)} b^j$, then the loss of path $i \in \mathcal{P}$ at time t is given by

$$\ell_{i,t} = \langle i, \ell_t^{(E)} \rangle = \sum_{j=1}^b \alpha_{b^j}^{(i,B)} \langle b^j, \ell_t^{(E)} \rangle = \sum_{j=1}^b \alpha_{b^j}^{(i,B)} \ell_{b^j,t} \tag{18}$$

where $\langle \cdot,\cdot \rangle$ denotes the standard inner product in $\mathbb{R}^{|E|}$. In the algorithm we obtain estimates $\tilde{\ell}_{b^j,t}$ of the losses of the basis paths and use (18) to estimate the loss of any $i \in \mathcal{P}$ as

$$\tilde{\ell}_{i,t} = \sum_{j=1}^b \alpha_{b^j}^{(i,B)} \tilde{\ell}_{b^j,t} \ . \tag{19}$$

It is algorithmically advantageous to calculate the estimated path losses $\tilde{\ell}_{i,t}$ from an intermediate estimate of the individual edge losses. Let B^+ denote the Moore-Penrose inverse of B defined by

$B^+ = B^T(BB^T)^{-1}$, where B^T denotes the transpose of B and BB^T is invertible since the rows of B are linearly independent. (Note that $BB^+ = I_b$, the $b \times b$ identity matrix, and $B^+ = B^{-1}$ if $b = |E|$.) Then letting $\tilde{\ell}_t^{(B)} = (\tilde{\ell}_{b^1,t}, \ldots, \tilde{\ell}_{b^b,t})^T$ and

$$\tilde{\ell}_t^{(E)} = B^+ \tilde{\ell}_t^{(B)}$$

it is easy to see that $\tilde{\ell}_{i,t}$ in (19) can be obtained as $\tilde{\ell}_{i,t} = \langle i, \tilde{\ell}_t^{(E)} \rangle$, or equivalently

$$\tilde{\ell}_{i,t} = \sum_{e \in i} \tilde{\ell}_{e,t}.$$

This form of the path losses allows for an efficient implementation of exponential weighting via dynamic programming Takimoto and Warmuth (2003).

To analyze the algorithm we need an upper bound on the magnitude of the coefficients $\alpha_{b^j}^{(i,B)}$. For this, we invoke the definition of a barycentric spanner from Awerbuch and Kleinberg (2004): the basis B is called a *C-barycentric spanner* if $|\alpha_{b^j}^{(i,B)}| \le C$ for all $i \in \mathcal{P}$ and $j = 1, \ldots, b$. Awerbuch and Kleinberg (2004) show that a 1-barycentric spanner exists if B is a square matrix (i.e., $b = |E|$) and give a low-complexity algorithm which finds a C-barycentric spanner for $C > 1$. We use their technique to show that a 1-barycentric spanner also exists in case of a non-square B, when the basis is chosen to maximize the absolute value of the determinant of BB^T. As before, b denotes the maximum number of linearly independent vectors (paths) in $\mathcal{P}$.

Lemma 10 *For a directed acyclic graph, the set of paths $\mathcal{P}$ between two dedicated nodes has a 1-barycentric spanner. Moreover, let B be a $b \times |E|$ matrix with rows from $\mathcal{P}$ such that* $\det[BB^T] \neq 0$. *If $B_{-j,i}$ is the matrix obtained from B by replacing its jth row by $i \in \mathcal{P}$ and*

$$\left|\det\left[B_{-j,i}B_{-j,i}^T\right]\right| \le C^2 \left|\det\left[BB^T\right]\right| \tag{20}$$

for all $j = 1, \ldots, b$ and $i \in \mathcal{P}$, then B is a C-barycentric spanner.

Proof Let B be a basis of $\mathcal{P}$ with rows $b^1, \ldots, b^b \in \mathcal{P}$ that maximizes $|\det[BB^T]|$. Then, for any path $i \in \mathcal{P}$, we have $i = \sum_{j=1}^b \alpha_{b^j}^{(i,B)} b^j$ for some coefficients $\{\alpha_{b^j}^{(i,B)}\}$. Now for the matrix $B_{-1,i} = [i^T, (b^2)^T, \ldots, (b^b)^T]^T$ we have

$$\begin{aligned}
&\left|\det\left[B_{-1,i}B_{-1,i}^T\right]\right| \\
&= \left|\det\left[B_{-1,i}i^T, B_{-1,i}(b^2)^T, B_{-1,i}(b^3)^T, \ldots, B_{-1,i}(b^b)^T\right]\right| \\
&= \left|\det\left[\left(\sum_{j=1}^b \alpha_{b^j}^{(i,B)} B_{-1,i} b^j\right)^T, B_{-1,i}(b^2)^T, B_{-1,i}(b^3)^T, \ldots, B_{-1,i}(b^b)^T\right]\right| \\
&= \left|\sum_{j=1}^b \alpha_{b^j}^{(i,B)} \det\left[B_{-1,i}(b^j)^T, B_{-1,i}(b^2)^T, B_{-1,i}(b^3)^T, \ldots, B_{-1,i}(b^b)^T\right]\right| \\
&= |\alpha_{b^1}^{(i,B)}| \left|\det\left[B_{-1,i}B^T\right]\right| \\
&= \left(\alpha_{b^1}^{(i,B)}\right)^2 \left|\det\left[BB^T\right]\right|
\end{aligned}$$

where last equality follows by the same argument the penultimate equality was obtained. Repeating the same argument for $B_{-j,i}$, $j = 2, \ldots, b$ we obtain

$$\left|\det\left[B_{-j,i}B_{-j,i}^T\right]\right| = \left(\alpha_{b^j}^{(i,B)}\right)^2 \left|\det\left[BB^T\right]\right|. \tag{21}$$

Thus the maximal property of $|\det[BB^T]|$ implies $|\alpha_{b^j}^{(i,B)}| \le 1$ for all $j = 1, \ldots, b$. The second statement follows trivially from (20) and (21). □

Awerbuch and Kleinberg (2004, Proposition 2.4) also present an iterative algorithm to find a C-barycentric spanner if B is a square matrix. Their algorithm has two parts. First, starting from the identity matrix, the algorithm replaces sequentially the rows of the matrix in each step by maximizing the determinant with respect to the given row. This is done by calling b times an optimization oracle, to compute $\arg\max_{i\in\mathcal{P}} |\det[B_{-j,i}]|$ for $j = 1, 2, \ldots, b$. In the second part the algorithm replaces an arbitrarily row j of the matrix in each iteration with some $i \in \mathcal{P}$ if $|\det[B_{-j,i}]| > C|\det[B]|$. It is shown that the oracle is called in the second part $O(b\log_C b)$ times for $C > 1$. In case B is not a square matrix, the algorithm carries over if we have access to an alternative optimization oracle that can compute $\arg\max_{i\in\mathcal{P}} |\det[B_{-j,i}B_{-j,i}^T]|$: In the first b steps, all the rows of the matrix are replaced (first part), then we can iteratively replace one row in each step, using the oracle, to maximize the determinant $|\det[B_{-j,i}B_{-j,i}^T]|$ in i until (20) is satisfied for all j and i. By Lemma 10, this results is a C-barycentric spanner. Similarly to Awerbuch and Kleinberg (2004, Lemma 2.5), it can be shown that the alternative optimization oracle is called $O(b\log_C b)$ times for $C > 1$.

For simplicity (to avoid carrying the constant C), assume that we have a 2-barycentric spanner B. Based on the ideas of label efficient prediction, the next algorithm, shown in Figure 6, gives a simple solution to the restricted shortest path problem. The algorithm is very similar to that of the algorithm in the label efficient case, but here we cannot estimate the edge losses directly. Therefore, we query the loss of a (random) basis vector from time to time, and create unbiased estimates $\tilde{\ell}_{b^j,t}$ of the losses of basis paths $\ell_{b^j,t}$, which are then transformed into edge-loss estimates.

The performance of the algorithm is analyzed in the next theorem. The proof follows the argument of Cesa-Bianchi et al. (2005), but we also have to deal with some additional technical difficulties. Note that in the theorem we do not assume that all paths between u and v have equal length.

Theorem 11 *Let K denote the length of the longest path in the graph. For any $\delta \in (0,1)$, parameters $0 < \varepsilon \le \frac{1}{K}$ and $\eta > 0$ satisfying $\eta \le \varepsilon^2$, and $n \ge \frac{8b}{\varepsilon^2}\ln\frac{4bN}{\delta}$, the performance of the algorithm defined above can be bounded, with probability at least $1-\delta$, as*

$$\widehat{L}_n - \min_{i\in\mathcal{P}} L_{i,n} \le K\left(\frac{\eta b}{\varepsilon}Kn + \sqrt{\frac{n}{2}\ln\frac{4}{\delta}} + n\varepsilon + \frac{\sqrt{2n\varepsilon\ln\frac{4}{\delta}}}{K} + \frac{16}{3}b\sqrt{2n\frac{b}{\varepsilon}\ln\frac{4bN}{\delta}}\right) + \frac{\ln N}{\eta}$$

In particular, choosing

$$\varepsilon = \left(\frac{Kb}{n}\ln\frac{4bN}{\delta}\right)^{1/3} \quad \text{and} \quad \eta = \varepsilon^2$$

we obtain

$$\widehat{L}_n - \min_{i\in\mathcal{P}} L_{i,n} \le 9.1K^2 b\left(Kb\ln(4bN/\delta)\right)^{1/3} n^{2/3}.$$

Parameters: $0 < \varepsilon, \eta \leq 1$.
Initialization: Set $w_{e,0} = 1$ for each $e \in E$, $\overline{w}_{i,0} = 1$ for each $i \in \mathcal{P}$, $\overline{W}_0 = N$. Fix a basis B, which is a 2-barycentric spanner. For each round $t = 1, 2, \ldots$

(a) Draw a Bernoulli random variable S_t such that $\mathbb{P}(S_t = 1) = \varepsilon$;

(b) If $S_t = 1$, then choose the path I_t uniformly from the basis B. If $S_t = 0$, then choose I_t according to the distribution $\{p_{i,t}\}$, defined by

$$p_{i,t} = \frac{\overline{w}_{i,t-1}}{\overline{W}_{t-1}} .$$

(c) Calculate the estimated loss of all edges according to

$$\tilde{\ell}_t^{(E)} = B^+ \tilde{\ell}_t^{(B)} ,$$

where $\tilde{\ell}_t^{(E)} = \{\tilde{\ell}_{e,t}^{(E)}\}_{e \in E}$, and $\tilde{\ell}_t^{(B)} = (\tilde{\ell}_{b^1,t}^{(B)}, \ldots, \tilde{\ell}_{b^b,t}^{(B)})$ is the vector of the estimated losses

$$\tilde{\ell}_{b^j,t} = \frac{S_t}{\varepsilon} \ell_{b^j,t} \mathbb{1}_{\{I_t = b^j\}} b$$

for $j = 1, \ldots, b$.

(d) Compute the updated weights

$$\begin{aligned} w_{e,t} &= w_{e,t-1} e^{-\eta \tilde{\ell}_{e,t}} , \\ \overline{w}_{i,t} &= \prod_{e \in i} w_{e,t} = \overline{w}_{i,t-1} e^{-\eta \sum_{e \in i} \tilde{\ell}_{e,t}} , \end{aligned}$$

and the sum of the total weights of the paths

$$\overline{W}_t = \sum_{i \in \mathcal{P}} \overline{w}_{i,t} .$$

Figure 6: A bandit algorithm for the restricted shortest path problem

The theorem is proved using the following two lemmas. The first one is an easy consequence of Bernstein's inequality:

Lemma 12 *Under the assumptions of Theorem 11, the probability that the algorithm queries the basis more than* $n\varepsilon + \sqrt{2n\varepsilon \ln \frac{4}{\delta}}$ *times is at most* $\delta/4$.

Using the estimated loss of a path $i \in \mathcal{P}$ given in (19), we can estimate the cumulative loss of i up to time n as

$$\tilde{L}_{i,n} = \sum_{t=1}^{n} \tilde{\ell}_{i,t} .$$

The next lemma demonstrates the quality of these estimates.

Lemma 13 *Let* $0 < \delta < 1$ *and assume* $n \geq \frac{8b}{\varepsilon} \ln \frac{4bN}{\delta}$. *For any* $i \in \mathcal{P}$, *with probability at least* $1 - \delta/4$,

$$\sum_{t=1}^{n} \sum_{i \in \mathcal{P}} p_{i,t} \ell_{i,t} - \sum_{t=1}^{n} \sum_{i \in \mathcal{P}} p_{i,t} \tilde{\ell}_{i,t} \leq \frac{8}{3} b \sqrt{2n \frac{bK^2}{\varepsilon} \ln \frac{4b}{\delta}} \, .$$

Furthermore, with probability at least $1 - \delta/(4N)$,

$$\tilde{L}_{i,n} - L_{i,n} \leq \frac{8}{3} b \sqrt{2n \frac{bK^2}{\varepsilon} \ln \frac{4bN}{\delta}} \, .$$

Proof We may write

$$\begin{aligned}
\sum_{t=1}^{n} \sum_{i \in \mathcal{P}} p_{i,t} \ell_{i,t} - \sum_{t=1}^{n} \sum_{i \in \mathcal{P}} p_{i,t} \tilde{\ell}_{i,t} &= \sum_{t=1}^{n} \sum_{i \in \mathcal{P}} p_{i,t} \sum_{j=1}^{b} \alpha_{b^j}^{(i,B)} \left(\ell_{b^j,t} - \tilde{\ell}_{b^j,t} \right) \\
&= \sum_{j=1}^{b} \sum_{t=1}^{n} \left[\sum_{i \in \mathcal{P}} p_{i,t} \alpha_{b^j}^{(i,B)} \left(\ell_{b^j,t} - \tilde{\ell}_{b^j,t} \right) \right] \\
&\stackrel{\text{def}}{=} \sum_{j=1}^{b} \sum_{t=1}^{n} X_{b^j,t} \, . \qquad (22)
\end{aligned}$$

Note that for any b^j, $X_{b^j,t}$, $t = 1, 2, \ldots$ is a martingale difference sequence with respect to (I_t, S_t), $t = 1, 2, \ldots$ as $\mathbb{E}_t \tilde{\ell}_{b^j,t} = \ell_{b^j,t}$. Also,

$$\mathbb{E}_t[X_{b^j,t}^2] \leq \left(\sum_{i \in \mathcal{P}} p_{i,t} \alpha_{b^j}^{(i,B)} \right)^2 \mathbb{E}_t \left[\left(\tilde{\ell}_{b^j,t} \right)^2 \right] \leq \sum_{i \in \mathcal{P}} p_{i,t} \left(\alpha_{b^j}^{(i,B)} \right)^2 \frac{K^2 b}{\varepsilon} \leq 4 \frac{K^2 b}{\varepsilon} \qquad (23)$$

and

$$|X_{b^j,t}| \leq \left| \sum_{i \in \mathcal{P}} p_{i,t} \alpha_{b^j}^{(i,B)} \right| \left| \ell_{b^j,t} - \tilde{\ell}_{b^j,t} \right| \leq \sum_{i \in \mathcal{P}} p_{i,t} \left| \alpha_{b^j}^{(i,B)} \right| \frac{Kb}{\varepsilon} \leq 2 \frac{Kb}{\varepsilon} \qquad (24)$$

where the last inequalities in both cases follow from the fact that B is a 2-barycentric spanner. Then, using Bernstein's inequality for martingale differences (Lemma 14), we have, for any fixed b^j,

$$\mathbb{P} \left[\sum_{t=1}^{n} X_{b^j,t} \geq \frac{8}{3} \sqrt{2n \frac{bK^2}{\varepsilon} \ln \frac{4b}{\delta}} \right] \leq \frac{\delta}{4b}$$

where we used (23), (24) and the assumption of the lemma on n. The proof of the first statement is finished with an application of the union bound and its combination with (22).

For the second statement we use a similar argument, that is,

$$\begin{aligned}
\sum_{t=1}^{n} (\tilde{\ell}_{i,t} - \ell_{i,t}) = \sum_{j=1}^{b} \alpha_{b^j}^{(i,B)} \sum_{t=1}^{n} (\tilde{\ell}_{b^j,t} - \ell_{b^j,t}) &\leq \sum_{j=1}^{b} \left| \alpha_{b^j}^{(i,B)} \right| \left| \sum_{t=1}^{n} (\tilde{\ell}_{b^j,t} - \ell_{b^j,t}) \right| \\
&\leq 2 \sum_{j=1}^{b} \left| \sum_{t=1}^{n} (\tilde{\ell}_{b^j,t} - \ell_{b^j,t}) \right| \, . \qquad (25)
\end{aligned}$$

Now applying Lemma 14 for a fixed b^j we get

$$\mathbb{P}\left[\sum_{t=1}^{n}(\tilde{\ell}_{b^j,t}-\ell_{b^j,t})\geq\frac{4}{3}\sqrt{2n\frac{K^2b}{\varepsilon}\ln\frac{4bN}{\delta}}\right]\leq\frac{\delta}{4bN} \tag{26}$$

because of $\mathbb{E}_t[(\tilde{\ell}_{b^j,t}-\ell_{b^j,t})^2]\leq\frac{K^2b}{\varepsilon}$ and $-K\leq\tilde{\ell}_{b^j,t}-\ell_{b^j,t}\leq K\left(\frac{b}{\varepsilon}-1\right)$. The proof is completed by applying the union bound to (26) and combining the result with (25). □

Proof of Theorem 11. Similarly to earlier proofs, we follow the evolution of the term $\ln\frac{\overline{W}_n}{\overline{W}_0}$. In the same way as we obtained (5) and (7), we have

$$\ln\frac{\overline{W}_n}{\overline{W}_0}\geq-\eta\min_{i\in\mathcal{P}}\widetilde{L}_{i,n}-\ln N$$

and

$$\ln\frac{\overline{W}_n}{\overline{W}_0}\quad\leq\quad\sum_{t=1}^{n}\left(-\eta\sum_{i\in\mathcal{P}}p_{i,t}\tilde{\ell}_{i,t}+\frac{\eta^2}{2}\sum_{i\in\mathcal{P}}p_{i,t}\tilde{\ell}_{i,t}^2\right)\,.$$

Combining these bounds, we obtain

$$\begin{aligned}-\min_{i\in\mathcal{P}}\widetilde{L}_{i,n}-\frac{\ln N}{\eta}\quad&\leq\quad\sum_{t=1}^{n}\left(-\sum_{i\in\mathcal{P}}p_{i,t}\tilde{\ell}_{i,t}+\frac{\eta}{2}\sum_{i\in\mathcal{P}}p_{i,t}\tilde{\ell}_{i,t}^2\right)\\&\leq\quad\left(-1+\frac{\eta Kb}{\varepsilon}\right)\sum_{t=1}^{n}\sum_{i\in\mathcal{P}}p_{i,t}\tilde{\ell}_{i,t}\,,\end{aligned}$$

because $0\leq\tilde{\ell}_{i,t}\leq\frac{2Kb}{\varepsilon}$. Applying the results of Lemma 13 and the union bound, we have, with probability $1-\delta/2$,

$$\begin{aligned}&-\min_{i\in\mathcal{P}}L_{i,n}-\frac{8}{3}b\sqrt{2n\frac{K^2b}{\varepsilon}\ln\frac{4bN}{\delta}}\\&\quad\leq\quad\left(-1+\frac{\eta Kb}{\varepsilon}\right)\left(\sum_{t=1}^{n}\sum_{i\in\mathcal{P}}p_{i,t}\ell_{i,t}-\frac{8}{3}b\sqrt{2n\frac{K^2b}{\varepsilon}\ln\frac{4b}{\delta}}\right)+\frac{\ln N}{\eta}\\&\quad\leq\quad\left(-1+\frac{\eta Kb}{\varepsilon}\right)\sum_{t=1}^{n}\sum_{i\in\mathcal{P}}p_{i,t}\ell_{i,t}+\frac{8}{3}b\sqrt{2n\frac{K^2b}{\varepsilon}\ln\frac{4b}{\delta}}+\frac{\ln N}{\eta}\,.\end{aligned}\tag{27}$$

Introduce the sets

$$\mathcal{T}_n\stackrel{\text{def}}{=}\{t:1\leq t\leq n\text{ and }S_t=0\}\qquad\text{and}\qquad\overline{\mathcal{T}}_n\stackrel{\text{def}}{=}\{t:1\leq t\leq n\text{ and }S_t=1\}$$

of "exploitation" and "exploration" steps, respectively. Then, by the Hoeffding-Azuma inequality (Hoeffding, 1963) we obtain that, with probability at least $1-\delta/4$,

$$\sum_{t\in\mathcal{T}_n}\sum_{i\in\mathcal{P}}p_{i,t}\ell_{i,t}\geq\sum_{t\in\mathcal{T}_n}\ell_{I_t,t}-\sqrt{\frac{|\mathcal{T}_n|K^2}{2}\ln\frac{4}{\delta}}\,.$$

Note that for the exploration steps $t \in \overline{\mathcal{T}}_n$, as the algorithm plays according to a uniform distribution instead of $p_{i,t}$, we can only use the trivial lower bound zero on the losses, that is,

$$\sum_{t\in\overline{\mathcal{T}}_n}\sum_{i\in\mathcal{P}} p_{i,t}\ell_{i,t} \geq \sum_{t\in\overline{\mathcal{T}}_n} \ell_{I_t,t} - K|\overline{\mathcal{T}}_n| \; .$$

The last two inequalities imply

$$\sum_{t=1}^{n}\sum_{i\in\mathcal{P}} p_{i,t}\ell_{i,t} \geq \widehat{L}_n - \sqrt{\frac{|\mathcal{T}_n|K^2}{2}\ln\frac{4}{\delta}} - K|\overline{\mathcal{T}}_n| \; . \tag{28}$$

Then, by (27), (28) and Lemma 12 we obtain, with probability at least $1-\delta$,

$$\begin{aligned}
&\widehat{L}_n - \min_{i\in\mathcal{P}} L_{i,n} \\
&\quad\leq \quad K\left(\frac{\eta b}{\varepsilon}Kn + \sqrt{\frac{n}{2}\ln\frac{4}{\delta}} + n\varepsilon + \frac{\sqrt{2n\varepsilon\ln\frac{4}{\delta}}}{K} + \frac{16}{3}b\sqrt{2n\frac{b}{\varepsilon}\ln\frac{4bN}{\delta}}\right) + \frac{\ln N}{\eta}
\end{aligned}$$

where we used $\widehat{L}_n \leq Kn$ and $|\mathcal{T}_n| \leq n$. Substituting the values of ε and η gives

$$\begin{aligned}
\widehat{L}_n - \min_{i\in\mathcal{P}} L_{i,n} &\leq K^2 bn\varepsilon + \frac{1}{4}Kn\varepsilon + Kn\varepsilon + \frac{1}{2}n\varepsilon + \frac{16}{3}b\sqrt{K}n\varepsilon + n\varepsilon \\
&\leq 9.1K^2 bn\varepsilon
\end{aligned}$$

where we used $\sqrt{\frac{n}{2}\ln\frac{4}{\delta}} \leq \frac{1}{4}n\varepsilon$, $\sqrt{2n\varepsilon\ln\frac{4}{\delta}} \leq \frac{1}{2}n\varepsilon$, $\sqrt{n\frac{bK}{\varepsilon}\ln\frac{4N}{\delta}} = n\varepsilon$, and $\frac{\ln N}{\eta} \leq n\varepsilon$ (from the assumptions of the theorem). □

8. Simulation Results

To further investigate our new algorithms, we have conducted some simple simulations. As the main motivation of this work is to improve earlier algorithms in case the number of paths is exponentially large in the number of edges, we tested the algorithms on the small graph shown in Figure 1 (b), which has one of the simplest structures with exponentially many (namely $2^{|E|/2}$) paths.

The losses on the edges were generated by a sequence of independent and uniform random variables, with values from $[0,1]$ on the upper edges, and from $[0.32,1]$ on the lower edges, resulting in a (long-term) optimal path consisting of the upper edges. We ran the tests for $n = 10000$ steps, with confidence value $\delta = 0.001$. To establish baseline performance, we also tested the EXP3 algorithm of Auer et al. (2002) (note that this algorithm does not need edge losses, only the loss of the chosen path). For the version of our bandit algorithm that is informed of the individual edge losses (edge-bandit), we used the simple 2-element cover set of the paths consisting of the upper and lower edges, respectively (other 2-element cover sets give similar performance). For our restricted shortest path algorithm (path-bandit) the basis $\{uuuuu, uuuul, uuull, uulll, ullll, lllll\}$ was used, where u (resp. l) in the kth position denotes the upper (resp. lower) edge connecting v_{k-1} and v_k. In this example the performance of the algorithm appeared to be independent of the actual choice of the basis; however, in general we do not expect this behavior. Two versions of the algorithm of

Awerbuch and Kleinberg (2004) were also simulated. With its original parameter setting (AwKl), the algorithm did not perform well. However, after optimizing its parameters off-line (AwKl tuned), substantially better performance was achieved. The normalized regret of the above algorithms, averaged over 30 runs, as well as the regret of the fixed paths in the graph are shown in Figure 7.

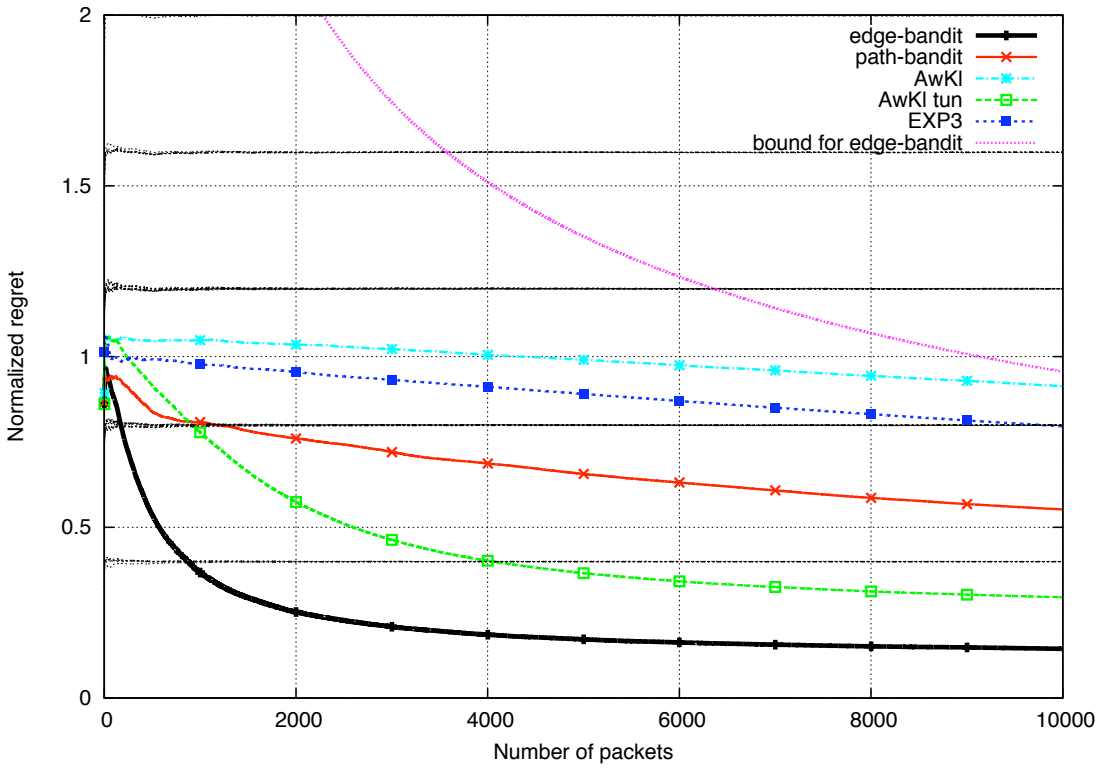


Figure 7: Normalized regret of several algorithms for the shortest path problem. The gray dotted lines show the normalized regret of fixed paths in the graph.

Although all algorithms showed better performance than the bound for the edge-bandit algorithm, the latter showed the expected superior performance in the simulations. Furthermore, our algorithm for the restricted shortest path problem outperformed Awerbuch and Kleinberg's (AwKl) algorithm, while being inferior to its off-line tuned version (AwKl tuned). It must be noted that similar parameter optimization did not improve the performance of our path-bandit algorithm, which showed robust behavior with respect to parameter tuning.

9. Conclusions

We considered different versions of the on-line shortest path problem with limited feedback. These problems are motivated by realistic scenarios, such as routing in communication networks, where the vertices do not have all the information about the state of the network. We have addressed the problem in the adversarial setting where the edge losses may vary in an arbitrary way; in particular, they may depend on previous routing decisions of the algorithm. Although this assumption may

neglect natural correlation in the loss sequence, it suits applications in mobile ad-hoc networks, where the network topology changes dynamically in time, and also in certain secure networks that has to be able to handle denial of service attacks.

Efficient algorithms have been provided for the multi-armed bandit setting and in a combined label efficient multi-armed bandit setting, provided the individual edge losses along the chosen path are revealed to the algorithms. The normalized regrets of the algorithms, compared to the performance of the best fixed path, converge to zero at an $O(1/\sqrt{n})$ rate as the time horizon n grows to infinity, and increases only polynomially in the number of edges (and vertices) of the graph. Earlier methods for the multi-armed bandit problem either do not have the right $O(1/\sqrt{n})$ convergence rate, or their regret increase exponentially in the number of edges for typical graphs. The algorithm has also been extended so that it can compete with time varying paths, that is, to handle situations when the best path can change from time to time (for consistency, the number of changes must be sublinear in n).

In the restricted version of the shortest path problem, where only the losses of the whole paths are revealed, an algorithm with a worse $O(n^{-1/3})$ normalized regret was provided. This algorithm has comparable performance to that of the best earlier algorithm for this problem Awerbuch and Kleinberg (2004), however, our algorithm is significantly simpler. Simulation results are also given to assess the practical performance and compare it to the theoretical bounds as well as other competing algorithms.

It should be noted that the results are not entirely satisfactory in the restricted version of the problem, as it remains an open question whether the $O(1/\sqrt{n})$ regret can be achieved without the exponential dependence on the size of the graph. Although we expect that this is the case, we have not been able to construct an algorithm with such a proven performance bound.

Acknowledgments

This research was supported in part by the János Bolyai Research Scholarship of the Hungarian Academy of Sciences, the Mobile Innovation Center of Hungary, by the Hungarian Scientific Research Fund (OTKA F60787), by the Natural Sciences and Engineering Research Council (NSERC) of Canada, by the Spanish Ministry of Science and Technology grant MTM2006-05650, by Fundación BBVA, by the PASCAL Network of Excellence under EC grant no. 506778 and by the High Speed Networks Laboratory, Department of Telecommunications and Media Informatics, Budapest University of Technology and Economics. Parts of this paper have been presented at COLT'06.

Appendix A.

First we describe a simple algorithm that, given a directed acyclic graph (V,E) with a source vertex u and destination vertex v, constructs a graph by adding at most $(K-2)(|V|-2)+1$ vertices and edges (with constant weight zero) to the graph without modifying the weights of the paths between u and v such that each path from u to v is of length K, where K denotes the length of the longest path of the original graph.

Order the vertices v_i, $i = 1,\ldots,|V|$ of the graph such that if $(v_i,v_j) \in E$ then $i < j$. Replace the destination vertex $v = v_{|V|}$ with a chain of K vertices $v_{|V|,0},\ldots,v_{|V|,K-1}$ and vertices v_i, $i = 3,\ldots,|V|-1$ with a chain of $K-1$ vertices $v_{i,0},\ldots,v_{i,K-2}$ such that in the chains the only edges

are of the form $(v_{i,k+1}, v_{i,k})$ (for each possible value of k), and these edges are of constant weight zero. Furthermore, if $(v_i, v_j) \in E$ is such that the length of the longest path from v_i (resp. v_j) to the destination is K_i (resp. K_j), then this edge is replaced in the new graph by $(v_{i,0}, v_{j,K_j-K_i-1})$ whose weight equals that of the original at each time instant. (Note that here $v_{1,0} = v_1 = u$ and $v_{2,0} = v_2$ and $K_i < K_j$.) It is easy to see that each path from source to destination is of length K in the new graph and the weights of the new paths are the same as those of the corresponding originals.

Next we recall a martingale inequality used in the proofs:

Lemma 14 *(Bernstein's inequality for martingale differences (Freedman, 1975).) Let $X_1, \dots, X_n$ be a martingale difference sequence such that $X_t \in [a,b]$ with probability one ($t = 1, \dots, n$). Assume that, for all t,*

$$\mathbb{E}\left[X_t^2 | X_{t-1}, \dots, X_1\right] \le \sigma^2 \ a.s.$$

Then, for all $\varepsilon > 0$,

$$\mathbb{P}\left\{\sum_{t=1}^{n} X_t > \varepsilon\right\} \le e^{\frac{-\varepsilon^2}{2n\sigma^2 + 2\varepsilon(b-a)/3}}$$

and therefore

$$\mathbb{P}\left\{\sum_{t=1}^{n} X_t > \sqrt{2n\sigma^2 \ln \delta^{-1}} + 2\ln\delta^{-1}(b-a)/3\right\} \le \delta.$$

References

C. Allenberg, P. Auer, L. Györfi, and Gy. Ottucsák. Hannan consistency in on-line learning in case of unbounded losses under partial monitoring. In *Proceedings of 17th International Conference on Algorithmic Learning Theory, ALT 2006, Lecture Notes in Computer Science 4264*, pages 229–243, Barcelona, Spain, Oct. 2006.

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, 2002.

P. Auer and M. Warmuth. Tracking the best disjunction. *Machine Learning*, 32(2):127–150, 1998.

P. Auer and Gy. Ottucsák. Bound on high-probability regret in loss-bandit game. Preprint, 2006. http://www.szit.bme.hu/õti/green.pdf.

B. Awerbuch, D. Holmer, H. Rubens, and R. Kleinberg. Provably competitive adaptive routing. In *Proceedings of IEEE INFOCOM 2005*, volume 1, pages 631–641, March 2005.

B. Awerbuch and R. D. Kleinberg. Adaptive routing with end-to-end feedback: Distributed learning and geometric approaches. In *Proceedings of the 36th Annual ACM Symposium on the Theory of Computing, STOC 2004*, pages 45–53, Chicago, IL, USA, Jun. 2004. ACM Press.

D. Blackwell. An analog of the minimax theorem for vector payoffs. *Pacific Journal of Mathematics*, 6:1–8, 1956.

O. Bousquet and M. K. Warmuth. Tracking a small set of experts by mixing past posteriors. *Journal of Machine Learning Research*, 3:363–396, Nov. 2002.

N. Cesa-Bianchi, Y. Freund, D. P. Helmbold, D. Haussler, R. Schapire, and M. K. Warmuth. How to use expert advice. *Journal of the ACM*, 44(3):427–485, 1997.

N. Cesa-Bianchi and G. Lugosi. *Prediction, Learning, and Games*. Cambridge University Press, Cambridge, 2006.

N. Cesa-Bianchi, G. Lugosi, and G. Stoltz. Minimizing regret with label efficient prediction. *IEEE Trans. Inform. Theory*, IT-51:2152–2162, June 2005.

D.A. Freedman. On tail probabilities for martingales. *Annals of Probability*, 3:100–118, 1975.

E. Gelenbe, M. Gellman, R. Lent, P. Liu, and P. Su. Autonomous smart routing for network QoS. In *Proceedings of First International Conference on Autonomic Computing*, pages 232–239, New York, May 2004. IEEE Computer Society.

E. Gelenbe, R. Lent, and Z. Xhu. Measurement and performance of a cognitive packet network. *Journal of Computer Networks*, 37:691–701, 2001.

A. György, T. Linder, and G. Lugosi. Efficient algorithms and minimax bounds for zero-delay lossy source coding. *IEEE Transactions on Signal Processing*, 52:2337–2347, Aug. 2004a.

A. György, T. Linder, and G. Lugosi. A "follow the perturbed leader"-type algorithm for zero-delay quantization of individual sequences. In *Proc. Data Compression Conference*, pages 342–351, Snowbird, UT, USA, Mar. 2004b.

A. György, T. Linder, and G. Lugosi. Tracking the best of many experts. In *Proceedings of the 18th Annual Conference on Learning Theory, COLT 2005, Lecture Notes in Computer Science 3559*, pages 204–216, Bertinoro, Italy, Jun. 2005a. Springer.

A. György, T. Linder, and G. Lugosi. Tracking the best quantizer. In *Proceedings of the IEEE International Symposium on Information Theory*, pages 1163–1167, Adelaide, Australia, June-July 2005b.

A. György and Gy. Ottucsák. Adaptive routing using expert advice. *The Computer Journal*, 49(2): 180–189, 2006.

J. Hannan. Approximation to Bayes risk in repeated plays. In M. Dresher, A. Tucker, and P. Wolfe, editors, *Contributions to the Theory of Games*, volume 3, pages 97–139. Princeton University Press, 1957.

D.P. Helmbold and S. Panizza. Some label efficient learning results. In *Proceedings of the 10th Annual Conference on Computational Learning Theory*, pages 218–230. ACM Press, 1997.

M. Herbster and M. K. Warmuth. Tracking the best expert. *Machine Learning*, 32(2):151–178, 1998.

M. Herbster and M. K. Warmuth. Tracking the best linear predictor. *Journal of Machine Learning Research*, 1:281–309, 2001.

W. Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58:13–30, 1963.

A. Kalai and S Vempala. Efficient algorithms for the online decision problem. In B. Schölkopf and M. Warmuth, editors, *Proceedings of the 16th Annual Conference on Learning Theory and the 7th Kernel Workshop, COLT-Kernel 2003, Lecture Notes in Computer Science 2777*, pages 26–40, New York, USA, Aug. 2003. Springer.

N. Littlestone and M. K. Warmuth. The weighted majority algorithm. *Information and Computation*, 108:212–261, 1994.

H. B. McMahan and A. Blum. Online geometric optimization in the bandit setting against an adaptive adversary. In *Proceedings of the 17th Annual Conference on Learning Theory, COLT 2004, Lecture Notes in Computer Science 3120*, pages 109–123, Banff, Canada, Jul. 2004. Springer.

M. Mohri. General algebraic frameworks and algorithms for shortest distance problems. Technical Report 981219-10TM, AT&T Labs Research, 1998.

R. E. Schapire and D. P. Helmbold. Predicting nearly as well as the best pruning of a decision tree. *Machine Learning*, 27:51–68, 1997.

E. Takimoto and M. K. Warmuth. Path kernels and multiplicative updates. *Journal of Machine Learning Research*, 4:773–818, 2003.

V. Vovk. Aggregating strategies. In *Proceedings of the Third Annual Workshop on Computational Learning Theory*, pages 372–383, Rochester, NY, Aug. 1990. Morgan Kaufmann.

V. Vovk. Derandomizing stochastic prediction strategies. *Machine Learning*, 35(3):247–282, Jun. 1999.

Journal of Machine Learning Research 8 (2007) 2405-2441 Submitted 3/07; Published 10/07

The Locally Weighted Bag of Words Framework for Document Representation

Guy Lebanon LEBANON@STAT.PURDUE.EDU
Yi Mao YMAO@ECE.PURDUE.EDU
Joshua Dillon JVDILLON@ECE.PURDUE.EDU
Department of Statistics and
School of Electrical and Computer Engineering
Purdue University - West Lafayette, IN, USA

Editor: Andrew McCallum

Abstract

The popular bag of words assumption represents a document as a histogram of word occurrences. While computationally efficient, such a representation is unable to maintain any sequential information. We present an effective sequential document representation that goes beyond the bag of words representation and its n-gram extensions. This representation uses local smoothing to embed documents as smooth curves in the multinomial simplex thereby preserving valuable sequential information. In contrast to bag of words or n-grams, the new representation is able to robustly capture medium and long range sequential trends in the document. We discuss the representation and its geometric properties and demonstrate its applicability for various text processing tasks.

Keywords: text processing, local smoothing

1. Introduction

Modeling text documents is an essential component in a wide variety of text processing applications, including the classification, segmentation, visualization and retrieval of text. A crucial part of the modeling process is choosing an appropriate representation for documents. In this paper we demonstrate a new representation that considers documents as smooth curves in the multinomial simplex. The new representation goes beyond standard alternatives such as the bag of words and n-grams and captures sequential content at a certain resolution determined by a given local smoothing operator.

We consider documents as finite sequences of words

$$y = \langle y_1, \ldots, y_N \rangle \qquad y_i \in V \tag{1}$$

where V represents finite vocabulary which for simplicity is assumed to be a set of integers $V = \{1, \ldots, |V|\} = \{1, \ldots, V\}$. The slight abuse of notation of using V once as a set and once as an integer will not cause confusion later on and serves to simplify the notation. Due to the categorical or nominal nature of V, a document should be considered as a categorical valued time series. In typical cases, we have $1 < N \ll V$ which precludes using standard tools from categorical time series analysis. Instead, the standard approach in text processing is to "vectorize" the data by keeping track of occurrences of length-n word-patterns irrespective of where they appear in the document. This approach, called the n-gram representation, has the benefit of embedding sequential documents

in a Euclidean space $\mathbb{R}^{V^n}$ which is a convenient representation, albeit high dimensional, for many machine learning and statistical models. The specific case of $n = 1$, also called bag of words or bow representation, is perhaps the most frequent document representation due to its relative robustness in sparse situations.

Formally, the n-gram approach represents a document $y = \langle y_1, \ldots, y_N \rangle, y_i \in V$ as $x \in \mathbb{R}^{V^n}$, defined by

$$x_{(j_1,\ldots,j_n)} = \frac{1}{N-n+1} \sum_{i=1}^{N-n+1} \delta_{y_i,j_1} \delta_{y_{i+1},j_2} \cdots \delta_{y_{i+n-1},j_n}, \tag{2}$$

where $\delta_{a,b} = 1$ if $a = b$ and 0 otherwise. In the case of 1-gram or bag of words the above representation reduces to

$$x_j = \frac{1}{N} \sum_{i=1}^{N} \delta_{y_i,j}$$

which is simply the relative frequencies of different vocabulary words in the document.

A slightly more general outlook is to consider smoothed versions of (2) in order to avoid the otherwise overwhelmingly sparse frequency vector (since $N \ll V$, only a small subset of the vocabulary appears in any typical document). For example, a smoothed 1-gram representation is

$$x_j = \frac{1}{Z} \sum_{i=1}^{N} (\delta_{y_i,j} + c), \quad c \geq 0 \tag{3}$$

where Z is a constant that ensures normalization $\sum x_j = 1$. The smoothed representation (3) has a Bayesian interpretation as a the maximum posterior estimate for a multinomial model with Dirichlet prior and setting $c = 0$ in (3) reduces it to the standard word histogram or 1-gram. Recent comparative studies of various n-gram smoothing methods in the contexts of language modeling and information retrieval may be found in Chen and Rosenfeld (2000) and Zhai and Lafferty (2001).

Conceptually, we may consider the n-gram representation for $n = N$ in which case the full original sequential information is maintained. In practice, however, n is typically chosen to be much smaller than N, often taking the values 1, 2, or 3. In these cases, frequently occurring word patterns are kept allowing some limited amount of word-sense disambiguation. On the other hand, almost all of the sequential content, including medium and long range sequential trends and position information is lost.

The paper's main contribution is a new sequential representation called locally weighted bag of words or lowbow. This representation, first introduced in Lebanon (2005), generalizes bag of words by considering the collection of local word histograms throughout the document. In contrast to n-grams, which keep track of frequently occurring patterns independent of their positions, lowbow keeps track of changes in the word histogram as it sweeps through the document from beginning to end. The collection of word histograms is equivalent to a smooth curve which facilitates the differential analysis of the document's sequential content. The use of bag of words rather than n-grams with $n > 1$ is made here for simplicity purpose only. The entire lowbow framework may be generalized to define locally weighted n-grams in a straightforward manner.

The next section presents a detailed explanation of the locally weighted bag of words framework. Section 3 describes the mechanics of using the lowbow framework in document modeling. Section 4 discusses the tradeoff in choosing the amount of temporal smoothing through a bias-variance analysis and generalization error bounds. Section 5 outlines several experiments, followed

by related work and discussion. Since our presentation makes frequent use of the geometry of the multinomial simplex, which is not common knowledge in the machine learning community, we provide a brief summary of it in Appendix A.

2. Locally Weighted Bag of Words

As mentioned previously, the original word sequence (1) is categorical, high dimensional and sparse. The smoothing method employed by the bag of words representation (3) is categorical in essence rather than temporal since no time information is preserved. In contrast to (3) or its variants, temporal smoothing such as the one used in local regression or kernel density estimation (e.g., Wand and Jones, 1995) is performed across a continuous temporal or spatial dimension. Temporal smoothing has far greater potential than categorical smoothing since a word can be smoothed out to varying degrees depending on the temporal difference between the two document positions. The main idea behind the locally weighted bag of words framework is to use a local smoothing kernel to smooth the original word sequence temporally. In other words, we borrow the presence of a word at a certain location in the document to a neighboring location but discount its contribution depending on the temporal distance between the two locations.

Since temporal smoothing of words results in several words occupying one location we need to consider the following broader definition of a document.

Definition 1 *A document x of length N is a function $x : \{1,\ldots,N\} \times V \to [0,1]$ such that*

$$\sum_{j \in V} x(i,j) = 1 \quad \forall i \in \{1,\ldots,N\}.$$

The set of documents (of all lengths) is denoted by $\mathfrak{X}$.

For a document $x \in \mathfrak{X}$ the value $x(i,j)$ represent the weight of the word $j \in V$ at location i. Since the weights sum to one at any location we can consider Definition 1 as providing a local word histogram or distribution associated with each document position. The standard way to represent a word sequence as a document in $\mathfrak{X}$ is to have each location host the appropriate single word with constant weight, which corresponds to the δ_c representation defined below with $c = 0$.

Definition 2 *The standard representation $\delta_c(y) \in \mathfrak{X}$, where $c \geq 0$, of a word sequence $y = \langle y_1,\ldots,y_N \rangle$ is*

$$\delta_c(y)(i,j) = \begin{cases} \frac{c}{1+c|V|} & y_i \neq j \\ \frac{1+c}{1+c|V|} & y_i = j \end{cases}. \tag{4}$$

Equation (4) is consistent with Definition 1 since $\sum_{j \in V} \delta_c(y)(i,j) = \frac{1+c|V|}{1+c|V|} = 1$. The parameter c in the above definition injects categorical smoothing as in (3) to avoid zero counts in the δ_c representation.

The standard representation δ_c assumes that each word in the sequence $y = \langle y_1,\ldots,y_N \rangle$ occupies a single temporal location $1,\ldots,N$. In general, however, Definition 1 lets several words occupy the same location by smoothing the influence of words y_j across different document positions. Doing so is central in converting the discrete-time standard representation to a continuous representation that is much more convenient for modeling and analysis.

Definition 1 is problematic since according to it, two documents of different lengths are considered as fundamentally different objects. It is not clear, for example, how to compare two documents $x_1 : \{1,\ldots,N_1\} \times V \to [0,1]$, $x_2 : \{1,\ldots,N_2\} \times V \to [0,1]$ of varying lengths $N_1 \neq N_2$. To allow a unified treatment and comparison of documents of arbitrary lengths we map the set $\{1,\ldots,N\}$ to a continuous canonical interval, which we arbitrarily choose to be $[0,1]$.

Definition 3 *A length-normalized document x is a function $x : [0,1] \times V \to [0,1]$ such that*

$$\sum_{j \in V} x(t,j) = 1, \qquad \forall t \in [0,1].$$

The set of length-normalized documents is denoted $\mathfrak{X}'$.

A simple way of converting a document $x \in \mathfrak{X}$ to a length-normalized document $x' \in \mathfrak{X}'$ is expressed by the length-normalization function defined below.

Definition 4 *The length-normalization of a document $x \in \mathfrak{X}$ of length N is the mapping*

$$\varphi : \mathfrak{X} \to \mathfrak{X}' \quad \varphi(x)(t,j) = x(\lceil tN \rceil, j)$$

where $\lceil r \rceil$ is the smallest integer greater than or equal to r.

The length-normalization process abstracts away from the actual document length and focuses on the sequential variations within the document relative to its length. In other words, we treat two documents with similar sequential contents but different lengths in a similar fashion. For example the two documents $\langle y_1, y_2, \ldots, y_N \rangle$ and $\langle y_1, y_1, y_2, y_2, \ldots, y_N, y_N \rangle$ or the more realistic example of a news story and its summary would be mapped to the same length-normalized representation. The assumption that the actual length does not matter and sequential trends should be considered relative to the total length may not hold in some cases. We comment on this assumption further and on how to relax it in Section 7.

We formally define bag of words as the integral of length-normalized documents with respect to time. As we show later, this definition is equivalent to the popular definition of bag of words expressed in Equation (3).

Definition 5 *The bag of words or bow representation of a document y is $\rho(\varphi(\delta_c(y)))$ defined by*

$$\rho : \mathfrak{X}' \to \mathbb{P}_{V-1} \quad \textit{where} \quad [\rho(x)]_j = \int_0^1 x(t,j)\, dt, \tag{5}$$

and $[\cdot]_j$ denotes the j-th component of a vector.

Above, $\mathbb{P}_{V-1}$ stands for the multinomial simplex

$$\mathbb{P}_{V-1} = \left\{ \theta \in \mathbb{R}^V : \forall i\, \theta_i \geq 0, \sum_{j=1}^{V} \theta_j = 1 \right\}$$

which is the subset of $\mathbb{R}^V$ representing the set of all distributions on V events. The subscript $V-1$ is used in $\mathbb{P}_{V-1}$ rather than V in order to reflect its intrinsic dimensionality. The simplex and its Fisher or information geometry are a central part of this paper. Appendix A contains a brief overview of

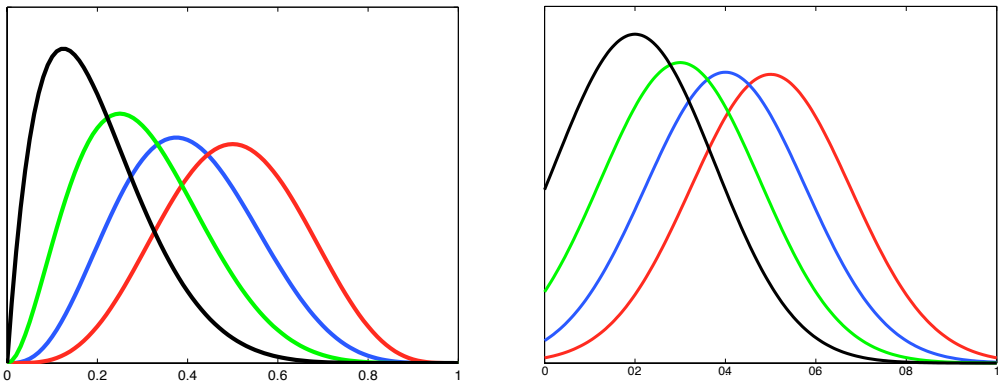


Figure 1: Beta (left) and bounded Gaussian (right) smoothing kernels for $\mu = 0.2, 0.3, 0.4, 0.5$.

the necessary background and further details may be found in Kass and Voss (1997), Amari and Nagaoka (2000) and Lebanon (2005). Note that the function ρ in Definition 5 is well defined since

$$\sum_{j\in V} [\rho(x)]_j = \sum_{j\in V} \int_0^1 x(t,j)\,dt = \int_0^1 \sum_{j\in V} x(t,j)\,dt = \int_0^1 1\,dt = 1 \quad \Longrightarrow \quad \rho(x) \in \mathbb{P}_{V-1}.$$

A local alternative to the bag of words is obtained by integrating a length-normalized document with respect to a non-uniform measure on $[0,1]$. In particular, integrating with respect to a measure that is concentrated around a particular location $\mu \in [0,1]$ provides a smoothed characterization of the local word histogram. In accordance with the statistical literature of non-parametric smoothing we refer to such a measure as a smoothing kernel. Formally, we define it as a function $K_{\mu,\sigma} : [0,1] \to \mathbb{R}$ parameterized by a location parameter $\mu \in [0,1]$ and a scale parameter $\sigma \in (0,\infty)$. The parameter μ represents the (length-normalized) document location at which the measure is concentrated and σ represents its spread or amount of smoothing. We further assume that $K_{\mu,\sigma}$ is smooth in t, μ and is normalized, that is, $\int_0^1 K_{\mu,\sigma}(t)\,dt = 1$.

One example of a smoothing kernel on $[0,1]$ is the Gaussian pdf restricted to $[0,1]$ and re-normalized

$$K_{\mu,\sigma}(x) = \begin{cases} \frac{N(x;\mu,\sigma)}{\Phi((1-\mu)/\sigma) - \Phi(-\mu/\sigma)} & x \in [0,1] \\ 0 & x \notin [0,1] \end{cases} \tag{6}$$

where $N(x;\mu,\sigma)$ is the Gaussian pdf with mean μ and variance σ^2 and Φ is the cdf of $N(x;0,1)$. Another example is the beta distribution pdf

$$K_{\mu,\sigma}(x) = \text{Beta}\left(x\,;\, \beta\frac{\mu}{\sigma}\,,\, \beta\frac{1-\mu}{\sigma}\right) \tag{7}$$

where β is selected so that the two parameters of the beta distribution will be greater than 1. The above beta pdf has expectation μ and variance that is increasing in the scale parameter σ. The bounded Gaussian and beta kernels are illustrated in Figure 1.

Definition 6 *The locally weighted bag of words or lowbow representation of the word sequence y is $\gamma(y) = \{\gamma_\mu(y) : \mu \in [0,1]\}$ where $\gamma_\mu(y) \in \mathbb{P}_{V-1}$ is the local word histogram at μ defined by*

$$[\gamma_\mu(y)]_j = \int_0^1 \varphi(\delta_c(y))(t,j)\, K_{\mu,\sigma}(t) dt. \tag{8}$$

Equation (8) indeed associates a document location with a local histogram or a point in the simplex $\mathbb{P}_{V-1}$ since

$$\begin{aligned}\sum_{j\in V} [\gamma_\mu(y)]_j &= \sum_{j\in V} \int_0^1 \varphi(\delta_c(y))(t,j) K_{\mu,\sigma}(t)\, dt = \int_0^1 K_{\mu,\sigma}(t) \sum_{j\in V} \varphi(\delta_c(y))(t,j)\, dt \\ &= \int_0^1 K_{\mu,\sigma}(t) \cdot 1\, dt = 1.\end{aligned}$$

Geometrically, the lowbow representation of documents is equivalent to parameterized curves in the simplex. The following theorem establishes the continuity and smoothness of these curves which enables the use of differential geometry in the analysis of the lowbow representation and its properties.

Theorem 1 *The lowbow representation is a continuous and differentiable parameterized curve in the simplex, in both the Euclidean and the Fisher geometry.*

Proof We prove below only the continuity of the lowbow representation. The proof of differentiability proceeds along similar lines. Fixing y, the mapping $\mu \mapsto \gamma_\mu(y)$ maps $[0,1]$ into the simplex $\mathbb{P}_{V-1}$. Since $K_{\mu,\sigma}(t)$ is continuous on a compact region $(\mu,t) \in [0,1]^2$, it is also uniformly continuous and we have

$$\begin{aligned}\lim_{\varepsilon\to 0} |[\gamma_\mu(y)]_j - [\gamma_{\mu+\varepsilon}(y)]_j| &= \lim_{\varepsilon\to 0} \Big| \int_0^1 \varphi(\delta_c(y))(t,j) K_{\mu,\sigma}(t) - \varphi(\delta_c(y))(t,j) K_{\mu+\varepsilon,\sigma}(t) dt \Big| \\ &\le \lim_{\varepsilon\to 0} \int_0^1 \varphi(\delta_c(y))(t,j) |K_{\mu,\sigma}(t) - K_{\mu+\varepsilon,\sigma}(t)|\, dt \\ &\le \lim_{\varepsilon\to 0} \sup_{t\in[0,1]} |K_{\mu,\sigma}(t) - K_{\mu+\varepsilon,\sigma}(t)| \int_0^1 \varphi(\delta_c(y))(t,j)\, dt \\ &= \lim_{\varepsilon\to 0} \sup_{t\in[0,1]} |K_{\mu,\sigma}(t) - K_{\mu+\varepsilon,\sigma}(t)| = 0.\end{aligned}$$

As a result,

$$\lim_{\varepsilon\to 0} \|\gamma_\mu(y) - \gamma_{\mu+\varepsilon}(y)\|_2 = \sqrt{\sum_{j\in V} |[\gamma_\mu(y)]_j - [\gamma_{\mu+\varepsilon}(y)]_j|^2} \to 0$$

proving the continuity of $\gamma_\mu(y)$ in the Euclidean geometry. Continuity in the Fisher geometry follows since it shares the same topology as the Euclidean geometry. ■

It is important to note that the parameterized curve that corresponds to the lowbow representation consists of two parts: the geometric figure $\{\gamma_\mu(y) : \mu \in [0,1]\} \subset \mathbb{P}_{V-1}$ and the parameterization function $\mu \mapsto \gamma_\mu(y)$ that ties the local histogram to a location μ in the normalized document. While

it is easy to ignore the parameterization function when dealing with parameterized curves, one must be aware that different lowbow representations may share similar geometric figures but possess different parameterization speeds. Thus it is important to keep track of the parameterization speed as well as the geometric figure.

The geometric properties of the curve depend on the word sequence, the kernel shape and the kernel scale parameter. The kernel scale parameter is especially important as it determines the amount of temporal smoothing employed. As the following theorem shows, if $\sigma \to \infty$ the lowbow curve degenerates into a single point corresponding to the bow representation. As a consequence we view the popular bag of words representation (3) as a special case of the lowbow representation.

Theorem 2 *Let $K_{\mu,\sigma}$ be a smoothing kernel such that when $\sigma \to \infty$, $K_{\mu,\sigma}(x)$ is constant in μ, x. Then for $\sigma \to \infty$, the lowbow curve $\gamma(y)$ degenerates into a single point corresponding to the bow representation of* (3).

Proof Since the kernel is both constant and normalized over $[0,1]$, we have $K_{\mu,\sigma}(t) = 1$ for all $\mu, t \in [0,1]$. For all $\mu \in [0,1]$,

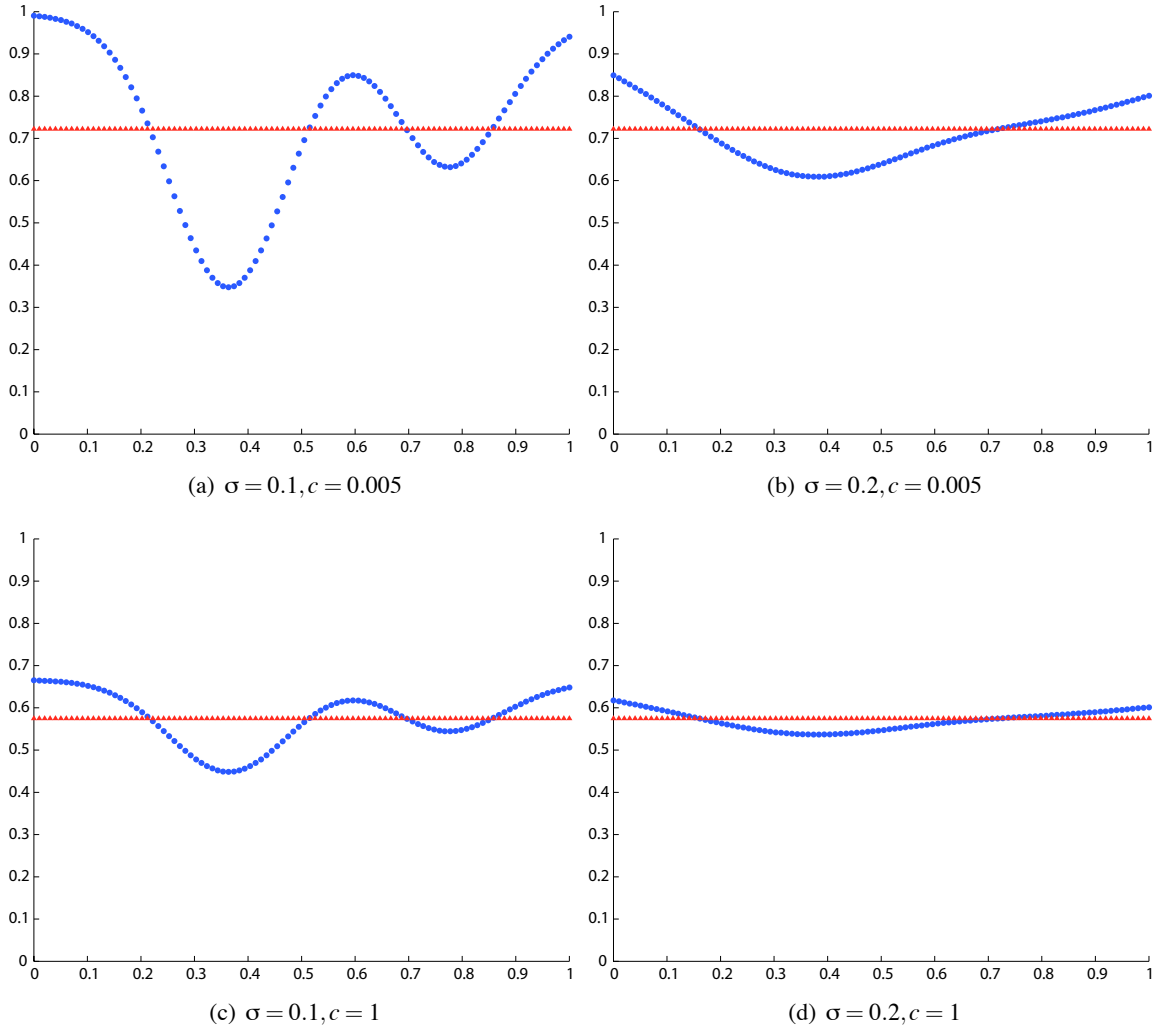
$$\begin{aligned}
[\gamma_\mu(y)]_j &= \int_0^1 \varphi(\delta_c(y))(t,j) K_{\mu,\sigma}(t)\, dt = \int_0^1 \varphi(\delta_c(y))(t,j)\, dt \\
&= \sum_{i=1}^{N} \frac{1}{N} \left(\delta_{y_i,j} \frac{1+c}{1+c|V|} + (1-\delta_{y_i,j}) \frac{c}{1+c|V|} \right) \\
&\propto \sum_{i=1}^{N} \delta_{y_i,j}(1+c) + (1-\delta_{y_i,j})c \propto \sum_{i=1}^{N} (\delta_{y_i,j} + c).
\end{aligned}$$

■

Intuitively, small σ will result in a simplicial curve that quickly moves between the different corners of the simplex as the words $y_1, y_2, \ldots, y_N$ are encountered. The extreme case of $\sigma \to 0$ represents a discontinuous curve equivalent to the original word sequence representation (1). It is unlikely that either of the extreme cases $\sigma \to \infty$ or $\sigma \to 0$ will be an optimal choice from a modeling perspective. By varying σ between 0 and ∞, the lowbow representation interpolates between these two extreme cases and captures sequential detail at different resolutions. Selecting an appropriate scale $0 < \sigma < \infty$ we obtain a sequential resolution that captures sequential trends at a certain resolution while smoothing away finer temporal details.

Figures 2-3 illustrate the curve resulting from the lowbow representation and its dependency on the kernel scale parameter and the smoothing coefficient. Notice how the curve shrinks as σ increases until it reaches the single point that is the bow model. Increasing c, on the other hand, pushes the geometric figure towards the center of the simplex.

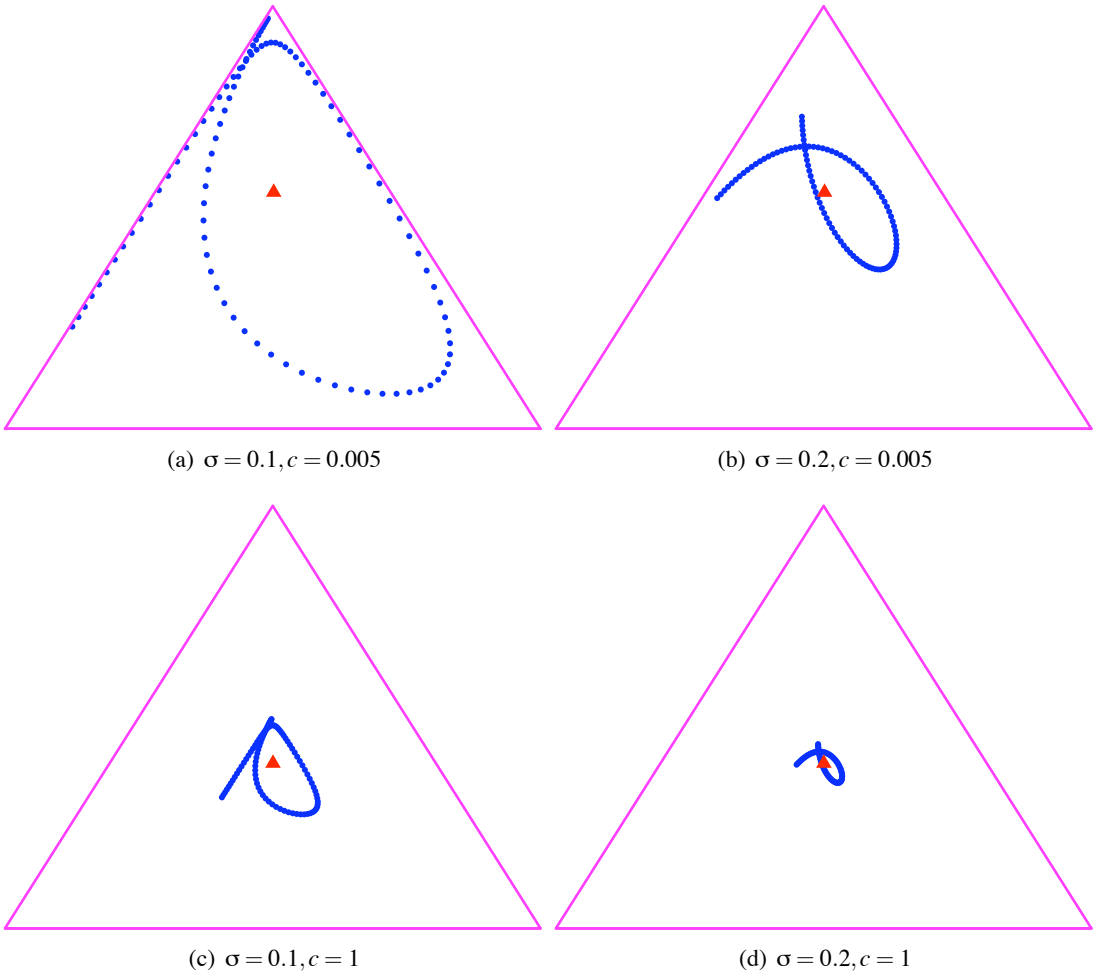
It is useful to have a quantitative characterization of the complexity of the lowbow representation as a function of the chosen kernel and σ. To this end, the kernel's complexity, defined below, serves as a bound for variations in the lowbow curve.



(a) $\sigma = 0.1, c = 0.005$ (b) $\sigma = 0.2, c = 0.005$

(c) $\sigma = 0.1, c = 1$ (d) $\sigma = 0.2, c = 1$

Figure 2: The curve in $\mathbb{P}_1$ resulting from the lowbow representation of the word sequence $\langle 1\ \ 1\ \ 1\ \ 2\ \ 2\ \ 1\ \ 1\ \ 1\ \ 2\ \ 1\ \ 1 \rangle$. Since $[\gamma_\mu(y)]_2 = 1 - [\gamma_\mu(y)]_1$ we visualize the curve by graphing $[\gamma_\mu(y)]_1$ as a function of μ. The figures illustrate the differences as the scale parameter of the Gaussian kernel σ increases from 0.1 to 0.2 (left vs. right column) and the smoothing coefficient c varies from 0.005 to 1 (first vs. second row). Increasing the kernel scale causes some local features to vanish, for example the second local minimum. In addition, increasing σ shrinks the figure towards the single bow point (represented by the horizontal line). Increasing the smoothing coefficient c causes the figure to stay away from the boundary of the simplex and concentrate in the center. Since the curves are composed of 100 dots, the distances between the dots indicate the parameterization speed of the curves.



(a) $\sigma = 0.1, c = 0.005$ (b) $\sigma = 0.2, c = 0.005$

(c) $\sigma = 0.1, c = 1$ (d) $\sigma = 0.2, c = 1$

Figure 3: The curve in $\mathbb{P}_2$ resulting from the lowbow representation of the word sequence $\langle 1\ 3\ 3\ 3\ 2\ 2\ 1\ 3\ 3 \rangle$. In this case $\mathbb{P}_2$ is visualized as a triangle in $\mathbb{R}^2$ (see Figure 15 for visualizing $\mathbb{P}_2$). The figures illustrate the differences as the scale parameter of the Gaussian kernel σ increases from 0.1 to 0.2 (left vs. right column) and the smoothing coefficient c varies from 0.005 to 1 (first vs. second row). Increasing the kernel scale causes some local features to vanish, for example the tail in the bottom left corner of $\mathbb{P}_2$. In addition, increasing σ shrinks the figure towards the single bow point (represented by the triangle). Increasing the smoothing coefficient c causes the figure to stay away from the boundary of the simplex and concentrate in the center. Since the curves are composed of 100 dots, the distances between the dots indicate the parameterization speed of the curves.

Definition 7 *Let $K_{\mu,\sigma}(t)$ be a kernel that is Lipschitz continuous*[1] *in μ with a Lipschitz constant $C_K(t)$. The kernel's complexity is defined as*

$$O(K) = \sqrt{V} \int_0^1 C_K(t)\, dt.$$

The theorem below proves that the lowbow curve is Lipschitz continuous with a Lipschitz constant $O(K)$, thus connecting the curve complexity with the shape and the scale of the kernel.

Theorem 3 *The lowbow curve $\gamma(y)$ satisfies*

$$\|\gamma_\mu(y) - \gamma_\tau(y)\|_2 \;\le\; |\mu - \tau|\; O(K), \quad \forall \mu, \tau \in [0,1].$$

Proof

$$\begin{aligned} |[\gamma_\mu(y)]_j - [\gamma_\tau(y)]_j| &\le \int_0^1 \varphi(\delta_c(y))(t,j) |K_{\mu,\sigma}(t) - K_{\tau,\sigma}(t)|\, dt \\ &\le \int_0^1 |K_{\mu,\sigma}(t) - K_{\tau,\sigma}(t)|\, dt \\ &\le |\mu - \tau| \int_0^1 C_K(t)\, dt \end{aligned}$$

and so $\|\gamma_\mu(y) - \gamma_\tau(y)\|_2 = \sqrt{\sum_{j \in V} |[\gamma_\mu(y)]_j - [\gamma_\tau(y)]_j|^2} \le |\mu - \tau| O(K)$. ■

3. Modeling of Simplicial Curves

Modeling functional data such as lowbow curves is known in the statistics literature as functional data analysis (e.g., Ramsay and Dalzell, 1991; Ramsay and Silverman, 2005). Previous work in this area focused on low dimensional functional data such as one dimensional or two dimensional curves. In this section we discuss some issues concerning generative and conditional modeling of lowbow curves. Additional information regarding the practical use of lowbow curves in a number of text processing tasks may be found in Section 5.

Geometrically, a lowbow curve is a point in an infinite product of simplices $\mathbb{P}_{V-1}^{[0,1]}$ that is naturally equipped with the product topology and geometry of the individual simplices. In practice, maintaining a continuous representation is often difficult and unnecessary. Sampling the path at representative points $\mu_1, \ldots, \mu_l \in [0,1]$ provides a finite dimensional lowbow representation equivalent to a point in the product space $\mathbb{P}_{V-1}^{l}$. Thus, even though we proceed below to consider continuous curves and infinite dimensional spaces $\mathbb{P}_{V-1}^{[0,1]}$, in practice we will typically discretize the curves and replace integrals with appropriate summations.

Given a Riemannian metric g on the simplex, its product form

$$g'_\theta(u,v) = \int_0^1 g_{\theta(t)}(u(t), v(t))\, dt$$

defines a corresponding metric on lowbow curves. As a result, geometric structures compatible with the base metric g, such as distance or curvature, give rise to analogous product versions. For

1. A Lipschitz continuous function f satisfies $|f(x) - f(y)| \le C|x - y|$ for some constant C called the Lipschitz constant.

example, the distance between lowbow representations of two word sequences $\gamma(y), \gamma(z) \in \mathbb{P}_m^{[0,1]}$ is the average distance between the corresponding time coordinates

$$d(\gamma(y), \gamma(z)) = \int_0^1 d(\gamma_\mu(y), \gamma_\mu(z))\, d\mu \tag{9}$$

where $d(\gamma_\mu(y), \gamma_\mu(z))$ depends on the simplex geometry under consideration, e.g. Equation (21) in the case of the Fisher geometry or $d(\gamma_\mu(y), \gamma_\mu(z)) = \|\gamma_\mu(y) - \gamma_\mu(z)\|_2$ in the case of Euclidean geometry.

Using the integrated distance formula (9) we can easily adapt distance-based algorithms to the lowbow representation. For example, k-nearest neighbor classifiers are adapted by replacing standard distances such as the Euclidean distance or cosine similarity with the integrated distance (9) or its discretized version.

In contrast to the base distance on $\mathbb{P}_{V-1}$ which is used in the bow representation, the integrated distance (9) captures local differences in text sequences. For example, it compares the beginning of document y with the beginning of document z, the middle with the middle, and the end with the end. While it may be argued that the above is not expected to always accurately model differences between documents, it does hold in some cases. For example, news articles have a natural semantic progression starting with a brief summary at the beginning and delving into more detail later on, often in a chronological manner. Similarly, other documents such as web pages and emails share a similar sequential structure. Section 5.3 provides some experimental support for this line of thought and also describes some alternatives.

In a similar way, we can also apply kernel-based algorithms such as SVM to documents using the lowbow representation by considering a kernel over $\mathbb{P}_{V-1}^{[0,1]}$. For example, the product geometry may be used to define a product diffusion process whose kernel can conveniently capture local relationships between documents. Assuming a base Fisher geometry we obtain the approximated diffusion kernel

$$K_t(\gamma(y), \gamma(z)) \;\propto\; \exp\left(-\frac{1}{t}\left(\int_0^1 \arccos\left(\sum_{j \in V} \sqrt{[\gamma_\mu(y)]_j [\gamma_\mu(z)]_j}\right) d\mu\right)^2\right) \tag{10}$$

using the parametrix expansion described in Berger et al. (1971). We omit the details as they are closely related to the derivations of Lafferty and Lebanon (2005). Alternative kernels can be obtained using the mechanism of Hilbertian metrics developed by Christensen et al. (1984) and Hein and Bousquet (2005).

The Fisher diffusion kernel of Lafferty and Lebanon (2005) achieves excellent performance in standard text classification experiments. We show in Section 5 that its lowbow version (10) further improves upon those results. In addition to classification, the lowbow diffusion kernel may prove useful for other tasks such as dimensionality reduction using kernel PCA, support vector regression, and semi-supervised learning.

The lowbow representation may also be used to construct generative models for text that generalize the naive Bayes or multinomial model. By estimating the probability $p(y)$ associated with a given text sequence y, such models serve an important role in machine translation, speech recognition and information retrieval. In contrast to the multinomial model which ignores the sequential progression in a document, lowbow curves γ may be considered as a semiparametric generative model assigning the probability vector γ_μ to the generation of words around the document location μN. Formally this amounts to the following process:

Step 1 Draw a document length N from some distribution on positive integers.

Step 2 Generate the words $y_1, \ldots, y_N$ according to

$$y_i \sim \text{Mult}(\theta_{i1}, \ldots, \theta_{iV}) \quad \text{where} \quad \theta_{ij} \propto \int_{(i-1)/N}^{i/N} [\gamma_\mu]_j \, d\mu.$$

The above model can also be used to describe situations in which the underlying document distribution changes with time (e.g., Forman, 2006). Lebanon and Zhao (2007) describe a local likelihood model that is essentially equivalent to the generative lowbow model described above. In contrast to the model of Blei and Lafferty (2006) the lowbow generative model is not based on latent topics and is inherently smooth.

The differential characteristics of the lowbow curve convey significant information and deserve closer inspection. As pointed out by Ramsay and Silverman (2005), applying linear differential operators L_α to functional data $L_\alpha f = \sum_i \alpha_i D^i f$ (where D^i is the i-th derivative operator) often reveals interesting features and properties that are normally difficult to detect. The simplest such operator is the first derivative or velocity $D\gamma_\mu = \dot{\gamma}_\mu$ (defined by $[\dot{\gamma}_\mu]_j = \mathrm{d}[\gamma_\mu]_j / \mathrm{d}\mu$) which reveals the instantaneous direction of the curve at a certain time point as well as the current speed through its norm $\|\dot{\gamma}_\mu\|$. More specifically, we can obtain a tangent vector field $\dot{\gamma}$ along the curve that describes sequential topic trends and their change. Higher order differential operators such as the curvature reveal the amount of curve variation or deviation from a straight line. Integrating the norm of the curvature tensor over $\mu \in [0,1]$ provides a measure of the sequential topic complexity or variability throughout the document. We demonstrate such differential operators and their use in visualization and segmentation of documents in Section 5. Further details concerning differential operators and their role in visualizing lowbow curves may be found in Mao et al. (2007).

In general, it is fair to say that modeling curves is more complicated than modeling points. However, if done correctly it has the potential to capture information that otherwise would remain undetected. Keeping in mind that we can control the amount of variability by changing σ, thereby interpolating between $\langle y_1, \ldots, y_N \rangle$ and (3), we are able to effectively model sequential trends in documents. The choice of σ controls the amount of smoothing and as in non-parametric density estimation, an appropriate choice is crucial to the success of the model. This notion is explored in greater detail in the next section.

4. Kernel Smoothing, Bias-Variance Tradeoff and Generalization Error Bounds

The choice of the kernel scale parameter σ or the amount of smoothing is essential for the success of the lowbow framework. Choosing a σ that is too large would result in a function class that is relatively weak and will not be able to express the desired sequential content. Choosing a σ that is too small would result in a rich function class that is destined to overfit the available training set. This central tradeoff has been analyzed in statistics through the bias and variance of the estimated parameters and in computational learning theory through generalization error bounds. In this section, we discuss this tradeoff from both viewpoints. Further details concerning the statistical properties of the lowbow estimator as a local likelihood model for streaming data may be found in Lebanon and Zhao (2007). Practical aspects concerning the selection of σ appear in Section 5.

4.1 Bias and Variance Tradeoff

We discuss the bias and variance of the lowbow model $\gamma(y)$ as an estimator for an underlying semi-parametric model $\{\theta_t : t \in [0,1]\} \subset \mathbb{P}_{V-1}$ which we assume generated the observed document y. The model assigns a local multinomial θ_t to different locations t and proceeds to generate the words $y_i, i = 1,\ldots,N$ according to $y_i \sim_{iid} \theta_{i/N}$. Note that the iid sampling assumption simply implies that the sampling of the words from their respective multinomials are independent. It does not prevent the assumption of a higher order structure, Markovian or otherwise, on the relationship between the multinomials generating adjacent words $\theta_{i/N}, \theta_{(i+1)/N}$.

The bias and variance of the the lowbow estimator $\gamma(y) = \hat{\theta}(y)$, reveal the expected tradeoff by considering their dependence on the kernel scale σ. We start by writing the components of $\gamma(y) = \hat{\theta}$ as a weighted combination of the sampled words

$$\hat{\theta}_{\mu j} = \int_0^1 y(t,j) K_{\mu,\sigma}(t)\, dt = \sum_{i=1}^{N} y(i,j) \int_{(i-1)/N}^{i/N} K_{\mu,\sigma}(t)\, dt = \sum_{\tau \in J} w_{\mu-\tau} y(\mu-\tau, j)$$

where $y \in \mathcal{X}'$, $w_i = \int_{(i-1)/N}^{i/N} K_{\mu,\sigma}(t)\, dt$ and $J = \{\mu - N, \ldots, \mu - 1\}$. It is relatively simple to show that $\hat{\theta}_{\mu j}$ is a consistent estimator of $\theta_{\mu j}$ under conditions that ensure the weight function w approaches a delta function at μ as the number of samples goes to infinity (e.g., Wand and Jones, 1995). In our case, the number of samples is fixed and is dictated by the number of words in the document. However, despite the lack of an asymptotic trend $N \to \infty$ we can still gain insight from analyzing the dependency of the bias and variance of the lowbow estimator as a function of the kernel scale parameter σ.

Using standard results concerning the expectation and variance of Bernoulli random variables we have

$$\begin{aligned}
\mathsf{bias}\,(\hat{\theta}_{\mu j}) &= \mathsf{E}\,(\hat{\theta}_{\mu j} - \theta_{\mu j}) = \sum_{\tau \in J} w_{\mu-\tau} \mathsf{E}\,(y(\mu-\tau, j)) - \theta_{\mu j} \\
&= \sum_{\tau \in J} w_{\mu-\tau} (\theta_{\mu-\tau,j} - \theta_{\mu j}). \qquad (11) \\
\mathsf{Var}\,(\hat{\theta}_{\mu j}) &= \mathsf{E}\,(\hat{\theta}_{\mu j} - \mathsf{E}\,\hat{\theta}_{\mu j})^2 = \mathsf{E}\left(\sum_{\tau \in J} w_{\mu-\tau} (y(\mu-\tau, j) - \theta_{\mu-\tau,j}) \right)^2 \\
&= \sum_{\tau \in J} \sum_{\tau' \in J} w_{\mu-\tau} w_{\mu-\tau'} \mathsf{E}\,(y(\mu-\tau, j) - \theta_{\mu-\tau})(y(\mu-\tau', j) - \theta_{\mu-\tau',j}) \\
&= \sum_{\tau \in J} w_{\mu-\tau}^2 \mathsf{Var}\,(y(\mu-\tau, j)) \\
&= \sum_{\tau \in J} w_{\mu-\tau}^2 \theta_{\mu-\tau,j} (1 - \theta_{\mu-\tau,j}). \qquad (12)
\end{aligned}$$

The bias term clearly depends on the weight vector w and on the rate of local changes in the true parameter $\theta_{\mu j}$. For a certain fixed model $\hat{\theta}_{\mu j}$, the bias clearly decreases as the weight distribution approaches a delta function at μ, that is, $w_i = 1$ if $i = \mu$ and 0 otherwise. In fact, in the limiting case of $w_i = \delta_{i\mu}$, $\mathsf{bias}\,(\hat{\theta}_{\mu j}) = 0$ and $\mathsf{Var}\,(\hat{\theta}_{\mu j}) = \theta_{\mu j}(1 - \theta_{\mu j})$. As the weight distribution becomes less localized, the bias will increase (in the absolute value) and the variance will typically decrease due to the shape of the function $f(w_i) = w_i^2$ for $w_i \in [0,1]$. The precise characterization of the variance

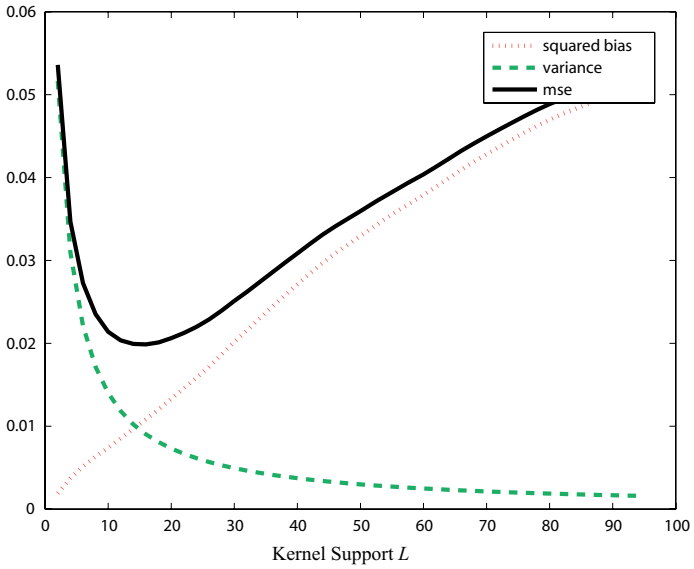


Figure 4: Squared bias, variance and mean squared error of the lowbow estimator $\hat{\theta}_{ij}$ as a function of a triangular kernel support, that is, L in (13). The curve was generated by averaging over synthetic data θ_{ij} drawn from a bounded Wiener process on $[0,1]$.

reduction depends on the model $\theta_{\mu j}$ and the functional form of the kernel. Figure 4 contains an illustration of the squared bias, variance and mean squared error for the discretized triangular kernel

$$w_i = \frac{1}{Z}\left(1 - \frac{2}{L}|i|\right) \qquad i = -L/2, \ldots, L/2 \tag{13}$$

where L defines the kernel support and Z ensures normalization. In the figure, we used synthetic data $\theta_{ij}, i = 1, \ldots, 100$ generated from a bounded Wiener process on $[0,1]$ (i.e., a bounded random walk with Gaussian increments). To avoid phenomena that correspond to a particular sample path we averaged the bias and variance over 200 samples from the process.

The problem of selecting a particular weight vector w or kernel K for the lowbow estimator that minimizes the mean squared error is related to the problem of bandwidth kernel selection in local regression and density estimation. The simple estimate obtained from the plug-in rule for the bias and variance (i.e., $\theta_{\mu j} \mapsto \hat{\theta}_{\mu j}$ in Equations (11)-(12)) is usually not recommended due to the poor estimation performance of plug-in rules (e.g., Cleveland and Loader, 1996). More sophisticated estimates exist, including adaptive estimators that may select different bandwidths or kernels at different points. An alternative approach, which we adopted in our experiments, is to use cross validation or bootstrapping in the selection process.

4.2 Large Deviation Bounds and Covering Numbers

An alternative approach to bias-variance analysis is to characterize the kernel scale tradeoff through the study of generalization error bounds. Such bounds use large deviation techniques to characterize

the difference between the empirical risk or training error and the expected risk uniformly over a class of functions $L = \{f_\alpha : \alpha \in I\}$. These bounds are expressed probabilistically and usually take the following form (Anthony and Bartlett, 1999)

$$P\left(\sup_{\alpha\in I} |\mathsf{E}_p(\mathcal{L}(f_\alpha(Z))) - \mathsf{E}_{\tilde{p}}(\mathcal{L}(f_\alpha(Z)))| \geq \varepsilon\right) \leq C(\mathcal{L}, L, n, \varepsilon). \tag{14}$$

Above, Z represents any sequence of n examples - either X in the unsupervised scenario or (X,Y) in the supervised scenario and $\mathsf{E}_p, \mathsf{E}_{\tilde{p}}$ represent the expectation over the sampling distribution and the empirical distribution $\tilde{p}(z) = \frac{1}{n}\sum_{i=1}^{n} \delta_{z,z_i}$. $\mathcal{L}$ represents some loss function, for example classification error rate and the function C measures the rate of uniform convergence of the empirical risk $\mathsf{E}_{\tilde{p}}(\mathcal{L}(f_\alpha(Z)))$ to the true risk $\mathsf{E}_p(\mathcal{L}(f_\alpha(Z)))$ over the function class $L = \{f_\alpha : \alpha \in I\}$.

To obtain a model with a small expected risk we need to balance the following two goals. On the one hand, we need to minimize the empirical risk $\mathsf{E}_{\tilde{p}}(\mathcal{L}(\alpha, Z))$ since the expected risk is typically close to the empirical risk (by the uniform law of large numbers). On the other hand, we need to tighten the bound (14) by selecting a function class L that results in a small value of the function C. This tradeoff, presented by Vapnik (1998) under the name structural risk minimization, is the computational learning theory analog of the statistical bias-variance concept.

A lowbow representation with a small σ would lead to a low empirical risk since it results in a richer and more accurate expression of the data. Increasing σ forms a lossy transformation and hence leads to essential loss of data features and higher training error but would reduce C and therefore also the bound on the expected error.

The most frequent way to bound C is through the use of the covering number which measures the size of a function class (Dudley, 1984; Anthony and Bartlett, 1999). The covering number enables several ways of determining the rate of uniform convergence C in (14), for example see Theorem 1 and 2 in Zhang (2002).

Definition 8 *Let $x = x_1, \ldots, x_n \in \mathcal{X}$ be a set of observations and $f_\alpha : \mathcal{X} \to \mathbb{R}$ be a parameterized function. The covering number in p-norm $\mathcal{N}_p(f, \varepsilon, (x_1, \ldots, x_n))$ is the minimum number m of vectors $v_1, \ldots, v_m \in \mathbb{R}^n$ for which*

$$\forall \alpha \; \exists v_j \quad \textit{such that} \quad \left(\frac{1}{n}\sum_{i=1}^{n} |f_\alpha(x_i) - v_{ji}|^p\right)^{1/p} \leq \varepsilon.$$

In other words, the set $\{f_\alpha(x) : \alpha \in I\} \subset \mathbb{R}^n$ is covered by m ε-balls centered at $v_1, \ldots, v_m$.

Definition 9 *The uniform covering number $\mathcal{N}_p(f, \varepsilon, n)$ is defined as*

$$\mathcal{N}_p(f, \varepsilon, n) = \sup_{x_1,\ldots,x_n} \mathcal{N}_p(f, \varepsilon, x_1, \ldots, x_n).$$

The covering numbers themselves are difficult to compute precisely and are usually bounded themselves. Recent research results that bound the covering numbers for important function classes such as neural networks, support vector machines, boosting and logistic regression may be found in Williamson et al. (2001), Guo et al. (2002) and Zhang (2002). We focus on the covering number bounds in Zhang (2002) for linear classifiers as they are relatively easy to express in terms of the kernel scale parameter. The theorem and bounds below are expressed for continuous lowbow representation and continuous linear classifiers. The same results hold with analogous proofs in the more practical case of finite dimensional linear classifiers and discretized lowbow representations.

Theorem 4 *For the class of continuous linear classifiers $L = \{f_\alpha(\gamma(y)) : \|\alpha\|_2 \le a\}$ operating on continuous lowbow representation*

$$f_\alpha(\gamma(y)) = \sum_{j\in V} \int_0^1 \alpha_j(\mu)[\gamma_\mu(y)]_j \, d\mu$$

we have the following bounds on the L_2 and L_∞ covering numbers

$$\mathcal{N}_2(L,\varepsilon,n) \le 2^{\lceil a^2b^2/\varepsilon^2\rceil \log_2(2n+1)}$$
$$\mathcal{N}_\infty(L,\varepsilon,n) \le 2^{36(a^2b^2/\varepsilon^2)\log_2(2\lceil 4ab/\varepsilon+2\rceil n+1)}$$

where $b = \min(1, |||\mathbf{K}_\sigma|||_2)$.

Above, α represents a vector of weight functions $\alpha = (\alpha_1,\ldots,\alpha_V), \alpha_i : [0,1] \to \mathbb{R}$ that parameterize linear operators on $\gamma(y)$. The norm $\|\alpha\|_2$ is defined as $\sqrt{\sum_j \int \alpha_j^2(t)\,dt}$. $\mathbf{K}_\sigma$ is an operator on $f : [0,1] \mapsto [0,1]$ such that $\left(\mathbf{K}_\sigma f\right)(\mu) = \int K_{\mu,\sigma}(t) f(t)\,dt$. The induced 2-norm of the operator is (e.g., Horn and Johnson, 1990)

$$|||\mathbf{K}_\sigma|||_2 = \sup_{\|f\|_2=1} \|\mathbf{K}_\sigma f\|_2 = \sup_{\|f\|_2=1} \sqrt{\int \left(\int K_{\mu,\sigma}(t) f(t)\,dt\right)^2 d\mu} \tag{15}$$

where $\|f\|_2 = \sqrt{\int f^2(t)\,dt}$.

Proof First note that the L_2 norm of the lowbow representation can be bounded by the constant 1

$$\begin{aligned}
\|\gamma(y)\|_2^2 &= \sum_{j\in V} \int_0^1 ([\gamma_\mu(y)]_j)^2 \, d\mu = \sum_{j\in V} \int_0^1 \left(\int_0^1 x(t,j) K_{\mu,\sigma}(t)\,dt\right)^2 d\mu \\
&= \sum_{j\in V} \int_0^1 \left(\iint_{[0,1]^2} x(t,j)x(t',j) K_{\mu,\sigma}(t) K_{\mu,\sigma}(t')\,dt\,dt'\right) d\mu \\
&\le \iiint_{[0,1]^3} \left(\sum_{j\in V} x^2(t,j)\right)^{1/2} \left(\sum_{j\in V} x^2(t',j)\right)^{1/2} K_{\mu,\sigma}(t) K_{\mu,\sigma}(t')\,dt\,dt'\,d\mu \\
&\le \iiint_{[0,1]^3} \left(\sum_{j\in V} x(t,j)\right)^{1/2} \left(\sum_{j\in V} x(t',j)\right)^{1/2} K_{\mu,\sigma}(t) K_{\mu,\sigma}(t')\,dt\,dt'\,d\mu \\
&= \int_0^1 \left(\int_0^1 K_{\mu,\sigma}(t)\,dt\right)^2 d\mu = 1.
\end{aligned}$$

Alternatively, an occasionally tighter bound that depends on the operator norm and therefore on the kernel's scale parameter is

$$\begin{aligned}
\|\gamma(y)\|_2^2 &= \sum_{j\in V} \int_0^1 ([\gamma_\mu(y)]_j)^2 d\mu = \sum_{j\in V} \int_0^1 \left(\int_0^1 K_{\mu,\sigma}(t) x(t,j) dt\right)^2 d\mu = \sum_{j\in V} \|\mathbf{K}_\sigma x(\cdot,j)\|_2^2 \\
&\le \sum_{j\in V} |||\mathbf{K}_\sigma|||_2^2 \|x(\cdot,j)\|_2^2 = |||\mathbf{K}_\sigma|||_2^2 \left(\sum_{j\in V} \|x(\cdot,j)\|_2^2\right) \le |||\mathbf{K}_\sigma|||_2^2 \left(\sum_{j\in V} \|x(\cdot,j)\|_1\right) \\
&= |||\mathbf{K}_\sigma|||_2^2 \left(\int_0^1 \sum_{j\in V} x(t,j) dt\right) = |||\mathbf{K}_\sigma|||_2^2
\end{aligned}$$

where the last two inequalities follow from the definition of the induced operator norm (Horn and Johnson, 1990) and the fact that $\|x(\cdot,j)\|_2^2 = \int_0^1 x(t,j)^2 dt \leq \int_0^1 x(t,j)dt = \|x(\cdot,j)\|_1$.

The proof is concluded by plugging in the above bound into Corollary 3 and Theorem 4 of Zhang (2002):

$$\|x\|_2 \leq b, \|w\|_2 \leq a \quad \Rightarrow \quad \log_2 \mathcal{N}_2(L,\varepsilon,n) \leq \lceil a^2 b^2/\varepsilon^2 \rceil \log_2(2n+1), \tag{16}$$

$$\|x\|_2 \leq b, \|w\|_2 \leq a \quad \Rightarrow \quad \log_2 \mathcal{N}_\infty(L,\varepsilon,n) \leq 36\frac{a^2 b^2}{\varepsilon^2} \log_2(2\lceil 4ab/\varepsilon + 2\rceil n + 1). \tag{17}$$

Note that since the bounds in (16)-(17) do not depend on the dimensionality of the data x they hold for any dimensionality, as well as in the limit of continuous data and continuous linear operators as above. ■

The theorem above remains true (and actually it is closer to the original statements in Zhang, 2002) for discretized lowbow representation $\{\gamma_\mu(y) : \mu \in T\}$ where T is a finite set which reduces the lowbow representation and α to a matrix form and discretize the linear operator $\langle \alpha, \gamma(y) \rangle = \sum_{j \in V} \sum_{\mu \in T} \alpha_{j\mu} [\gamma_\mu(y)]_j$. The covering number bounds in Theorem 4 may be directly applied, using either the continuous or the discretized versions, to bound the classification expected error rate for linear classifiers such as support vector machines, Fisher's linear discriminant, boosting, and logistic regression. We do not reproduce these results here since active research in this area frequently improves the precise form of the bound and the constants involved.

As the kernel scale parameter σ decreases, the kernel becomes less uniform thus increasing the possible variability in the data representation $\|\gamma(y)\|_2^2$ and the covering number bound. In the case of the bounded Gaussian kernel (6) we compute $|||\mathbf{K}_\sigma|||_2$ as a function of the kernel scale parameter σ which is illustrated in Figure 5.

5. Experiments

In this section, we demonstrate lowbow's applicability to several text processing tasks, including text classification using nearest neighbor and support vector machines, text segmentation, and document visualization. All experiments use real world data.

5.1 Text Classification using Nearest Neighbor

We start by examining lowbow and its properties in the context of text classification using a nearest neighbor classifier. We report experimental results for the WebKB faculty vs. course task and the Reuters-21578 top ten categories (1 vs. all) using the standard mod-apte training-testing split. In the WebKB task we repeatedly sampled subsets for training and testing with equal positive and negative examples. In the Reuters task we randomly sampled subsets of the mod-apte split for training and testing data which resulted in unbalanced train and test sets containing more negative than positive examples. Sampling training sets of different sizes from the mod-apte split enabled us to examine the behavior of the classifiers as a function of the size of the training set.

The continuous quantities in the lowbow calculation were approximated by a discrete sample of 5 equally spaced points in the interval $[0,1]$ turning the integrals into simple sums. As a result, the computational complexity associated with the lowbow representation is simply the number of sampling points times the complexity of the corresponding bow classifier. Choosing 5 sampling

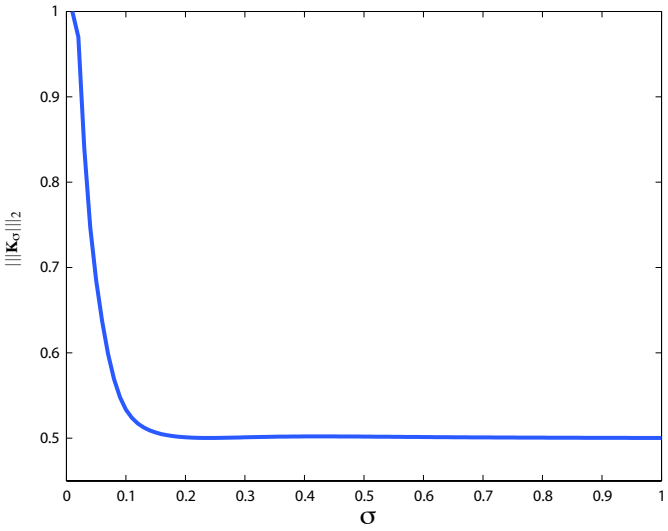


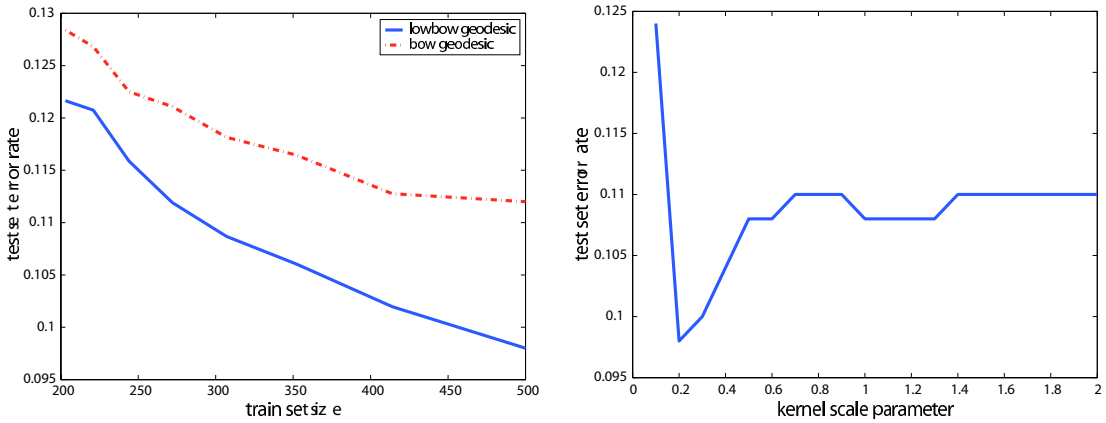
Figure 5: $|||\mathbf{K}_\sigma|||_2$ for the bounded Gaussian kernel (6) as a function of the kernel scale parameter σ. The continuous 2-norm definition in (15) is approximated by 5 equally spaced samples for μ and 20 equally spaced samples for t.

points is rather arbitrary in our case and we did not find it critical to the nature of the experimental results. Throughout the experiments we used the bounded Gaussian kernel (6) and computed several alternatives for the kernel scale parameter σ and chose the best one. While not entirely realistic, this setting enables us to examine lowbow's behavior in the optimistic scenario of being able to find the best scale parameter. The next section includes similar text classification experiments using SVM that explore further the issue of automatically selecting the scale parameter σ.

Figure 6 (top) displays results for nearest neighbor classification using the Fisher geodesic distance on the WebKB data. The left graph is a standard train-set size vs. test set error rate comparing the bow geodesic (lowbow with $\sigma \to \infty$) (dashed) and the lowbow geodesic distance. The right graph displays the dependency of the test set error rate on the scale parameter indicating an optimal scale at around $\sigma = 0.2$ (for repeated samplings of 500 training examples). In both cases, the performances of standard bow techniques such as tf cosine similarity or Euclidean distance were significantly inferior (20-40% higher error rate) than the Fisher geodesic distances and as a result are not displayed.

Figure 6 (bottom) displays test set error rates for the Reuters-21578 task. The 10 rows in the table indicate the classification task of identifying each of the 10 most popular classes in the Reuters collection. The columns represent varying training set sizes sampled from the mod-apte split. The lowbow geodesic distance for an intermediate scale is denoted by err_1 and for $\sigma \to \infty$ is denoted by err_2. Tf-cosine similarity and Euclidean distance for bow are denoted by err_3 and err_4.

The experiments indicate that lowbow geodesic clearly outperforms, for most values of σ, the standard tf-cosine similarity and Euclidean distance for bow (represented by $\text{err}_3, \text{err}_4$). In addition they also indicate that in general, the best scale parameter for lowbow is an intermediate one, rather than the standard bow model $\sigma \to \infty$ thus validating the hypothesis that we can leverage sequential



	Train Size = 100				Train Size = 200				Train Size = 400			
class	err_1	err_2	err_3	err_4	err_1	err_2	err_3	err_4	err_1	err_2	err_3	err_4
1	**9.9**	10.6	11.2	11.0	**8.1**	9.7	8.2	10.7	**6.7**	7.3	11.2	9.0
2	**11.6**	12.8	17.6	22.4	**9.4**	9.7	17.6	19.9	7.9	**7.8**	17.2	17.9
3	**6.8**	7.6	6.9	12.9	**5.8**	7.2	7.8	16.9	5.4	**5.3**	10.2	12.6
4	5.6	6.5	6.5	**5.5**	**4.8**	**4.8**	7.1	7.0	**4.5**	4.7	8.5	7.5
5	6.6	**6.2**	9.0	11.4	**5.7**	6.8	6.7	10.3	**5.0**	5.6	5.8	7.4
6	**5.7**	5.8	5.8	10.8	**5.2**	5.3	5.3	10.0	**4.8**	5.4	5.6	11.3
7	**4.2**	5.1	7.0	12.9	**4.2**	4.3	7.9	9.0	**3.9**	4.3	5.8	7.5
8	**3.0**	3.2	4.7	7.6	**3.0**	3.3	3.4	3.4	**2.6**	2.9	3.2	3.9
9	**2.8**	4.0	4.9	7.9	3.1	**3.0**	6.4	2.8	**2.9**	3.2	4.7	5.1
10	2.7	2.9	3.6	**2.6**	**2.6**	3.0	5.8	3.1	2.3	2.6	3.7	**2.2**

Figure 6: Experimental test set error rates for WebKB course vs. faculty task (top) and Reuters top 10 classes using samples from mod-apte split (bottom). err_1 is obtained using the lowbow geodesic distance with the optimal kernel scale. err_2–err_4 denote using geodesic distance, tf-Cosine similarity and Euclidean distance for bow.

information using the lowbow framework to improve on global bow models. The next section describes similar experiments using SVM on the RCV1 data set which include automatic selection of the scale parameter σ.

5.2 Text Classification using Support Vector Machine

We extended our WebKB and Reuters-21578 text classification experiments to the more recently released and larger RCV1 data set (Lewis et al., 2004). In particular, we focused on the 1 vs. all classification tasks for topics that correspond to leaf nodes in the topic hierarchy and contain no less than 5000 documents. This results in a total of 43 topic codes displayed in Table 1. For further description of the topic hierarchy of the RCV1 data set refer to Lewis et al. (2004).

In our experiments we examined the classification performance of SVM with the Fisher diffusion kernel for bow (Lafferty and Lebanon, 2005) and its corresponding product version for lowbow (10) (which reverts to the kernel of Lafferty and Lebanon (2005) for $\sigma \to \infty$). Our experiments validate the findings in Lafferty and Lebanon (2005) which indicate a significantly poorer performance for linear or RBF kernels. We therefore omit these results and concentrate on comparing the SVM performance for the kernel (10) using various values of σ.

We report the classification performance of SVM using the kernel (10) for three different values of σ: (i) $\sigma \to \infty$ represents the standard bow diffusion kernel of Lafferty and Lebanon (2005) (ii) σ_{opt} represents the best performing scale parameter in terms of test set error rate, and (iii) $\hat{\sigma}_{\text{opt}}$ represents an automatically selected scale parameter based on minimizing the leave-one-out cross validation (loocv) train-set error estimate computed by the SVM-light toolkit. In case of ties, we pick the σ with the smallest value, thus favoring less local smoothing. The loocv estimate is computed at no extra cost and is a convenient way to adaptively estimate σ_{opt}. In all of our experiments below we ignore the role of the diffusion time t in (10) and simply try several different values and choose the best one.

Table 1 reports the test set error rates and standard errors corresponding to the three scales $\hat{\sigma}_{\text{opt}}, \sigma \to \infty, \sigma_{\text{opt}}$ for the selected RCV1 1 vs. all classification tasks. Notice that in general, the lowbow $\hat{\sigma}_{\text{opt}}$ significantly outperforms the standard bow approach. The performance of σ_{opt} further improves on that indicating that a more intelligent scale selection method could result in even lower error rates. Table 1 is also displayed graphically in Figure 8 for $\hat{\sigma}_{\text{opt}}$ and $\sigma \to \infty$. Figure 7 shows the corresponding train set loocv error rates and standard errors.

In our experiments, the sampling of the train and test sets were balanced, with equal number of positive and negative examples. Selections of the optimal σ_{opt} and the estimated $\hat{\sigma}_{\text{opt}}$ were done based on the following set of possible values $\{0.1, 0.15, 0.2, 0.25, 0.3, 0.35, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1, 2, 4, 10, 100\}$. In all the classification tasks, lowbow performs substantially better than bow. The error bars indicate one standard deviation from the mean, and support experimentally the assertion that lowbow has lower variance.

Figure 9 compares the performance of lowbow for $\hat{\sigma}_{\text{opt}}$, $\sigma \to \infty$, and σ_{opt} as a function of the train set size (with the testing size being fixed as 200). As pointed out earlier, the performance of $\hat{\sigma}_{\text{opt}}$ is consistently better than bow with some room for improvement represented by the σ_{opt}.

5.3 Dynamic Time Warping of Lowbow Curves

As presented in the previous sections, the lowbow framework normalizes the time interval $[1, N]$ to $[0, 1]$ thus achieving an embedding of documents of varying lengths in $\mathbb{P}_{V-1}^{[0,1]}$. Proceeding with the

	$\hat{\sigma}_{opt}$	$\sigma \to \infty$ (bow)	σ_{opt}
C11	0.1021 ± 0.0122	0.1234 ± 0.0198	0.0755 ± 0.0071
C12	0.0519 ± 0.0099	0.0664 ± 0.0161	0.0324 ± 0.0072
C13	0.1316 ± 0.0156	0.1527 ± 0.0232	0.1008 ± 0.0088
C14	0.0359 ± 0.0070	0.0537 ± 0.0151	0.0190 ± 0.0050
C1511	0.0494 ± 0.0105	0.0636 ± 0.0163	0.0296 ± 0.0067
C152	0.0860 ± 0.0117	0.1129 ± 0.0239	0.0660 ± 0.0075
C171	0.0522 ± 0.0113	0.0662 ± 0.0185	0.0310 ± 0.0067
C172	0.0313 ± 0.0077	0.0491 ± 0.0134	0.0175 ± 0.0053
C174	0.0066 ± 0.0044	0.0138 ± 0.0080	0.0003 ± 0.0011
C181	0.0634 ± 0.0105	0.0879 ± 0.0174	0.0444 ± 0.0066
C183	0.0283 ± 0.0083	0.0400 ± 0.0135	0.0126 ± 0.0036
C21	0.1269 ± 0.0151	0.1541 ± 0.0298	0.0985 ± 0.0105
C22	0.0614 ± 0.0121	0.0839 ± 0.0235	0.0400 ± 0.0063
C24	0.1009 ± 0.0147	0.1192 ± 0.0267	0.0725 ± 0.0078
C312	0.0494 ± 0.0097	0.0684 ± 0.0176	0.0299 ± 0.0059
C411	0.0321 ± 0.0084	0.0456 ± 0.0109	0.0156 ± 0.0050
C42	0.0550 ± 0.0132	0.0745 ± 0.0211	0.0347 ± 0.0071
E11	0.0356 ± 0.0088	0.0489 ± 0.0174	0.0196 ± 0.0049
E131	0.0213 ± 0.0066	0.0320 ± 0.0115	0.0083 ± 0.0038
E211	0.0372 ± 0.0079	0.0526 ± 0.0151	0.0233 ± 0.0045
E212	0.0293 ± 0.0077	0.0441 ± 0.0142	0.0150 ± 0.0038
E512	0.0568 ± 0.0091	0.0694 ± 0.0186	0.0339 ± 0.0054
E71	0.0051 ± 0.0042	0.0106 ± 0.0084	0.0000 ± 0.0000
G154	0.0155 ± 0.0067	0.0238 ± 0.0127	0.0043 ± 0.0033
GCRIM	0.0533 ± 0.0111	0.0730 ± 0.0164	0.0315 ± 0.0059
GDEF	0.0373 ± 0.0106	0.0526 ± 0.0164	0.0221 ± 0.0047
GDIP	0.0485 ± 0.0112	0.0694 ± 0.0180	0.0309 ± 0.0058
GDIS	0.0306 ± 0.0065	0.0451 ± 0.0190	0.0145 ± 0.0052
GENV	0.0466 ± 0.0106	0.0626 ± 0.0155	0.0301 ± 0.0045
GHEA	0.0298 ± 0.0088	0.0406 ± 0.0154	0.0148 ± 0.0045
GJOB	0.0512 ± 0.0117	0.0628 ± 0.0169	0.0308 ± 0.0057
GPOL	0.0675 ± 0.0099	0.0800 ± 0.0178	0.0434 ± 0.0073
GPRO	0.0624 ± 0.0094	0.0800 ± 0.0204	0.0414 ± 0.0068
GSPO	0.0035 ± 0.0032	0.0095 ± 0.0068	0.0000 ± 0.0000
GVIO	0.0359 ± 0.0080	0.0483 ± 0.0136	0.0185 ± 0.0044
GVOTE	0.0274 ± 0.0076	0.0415 ± 0.0130	0.0126 ± 0.0045
M11	0.0395 ± 0.0099	0.0602 ± 0.0165	0.0213 ± 0.0055
M12	0.0366 ± 0.0096	0.0495 ± 0.0120	0.0200 ± 0.0051
M131	0.0343 ± 0.0087	0.0485 ± 0.0131	0.0184 ± 0.0054
M132	0.0300 ± 0.0085	0.0401 ± 0.0134	0.0141 ± 0.0042
M141	0.0236 ± 0.0070	0.0379 ± 0.0122	0.0106 ± 0.0044
M142	0.0181 ± 0.0061	0.0311 ± 0.0102	0.0065 ± 0.0036
M143	0.0200 ± 0.0067	0.0320 ± 0.0101	0.0076 ± 0.0038

Table 1: Mean and standard error of the test set error rate over 40 realizations of 200 testing and 500 training documents for RCV1 C, E, G, M categories that also appear in Figure 8. Best achievable error rates for lowbow are also reported in the third column.

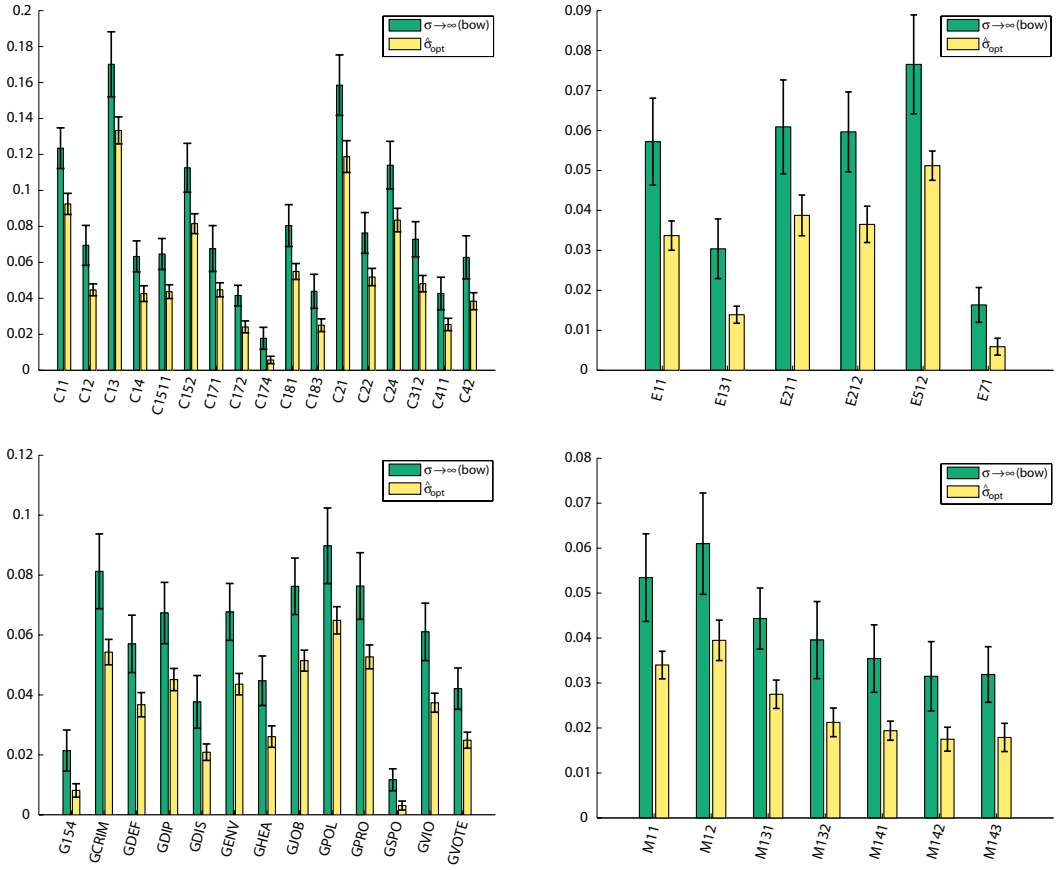


Figure 7: Mean and standard error of train set leave-one-out cross-validation (loocv) error rates. Results are averaged over 40 realizations of 500 training documents with a balanced positive and negative sampling. Lowbow results correspond to $\hat{\sigma}_{\text{opt}}$.

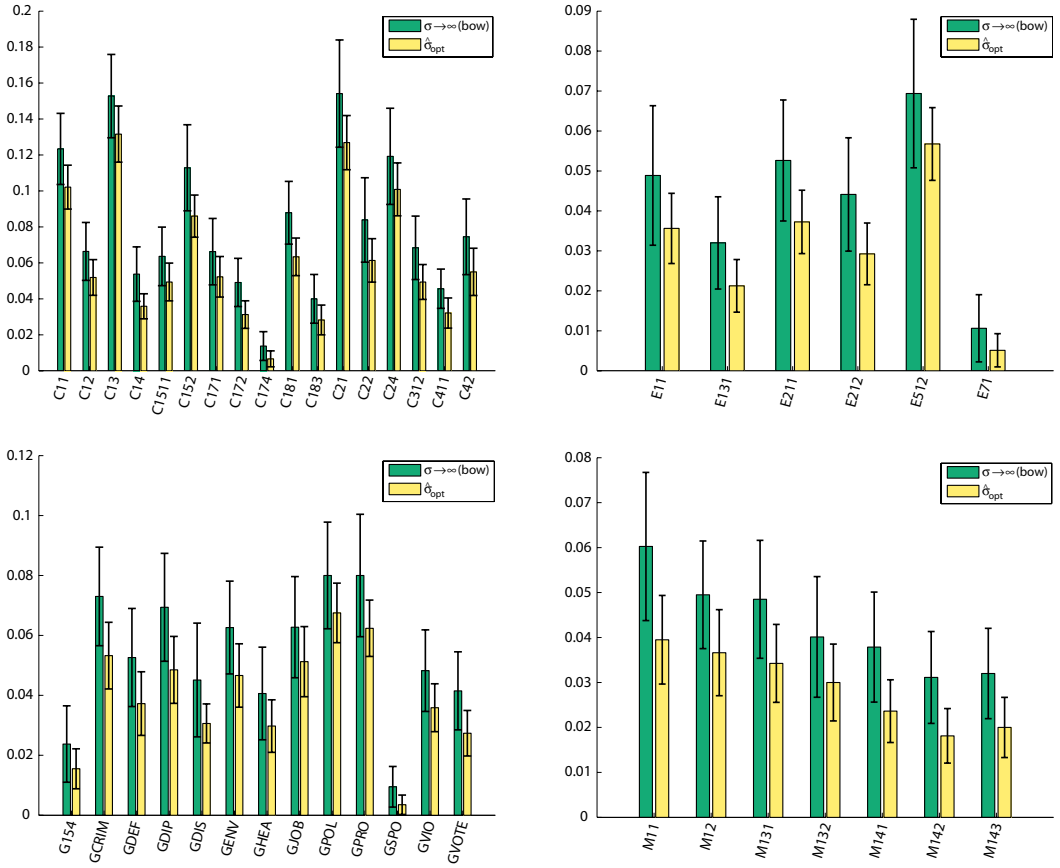


Figure 8: Mean and standard error of test set error rates. Results are averaged over 40 realizations of 500 training and 200 testing documents with a balanced positive and negative sampling. Lowbow results correspond to $\hat{\sigma}_{\text{opt}}$. See Table 1 for associated values.

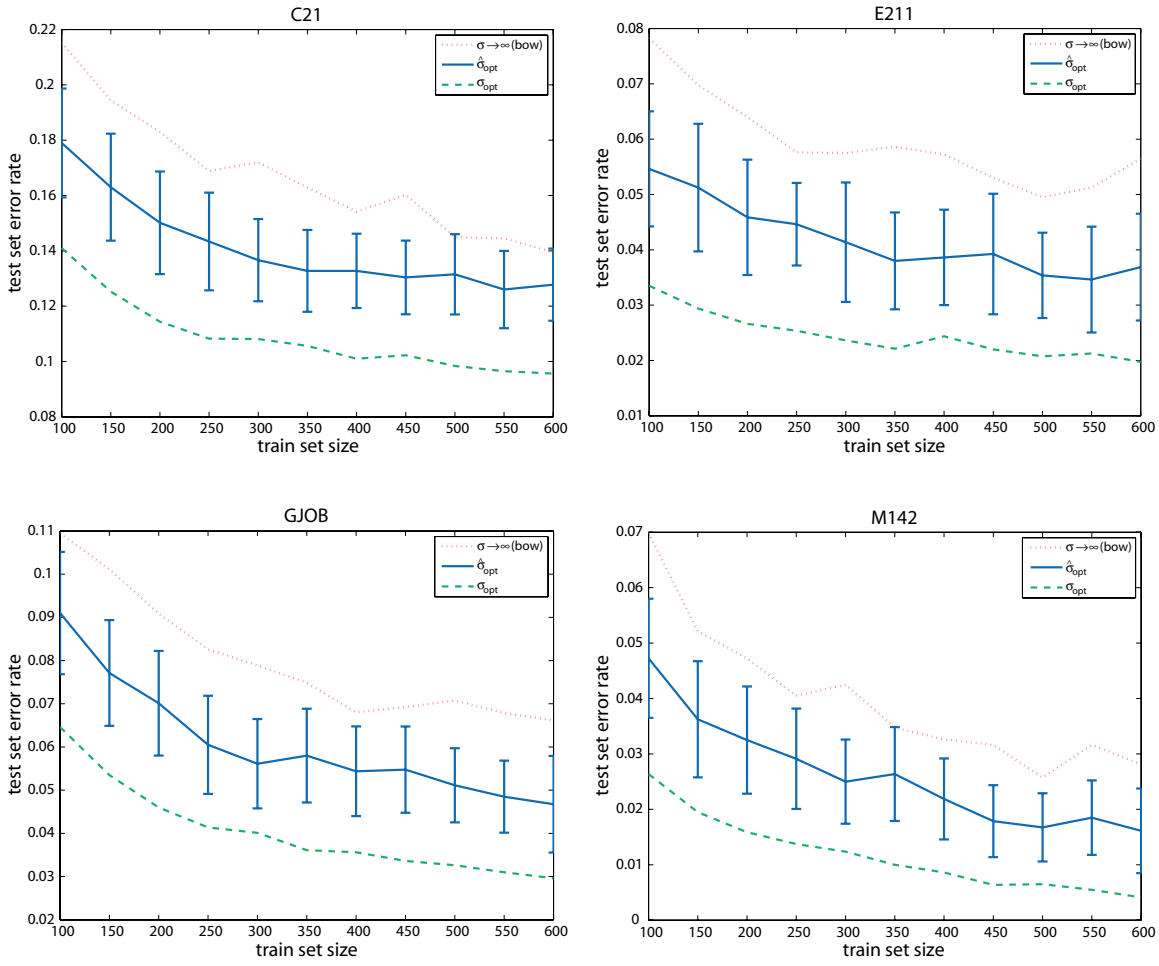


Figure 9: Test set error rate as a function of training size averaged over 40 realizations for RCV1 tasks C21, E211, GJOB and M142 (1 vs. all).

assumption of a product geometry, lowbow representations corresponding to different documents y, z relate to each other by comparing $\gamma_\mu(y)$ to $\gamma_\mu(z)$ for all $\mu \in [0,1]$, for example as is the case in the integrated distance

$$d(\gamma(y),\gamma(z)) = \int_0^1 d(\gamma_\mu(y),\gamma_\mu(z))\, d\mu. \tag{18}$$

This seems reasonable if the two documents y, z share a sequential progression of a similar rate, after normalizing for document length. However, such an assumption seems too restrictive in general as documents of different nature such as news stories and personal webpages are unlikely to posses such similar sequential progression. This assumption also seems untrue to a lesser extent for two documents written by different authors who posses their own individual styles. Such cases can be modeled by introducing time-warping or re-parameterization functions that match the individual temporal domains of lowbow curves to a unique canonical parameterization. Before proceeding to discuss such re-parameterization in the context of lowbow curves we briefly review their use in speech recondition and functional data analysis.

In speech recognition such re-parameterization functions are used to align the time axes corresponding to two speech signals uttered by different individuals or by the same individual under different circumstances. These techniques, commonly referred to as dynamic time warping (DTW) (Sakoe and Chiba, 1978), define the distance between two signals s, r as

$$d(s,r) = \min_{\iota_1,\iota_2 \in I} \int d(s(\iota_1(t)), r(\iota_2(t)))\, dt \tag{19}$$

where I represent the class of smooth monotonic increasing bijections $\iota : [0,1] \to [0,1]$. Using dynamic programming the discretized minimization problem corresponding to (19) can be efficiently computed, resulting in the wide spread use of DTW in the speech recognition community.

Similarly, such time parameterization techniques have been studied in functional data analysis under the name curve registration (Ramsay and Silverman, 2005). In contrast to dynamic time warping, curve registration is usually performed by an iterative procedure aimed at aligning salient features of the data and minimizing the post-alignment residual.

In contrast to the smoothness and monotonic nature of the re-parameterization class I in speech recognition and functional data analysis, it seems reasonable to allow some amount of discontinuity in lowbow re-parameterization. For example, while one document may posses a certain sequential progression, a second document may reverse the appearance of some of the sequential trends. Adjusting the original DTW definition of the re-parameterization family I we obtain the following modified characterization of the class of admissible re-parameterization.

Bijection Re-parameterization $\iota \in I$ are a bijection from $[0,1]$ onto itself.

Piecewise smoothness The re-parameterization functions $\iota \in I$ are piecewise smooth and monotonic, that is, given two partitions of $[0,1]$ to sequences of disjoint intervals $A_1, \ldots, A_r$ with $\cup A_j = [0,1]$ and $B_1, \ldots, B_r$ with $\cup B_j = [0,1]$ we have that for some permutation π over r items, $\iota : A_j \to B_{\pi(j)}$ is a smooth monotonic increasing bijection for all $j = 1, \ldots, r$.

The requirement above of piecewise continuity seems reasonable as it is natural to expect some re-ordering among sections or paragraphs of similar documents. Using a combination of dynamic programming similar to the one of Sakoe and Chiba (1978) and a variation of earth mover distance

(Rubner et al., 2000) known as the Hungarian algorithm (Munkres, 1957), the minimization problem (19) over the class I described above may be computed efficiently.

We conducted a series of experiments examining the benefit in introducing dynamic time warping or registration in text classification. Somewhat surprisingly, adding dynamic time warping or registration to lowbow classification resulted in only a marginal modification of the distances and consequently only a marginal improvement in classification performance. There are two reasons for this relatively minor effect introduced by the dynamic time warping. First, the RCV1 corpus for which these experiments were conducted consists of documents containing a fairly homogeneous semantic structure and presentation. As such, the curves can reasonably be compared by using integrated distances or kernels without a need for re-parameterization. Second, the local smoothing inherent in the lowbow representation makes it fairly robust to some amount of temporal misalignment. In particular, by selecting the kernel scale parameter appropriately we are able to prevent unfortunate effects due to different sequential parameterizations. Although surprising, this is indeed a positive result as it indicates that the lowbow representation is relatively robust to different time parameterization, at least when applied to documents sharing similar structure such as news stories in RCV1 corpus or webpages in the WebKB data set.

5.4 Text Segmentation

Text segmentation is the task of discovering topical boundaries inside documents, for example transcribed news-wire data. In general, this task is hard to accomplish using low order n-gram information. Most methods use a combination of longer range n-grams and other sequential features such as trigger pairs. Our approach in this section is not to carefully construct a state-of-the-art text segmentation system but rather to demonstrate the usefulness of the continuous lowbow representation in this context. More information on text segmentation and a recent exponential model-based approach may be found in Beeferman et al. (1999).

The boundaries between different text segments, by definition, separate document parts containing different word distributions. In the context of lowbow curves, this would correspond to sudden dramatic shifts in the curve location. Due to the continuity of the lowbow curves, such sudden movements may be discovered by considering the gradient vector field $\dot{\gamma}_\mu$ along the lowbow curve. In addition to containing predictive information that can be used in segmentation models, the gradient enables effective visualization of the instantaneous change that is central to human-assisted segmentation techniques.

To illustrate the role of the gradient $\dot{\gamma}_\mu(y)$ in segmentation we examine its behavior for a document y containing clear and pre-determined segments. Following Beeferman et al. (1999), we consider documents y created by concatenating news stories which resemble the continuous transcription of news stories. We examine the behavior of the lowbow curve and its gradient for two documents created in this fashion. The curves were sampled at 100 equally spaced points and the gradient is approximated by the finite difference in word histograms localized at adjacent time points. Generally speaking, for purpose of visualization the number of sampling points should be proportional to the length of the document in order to accurately capture the change in the local word histogram. This is different from the classification task which is not sensitive to the choice of the number of samples and thus favors a small value in the interest of lowering the computational complexity associated with classification.

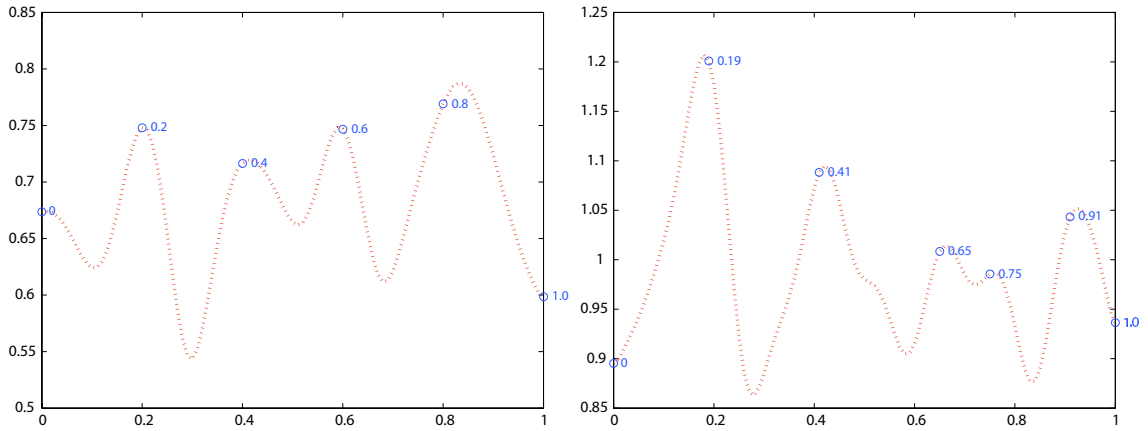


Figure 10: Velocity of the lowbow curve as a function of t. Left: five randomly sampled new stories of equal size ($\sigma = 0.08$). Right: three successive RCV1 news articles of varying lengths ($\sigma = 0.065$).

The first document was created by concatenating five randomly sampled news stories from the Wall Street Journal data set. To ensure that the different segments will be of equal length, we removed the final portions of the longer stories thus creating predetermined segment borders at $\mu = 0.2, 0.4, 0.6, 0.8$. The gradient norm $\|\dot{\gamma}_\mu(y)\|_2$ of this document is displayed in the the left panel of Figure 10. Notice how the 4 equally spaced internal segment borders (displayed by the numbered circles in the figure), correspond almost precisely, to the local maxima of the gradient norm.

The second document, represents a more realistic scenario where the segments correspond to successive news stories of varying lengths. We created it by randomly picking three successive news articles from the RCV1 collection (document id: 18101, 18102 and 18103) and concatenating them into a single document. The two internal segment borders occur at $\mu = 0.19$ and $\mu = 0.41$ (the last story is obviously longer than the first two stories). The right panel of Figure 10 displays the gradient norm $\|\dot{\gamma}_\mu\|_2$ for the corresponding lowbow curve. The curve has five local maxima, with the largest two local maxima corresponding almost precisely to the segment borders at $\mu = 0.19$ and $\mu = 0.41$. The three remaining local maxima correspond to internal segment boundaries within the third story. Indeed, the third news story begins with discussion of London shares and German stocks; it then switches to discuss French stocks at point $\mu = 0.65$ before switching again at $\mu = 0.75$ to talk about how the Bank of Japan's quarterly corporate survey affects the foreign exchange. The story moves on at $\mu = 0.95$ to discuss statistics of today's currencies and stock market. As with the different news story boundaries, the internal segment boundaries of the third story closely match the local maxima of the gradient norm.

The lowbow curve itself carries additional information beyond the gradient norm for segmentation purposes. Portions of the curve corresponding to different segments will typically contain different local word histograms and will therefore occupy different parts of the simplex. Sampling the curve at an equally spaced time grid $\{\mu_1, \ldots, \mu_k\} \subset [0,1]$ and clustering the points $\{\gamma_{\mu_1}(y), \ldots, \gamma_{\mu_k}(y)\}$ reveals distinct segments as distinct clusters. A similar approach uses partial human feedback to present to a user a low dimensional embedding of $\{\gamma_{\mu_1}(y), \ldots, \gamma_{\mu_k}(y)\}$ given his or her choice of the scale parameter. The benefit in doing so is that it is typically much easier to visualize graphics than

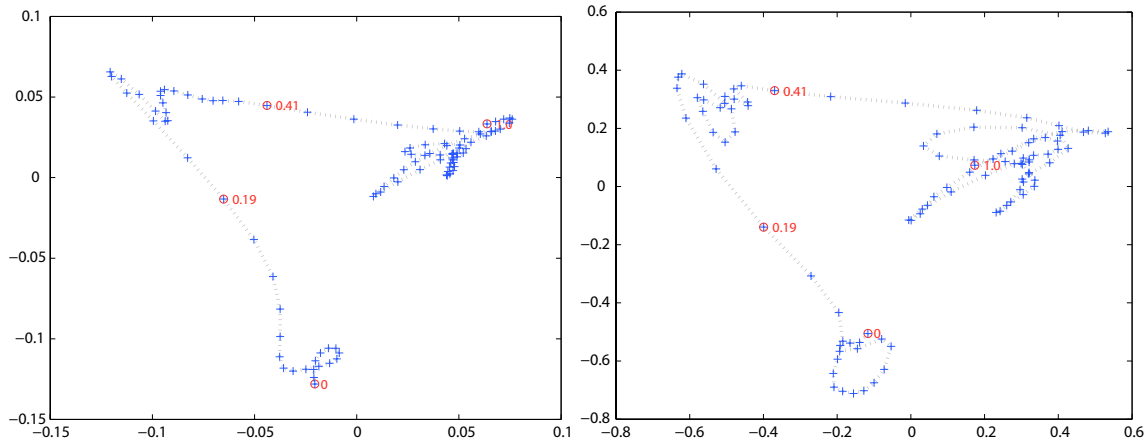


Figure 11: 2D embeddings of the lowbow curve representing the three successive RCV1 stories (see text for more details) using PCA (left, $\sigma = 0.02$) and MDS (right, $\sigma = 0.01$).

text content. Such techniques for rapid document visualization and browsing are also illustrated in the next section.

Figure 11 shows the 2D projection of the lowbow curve for the three concatenated RCV1 stories mentioned above. To embed the high dimensional curve in two dimensions we used principal component analysis (PCA) (left panel) and multidimensional scaling using the Fisher geodesic distance (right). The blue crosses indicate the positions of the sampled points in the low dimensional embedding while the red circles correspond to the segment boundaries of the three RCV1 documents. In both figures, $\{\gamma_{\mu_1}(y), \ldots, \gamma_{\mu_k}(y)\}$ are naturally grouped into three clusters, indicating the presence of three different segments. The distance between successive points near the segment boundaries is relatively large which demonstrates the high speed of the lowbow curve at these points (compare it with the gradient norm in right panel of Figure 10).

5.5 Text Visualization

We conclude the experiments with a text visualization demonstration based on the current journal article. Visualizing this document has the added benefit that the reader is already familiar with the text, hopefully having read it carefully thus far. Additional visualization applications of the lowbow framework may be found in Mao et al. (2007).

The gradient norm $\|\dot{\gamma}(t)\|_2$ of the lowbow curve of this paper is displayed in Figure 12. The marks indicate the beginning of each section that are identified by numbers in parentheses, for example, Section 2 begins at $\mu = 0.085$. Almost all of the local maxima of the curve correspond precisely to the section boundaries, with some interesting exceptions. For example, the global maximum occurs near $\mu = 0.17$ where we finish the lowbow definition (Definition 6) and start proving its properties (Theorem 1). Interestingly, the gradient speed does not distinguish between the two subsections concerning nearest neighbor and SVM classification experiments ($\mu = 0.61$). In performing the above experiment, the abstract, references and this subsection (Section 5.5) are excluded from the generation of the lowbow curve and equations were replaced by special markers.

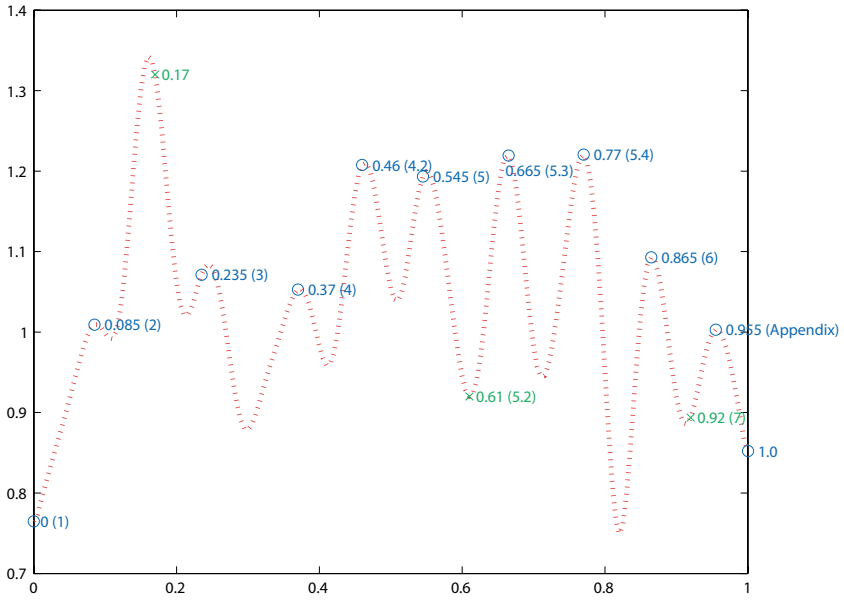


Figure 12: Velocity of the lowbow curve computed for this paper as a function of μ ($\sigma = 0.04$). Abstract, references and Section 5.5 are excluded from curve generation. The marks indicate the beginning of each section that are identified by numbers in parentheses, for example, Section 2 begins at $\mu = 0.085$.

Figure 13 depicts 2D projections of the lowbow curve corresponding to Sections 5.1–5.4 (section boundaries occurring at $\mu = \{0.2, 0.38, 0.72\}$) using PCA (left) and MDS (right) based on Fisher geodesic distance. As previously demonstrated, the lowbow curve nicely reveals three clusters corresponding to the different subsections with the exception of not distinguishing between the nearest neighbor and SVM experiments. Using interactive graphics it is possible to extract more information from the lowbow curves by examining the 3D PCA projection, displayed in Figure 14. The dense clustering of the points at the beginning of the 2D curve is separated in the 3D figure, however, there is no way to separate the crossing at the end of the curve in both 3D and 2D.

6. Related Work

The use of n-gram and bow has a long history in speech recognition, language modeling, information retrieval and text classification. Recent monographs surveying these areas are Jelinek (1998), Manning and Schutze (1999), and Baeza-Yates and Ribeiro-Neto (1999). In speech recognition and language modeling n-grams are used typically with $n = 1, 2, 3$. In classification, on the other hand, 1-grams or bow are the preferred option. While some attempts have been made to use bi-grams and tri-grams in classification as well as to incorporate richer representations, the bow representation is still by far the most popular.

Comparisons of several statistical methods for text classification using the bow representation may be found in Yang (1999). Joachims (2000) applies support vector machines to text classification using various 1-gram representations. Zhang and Oles (2001) consider several regularized linear

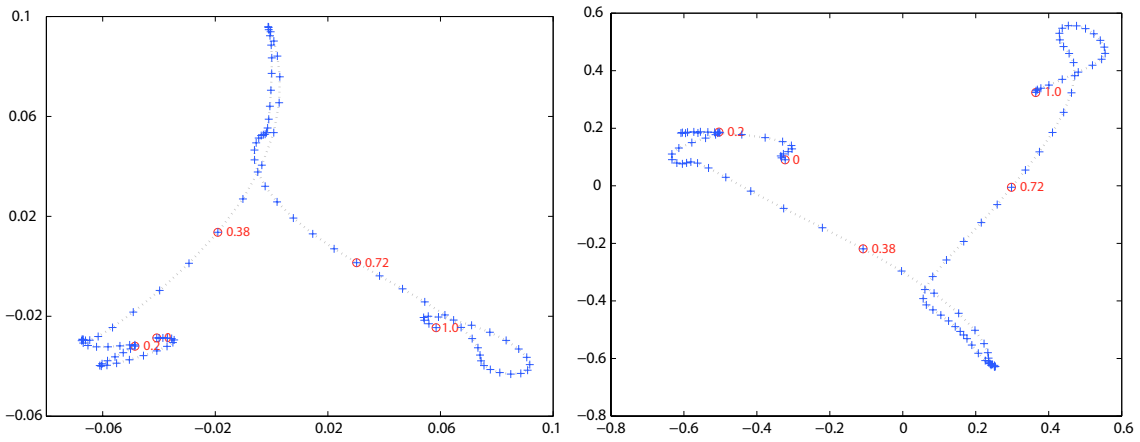


Figure 13: 2D embeddings of the lowbow curve computed for Section 5.1–5.4 using PCA (left, $\sigma = 0.03$) and MDS (right, $\sigma = 0.02$).

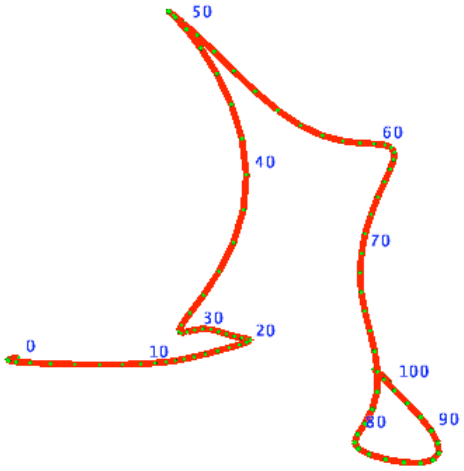


Figure 14: 3D embeddings of the lowbow curve computed for Section 5.1–5.4 using PCA ($\sigma = 0.03$). The numbers are $\mu \times 100$.

classifiers and Schapire and Singer (2000) experiment with AdaBoost. In general, most papers use a representation that is based on the word histogram with L_2 or L_1 normalization, binary word appearance events, or tfidf. The differences between the above representations tend to be minor and there is no clear conclusion which precise representation works best. Once the representation has been fixed, it is generally accepted that support vector machines with linear or rbf kernels result in the state-of-the-art performance, with logistic regression slightly trailing behind.

A geometric point of view considering the bow representation as a point in the multinomial simplex is expressed in Lebanon (2005) and Lafferty and Lebanon (2005). A recent overview of the geometrical properties of probability spaces is provided in Kass (1989) and Amari and Nagaoka (2000). The use of simplicial curves in text modeling is a relatively new approach but has been previously considered by Gous (1998) and Hall and Hofmann (2000). However, in contrast to these papers we represent a single document, rather than a corpus, as a curve in the simplex. The use of the heat or diffusion kernel in machine learning appeared first in Kondor and Lafferty (2002) in the context of graphs and later in Lafferty and Lebanon (2003) in the context of Riemannian manifolds. Cuturi (2005) describes some related ideas that lead to a non-smooth multi-scale view of images. These ideas were later expanded (Cuturi et al., 2007) to consider dynamic time warping which is highly relevant to the problem of matching two lowbow curves.

Modeling functional data such as lowbow curves in statistics has been studied in the context of functional data analysis. The recent monograph by Ramsay and Silverman (2005) provides an interesting survey and advocates the use of continuous representations even for data that is normally obtained in a discrete form. Wand and Jones (1995) and Loader (1999) provide a recent overview of local smoothing in statistics which is closely related to the lowbow framework.

Document visualization solutions were generally considered for visualizing a corpus of documents rather than visualizing a single document. Typical approaches include dimensionality reduction of the bow representation using methods such as multi-dimensional scaling and PCA. For examples see Fortuna et al. (2005) and Havre et al. (2002). IN-SPIRE is a document visualization tool[2] developed at Pacific Northwest National Lab that uses related ideas for corpus visualization. Blei and Lafferty (2006) use a dynamic extension of latent Dirichlet allocation (Blei et al., 2003) to explore and visualize temporal changes in a corpus of time-stamped documents. In contrast to most of the studies mentioned above, we are concerned with the sequentially modeling of a single document, rather than a corpus, at one or more sequential resolutions.

7. Discussion

The lowbow representation is a promising new direction in text modeling. By varying σ it interpolates between the standard word sequence representation $\langle y_1, \ldots, y_N \rangle$ and bow. In contrast to n-gram, it captures topical trends and incorporates long range information. On the other hand, the lowbow novelty is orthogonal to n-gram as it is possible to construct lowbow curves over n-gram counts.

Under our current model, two different lowbow curves are compared in a point-wise manner. When an attempt was made to register the curves by constructing a time warping function to synchronize the two curves, little or no improvement was found. This is due partly to the nature of the data and partly to the robustness of the lowbow representation being insensible to time-misalignment by adapting the scale parameter.

2. IN-SPIRE can be found at `http://in-spire.pnl.gov/`.

In this paper we have focused on analyzing lowbow curves at one particular scale σ. An alternative and potentially more powerful approach is to consider a family of lowbow curves corresponding to a collection of scale parameters σ. The resulting family of curves constitute a multiresolution representation of documents similar to the use of Gabor filters and wavelets in signal processing. Exploring such sequential multiresolution representation and their relationship to wavelets should be more closely examined.

The lowbow framework is aesthetically pleasing, and achieves good results both in terms of numeric classification accuracy and in terms of presenting a convenient visual text representation to a user. Using a smoothing kernel, it naturally interpolates between bow and complete sequential information. In contrast to categorical smoothing methods employed by n-grams (such as back-off and interpolation) lowbow employs temporal smoothing which is potentially more powerful due to its ordinal nature. The correspondence with smooth curves in the simplex enables the use of a wide array of tools from differential geometry and analysis that are otherwise unavailable to standard discrete text representations.

Acknowledgments

The authors wish to thank Jeff Bilmes for his interesting comments regarding the lowbow representation, and the anonymous reviewers for their helpful suggestions. This work was supported in part by NSF grant DMS-0604486.

Appendix A. The Multinomial Simplex and its Geometry

In this section, we present a brief description of the multinomial simplex and its information geometry. Since the simplex is the space of bow representations, its geometry is crucial to the lowbow representation. The brief description below uses some concepts from Riemannian geometry. For additional information concerning the geometry of the simplex refer to Kass and Voss (1997), Amari and Nagaoka (2000), or Lebanon (2005). Standard textbooks on differential and Riemannian geometry are Spivak (1975), Lee (2002), and Boothby (2003).

The multinomial simplex $\mathbb{P}_m$ for $m > 0$ is the m-dimensional subset of $\mathbb{R}^{m+1}$ of all probability vectors or histograms over $m+1$ objects

$$\mathbb{P}_m = \left\{ \theta \in \mathbb{R}^{m+1} : \forall i \; \theta_i \geq 0, \sum_{j=1}^{m+1} \theta_j = 1 \right\}.$$

Its connection to the multinomial distribution is that every $\theta \in \mathbb{P}_m$ corresponds to a multinomial distribution over $m+1$ items.

The simplex definition above includes the boundary of vectors with zero probabilities. In order to formally consider the simplex as a differentiable manifold we need to remove that boundary and consider only strictly positive probability vectors. A discussion concerning the positivity restriction and its relative unimportance in practice may be found in Lafferty and Lebanon (2005).

The topological structure of $\mathbb{P}_m$, which determines the notions of convergence and continuity, is naturally inherited from the standard topological structure of the embedding space $\mathbb{R}^{m+1}$. The geometrical structure of $\mathbb{P}_m$ that determines the notions of distance, angle and curvature is determined by a local inner product $g_\theta(\cdot,\cdot), \theta \in \mathbb{P}_m$, called the Riemannian metric. The most obvious

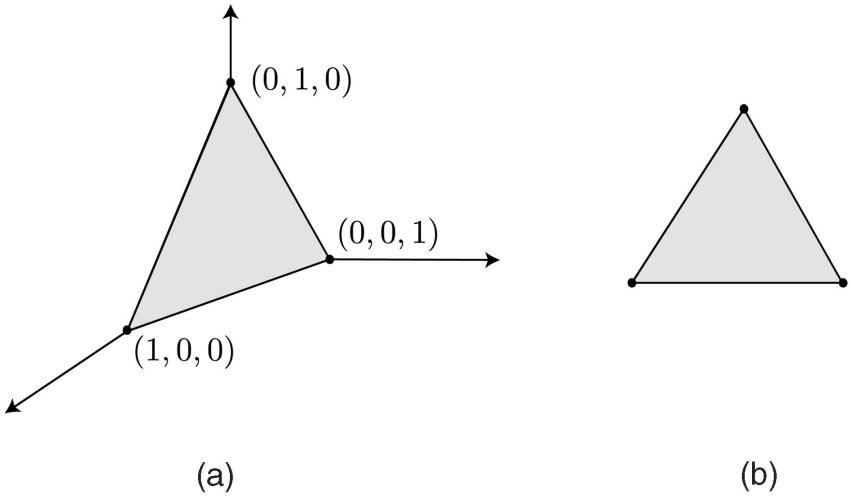


Figure 15: The 2-simplex $\mathbb{P}_2$ may be visualized as a surface in $\mathbb{R}^3$ (left) or as a triangle in $\mathbb{R}^2$ (right).

choice, perhaps, is the standard Euclidean inner product $g_\theta(u,v) = \sum u_i v_i$. However, such a choice is problematic from several aspects (Lebanon, 2005). A more motivated choice for a local inner product on the simplex is the Fisher information metric

$$\begin{aligned} g_\theta(u,v) &= \sum_{ij} u_i v_j \mathsf{E}_{p_\theta}\left(\frac{\partial \log p_\theta(x)}{\partial \theta_i}\frac{\partial \log p_\theta(x)}{\partial \theta_j}\right) \\ &= \sum_{i=1}^{m+1} \frac{u_i v_i}{\theta_i} \end{aligned} \tag{20}$$

where $p_\theta(x)$ above is the multinomial probability associated with the parameter θ. It can be shown that the Fisher information metric is the only invariant metric under sufficient statistics transformations (Čencov, 1982; Campbell, 1986). In addition, various recent results motivate the Fisher geometry from a practical perspective (Lafferty and Lebanon, 2005).

The inner product (20) defines the geometric properties of distance, angle and curvature on $\mathbb{P}_m$ in a way that is quite different from the Euclidean inner product. The distance function $d : \mathbb{P}_m \times \mathbb{P}_m \to [0, \pi/2]$ corresponding to (20) is

$$d(\theta,\eta) = \arccos\left(\sum_{i=1}^{m+1}\sqrt{\theta_i \eta_i}\right) \quad \theta,\eta \in \mathbb{P}_m. \tag{21}$$

The distance function (21) and the Euclidean distance function $d(\theta,\eta) = \sqrt{\sum(\theta_i - \eta_i)^2}$ resulting from the inner product $g_\theta(u,v) = \sum u_i v_i$ on $\mathbb{P}_2$ are illustrated in Figures 15-16.

By determining geometric properties such as distance, the choice of metric for $\mathbb{P}_m$ is of direct importance to the bow representation of documents and its modeling. For example, while the Euclidean metric is homogeneous across the simplex, the Fisher metric (20) emphasizes the area close to the boundary. In addressing the question of modeling the lowbow curves, the geometry of the simplex plays a central role. It dictates notions such as the distance between two curves,

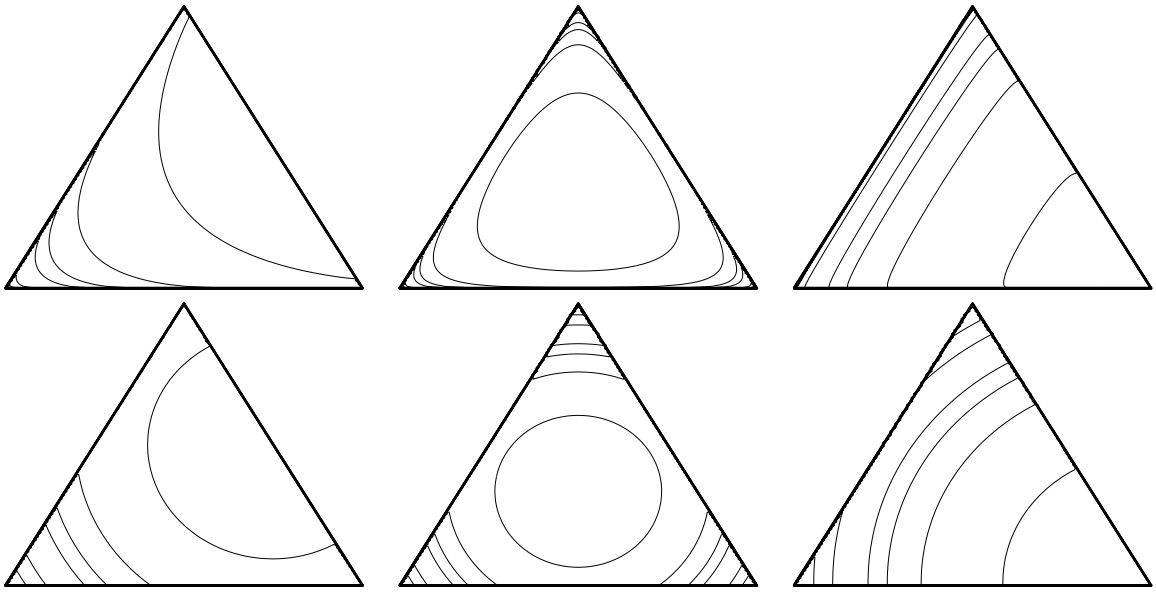


Figure 16: Equal distance contours on $\mathbb{P}_2$ from the upper right edge (left column), the center (center column), and lower right corner (right column). The distances are computed using the Fisher information metric (top row) and the Euclidean metric (bottom row).

the instantaneous direction of a curve, and the curvature or complexity of a curve. Understanding the relationship between these geometric notions and g_θ is a necessary prerequisite for modeling documents using the lowbow representation.

References

S.-I. Amari and H. Nagaoka. *Methods of Information Geometry*. American Mathematical Society, 2000.

M. Anthony and P. Bartlett. *Neural Network Learning: Theoretical Foundations*. Cambridge University Press, 1999.

R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval*. Addison Wesley, 1999.

D. Beeferman, A. Berger, and J. D. Lafferty. Statistical models for text segmentation. *Machine Learning*, 34(1-3):177–210, 1999.

M. Berger, P. Gauduchon, and E. Mazet. Le spectre d'une varieté Riemannienne. *Lecture Notes in Mathematics*, Vol. 194, Springer-Verlag, 1971.

D. Blei and J. Lafferty. Dynamic topic models. In *Proceedings of the Twenty-Third International Conference on Machine Learning*, pages 113–120, 2006.

D. Blei, A. Ng, , and M. Jordan. Latent Dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, 2003.

W. M. Boothby. *An Introduction to Differentiable Manifolds and Riemannian Geometry*. Academic Press, 2003.

L. L. Campbell. An extended Čencov characterization of the information metric. *Proceedings of the American Mathematical Society*, 98(1):135–141, 1986.

N. N. Čencov. *Statistical Decision Rules and Optimal Inference*. American Mathematical Society, 1982.

S. Chen and R. Rosenfeld. A survey of smoothing techniques for ME models. *IEEE Transactions on Speech and Audio Processing*, 8(1), 2000.

J. P. R. Christensen, C. Berg, and P. Ressel. *Harmonic Analysis on Semi-Groups*. Springer, 1984.

W. S. Cleveland and C. L. Loader. *Statistical Theory and Computational Aspects of Smoothing*, chapter Smoothing by Local Regression: Principles and Methods, pages 10–49. Springer, 1996.

M. Cuturi. *Learning from Structured Objects with Semigroup Kernels*. PhD thesis, Ecole des Mines de Paris, 2005.

M. Cuturi, J.-P, Vert, O. Birkenes, and T. Matsui. A kernel for time series based on global alignments. In *Proc. of the 32nd IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 2, pages 413–416, 2007.

R. M. Dudley. A course on empirical processes. *Lecture Notes in Mathematics*, 1097:2–142, 1984.

G. Forman. Tackling concept drift by temporal inductive transfer. In *Proc. of the ACM SIGIR Conference*, 2006.

B. Fortuna, M. Grobelnik, and D. Mladenic. Visualization of text document corpus. *Informatica*, 29(4):497–504, 2005.

A. Gous. *Exponential and Spherical Subfamily Models*. PhD thesis, Stanford University, 1998.

Y. Guo, P. L. Bartlett, J. Shawe-Taylor, and R. C. Williamson. Covering numbers for support vector machines. *IEEE Transaction on Information Theory*, 48(1):239–250, 2002.

K. Hall and T. Hofmann. Learning curved multinomial subfamilies for natural language processing and information retrieval. In *Proc. of the 17th International Conference on Machine Learning*, pages 351–358, 2000.

S. Havre, E. Hetzler, P. Whitney, and L. Nowell. Themeriver: Visualizing thematic changes in large document collections. *IEEE Transactions on Visualization and Computer Graphics*, 8(1):9–20, 2002.

M. Hein and O. Bousquet. Hilbertian metrics and positive definite kernels on probability measures. In *Proceedings of AI and Statistics*, pages 136–143, 2005.

R. A. Horn and C. R. Johnson. *Matrix Analysis*. Cambridge University Press, 1990.

F. Jelinek. *Statistical Methods for Speech Recognition*. MIT Press, 1998.

T. Joachims. *The Maximum Margin Approach to Learning Text Classifiers Methods, Theory and Algorithms*. PhD thesis, Dortmund University, 2000.

R. E. Kass. The geometry of asymptotic inference. *Statistical Science*, 4(3):188–234, 1989.

R. E. Kass and P. W. Voss. *Geometrical Foundation of Asymptotic Inference*. John Wiley & Sons, 1997.

R. Kondor and J. Lafferty. Diffusion kernels on graphs and other discrete structures. In *Proceedings of the 19th International Conference on Machine Learning*, 2002.

J. Lafferty and G. Lebanon. Information diffusion kernels. In *Advances in Neural Information Processing, 15*. MIT Press, 2003.

J. Lafferty and G. Lebanon. Diffusion kernels on statistical manifolds. *Journal of Machine Learning Research*, 6:129–163, January 2005.

G. Lebanon. *Riemannian Geometry and Statistical Machine Learning*. PhD thesis, Carnegie Mellon University, 2005.

G. Lebanon and Y. Zhao. Local likelihood modeling of the concept drift phenomenon. Technical Report 07-10, Statistics Department, Purdue University, 2007.

J. M. Lee. *Introduction to Smooth Manifolds*. Springer, 2002.

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.

C. Loader. *Local Regression and Likelihood*. Springer, 1999.

C. D. Manning and H. Schutze. *Foundations of Statistical Natural Language Processing*. MIT Press, 1999.

Y. Mao, J. Dillon, and G. Lebanon. Sequential document visualization. *IEEE Transactions on Visualization and Computer Graphics*, 13(6), 2007.

J. Munkres. Algorithms for the assignment and transportation problems. *Journal of the Society for Industrial and Applied Mathematics*, 5(1):32–38, 1957.

J. Ramsay and B. W. Silverman. *Functional Data Analysis*. Springer, second edition, 2005.

J. O. Ramsay and C. J. Dalzell. Some tools for functional data analysis. *Journal of the Royal Statistical Society B*, 53(3):539–572, 1991.

Y. Rubner, C. Tomasi, and L. J. Guibas. The earth mover's distance as a metric for image retrieval. *International Journal of Computer Vision*, 40(2), 2000.

H. Sakoe and S. Chiba. Dynamic programming algorithm optimization for spoken word recognition. *IEEE Trans. Acoust. Speech Signal Process*, 26(1):43–49, 1978.

R. E. Schapire and Y. Singer. Boostexter: A boosting-based system for text categorization. *Machine Learning*, 39(2/3):135–168, 2000.

M. Spivak. *A Comprehensive Introduction to Differential Geometry*, volume 1-5. Publish or Perish, 1975.

V. N. Vapnik. *Statistical Learning Theory*. John Wiley & Sons, 1998.

M. P. Wand and M. C. Jones. *Kernel Smoothing*. Chapman and Hall/CRC, 1995.

R. C. Williamson, A. J. Smola, and B. Schölkopf. Generalization performance of regularization networks and support vector machines via entropy numbers of compact operators. *IEEE Transactions on Information Theory*, 47(6):2516–2532, 2001.

Y. Yang. An evaluation of statistical approaches to text categorization. *Journal of Information Retrieval*, 1(1/2):67–88, 1999.

Chengxiang Zhai and John Lafferty. A study of smoothing methods for language models applied to ad hoc information retrieval. In *Proc. of ACM-SIGIR conference*, 2001.

T. Zhang. Covering number bounds for certain regularized linear function classes. *Journal of Machine Learning Research*, 2:527–550, 2002.

T. Zhang and F. J. Oles. Text categorization based on regularized linear classification methods. *Information Retrieval*, 4:5–31, April 2001.

Journal of Machine Learning Research 8 (2007) 2443-2466 Submitted 7/07; Published 10/07

The Need for Open Source Software in Machine Learning

Sören Sonnenburg* SOEREN.SONNENBURG@FIRST.FRAUNHOFER.DE
Fraunhofer Institute FIRST
Kekulestr. 7
12489 Berlin, Germany

Mikio L. Braun* MIKIO@CS.TU-BERLIN.DE
Technical University Berlin
Franklinstr. 28/29
10587 Berlin, Germany

Cheng Soon Ong* CHENGSOON.ONG@TUEBINGEN.MPG.DE
Friedrich Miescher Laboratory
Max Planck Society
Spemannstr. 39
72076 Tübingen, Germany

Samy Bengio BENGIO@GOOGLE.COM
Google
1600 Amphitheatre Pkwy, Building 47-171D
Mountain View, CA 94043, USA

Leon Bottou LEON@BOTTOU.ORG
NEC Laboratories America, Inc.
4 Independence Way Suite 200
Princeton NJ 08540 , USA

Geoffrey Holmes GEOFF@CS.WAIKATO.AC.NZ
Department of Computer Science
University of Waikato
Hamilton, New Zealand

Yann LeCun YANN@CS.NYU.EDU
New York University
715 Broadway
New York, NY 10003, USA

Klaus-Robert Müller KRM@CS.TU-BERLIN.DE
Technical University Berlin
Franklinstr. 28/29
10587 Berlin, Germany

Fernando Pereira PEREIRA@CIS.UPENN.EDU
University of Pennsylvania
3330 Walnut Street
Philadelphia, PA 19104, USA

Carl Edward Rasmussen CER54@CAM.AC.UK
Department of Engineering
Trumpington Street
Cambridge, CB2 1PZ, United Kingdom

∗. The first three authors contributed equally.

Gunnar Rätsch GUNNAR.RAETSCH@TUEBINGEN.MPG.DE
Friedrich Miescher Laboratory
Max Planck Society
Spemannstr. 39
72076 Tübingen, Germany

Bernhard Schölkopf BS@TUEBINGEN.MPG.DE
Max Planck Institute for Biological Cybernetics
Spemannstr. 38
72076 Tübingen, Germany

Alexander Smola ALEX.SMOLA@GMAIL.COM
Australian National University and NICTA
Canberra, ACT 0200, Australia

Pascal Vincent VINCENTP@IRO.UMONTREAL.CA
Université de Montréal
Dept. IRO, CP 6128, Succ. Centre-Ville
Montréal, Québec, Canada

Jason Weston JASONW@NEC-LABS.COM
NEC Laboratories America, Inc.
4 Independence Way Suite 200
Princeton NJ 08540 , USA

Robert C. Williamson BOB.WILLIAMSON@ANU.EDU.AU
Australian National University and NICTA
Canberra, ACT 0200, Australia

Editor: David Cohn

Abstract

Open source tools have recently reached a level of maturity which makes them suitable for building large-scale real-world systems. At the same time, the field of machine learning has developed a large body of powerful learning algorithms for diverse applications. However, the true potential of these methods is not used, since existing implementations are not openly shared, resulting in software with low usability, and weak interoperability. We argue that this situation can be significantly improved by increasing incentives for researchers to publish their software under an open source model. Additionally, we outline the problems authors are faced with when trying to publish algorithmic implementations of machine learning methods. We believe that a resource of peer reviewed software accompanied by short articles would be highly valuable to both the machine learning and the general scientific community.

Keywords: machine learning, open source, reproducibility, creditability, algorithms, software

1. Introduction

The field of machine learning has been growing rapidly, producing a wide variety of learning algorithms for different applications. The ultimate value of those algorithms is to a great extent judged by their success in solving real-world problems. Therefore, algorithm replication and application to new tasks are crucial to the progress of the field.

However, few machine learning researchers currently publish the software and/or source code associated with their papers (Thimbleby, 2003). This contrasts for instance with the practices of the bioinformatics community, where open source software has been the foundation of further research (Strajich and Lapp, 2006). The lack of openly available algorithm implementations is a major obstacle to scientific progress in and beyond our community.

We believe that open source sharing of machine learning software can play a very important role in removing that obstacle. The open source model has many advantages which will lead to better reproducibility of experimental results: quicker detection of errors, innovative applications, and faster adoption of machine learning methods in other disciplines and in industry. However, incentives for polishing and publishing software are at present lacking. Published software *per se* does not have a standard, accepted means of citation in our field, and is thus invisible with respect to impact measurement tools like citation statistics: at present the only way of referring to it is by citing the paper which describes the *theory* associated with the code or alternatively by citing the user's manual which has been released in the form of some technical report, such as Benson et al. (2004). To address this difficulty, we propose a method for formal publication of machine learning software, similar to what the ACM Transactions on Mathematical Software provide for Numerical Analysis.

This paper is structured as follows: First, we briefly explain the idea behind open source software (Section 2). A widespread adoption of this publication model would have several positive effects which we outline in Section 3. Next, we discuss current obstacles, and propose possible changes in order to improve this situation (Section 4). Finally, we propose a new, separate, ongoing track for machine learning open source software in JMLR (JMLR-MLOSS) in Section 5. We provide an overview about open source licenses in Appendix A and guidelines for good machine learning software in Appendix B.

2. Open Source and Science

If I have seen further it is by standing on the shoulders of giants.

—Sir Isaac Newton (1642–1727)

The basic idea of open source software is very simple; programmers or users can read, modify and redistribute the source code of a piece of software (Gacek and Arief, 2004). While there are various licenses of open source software (cf. Appendix A; Lin et al., 2006; Välimäki, 2005) they all share a common ideal, which is to allow free exchange and use of information. The open source model replaces central control with collaborative networks of contributors. Every contributor can build on the work that has been done by others in the network, thus minimizing time spent "reinventing the wheel".

The Open Source Initiative (OSI)[1] defines open source software as work that satisfies the criteria spelled out in Table 1. These goals are very similar to the way research works (Bezroukov, 1999):

1. OSI can be found at `http://www.opensource.org`.

1.	Free redistribution
2.	Source code
3.	Derived works
4.	Integrity of the author's source code
5.	No discrimination against persons or groups
6.	No discrimination against fields of endeavor
7.	Distribution of license
8.	License must not be specific to a product
9.	License must not restrict other software
10.	License must be technology-neutral

Table 1: Attributes of Open Source Software from the Open Source Initative

researchers build upon work of other researchers to develop new methods, apply them to produce new results, and publish all of this work, always citing relevant previous work. It is well documented how the move to an "open science" or "open source" model in the Age of Enlightenment (Schaffner, 1994) greatly increased the efficiency of the experimental scientific method (Kronick, 1962) and opened the way for the significant economic growth of the Industrial Revolution (Mokyr, 2005).

However, scientific publications are also not as free as one may think. Major journals are not freely available to the general public since publishers limit access only to subscribers. A few pioneering journals such as the Journal of Machine Learning Research, the Journal of Artificial Intelligence Research, or the Public Library of Science Journals have begun publishing in the so called "open access" model.[2] Open-access literature is digital, online, free of charge, and free of most copyright and licensing restrictions. This model is enabled by low-cost distribution on the Internet, which was economically impossible in the age of print. The "journal pricing crisis" in which journal subscription fees have risen four times faster than inflation since 1986, strongly motivated the development of open access. In summary, open access (with certain limitations) removes *price barriers*, for instance, subscription and licensing fees, and *permission barriers*, that is, most copyright and licensing restrictions. An extensive overview and a time-line concerning this distribution model which our brief summary is also based on, is available from the SPARC Open Access Newsletter.[3] An open letter to the U.S. Congress, signed by 25 Nobel laureates, puts it succinctly:

Open access truly expands shared knowledge across scientific fields, it is the best path for accelerating multidisciplinary breakthroughs in research.[4]

—Open letter to the U.S. Congress, signed by 25 Nobel laureates, (August 26, 2004)

It is plausible that a similar boost could be expected from a more widespread adoption of open source publication practices in the machine learning field, in which the software implementing the methods would play a comparable role to the underlying theory in the advancement of science. To achieve this, the supporting *software and data* should be distributed under a suitable open source license along with the scientific paper. This is already common practice in some biomedical research, where protocols and biological samples are frequently made publicly available. In the area

2. A list of open access journals is currently maintained at `http://www.doaj.org`.
3. The newsletter can be obtained from `http://www.earlham.edu/~peters/fos/`.
4. The letter is available from `http://www.public-domain.org/?q=node/60`.

of machine learning, this is still rarely the case. However, some freely available benchmark data sets exist, for example, the UCI Repository,[5] the Delve repository,[6] the Caltech 101 data set[7] or Rätsch et al. (2001). Nonetheless, this small number of data sets has had a significant influence on the progress in machine learning, since challenging (in their size or complexity) data collections have helped to calibrate algorithms and to establish their relative merits. For instance, much of the progress of the pattern recognition group at AT&T was tracked in terms of the performance of their algorithms on the NIST and USPS data sets.

In Section 4, we will discuss possible reasons for the current situation in more depth. In the rest of this section, we would like to clarify the notion of "open source" by addressing a common misconception that opening the source makes commercial exploitation impossible. On the contrary, open source software has created numerous new opportunities for businesses (Riehle, 2007). Also, simply *using* an open source program on a day to day basis has little legal implications for a user provided they comply with the terms of their license. Users are free to copy and distribute the software as is. Most issues arise when users, playing the role of a developer, modify the software or incorporate it in their own programs and *distribute a modified product*.

A variety of open source licenses exists, which protect different aspects of the software with benefits for the initial developer or for developers creating derived work (Laurent, 2004). Therefore, there is some flexibility in choosing the license according to the specific needs of the developer, or employer. In the following we give suggestions on which license to choose for common scenarios. This oversimplified description is targeted at developers who just want to "get the program out there".

1. A developer who wants to give away the source code in exchange for *proper credit* for derivative works, even closed-source ones, could choose the *BSD license*. A typical example for this kind of developer would be a researcher who just wants to make his work available to the public, but does not want to prevent inclusion into closed-source software, and also does not rely on getting improvement back from the community. An example for a project using the BSD license is FreeBSD, on which Apple's operating system Mac OS X is partially based.

2. A developer who wants to give away the source code, is comfortable with his source being incorporated into a closed-source product but still wants to *receive bug-fixes and changes* that are necessary *to his source* when integrating the code could choose the *GNU Lesser General Public License (LGPL)*. This developer could be someone who wants to keep developing his software, and by publishing his software basically invites the community to contribute to the software. Using the software as-is in closed-source products is allowed. An example project using this license is the GNU C library, used by nearly all programs on a linux system.

3. A developer who wants to give away the source code and *make sure that his program stays open source*, that is, any extension (or integration) will require both the original and the derived code to be released as open source, could choose the *GNU General Public License (GPL)*. Here, the developer could be a researcher who has further plans with his software and wants to make sure that no closed-source product, not even one of his own *if it includes changes of external developers*, is benefiting from his software. An example of this is the GNU/Linux project.

5. This database is located at `http://mlearn.ics.uci.edu/MLRepository.html`.
6. The website is at `http://www.cs.toronto.edu/~delve/`.
7. The data set is available at `http://www.vision.caltech.edu/Image_Datasets/Caltech101/Caltech101.html`.

License	Apache	BSD/MIT	GPL	LGPL	MPL/CDDL	CPL/EPL
Closed source	Yes	Yes	No	Maybe	Yes	Yes
Commercial	Yes	Yes	No	Maybe	Yes	Yes
Modification release	No	No	Yes	Yes	Yes	Yes
Patent	Yes	No	No	No	Yes	Yes
Jurisdiction	Silent	Silent	Silent	Silent	California	New York
Freedom	PR	Free	PR	PR	Free	PR

Table 2: The rights of the developer to redistribute a modified product. A comparison of open source software licenses listed as "with strong communities" on `http://opensource.org/licenses/category`. The main questions are: whether code can be used in closed source projects (Closed source); whether a program that incorporates the code can be sold commercially (Commercial) without releasing the incorporating program under the same license; whether the source code to modifications must be released (Modification release); whether it provides an explicit license of patents covering the code (Patent); the legal jurisdiction the license falls under (Jurisdiction); freedom to adapt licence terms (Freedom) (PR = Permission Required from license drafter). Apache: License used by the Apache web server; BSD: License under which the BSD Unix variant is released; MIT: developed by the MIT; GPL/LGPL: (lesser) GNU General Public License; MPL: License used by the Mozilla web browser; CDDL: Common Development and Distribution License developed by Sun Microsystems based on the MPL; CPL: Common Public License published by IBM; EPL: Eclipse Public License used by the Eclipse Foundation, derived from the CPL.

All of the open source licenses allow for derivative works (item two in Table 1). In addition it is not possible to limit an open source product to a particular use, for example, to non-commercial or academic use, as it conflicts with item six in Table 1. In a brief summary of common open source licenses, Table 2 shows the rights of a developer to distribute a modified product. A more in-depth discussion about licenses can be found in Appendix A. For more details and a comparison of the various freedoms different licenses provide, see Lin et al. (2006).

Finally, note that the idea of "open source" is not limited to scientific publications and computer software. Authors of other creative works may also want to openly distribute their work. This has created a demand for "open source" type licenses applicable to other media, such as music or images. One of the most prominent movements addressing this demand are the Creative Commons (CC) licenses.[8] The CC project was started in 2001 to supply the analog to open source for less technical forms of expression (Coates, 2007) and extends to all kinds of media like text documents, photographs, video and music. All CC licenses allow copying, distribution, and public performance and display of the work without any license payments. However, CC common terms state that the licenses do not interfere with fair use rights (such as citations, private use etc.), first sale or the freedom of expression and it may restrict the use to, for instance, non-commercial purposes or that no derivative works are allowed (Lin et al., 2006; Välimäki, 2005). It therefore conflicts with the non-discrimination provision in the open source definition (Table 1). It should also be noted that in

8. The creative commons homepage is `http://creativecommons.org/`.

principle anyone can submit a new license to the Open Source Initiative to be certified to comply with the Open Source Definition. Creative Commons does not have such a process but was designed *top-down* (Välimäki, 2005). Applied to the area of science, Creative Commons advocates not only having open source methods, but also open source data and results. It should be noted that open access journals like PLoS use a CC license, namely the Creative Commons Attribution License.[9] The European Union supports a related project towards free exchange of scientific results and data sets.[10]

3. Open Source in Machine Learning

This section of the paper aims to provide a brief overview of open source software and its relationship to scientific activity, specifically machine learning. The reader may think that we are overly positive about the benefits of open source, and do not discuss negative views. The truth is that it is extremely difficult to obtain hard evidence on the debate between proprietary systems and open source software.[11] We argue from moral, ethical and social grounds that open source should be the preferred software publication option for machine learning research and refer the reader to the many advantages of the open source software development (Raymond, 2000). There are also a multitude of advantages of sharing of data and resources, as promulgated in the open science approach (Nature, 2005). Here, we focus on the specific advantages of open source software for machine learning research, which combines the needs and requirements both of being a scientific endeavor, as well as being a producer and consumer of software. They can be categorized into:

1. reproducibility of scientific results and fair comparison of algorithms;
2. uncovering problems;
3. building on existing resources (rather than re-implementing them);
4. access to scientific tools without cease;
5. combination of advances;
6. faster adoption of methods in different disciplines and in industry; and
7. collaborative emergence of standards.

We discuss these points in the following seven subsections.

3.1 Reproducibility and Fair Comparison of Methods

Reproducibility of experimental results is a cornerstone of science. In many areas of science it is only when an experiment has been corroborated independently by another group of researchers that it is generally accepted by the scientific community. It is often the case that experiments are quite hard to reproduce exactly, and in many fields (e.g., medicine) people go to great lengths to try to

9. See for example `http://www.plos.org/oa/definition.html`.
10. The Digital Repository Infrastructure Vision for European Research located at `http://www.driver-repository.eu`.
11. See Section 1.2 of `http://www.dwheeler.com/oss_fs_why.html`.

ensure this. Reproducibility would be quite easy to achieve in machine learning simply by sharing the full code used for experiments.

In the field of machine learning, numerical simulations are often used to provide experimental validation and comparison of methods. Ideally, such a comparison between methods would be based on a rigorous theoretical analysis. For various reasons however, it may not be possible to theoretically analyze a particular machine learning algorithm or to analytically compute its performance in contrast to another. As many methods seek to do well on some real-world problems where the underlying (true) model is unknown, it is very difficult to measure performance in any other way than empirically. In that sense, experiments play a different role than in the natural sciences, as for example physics or chemistry, where experiments are used to better understand certain aspects of nature, instead of algorithms constructed by humans. Nevertheless, the results of the experimental validations are equally important, as these may for instance provide the evidence that a method outperforms existing approaches (or not). Unfortunately, the current practice in the machine learning community is extremely sloppy, as papers get accepted, which are not detailed enough to allow replication.[12] In the pre-internet era, one could perhaps have argued, that for complex algorithms typically used in machine learning, describing every detail would be too lengthy for publication; but nowadays, there would seem to be no such constraints, as supplementary material could be made available online. Indeed, for many complex algorithms one can probably argue, that a clear and well documented program is perhaps the most convenient way of documenting the full details of a machine learning algorithm. So, it follows that an open source approach would be ideally suited to this challenge.

A survey[13] asking JMLR authors for the availability of the system they described in their JMLR papers concluded that about a third specifically said their systems were unavailable for the reasons discussed in Section 4.

> *My informal survey suggests some authors have a relaxed regard for scientific virtues: reproducibility, testability, and availability of data, methods and programs—the openness and attention to detail that supports other researchers. It's a widespread problem in computer science generally. I'm guilty, too. We programmers tend not to keep the equivalent of lab books, and reconstructing what we have done is often unnecessarily hard. As I wrote elsewhere (see Thimbleby, 2003) there can be problems with publishing work that is not rigorously supported. It is the computer science equivalent of fudging experimental data—whether this really matters for the progress of science is another question.*
>
> *—Harold Thimbleby, 2003*

Reproducing numerical results in order to compare methods is not trivial, as it is often not possible to re-implement a method based only on the information contained in publications. Methods often have a number of free parameters whose correct adjustment requires extensive experience with the specific algorithm, data set, or both. In this context it should be noted that all steps involved in data pre-processing are equally crucial in reproducing results.

The non-reproducibility of results is not merely a theoretical possibility. Consider the recent exchange of papers in this journal (Loosli and Canu, 2007; Tsang and Kwok, 2007). A comment has been published in which the authors document that they could not reproduce the results of another paper. The authors of the original paper defended their original results, blaming the differences

12. One may indeed go further, and ask whether such a practice lives up to the basic requirements of scientific work.
13. A summary is at `http://www.uclic.ucl.ac.uk/harold/srf/jmlr.html`.

on the operating system used to perform the experiments. Needless to say, such a situation is unsatisfactory. This example also reflects another benefit of making source code available: it allows us to uncover hidden tricks that remain typically undocumented (Orr and Müller, 1998). The reason a certain implementation of a machine learning method outperforms all other approaches with similar algorithms may be due to a number of functions that have been tuned to specific machine instructions.

Furthermore, instances of fraud or scientific misconduct can be more easily detected if all the code required to perform the experiment is made available. Thus, making algorithms, *including* the source code and data publicly available (such as the efforts mentioned in Section 2) significantly enhances the reproducibility and the feasibility of (fair) comparisons.

3.2 Quicker Detection and Correction of Bugs

An important feature that has contributed much to the success of open source software is that with the availability of the source code, it is much easier to spot and fix bugs in software. While not everyone would be inclined (or able) to satisfactorily resolve a bug himself, everybody has the possibility to inspect the source code, find the bug and submit a patch to the maintainers of the project. This observation has been summarized as "Given enough eyeballs, all bugs are shallow", known as Linus's Law (Raymond, 2000). Further, to paraphrase Al Viro,[14] all software contains bugs, be it open-source or proprietary. The only question is what can be done about a particular instance of software failure, and that is where having the source matters.

3.3 Faster Scientific Progress by Reduced Cost for Re-implementation of Methods

Scientific progress always builds on existing publications and methods. The field of machine learning is no exception. However, re-implementing existing methods in order to test them, use them as part of a larger project, or to extend them, is a large burden on the researcher. This is particularly true for method oriented research. As already discussed above, publications often do not contain all the information necessary to re-implement a method. The complexity of existing methods is often so large that re-implementing its algorithms can require prohibitive effort.

As a consequence, work on such methods is often restricted to a few groups who already have implementations, and newcomers to the field have to first redo the work of others. Alternatively, such a situation can lead to ignoring existing competitors since implementations are not available, and re-implementation seems infeasible. Therefore, the availability of open source implementations can help speed up scientific progress significantly.

3.4 Long Term Availability and Support

For the individual researcher, open source may provide a means of ensuring that he will be able to use his research even after changing his employer. Even the most generous institutions tend to introduce delays before giving *formal* approval for code reuse after the researcher moves. This is, however, *harmful for both researcher and employer*: obviously for the researcher since he loses access to the tools he has been working with but also for the institution since the piece of code

14. This quotation can be obtained from the linux kernel mailinglist `http://www.ussg.iu.edu/hypermail/linux/kernel/0404.3/1344.html`.

in question becomes *unsupported*. By releasing code under an open source license the chances of having long-term support are dramatically increased.

3.5 Combination of Advances

Scientific progress does not always occur as paradigm shifts (e.g., the emergence of Decision Trees, Neural Networks, Kernel Methods, Boosting, and Graphical Models) but it is much more likely to occur by incremental improvements over a given existing technique. Moreover, it is likely that several such changes occur simultaneously once a given topic reaches the mainstream. While this is, in principle, a good thing, it poses a rather unique problem: how to combine several of those advances into *one* joint implementation.

As a case in point, consider progress in kernel methods. There is currently no piece of code or even a publication which combines structured estimation, semiparametric methods, automatic margin adjustment, different types of regularization, methods for dealing with missing variables, methods for dealing with invariances, a large set of kernel functions, nonconvex approximations of the loss, leave-one-out estimators, or transductive estimation. While each of these modifications are well established and it is commonly accepted that they work, there is no publication indicating the performance of a combination of more than three of the ten aforementioned methods.

This is more than just a simple nuisance: it is not clear at all whether the combination of all of those "improvements" would really be beneficial and what their interactions might be. Do some of these methods effectively solve the same problem and derive their gains from a common change in the estimate? What are the computational limitations?

Without access to a common codebase *and* willingness of the community to improve upon it it will be next to impossible to address this issue, since it is likely to be too difficult for a single researcher to track and compare all modifications.

3.6 Faster Adoption in Machine Learning, Other Disciplines and Industry

Availability of high-quality open source implementations can ease adoption by other machine learning researchers, users in other disciplines and developers in industry for the following reasons:

1. Open source software can be used without cost in teaching.

2. If a method proves useful and its source code is available, it can be directly applied to related real world problems in other fields or in industry.

In areas such as bioinformatics, the expertise to implement advanced machine learning methods from scratch is often not available. While this situation might be perceived as desirable by some to ensure that machine learning experts are sought by the industry, hiring machine learning experts will become more desirable for companies as the field gains prominence. In fact, one may argue that it is the problem of *automatic* adjustment and deployment that machine learning theory should be addressing by suitable means of model selection. Having access to an extensive machine learning toolkit will allow us to compare model selection techniques in realistic settings.[15] Increased distribution of machine learning's end-product, software, will lead to more success stories of its use within industrial applications. Publishing software as open source might also be the

15. See, for example, the NIPS'04 workshop on the (Ab)Use of Bounds `http://www.hunch.net/~jl/conferences/abuse_of_bounds/abuse_of_bounds.html`.

only means to reach wide-spread distribution of your software if you lack the logistic infrastructure of big companies like Microsoft™. In addition, the adoption of machine learning methods in large-scale applications can have a very stimulating effect on the field itself, and lead to novel and interesting challenges. It still requires an expert with deep understanding of the method to adjust it to a particular application. There are also impressive precedents of open source software leading to the creation of multi-billion dollar companies and industries.[16]

3.7 Collaborative Moves towards Better Interoperability

The diversity of machine learning forbids a single, mono-cultural software framework satisfying all needs. However, even in areas where it is in principle feasible, most pieces of machine learning software do not inter-operate very well, because of differences in interfaces, data abstractions and work flows. Ultimately it would be desirable to agree to a set of standards which ensure, for example, that data sets can be exchanged between machine learning tools, and that classification algorithms can be interchanged seamlessly.

However, given the distributed nature of scientific work, it is unlikely that a centralized institution can be formed which develops such standards in a top-down manner. Now with the publication of toolboxes according to an open source model, it becomes possible for individual projects to move towards standardization in a collaborative, distributed manner.

This process has already begun, mostly with toolboxes incorporating other toolboxes or providing "glue" code to access functionality contained in other toolboxes. A typical example are the libraries for learning support vector machines, such as LIBSVM (Chang and Lin, 2001), SVMLin (Sindhwani and Keerthi, 2006), SVMTorch (Collobert and Bengio, 2001) and GPDT (Zanni et al., 2006). A small sample of larger frameworks which provide access to (among other features) one or more of these libraries include Elefant (Gawande et al., 2007), Orange (Demsar and Zupan, 2004), PLearn (Vincent et al.), RapidMiner[17], Shogun (Sonnenburg et al., 2006), Torch (Collobert et al., 2002) and the Weka (Witten and Frank, 2005) toolboxes.

In the future, instead of the constant repetition of work, standards should emerge, pushed either by library and/or toolbox developers, in order to make this integration much less difficult. A consensus could also emerge via dialog in journal or community websites.[18] Which standards will be adopted will depend on the popularity of the individual toolboxes or libraries.

We conclude this section by summarizing the advantages described above in Table 3.

4. Current Obstacles to an Open Source Community

While there exist many advantages to publishing implementations according to the open source model, this option is currently not taken often. We believe that there are six main reasons which will be discussed in greater detail in the next sections.

16. Perhaps the oldest, dating from the early 1970s, is SPICE (Simulation Program with Integrated Circuit Emphasis) (Wikipedia, 2007b), which has led to the foundation of Synopsys and Cadence Design Systems, and significantly grew the whole Electronic Design Automation Industry.
17. Former YALE toolbox, available from `http://www.rapidminer.com`.
18. We propose to use `http://mloss.org` as *the* platform for machine learning open source software (MLOSS) to openly discuss design decisions and to host and announce MLOSS.

1.	Reproducibility of scientific research is increased
2.	Algorithms implemented in same framework facilitate fair comparisons
3.	Problems can be uncovered much faster
4.	Bug fixes and extensions from external sources
5.	Methods are more quickly adopted by others
6.	Efficient algorithms become available
7.	Leverage existing resources to aid new research
8.	Wider use leads to wider recognition
9.	More complex machine learning algorithms can be developed
10.	Accelerates research
11.	Benefits newcomers and smaller research groups

Table 3: Eleven Advantages of Machine Learning Open Source Software

4.1 Publishing Software is Not Considered a Scientific Contribution

Some researchers may not consider the extra effort to create a usable piece of software out of machine learning methods to be science. However, machine learning is a synthetic discipline as well as an analytic one, and certainly if it is science it is in Simon's phrase, a "Science of the Artificial" (Simon, 1969), in which artifacts, specifically implemented algorithms, is one of the major outputs. In addition to the "pure" scientific pursuits, machine learning researchers also produce technological outputs. As such, the discipline could be considered to be mathematical engineering. In any case, as was pointed out in Section 3, the complexity of existing methods is growing such that reimplementing algorithms can easily take months. Some argue that if you want to really understand an algorithm and want to extend it—which is an important task for machine learning researchers—you have to implement it from scratch and thus it is not beneficial to have the software available. This is only partially true: one does not want to reimplement all the basic algorithms an advanced method builds on, but simply understand the high-level steps. After all, one has to build upon existing libraries, as for example the standard or math library, the Basic Linear Algebra Subprograms (BLAS) (Lawson et al., 1979), the Linear Algebra PACKage (LAPACK) (Anderson et al., 1999) to be productive. Only few people would want to re-implement, or would be able to generate a high quality implementation of, common sorting algorithms such as `qsort`, basic mathematical functions such as `sin`, or linear algebra operations such as `dgemm` or `dgesv`.

4.2 Misconception—Opening the Source Conflicts with Commercial Interests

As already discussed in Section 2, there is a common misconception that opening the source makes commercial use—licensing of commercial versions or use in industrial projects—impossible. It may, however, prevent the creation of closed-source products that include external open-source contributions. In reality, careful selection of a suitable open source license would satisfy the requirements of most researchers *and their employer*. For example, using the concept of *dual licensing* one could release the source code to the public under a open source license with strong reciprocal obligations (like the GNU GPL), and at the same time sell it commercially in a closed-source product. In Appendix A we give a few hints for choosing an appropriate license.

4.3 The Incentive for Publishing Open Source Software is not High Enough

Unlike writing a journal article, releasing a piece of software is only the beginning. Maintaining a software package, fixing bugs or writing further documentation requires time and commitment from the developer, and this contribution is also rarely acknowledged. Open source programmers often gain a good "reputation" among their peers, which in some situations may be worth more than citations (Kelty, 2001; Franck, 1999). But scientific success, especially in research institutions, is often determined by measures such as citation statistics. However, there exists no academic, widely accepted platform to publish software. As a result, researchers tend to not acknowledge software used in their published research, and the effort which has to be expended to turn a piece of code for personal research into a software product that can be used, understood, and extended by others is not sufficiently acknowledged. As just one example, a well-known structured classification method had 766 Google Scholar citations as of this writing, while the supporting software, which was released with an open-source license but no peer-reviewed publication, has only 78 citations. In contrast, published software descriptions for bioinformatics programs are cited in every published use of the program: the published description of one version of BLAST had 20540 Google Scholar citations, for instance.

4.4 Machine Learning Researchers are Not Good Programmers

While most machine-learning methods are implemented in some form, it does not follow that the best machine learning researchers are the best programmers. Opening up "research quality" code to the inspection and modification of others (who may be more skilled programmers) can certainly help to improve the quality of the code base. On the other hand, the initial developers may be reluctant to expose their programming practices to public criticism.

4.5 Sloppiness Hides Problems of Newly Proposed Methods and Eases Acceptance at Conferences and Journals.

A certain degree of sloppiness may be advantageous to someone trying to promote a new method. For example, many algorithms require the setting of parameters, decisions about convergence, and a multitude of other things, and it is perhaps not unusual that researchers inadvertently "help their new algorithms along", by carefully making sure that "nothing goes wrong" during the application of a method, and if something does go wrong, a suitable measure is taken, that is, reduction of a learning rate, restart with a new random seed etc. Thus, being absolutely precise about the algorithm, could help bring these issues to the surface, but this is currently only rarely done, presumably because such details are thought of as secondary, and not really part of the idea of the algorithm. Therefore, at first glance, making the source code for a particular machine learning paper public may seem counter-productive for the researcher, as other researchers can more easily find problems with the proposed method, and possibly even discredit the approach. The researcher may also lose a competitive advantage because competing groups can also use the software. However, the same argument holds for making research papers publicly available, and as discussed in Section 2 the move to an open science in the Age of Enlightenment sped up scientific progress and boosted economic growth. Therefore, the already altruistic behavior of publishing papers should be complemented by also providing open source code as the same great benefits can be expected if many other researchers follow this path and also distribute accompanying open source software.

4.6 Tradition—Reviewers Pass Papers of Similar Quality

Finally, there seems to exist a tradition, which let's people "get away" with less. When reviewers examine a paper, they have other similar papers (they passed) in mind. They therefore pass papers "for tradition", although the papers could have become a lot more valuable, if reviewers required that the source code of the algorithm had been provided.

These latter two issues are closely related to the question of how to design experiments in a way which ensures the ability to make strong statistical claims about the outcomes of experiments. One such attempt was made in the DELVE (Data for Evaluating Learning in Valid Experiments) archive. However, this archive never gained much popularity, presumably because its data sets are typically not very large, and it has proven to be difficult to reach statistically strong conclusions using relatively small data sets.

5. Proposal

In summary, providing open source code would help the whole community in accelerating research. Arguably, the best way to build an open source community of scientists in machine learning is to promote open source software through the existing reward system based on citation of archival sources (journals, conferences). Unfortunately, persuading people to publish the implementation together with their research paper is a long-term process, exacerbated by a potentially conflicting industrial interest. However, it is possible that a push in this direction could gather momentum, with peer pressure doing the rest.

We would like to initiate this process by giving researchers the opportunity to publish their machine learning open source software, thereby setting an example of how to deal with this kind of publication media. The proposed new JMLR track on machine learning open source software with review guidelines specially tailored to the needs of software is designed to serve that purpose.

We encourage submissions which are contributions related to implementations of non-trivial machine learning algorithms, toolboxes or even languages for scientific computing. As with the main JMLR papers, all published papers will be freely available online. The software must adhere to a recognized open source license (`http://www.opensource.org/licenses/`). Submissions should clearly indicate that they are intended for this special track in the cover letter of the submission.

Since we specifically want to honor the effort of turning a method into a highly usable piece of software, prior publication of the method is admissible, as long as the software has not been published elsewhere. As an inspiration we discuss in Appendix B basic software design principles and more machine learning (toolbox) related ideas. In summary, preparing research software for publication is a significant extra effort which should also be rewarded as such.

It is hoped that the open source track will motivate the machine learning community towards open science, where open access publishing, open data standards and open source software foster research progress.

5.1 Format

We invite submissions of descriptions of high quality machine learning open source software implementations. Submissions should at least include:

1. A cover letter stating that the submission is intended for the machine learning open source software section, the open source license the software is released under, the web address of the project, and the software version to be reviewed.

2. An up to four page description based on the JMLR format.

3. A zip or compressed tar-archive file containing the source code and documentation.

5.2 Review Criteria

The following guidelines would be used to review submissions. While ideally submissions should satisfy all the criteria below, they are not necessary requirements. Some examples of acceptable submissions which do not satisfy all criteria are: well designed open source toolboxes based on Matlab^{TM}; learning algorithms using commercial optimizers such as MOSEK or CPLEX as a back-end; or a teaching tool which has poor computational performance due to its didactic implementation.

1. The quality of the four page description.

2. The novelty and breadth of the contribution.

3. The clarity of design.

4. The freedom of the code (lack of dependence on proprietary software).

5. The breadth of platforms it can be used on (should include an open-source operating system).

6. The quality of the user documentation (should enable new users to quickly apply the software to other problems, including a tutorial and several non-trivial examples of how the software can be used).

7. The quality of the developer documentation (should enable easy modification and extension of the software, provide an API reference, provide unit testing routines).

8. The quality of comparison to previous (if any) related implementations, w.r.t. run-time, memory requirements, features, to explain that significant progress has been made.

After acceptance, the abstract including the link to the software project website, the four page description and the reviewed version of the software will be published on the JMLR-MLOSS website `http://www.jmlr.org/papers/mloss`. The authors can then make sure that the software is appropriately maintained and that the link to the project website is kept up-to-date.

6. Conclusion

We have argued that the adoption of the open source model of sharing information for implementations of machine learning software can be highly beneficial for the whole field. The open source model has many advantages, such as improved reproducibility of experimental results, quicker detection of errors, accelerated scientific progress, and faster adoption of machine learning methods in other disciplines and in the industry. As the incentives for publishing open source software are

currently insufficient, we outlined a platform for publishing software for machine learning. Additionally, we discussed desirable features of machine learning software which will ultimately lead to highly usable, flexible and scalable software. We invite all machine learning researchers developing machine learning algorithms to submit to the new JMLR track for machine learning software. Defining well-designed interfaces will prove crucial towards better interoperability, leading to a community built suite of high-quality machine learning software.

Researchers in machine learning should not be content with writing small pieces of software for personal use. If machine learning is to solve real scientific and technological problems, the community needs to build on each others' open source software tools. Hence, we believe that there is an urgent need for machine learning open source software. Such software will fulfill several concurrent roles: a better means for reproducing results; a mechanism for providing academic recognition for quality software implementations; and acceleration of the research process by allowing the standing on shoulders of others (not necessarily giants!).

Acknowledgments

The authors would like to acknowledge S.V.N. Vishwanathan, Torsten Werner and the attendees of the NIPS Workshop on Machine Learning Open Source Software 2006 for inspiring discussions. We thank Andre Noll and Sebastian Schultheiß whose careful reading and insights have improved this manuscript. We thank Evana Ushakoff for providing legal comments. We gratefully acknowledge partial support from the PASCAL Network of Excellence (EU #506778). C. S. Ong is also with Max Planck Institute for Biological Cybernetics, Spemannstr. 38, 72076 Tübingen, Germany and K. R. Müller with the Fraunhofer Institute FIRST, Kekulestr. 7, 12489 Berlin, Germany.

Appendix A. Which License to Choose?

As discussed in Section 2, most issues regarding the use of open source software arise when one wants to distribute a modified or derived product. In this section, we wish to discuss these issues in more depth.

With the proliferation of open source software, various licenses have been put forward, confusing a developer who just wants to release his program to the public. Whilst the choice of license might be considered a boring legal/management detail, it is actually very important to get it right—the choice of certain licenses may significantly limit the impact a piece of software may have.[19] In this section we briefly summarize some pertinent questions below as a guideline to some of the more popular licenses.[20]

The owner of the intellectual property present in the code (often the original author, but depending on employment contract, sometimes the employer) owns the copyright of the work and can

19. For example if the SPICE software had been released under a GPL-like license, it is extremely unlikely that it would have had the impact that it did, with multi-billion dollar companies being created on the basis of it because the value-add the companies created could not have been protected, and thus there would be no competitive advantage. On the other hand it is questionable whether the Linux kernel would have evolved into an open, full featured multi-platform kernel with thousands of developers continuously contributing if it was BSD licensed.
20. Disclaimer: This does not constitute legal advice. Since licensing *is* a legal issue, and since employers usually have an interest in the protection of what is usually their intellectual property, readers should always seek their own formal legal advice.

thus dictate the license under which it is released (Webbink, 2003). Different licenses protect different aspects of the software with benefits for the initial developer or developers creating derived work (Laurent, 2004). Significant licensing issues may arise when open source software (OSS) is combined with proprietary code. Depending on the license, the resulting product may have to be published as open source, including the formerly proprietary code. Licenses which demand that subsequent modifications of the software be released under the same license are called "copyleft" licenses Wikipedia (2007a), the most famous of which is the GNU General Public License (GPL).

For example, for developers creating derived works a BSD/MIT license is the most liberal, as it allows a developer to incorporate the software in his own product, without open sourcing the whole product later; and GPL is the most strict, trying to ensure that all subsequent derivatives of the software also stay open. From the viewpoint of the original developer, this situation is reversed: Using the BSD/MIT license, he may not benefit from patches with enhancements, while using the GPL license ensures that derived work will stay open, making future enhancements available to the original developer. Then there are the "in between" licenses, like Lesser GNU General Public

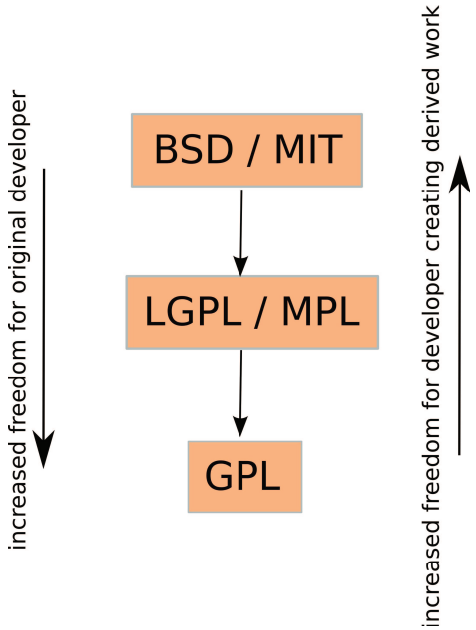


Figure 1: An illustration of open source licenses with respect to the rights for the initial developer and the developer creating derived works.

License (LGPL), the Common Public License (CPL) and the Mozilla Public License (MPL) that only require the changes to the code to be released. Hence the original author has access to any future modifications (bug fixes or new features) of his or her particular piece of software. Figure 1 visually illustrates license interdependencies.

Note that one can release one's own software under multiple licenses. This is referred to as *dual licensing* and allows a developer to release his code to the public under the GPL and at the same time sell it commercially in a closed-source product. However if one includes changes in a program that

other developers have made contributions, the agreement of all contributors is required to change a license (Laurent, 2004; Burnette, 2006; Fitzgerald and Bassett, 2003). A crude summary of some of the simple distinctions between some OSS licenses is given in Table 2. It should be noted that a simple table hides the complexity of some of the key issues (see below).

A.1 Some Complexities

As an illustration of some of the difficulties, let us consider the issue of conflicting open source licenses and the issue of reciprocal obligations.

A.1.1 OPEN SOURCE LICENSES MAY CONFLICT

When releasing a program as "open source" it is not obvious that although the program is now "open source" it still may have a license that conflicts with many other open source licenses. Licenses may have mutually conflicting requirements, for example with respect to jurisdiction, or including advertising clauses, such that one cannot legally combine the two programs into a new derived work (simply using both programs is usually possible, though). The OSI currently lists 60 open source licenses[21] and the consequence of this license proliferation[22] means that the simple inclusion BSD $\subset$ LGPL $\subset$ GPL as shown in Figure 1 does not hold for other licenses.[23] For example the MPL and CPL conflict with the most widely used licenses, which are the GPL (in use by about 70% of the OSS programs) and the LGPL (about 10% spread), and may even conflict with each other Figure 2. While this can be used to purposely generate conflicts, as a general rule, one should refrain from doing so as it will make code exchange between open source projects impossible and may limit distribution and thus success of a open source project. For a more in-depth discussion see Wheeler (2007). Researchers aspiring to a wide developer audience for their software should consider GPL compatible licenses,[24] or select one with a strong community.[25]

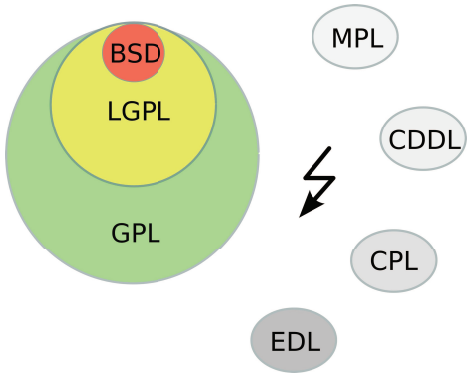


Figure 2: Open Source Licenses may conflict with each other.

21. The OSI license list can be found at `http://www.opensource.org/licenses/alphabetical`.
22. The license proliferation committee report is available at `http://opensource.org/osi3.0/proliferation-report`.
23. Note that this only holds for the 3-clause BSD license. Also note that this is a one-way street, that is, BSD licensed software cannot merge code from LGPL/GPL and LGPL cannot merge software from GPL projects
24. The Free Software Foundations GPL compatible license list is available at `http://www.gnu.org/philosophy/license-list.html`
25. Licenses with a strong community are listed at `http://opensource.org/licenses/category`.

A.2 Reciprocal Obligations

Another issue is the one of reciprocal obligations: any modifications to a piece of open source software may need to be available to the original authors. In the following, to give a hint of the complexity, reciprocal obligations are discussed for the following licenses:

- LGPL – applies the concept of "derivative works", which (confusingly) can include the combined work resulting from linking a LGPL-licensed Library and a non-LGPL "work that uses the Library". Problematically, the LGPL requires for such combined works that the source code of the "work that uses the Library" needs to be disclosed when the combined work is distributed (LGPL section 2, third last paragraph). This is a substantial limitation to the utility of the LGPL in enabling components to be further developed and distributed with proprietary code. The LGPL also tries to make some fairly complex and unclear distinctions between what constitutes a collective or derivative work to determine whether the LGPL attaches to licensee-created works.

- MPL – does not apply the concept of "derivative works", but talks instead of "modifications" to (i.e., additions to or deletions from) the Original Code as comprising part of the Covered Code (i.e., code to which the MPL applies). This makes the MPL more comprehensible to (some) legal audiences, and therefore more certain from that perspective. However, it also makes the MPL's reciprocal obligation more limited. The MPL permits Covered Code to be distributed within Larger Works in a combined work without the MPL attaching to the non-MPL code (as long as the distributor continues to apply the MPL to the Covered Code component of the Larger Work). This overcomes the over-inclusiveness aspect of the GPL and LGPL, and makes the MPL more friendly towards developers who may wish to combine MPL code with their own proprietary code that is not a "modification" of MPL code.

- CPL – like MPL, applies the concept of additions or changes from the original Contribution. However, the CPL arguably imposes more narrow reciprocity obligations than either GPL/LGPL or MPL, because the CPL explicitly exempts the reciprocity obligations from applying to a "separate module of software distributed in conjunction" with the original Contribution that is not a "derivative work". Put another way, the CPL reciprocity obligation only attaches to additions to the original contribution that are "derivative works" but not separate modules of software.

Appendix B. Guidelines for Good Machine Learning Software

Without claiming to be exhaustive, in this appendix we record some guidelines which, we believe, would lead to high quality machine learning software.

B.1 Good Software Practices

There is a significant difference between a piece of code which is intended to be used privately (either alone or within a small research group), and one which is intended to be made public and to be used (or even extended) by external users. While a certain lack of organization, documentation, and robustness can be tolerated when the software is used internally, it can make the software next to useless for others. The old rule that software is primarily written for other humans, and not only

for computers, is even more important when your audience is larger than colleagues with whom you closely collaborate.

1.	Software is useful and usable.
2.	Software is documented.
3.	Software is robust.
4.	Software has well-defined interfaces.
5.	Software uses existing interfaces and standards.
6.	Software has well established (unit) testing routines.

Table 4: Six features of useful machine learning software

Good machine learning software should first of all be a good piece of software (Table 4). There exist many books on software design. The inclined reader is referred to the books by Raymond (2004) or Hunt and Thomas (2000) for further information. Just putting your research software on your web-page will not be sufficient. One should follow general rules for developing open source software (see also the discussion by Levesque, 2004, which highlights common failure modes for open source software development):

- The software should be structured well and logically such that its usability is high.
- It should be documented well, such that you can learn to use the software quickly; for example, in the form of a tutorial, a reference, and examples; ideally, also include developer's documentation which explains the software's internals;
- It should be sufficiently robust, which means that it is as much as possible bug-free, but also tolerates incorrect inputs as well as providing meaningful error messages instead of breaking down silently.
- It should provide testing routines to verify automatically whether the code is correct. This reduces the likelihood that modifications of the code introduces bugs.

In any case, the main goal should be to maximize the re-usability of your software. Therefore you would want to make your software as flexible as possible such that it can deal with a large number of different types of data. You would also want to clearly define the interface to your software such that others can easily use it directly.

Ideally, the software also includes a number of unit tests. These are small programs which can be run automatically and test the individual parts of the program for correctness. Unit tests are an indispensable tool for ensuring that a small change does not introduce bugs which go unnoticed for a long time. Such tests therefore facilitate modification of software greatly.

Apart from these considerations that apply to any software design, there are several requirements that are specific to the domain of machine learning. Since these requirements are quite different depending on whether you are writing high-quality implementations of a specific algorithm (for example, support vector machines), or more general frameworks, we have split the discussion in two sections discussing these two extremes.

B.2 Guidelines for Single Machine Learning Algorithms

Consider the case that a researcher has special expertise in implementing a certain class of machine learning algorithms, and has developed, for example, a new implementation of support vector machines which is very efficient, or a clever implementation of a certain class of graphical models. There should exist several ways of using the program, for example, stand-alone from the command line, and as a library which can be linked to other programs. Reusing the interface of existing software solving the same problem is also very useful. Then, the software can be used as a drop-in replacement. If the algorithm can be practically applied to large data sets, it is desirable that the available main memory is not the limiting factor, but if the algorithms are designed such that they can also deal with data sets which reside on the hard disk, using the main memory as a cache. Finally, one should make sure that the software is able to read and write data formats in at least one commonly used data exchange standard.

B.3 Guidelines for Larger Machine Learning Frameworks

A completely different kind of endeavor is to build a framework, or an environment, which can be used for a large number of different machine learning tasks. Such a framework typically integrates a number of existing more specialized machine learning algorithms, or low-level numerical libraries.

Since frameworks should be suited for a wide range of applications—potentially including methods and data types which have not yet been invented—a clean design is particularly important. One approach to achieve this is to decompose the framework into several small modules with clearly defined interfaces so as to minimize the coupling between different parts of the framework. Then, individual modules can be modified or extended more easily.

For example, a framework which deals with vectorial data and matrices, could also provide access to a standard set of basic linear algebra routines, learning algorithms dealing with vectorial data like support vector machines, or least squares regression, and routines to store and read these standard data types. However, the interfaces between these components are sufficiently abstract that it is possible to replace the linear algebra routines by more efficient ones without affecting the rest of the framework.

But as machine learning deals with a large number of different kinds of data sets, frameworks could also support other data types like strings, sequences, trees, graphs, sparse vectors, et cetera. Likewise, tools for graphical models should allow for easy specification of the model, ability to save states, a variety of approximate samplers and solvers, convergence monitors, and flexible nonparametric message passing tools.

Beyond these basic features, the following methods would be nice to have: efficient optimization solvers; access to classical statistical methods and probability distribution; a good visualization library, that provides graphs of various kinds to help analyzing data and reporting results; various classification and regression algorithms, also with extensions to one-class and multi-class; clustering and structure learning algorithms; graphical models, and Bayesian inference, et cetera.

As clusters of machines become more and more affordable, it would be nice to provide simple ways to parallelize parts of the algorithms. Often machine learning algorithms are easy to parallelize and only the barrier of low-level parallel computing stops the designers from doing so. To achieve this goal parallel libraries such as OpenMP and MPI could be used.

References

Ed Anderson, Zhaojun Bai., Christian Bischof, Susan Blackford, James Demmel, Jack Dongarra., Jeremy Du Croz., Anne Greenbaum, Sven Hammarling, Alan McKenney, and Danny Sorensen. *LAPACK Users' Guide*. Society for Industrial and Applied Mathematics, Philadelphia, PA, third edition, 1999. ISBN 0-89871-447-8 (paperback).

Steven J. Benson, Lois Curfman-McInnes, Jorge Moré, and Jason Sarich. TAO user manual (revision 1.8). Technical Report ANL/MCS-TM-242, Mathematics and Computer Science Division, Argonne National Laboratory, 2004. `http://www.mcs.anl.gov/tao`.

Nikolai Bezroukov. Open source software development as a special type of academic research (critique of vulgar Raymondism). *First Monday*, 4(10), October 1999. `http://www.firstmonday.org/issues/issue4_10/bezroukov/index.html`.

Ed Burnette. How to pick an open source license. `http://blogs.zdnet.com/Burnette/?p=130` and `http://blogs.zdnet.com/Burnette/?p=131`, 2006.

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`.

Jessica Coates. Creative commons – the next generation: Creative commons licence use five years on. *SCRIPT-ed*, 4(1), 2007. `http://www.law.ed.ac.uk/ahrc/script-ed/vol4-1/coates.asp`.

Ronan Collobert and Samy Bengio. SVMTorch: Support vector machines for large-scale regression problems. *Journal of Machine Learning Research*, 1:143–160, 2001. ISSN 1533-7928.

Ronan Collobert, Samy Bengio, and J. Mariethoz. Torch: A modular machine learning software library. Technical report, IDIAP, 2002. IDIAP-RR 02-46.

Janez. Demsar and Blaz. Zupan. Orange: From experimental machine learning to interactive data mining, white paper `http://www.ailab.si/orange`. Technical report, Faculty of Computer and Information Science, University of Ljubljana., 2004.

Brian Fitzgerald and Graham Bassett. Legal issues relating to free and open source software. In *Legal Issues Relating to Free and Open Source Software*, pages 11–36. Queensland University of Technology, School of Law, 2003. `http://www.law.qut.edu.au/files/open_source_book.pdf`.

Georg Franck. Scientific communication – a vanity fair? *Science*, 286(5437):53–55, 1999.

Cristina Gacek and Budi Arief. The many meanings of open source. *IEEE Software*, 21(1):34–40, 2004.

Kishor Gawande, Christfried Webers, Alexander J. Smola, and S.V.N. Vishwanathan. ELEFANT: A python machine learning toolbox. In *SciPy Conference*, 2007.

Andrew Hunt and David Thomas. *The Pragmatic Programmer*. Addison-Wesley, 2000.

Christopher M. Kelty. Free software/free science. *First Monday*, 6(12), December 2001. http://www.firstmonday.org/issues/issue6_12/kelty/index.html.

David A. Kronick. *A history of scientific and technical periodicals: the origins and development of the scientific and technological press, 1665-1790*. Scarecrow Press, New York, 1962.

Andrew M. St. Laurent. *Open Source & Free Software Licensing*. O'Reilly Media, Inc, 2004. http://www.oreilly.com/catalog/osfreesoft/book/.

Charles L. Lawson, Richard J. Hanson, David Kincaid, and Fred T. Krogh. Basic linear algebra subprograms for fortran usage. *ACM Trans. Math. Soft.*, 5:308–323, 1979.

Michelle Levesque. Fundamental issues with open source software development. *First Monday*, 9 (4), April 2004. http://www.firstmonday.org/issues/issue9_4/levesque/index.html.

Yi-Hsuan Lin, Tung-Mei Ko, Tyng-Ruey Chuang, and Kwei-Jay Lin. Open source licenses and the creative commons framework: License selection and comparison. *Journal of Information Science and Engineering*, 22:1–17, 2006.

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.

Joel Mokyr. The intellectual origins of modern economic growth. *The Journal of Economic History*, 65(2):285–351, 2005.

Nature. Let data speak to data. *Nature*, 438(7068):531, 2005.

Open Source Initative. http://www.opensource.org/docs/osd.

Genevieve B. Orr and Klaus-Robert Müller, editors. *Neural Networks: Tricks of the Trade*, volume 1524 of *Lecture Notes in Computer Science*. Springer, 1998.

Gunnar Rätsch, Takashi Onoda, and Klaus-Robert Müller. Soft margins for AdaBoost. *Machine Learning*, 42(3):287–320, 2001. Data is hosted at http://ida.first.fraunhofer.de/projects/bench.

Eric S. Raymond. The cathedral & the bazaar. 2000. http://www.tuxedo.org/~esr.

Eric S. Raymond. *The Art of UNIX Programming*. Addison-Wesley, 2004.

Dirk Riehle. The economic motivation of open source software: Stakeholder perspectives. *IEEE Computer*, 40(4):25–32, 2007.

Ann C. Schaffner. The future of scientific journals: Lessons from the past. *Information Technology and Libraries*, 13(4):239–247, 1994.

Herbert A. Simon. *The Sciences of the Artificial*. MIT Press, Cambridge, Massachusetts, first edition, 1969.

Vikas Sindhwani and S. Sathiya. Keerthi. Large scale semi-supervised linear svms. In *SIGIR '06: Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 477–484, New York, NY, USA, 2006. ACM Press.

Sören Sonnenburg, Gunnar Rätsch, Christin Schäfer, and Bernhard Schölkopf. Large scale multiple kernel learning. *Journal of Machine Learning Research*, 7:1531–1565, July 2006.

Jason E. Strajich and Hilmar Lapp. Open source tools and toolkits for bioinformatics: significance, and where are we? *Briefings in Bioinformatics*, 7(3):287–296, 2006.

Harold Thimbleby. Explaining code for publication. *Software Practice and Experience*, 33(10): 975–1001, 2003.

Ivor W. Tsang and James T. Kwok. Author's reply to the "comments on the Core Vector Machines: Fast SVM Training on Very Large Data Sets". *Journal of Machine Learning Research*, 2007. submitted.

Mikko Välimäki. *The Rise of Open Source Licensing*. Turre Publishing, 2005.

Pascal Vincent, Yoshua Bengio, and Nicolas Chapados. `http://plearn.org`.

Mark Webbink. Licensing and open source software. In *Legal Issues Relating to Free and Open Source Software*, pages 1–11. Queensland University of Technology, School of Law, 2003. `http://www.law.qut.edu.au/files/open_source_book.pdf`.

David A. Wheeler. Make your open source software GPL-compatible. or else. `http://www.dwheeler.com/essays/gpl-compatible.html`, August 2007.

Wikipedia. Copyleft – Wikipedia, the free encyclopedia. `http://en.wikipedia.org/wiki/Copyleft`, 2007a. [Online; accessed 2-July-2007].

Wikipedia. SPICE – Wikipedia, the free encyclopedia. `http://en.wikipedia.org/wiki/SPICE`, 2007b. [Online; accessed 29-June-2007].

Ian H. Witten and Eibe Frank. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, San Francisco, 2005. 2nd Edition.

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, July 2006.

Journal of Machine Learning Research 8 (2007) 2467-2495 Submitted 7/06; Revised 4/07; Published 10/07

On the Representer Theorem and Equivalent Degrees of Freedom of SVR

Francesco Dinuzzo FRANCESCO.DINUZZO01@ATENEOPV.IT
Marta Neve MARTA.NEVE@UNIPV.IT
Giuseppe De Nicolao GIUSEPPE.DENICOLAO@UNIPV.IT
Dipartimento di Informatica e Sistemistica
Università di Pavia
Pavia, Italy

Ugo Pietro Gianazza GIANAZZA@IMATI.CNR.IT
Dipartimento di Matematica
Università di Pavia
Pavia, Italy

Editor: Ralf Herbrich

Abstract

Support Vector Regression (SVR) for discrete data is considered. An alternative formulation of the representer theorem is derived. This result is based on the newly introduced notion of pseudoresidual and the use of subdifferential calculus. The representer theorem is exploited to analyze the sensitivity properties of ε-insensitive SVR and introduce the notion of approximate degrees of freedom. The degrees of freedom are shown to play a key role in the evaluation of the optimism, that is the difference between the expected in-sample error and the expected empirical risk. In this way, it is possible to define a C_p-like statistic that can be used for tuning the parameters of SVR. The proposed tuning procedure is tested on a simulated benchmark problem and on a real world problem (Boston Housing data set).

Keywords: statistical learning, reproducing kernel Hilbert spaces, support vector machines, representer theorem, regularization theory

1. Introduction

Although Support Vector Machines are mainly used as classification algorithms, recent years have witnessed a growing interest for their application to regression problems as well. Among the advantages of SVR (Support Vector Regression), there are the sparseness property and the robustness against outliers.

The SVR estimator can be seen as the minimizer of a cost functional given by the sum of an ε-insensitive loss function and a regularization penalty. As such, it is a particular case of a larger class of kernel-based estimators that are obtained by applying regularization theory in Reproducing Kernel Hilbert Spaces (RKHS). Under mild assumptions, the solution of these problems can be written as a linear combination of kernel functions. This kind of result goes under the name of representer theorem. The first result of this type was due to Kimeldorf and Wahba (1979) for squared loss functions, see also Tikhonov and Arsenin (1977) for the application in the context of inverse problems.

The representer theorem was further generalized to differentiable loss functions (Cox and O' Sullivan, 1990; Poggio and Girosi, 1992) and even arbitrary monotonic ones (Schölkopf et al., 2001). Another important issue is the quantitative characterization of the coefficients a_i of the linear combination. For squared losses it is well known that the coefficients are obtained as the solution of a system of linear equations, see for example Wahba (1990) and Cucker and Smale (2001). An explicit characterization of the coefficients as the solution of a system of algebraic equations is still possible if the loss function is differentiable (Wahba, 1998). This result cannot be applied to ε-insensitive SVR because the loss function is not differentiable. The usual computational approach is to reformulate the original variational problem as a constrained minimization one whose dual Lagrangian formulation boils down to a finite dimensional quadratic programming problem (Vapnik, 1995).

Some recent contributions have approached the nondifferentiability issue by resorting to subdifferential calculus. More precisely, Steinwart (2003) has proven a quantitative representer theorem that, without using the dual problem, characterizes the coefficients by means of inclusions, when convex loss functions are considered. Various extensions can be found in De Vito et al. (2004). In particular, besides providing an alternative simpler proof of the quantitative representer theorem, De Vito and coworkers allow for the offset space and cover both regression and classification.

The contribution of the present paper is twofold. First of all, quantitative representation results are worked out for convex loss functions. Then, these results are specialized to SVR in order to study its sensitivity to data and develop a tuning method for its parameters.

Concerning the quantitative representation of the coefficients a_i, the paper provides a simple derivation of the quantitative representer theorem based on Fourier arguments (see Appendix A). Another result is a new formulation of the quantitative representer theorem that replaces inclusions with equations by using the newly introduced notion of pseudoresidual (Theorem 1). This result, not only gives insight into the relation between data and coefficients, but also puts the basis for the subsequent analysis of SVR properties. In particular, we give a complete characterization of the sensitivity of SVR coefficients and predictions with respect to the output data. Past work has focused on sensitivity with respect to the regularization parameter C, see for example Pontil and Verri (1998) and Hastie et al. (2004). As a byproduct of the sensitivity analysis, the degrees of freedom of SVR, defined as the trace of the sensitivity matrix, are found to be equal to the number of marginal support vectors. This analysis is instrumental to the last issue dealt with in the paper, that is the tuning of both the ε and C parameters of the SVR.

In the literature, the tuning of SVR has been addressed using various approaches. The interpretation of SVR as a Bayesian estimator provides a conceptually elegant framework for reformulating parameter tuning as a statistical estimation problem (Gao et al., 2002). The major drawback is the necessity of assuming the validity of the statistical prior underlying the Bayesian interpretation of SVR, an assumption that may not be appropriate in all cases.

As a matter of fact, the great majority of tuning approaches aims at the minimization of the prediction error. A powerful, though computationally expensive solution is to resort to k-fold cross validation. Alternatively, Chang and Lin (2005) strive for the minimization of an upper bound of the leave-one-out absolute error. Other authors have discussed the choice of ε observing that, asymptotically, the optimal ε depends linearly on the measurement error standard deviation (Smola et al., 1998; Kwok and Tsang, 2003). Finally, Schölkopf et al. (2000) have proposed modified SVR schemes that ease the tuning of the parameters.

A tuning method based on the extension of the *GCV* criterion to SVR has been proposed by Gunter and Zhu (2007).

In the present paper, a different approach is pursued which is based on the estimation of the so-called in-sample prediction error (Hastie et al., 2001). See also Cherkassky and Ma (2003) and Hastie et al. (2003) for a discussion on the merits and difficulties of this and other approaches to model selection. Herein, it is shown how the optimism, that is the difference between the in-sample prediction error and the expected empirical risk, depends on the sensitivity of the estimator (Theorem 3). This result opens the way to the estimation of the in-sample prediction error as a function of the measurement error variance and the degrees of freedom. This estimator can be seen as an extension to SVR of the so-called C_p statistic, a well known criterion for linear model order selection. A major advantage of the C_p statistic is that, differently from many other criteria, no assumption is made on the correctness of the model.

The paper is organized as follows. After some preliminaries (Section 2), the major results regarding the representation of the coefficients a_i and the sensitivity analysis of SVR are derived in Section 3, which ends with the definition of the degrees of freedom. The issue of parameter tuning is treated in Section 4, where the estimation of the in-sample prediction error by means of a suitable C_p statistic is addressed. Finally, the proposed parameter tuning procedure is illustrated in Section 5 by means of both a simulated problem and a real-world one. Some concluding remarks (Section 6) end the paper.

2. Preliminaries

Consider the problem of estimating the functional relationship existing between an input vector $\mathbf{x} \in \mathbb{R}^N$ and the output $y \in \mathbb{R}$ given the training set $\mathcal{D} = \{\mathbf{x}_i, y_i\}$ $(i = 1, 2, ..., \ell)$, where the input vectors $\mathbf{x}_i$ are all distinct. According to the Support Vector Regression approach, the function $\hat{f}(\mathbf{x}) : \mathbb{R}^N \to \mathbb{R}$ solving the aforementioned problem belongs to a Reproducing Kernel Hilbert Space (RKHS) $\mathcal{H}$ and minimizes the regularized risk:

$$\hat{f} = \arg\min_{f \in \mathcal{H}} H[f] = \arg\min_{f \in \mathcal{H}} \left(C \sum_{i=1}^{\ell} V(y_i, f(\mathbf{x}_i)) + \frac{1}{2} \|f\|_{\mathcal{H}}^2 \right). \tag{1}$$

The parameter C controls the relative importance given to the empirical risk and the regularization term $\|f\|_{\mathcal{H}}^2$, and must be properly tuned in order to obtain good performance. Among the possible convex loss functions V (quadratic, Laplace, etc) particular attention will be given to the ε-insensitive one:

$$V(y_i, f(\mathbf{x}_i)) = V_\varepsilon(y_i - f(\mathbf{x}_i)) = \begin{cases} 0, & |f(\mathbf{x}_i) - y_i| \leq \varepsilon \\ |f(\mathbf{x}_i) - y_i| - \varepsilon, & |f(\mathbf{x}_i) - y_i| > \varepsilon. \end{cases} \tag{2}$$

Such a function is known to produce sparse solutions, meaning that they depend only on a small number of training examples scattered in the input space. The positive scalar ε measures the extent of the "dead zone" (that is the interval over which the loss function is zero) and should be either fixed according to the desired resolution or tuned using an objective criterion.

The usual approach to the numerical computation of $\hat{f}$ calls for the solution of the dual quadratic programming problem, see for example Vapnik (1995). If the kernel is positive definite, the representer theorem states that the solution $\hat{f}$ can be written as

$$\hat{f}(\mathbf{x}) = \sum_{i=1}^{\ell} a_i K(\mathbf{x}_i, \mathbf{x}), \tag{3}$$

where a_i are suitable coefficients. If V is everywhere differentiable with respect to its second argument, it can be shown that

$$a_i = -C\partial_2 V(y_i, \hat{f}(\mathbf{x}_i)),$$

where ∂_2 denotes the partial derivative with respect to the second argument. Conversely, if V is a general measurable function convex with respect to its second argument, it is necessary to resort to subdifferential calculus (for a quick reference to the basic concepts of subdifferential calculus the interested reader may usefully refer to Steinwart (2003) and De Vito et al. (2004)). See also Borwein and Lewis (2000). In particular, Steinwart (2003) and De Vito et al. (2004) (Theorem 2) have shown that

$$a_i \in -C\partial_2 V(y_i, \hat{f}(\mathbf{x}_i)), \tag{4}$$

where, now, ∂_2 is the subdifferential with respect to the second argument. This result goes under the name of quantitative representer theorem. Note that (4) is no longer an equation but just an inclusion.

De Vito et al. (2004) studied also the so called continuous setting, that is measurements are taken on a continuous set rather than being taken as discrete samples. In Appendix A, we provide an alternative concise proof of the quantitative representer theorem based on Fourier arguments. The analysis developed in the next section differs from the representation results by Steinwart (2003) and De Vito et al. (2004) in that we show that the system of inclusions (4) can be replaced by a set of equations.

3. Quantitative Representation and Sensitivity Analysis

Hereafter, it is assumed that the loss function is of the type

$$V(y_i, f(\mathbf{x}_i)) = V(f(\mathbf{x}_i) - y_i),$$

where $V(\cdot)$ is a convex function and is twice differentiable everywhere except in a finite number of points γ_j, $j = 1, \ldots, N$. In the following, $D^-(\gamma)$ and $D^+(\gamma)$ will denote the left and right derivative of $V(\cdot)$ at γ:

$$D^-(\gamma) = \lim_{h \to 0^+} \frac{V(\gamma - h) - V(\gamma)}{-h},$$

$$D^+(\gamma) = \lim_{h \to 0^+} \frac{V(\gamma + h) - V(\gamma)}{h},$$

Letting $I = \{1, 2, \ldots, \ell\}$, define the *pseudoresiduals* as

$$\eta_i := y_i - \sum_{\substack{j \in I \\ j \neq i}} a_j K(\mathbf{x}_i, \mathbf{x}_j).$$

The following result holds

Theorem 1 *The coefficients a_i, $i = 1,\ldots,\ell$, that characterize the solution of problem (1), satisfy a system of algebraic equations*

$$a_i = S_i(\eta_i),$$

where $S_i(\eta_i)$ are monotone nondecreasing Lipschitz continuous functions. Moreover, when

$$\eta_i \in \left[-\left(\gamma_j + CK(\mathbf{x}_i,\mathbf{x}_i)D^+(\gamma_j)\right), -\left(\gamma_j + CK(\mathbf{x}_i,\mathbf{x}_i)D^-(\gamma_j)\right)\right], \quad j = 1,\ldots,N, \tag{5}$$

the functions $S_i(\eta_i)$ are affine and given by

$$S_i(\eta_i) = \frac{\eta_i + \gamma_j}{K(\mathbf{x}_i,\mathbf{x}_i)}.$$

Proof. By the definition of pseudoresidual,

$$\hat{f}(\mathbf{x}_i) - y_i = a_i K(\mathbf{x}_i,\mathbf{x}_i) - \eta_i. \tag{6}$$

Then,

$$a_i = \frac{\eta_i + \hat{f}(\mathbf{x}_i) - y_i}{K(\mathbf{x}_i,\mathbf{x}_i)}.$$

Now, there are two cases depending on whether $V(\cdot)$ is twice differentiable at $\gamma := \hat{f}(\mathbf{x}_i) - y_i$ or not. When $\gamma \neq \gamma_j$, $j = 1,\ldots,N$, $V(\gamma)$ is twice differentiable and its subdifferential is single-valued so that (4) yields

$$a_i = -CV'(\hat{f}(\mathbf{x}_i) - y_i) = -CV'(a_i K(\mathbf{x}_i,\mathbf{x}_i) - \eta_i). \tag{7}$$

Now, the Implicit Function Theorem can be used to prove that, locally, a_i is a monotone nondecreasing Lipschitz continuous function of η_i. In fact, by deriving with respect to η_i,

$$\frac{\partial a_i}{\partial \eta_i} = \frac{CV''(a_i K(\mathbf{x}_i,\mathbf{x}_i) - \eta_i)}{1 + CK(\mathbf{x}_i,\mathbf{x}_i)V''(a_i K(\mathbf{x}_i,\mathbf{x}_i) - \eta_i)}.$$

The denominator is always different from zero because, by convexity, $V'' \geq 0$ whenever it exists. Therefore, locally, a_i is a differentiable function of η_i:

$$a_i = \overline{S}(\eta_i).$$

The function $\overline{S}(\eta_i)$ is monotone nondecreasing and has bounded derivative because

$$0 \leq \frac{\partial a_i}{\partial \eta_i} < \frac{1}{K(\mathbf{x}_i,\mathbf{x}_i)}. \tag{8}$$

Now, let us consider the second case. When γ is fixed as $\gamma = \gamma_j$ for some j, $V(\cdot)$ is not twice differentiable at γ. Then, from (6),

$$a_i = S_i(\eta_i) = \frac{\gamma_j + \eta_i}{K(\mathbf{x}_i,\mathbf{x}_i)}, \tag{9}$$

so that a_i is an affine function of η_i in the interval

$$I_j := [\eta_j^L, \eta_j^R],$$

where

$$\begin{aligned} \eta_j^L &:= -\left(\gamma_j + CK(\mathbf{x}_i, \mathbf{x}_i) D^+(\gamma_j)\right), \\ \eta_j^R &:= -\left(\gamma_j + CK(\mathbf{x}_i, \mathbf{x}_i) D^-(\gamma_j)\right). \end{aligned}$$

On the other hand, recalling the properties of the subdifferential of a convex function,

$$a_i \in \left[-CD^+(\gamma_j), -CD^-(\gamma_j)\right]. \tag{10}$$

Hence, (5) follows from (9) and (10). Finally, since

$$\frac{\partial a_i}{\partial \eta_i} = \frac{1}{K(\mathbf{x}_i, \mathbf{x}_i)} > 0,$$

the functions $S_i(\eta_i)$ are locally monotone nondecreasing also in the second case. Combining this last inequality with the bound (8) that holds in differentiability points, we conclude that the derivative of $S_i(\eta_i)$ is bounded everywhere, possibly except for discontinuity points. Then, in order to prove Lipschitz continuity it suffices to show that $S_i(\eta_i)$ is continuous.

We now conclude the proof showing that the set of discontinuity points is actually empty. In this respect, the only points that must be analyzed are the boundaries of the intervals I_j. In fact, in the interior of I_j, $S_i(\eta_i)$ is infinitely differentiable because it is affine, while, outside, it has the same regularity of $V'(\cdot)$. Hence, it suffices to prove continuity at the left boundary η_j^L of I_j. Consider (9) and take the limit from the right:

$$\lim_{\eta_i \to (\eta_j^L)^+} S_i(\eta_i)\Big|_{\eta_i \in I_j} = \lim_{\eta_i \to (\eta_j^L)^+} \frac{\gamma_j + \eta_i}{K(\mathbf{x}_i, \mathbf{x}_i)} = -CD^+(\gamma_j).$$

Now, observe that, if a_i tends to $-CD^+(\gamma_j)$ from below, then η_i tends to η_j^L from the left. Indeed, taking the limit in (7) for $a_i \to -CD^+(\gamma_j)$ from below, we obtain that $\hat{f}(\mathbf{x}_i) - y_i \to \gamma_j$ from the right (recall that $V'(\cdot)$ is nondecreasing). In turn,

$$\begin{aligned} \lim_{a_i \to \left(-CD^+(\gamma_j)\right)^-} \eta_i &= \lim_{a_i \to \left(-CD^+(\gamma_j)\right)^-} \left(y_i - \hat{f}(\mathbf{x}_i) + a_i K(\mathbf{x}_i, \mathbf{x}_i)\right) \\ &= -\gamma_j - CD^+(\gamma_j) K(\mathbf{x}_i, \mathbf{x}_i) = \eta_j^L \end{aligned}$$

from the left. This proves the continuity of $S_i(\eta_i)$ at the left boundary η_j^L of I_j. ■

Hereafter, it will be assumed that $V = V_\varepsilon$ is the so-called ε-insensitive function. Hence, V_ε is not differentiable only at $\gamma_1 = -\varepsilon$ and $\gamma_2 = +\varepsilon$. The subdifferential of the loss function has a rather simple structure:

$$C\partial V_\varepsilon(\hat{f}(\mathbf{x}_i)-y_i) = \begin{cases} \{-C\} & \hat{f}(\mathbf{x}_i)-y_i < -\varepsilon, \\ [-C,0] & \hat{f}(\mathbf{x}_i)-y_i = -\varepsilon, \\ \{0\} & -\varepsilon < \hat{f}(\mathbf{x}_i)-y_i < \varepsilon, \\ [0,C] & \hat{f}(\mathbf{x}_i)-y_i = \varepsilon, \\ \{C\} & \hat{f}(\mathbf{x}_i)-y_i > \varepsilon. \end{cases}$$

For the subsequent derivation it is useful to define the following sets:

$$\begin{aligned} I_{in} &= \{i \in I : |\hat{f}(\mathbf{x}_i)-y_i| < \varepsilon\}, \\ I_C^+ &= \{i \in I : \hat{f}(\mathbf{x}_i)-y_i > \varepsilon\}, \\ I_C^- &= \{i \in I : \hat{f}(\mathbf{x}_i)-y_i < -\varepsilon\}, \\ I_M^+ &= \{i \in I : \hat{f}(\mathbf{x}_i)-y_i = \varepsilon\}, \\ I_M^- &= \{i \in I : \hat{f}(\mathbf{x}_i)-y_i = -\varepsilon\}, \\ I_{out} &= I_C^+ \cup I_C^- \qquad I_M = I_M^+ \cup I_M^-. \end{aligned}$$

Note that the set I_{in} identifies the data pairs $\{\mathbf{x}_i, y_i\}$ that belong to the so-called ε-tube, whereas I_{out} identifies the data outside the tube. The indices belonging to I_M correspond to data pairs lying on the boundary of the ε-tube, also called marginal support vectors. The union of I_{out} and I_M identifies the so-called support vectors.
In view of Theorem 1, the next corollary follows.

Corollary 1 *For the ε-insensitive loss function,*

$$a_i = S_i(\eta_i) = \begin{cases} -C & \eta_i \le -(\varepsilon + CK(\mathbf{x}_i,\mathbf{x}_i)), \\ \frac{\eta_i+\varepsilon}{K(\mathbf{x}_i,\mathbf{x}_i)} & -(\varepsilon + CK(\mathbf{x}_i,\mathbf{x}_i)) < \eta_i < -\varepsilon, \\ 0 & -\varepsilon \le \eta_i \le \varepsilon, \\ \frac{\eta_i-\varepsilon}{K(\mathbf{x}_i,\mathbf{x}_i)} & \varepsilon < \eta_i < (\varepsilon + CK(\mathbf{x}_i,\mathbf{x}_i)), \\ C & \eta_i \ge (\varepsilon + CK(\mathbf{x}_i,\mathbf{x}_i)). \end{cases}$$

Moreover,

- *If $i \in I_{in}$, $|\eta_i| \le \varepsilon$.*
- *If $i \in I_{out}$, $|\eta_i| \ge \varepsilon + CK(\mathbf{x}_i,\mathbf{x}_i)$.*
- *If $i \in I_M$, $\varepsilon \le |\eta_i| \le \varepsilon + CK(\mathbf{x}_i,\mathbf{x}_i)$.*

Corollary 1, which is illustrated in Fig. 1, is now used to evaluate the sensitivity of SVR with respect to the data.

Let $m = \#I_M$ denote the number of marginal vectors. We can assume without loss of generality that the indices I are ordered such that $I_M = \{1,\dots,m\}$, $I_{\bar{M}} := I_{out} \cup I_{in} = \{m+1,\dots,\ell\}$. Let the matrix $\mathbf{K} := [K(\mathbf{x}_i,\mathbf{x}_j)]$ be partitioned as

$$\mathbf{K} = \begin{pmatrix} \mathbf{K}_{MM} & \mathbf{K}_{M\bar{M}} \\ \mathbf{K}_{\bar{M}M} & \mathbf{K}_{\bar{M}\bar{M}} \end{pmatrix},$$

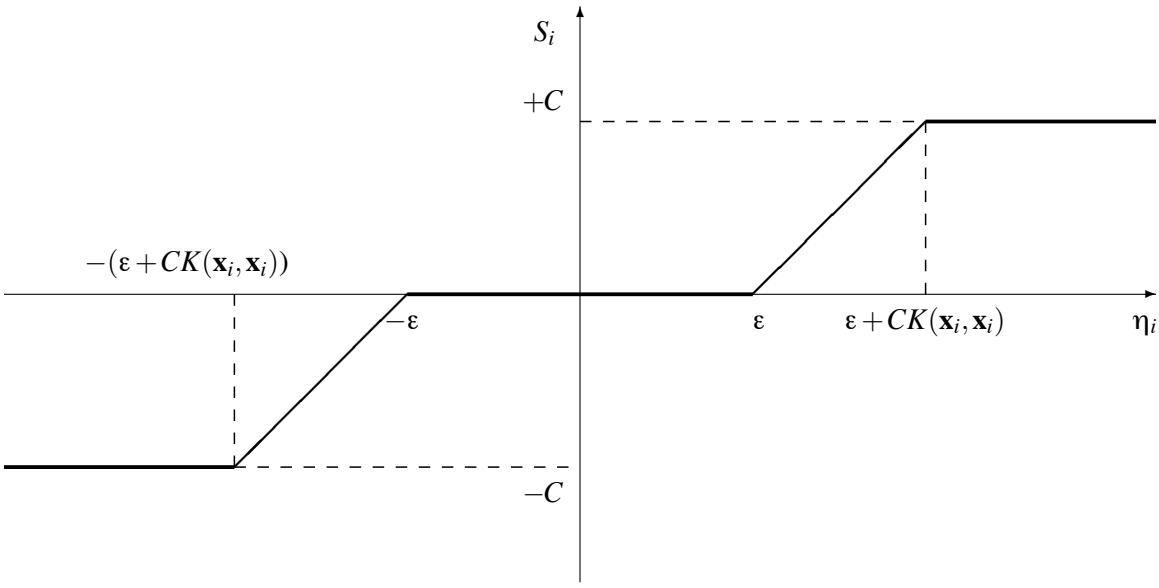


Figure 1: This function S_i gives the dependency of the coefficient a_i on the pseudoresidual η_i when the ε-insensitive loss function is used.

where $\mathbf{K}_{MM} \in \mathbb{R}^{m \times m}$. It is also useful to partition the coefficient vector $\mathbf{a} = (a_1, \ldots, a_\ell)^T$ as $(\mathbf{a}_M^T, \mathbf{a}_{\bar{M}}^T)^T$ and the data vector $\mathbf{y} = (y_1, \ldots, y_\ell)^T$ as $(\mathbf{y}_M^T, \mathbf{y}_{\bar{M}}^T)^T$, where $\mathbf{a}_M, \mathbf{y}_M \in \mathbb{R}^m$. Moreover, define

$$\mathbf{k}(\mathbf{x}) = \begin{pmatrix} K(\mathbf{x}, \mathbf{x}_1) \\ \ldots \\ K(\mathbf{x}, \mathbf{x}_\ell) \end{pmatrix}.$$

Proposition 1 *Assume that* $\eta_i \neq \pm\varepsilon$ *and* $\eta_i \neq \pm(\varepsilon + CK(\mathbf{x}_i, \mathbf{x}_i))$. *Then,*

$$\frac{\partial \hat{f}}{\partial y}(\mathbf{x}) = \mathbf{k}(\mathbf{x})^T \begin{pmatrix} \mathbf{K}_{MM}^{-1} & 0 \\ 0 & 0 \end{pmatrix}.$$

Proof. First of all, by (3)

$$\frac{\partial \hat{f}}{\partial y_k}(\mathbf{x}) = \sum_{j=1}^{\ell} \frac{\partial a_j}{\partial y_k} K(\mathbf{x}, \mathbf{x}_j).$$

In view of Corollary 1 and the definition of pseudoresidual η_i,

$$\frac{\partial a_i}{\partial y_k} = S_i'(\eta_i) \left(\delta_{ik} - \sum_{\substack{j \in I \\ j \neq i}} \frac{\partial a_j}{\partial y_k} K(\mathbf{x}_i, \mathbf{x}_j) \right), \forall i \tag{11}$$

where

$$S_i'(\eta_i) = \begin{cases} 0 & i \notin I_M, \\ \frac{1}{K(\mathbf{x}_i,\mathbf{x}_i)} & i \in I_M \end{cases}$$

and δ_{ik} is Kronecker's delta. Hence, $\forall i \notin I_M, \forall k$,

$$\frac{\partial a_i}{\partial y_k} = 0. \tag{12}$$

On the other hand, $\forall i \in I_M, \forall k$, (11) reads

$$\frac{\partial a_i}{\partial y_k} = \frac{1}{K(\mathbf{x}_i,\mathbf{x}_i)} \left(\delta_{ik} - \sum_{\substack{j \in I_M \\ j \neq i}} \frac{\partial a_j}{\partial y_k} K(\mathbf{x}_i,\mathbf{x}_j) \right),$$

whence

$$\sum_{j \in I_M} \frac{\partial a_j}{\partial y_k} K(x_i,x_j) = \delta_{ik}, \qquad \forall i \in I_M, \forall k \in I. \tag{13}$$

Equations (12) and (13) can be written as

$$\frac{\partial \mathbf{a}_{\bar{M}}}{\partial \mathbf{y}} = 0,$$

$$\mathbf{K}_{MM} \frac{\partial \mathbf{a}_M}{\partial \mathbf{y}} = \begin{pmatrix} I & 0 \end{pmatrix}.$$

Since the vectors $\mathbf{x}_i$ are all distinct, $\mathbf{K}_{MM}$ is a positive definite matrix and therefore

$$\frac{\partial \mathbf{a}}{\partial \mathbf{y}} = \begin{pmatrix} \mathbf{K}_{MM}^{-1} & 0 \\ 0 & 0 \end{pmatrix},$$

from which the thesis follows. ■

For linear-in-parameter regression it is usual to define the degrees of freedom of the estimator as the trace of the so-called "hat matrix," that maps the vector of output data into the corresponding predictions. Such degrees of freedom have a number of applications ranging from the computation of confidence intervals to model validation and model order selection, see for example Hastie and Tibshirani (1990) and Hastie et al. (2001).

Let $\hat{\mathbf{y}} = (\hat{y}_1, \ldots, \hat{y}_\ell)^T$, where $\hat{y}_i = \hat{f}(\mathbf{x}_i)$ denotes the SVR prediction at $\mathbf{x}_i$. The following Proposition provides the degrees of freedom of SVR. For an alternative proof, based on the dual problem formulation, see Gunter and Zhu (2007).

Proposition 2 *Let the degrees of freedom of the SVR be defined as*

$$q(\mathcal{D}) := tr\left(\frac{\partial \hat{\mathbf{y}}}{\partial \mathbf{y}} \right).$$

Then, under the assumption of Proposition 1, $q(\mathcal{D})$ is equal to the number m of marginal support vectors.

Proof. If $\mathbf{x} = \mathbf{x}_i$, the application of Proposition 1 yields

$$\frac{\partial \hat{\mathbf{y}}}{\partial \mathbf{y}} = \begin{pmatrix} I & 0 \\ \mathbf{K}_{\bar{M}M}\mathbf{K}_{MM}^{-1} & 0 \end{pmatrix}, \tag{14}$$

so that $q(\mathcal{D})$ is just equal to m.

Remark 1 *Note that the number m of marginal vectors can be evaluated by looking at the pseudoresiduals* η_i*. More precisely, the i-th observation* y_i *is a marginal vector if*

$$\varepsilon \leq |\eta_i| \leq \varepsilon + CK(\mathbf{x}_i, \mathbf{x}_i).$$

Remark 2 *The assumption on the value* η_i *made in Proposition 1 rules out the marginal support vectors whose coefficient* a_i *is either* 0 *or* $\pm C$*. This corresponds to experimental data that under a suitable infinitesimal perturbation leave the boundary of the* ε*-tube moving either inward (*$a_i = 0$*) or outward (*$a_i = \pm C$*). For such "transition data"* y_i*, the right and left derivatives* $\partial \hat{y}_i / \partial y_i$ *are different, so that the degrees of freedom would not be uniquely defined. Then, the degrees of freedom would range from the minimum to the maximum value of* $tr(\partial \hat{\mathbf{y}} / \partial \mathbf{y})$*. Alternatively, one could assign* $1/2$ *degree of freedom to each transition datum. Given that such pathological situation occurs on a zero-measure set, they will be removed from the analysis without appreciable consequences.*

4. Prediction Error Assessment via C_p Statistic

The goal of any regression method is to achieve good generalization performance. In this section, an index will be derived that assesses the generalization capabilities of SVR. In turn, this index can be used to tune the design parameters of the estimator.

In order to proceed, it is assumed that the training data are given by

$$y_i = f^0(x_i) + v_i, \tag{15}$$

where $f^0(x)$ is the "true function" to be estimated and the measurement error vector $v = [v_1 \dots v_\ell]^T$ is such that $E[v] = 0$, $Var[v] = diag(\sigma_1^2 \dots \sigma_\ell^2)$. Note that $f^0(x_i)$ can be seen as the conditional expectation of y_i given x_i ($f^0(x_i) = E[y_i|x_i]$) and $f^0(x)$ is also known as regression function.

In the following, the generalization performance will be measured in terms of the sum of squared errors. A first type of error is the empirical risk

$$\overline{err} := \frac{1}{\ell}\|y - \hat{y}\|^2.$$

This is not a valid measure of generalization because $\hat{y}$ depends on y. Usually, the generalization capabilities of the estimator are measured by the expected risk. Unfortunately, it is not easy to assess the value of the expected risk without introducing assumptions on the nature of $f^0(x_i)$. As an alternative, one can look for probabilistic upper bounds on the expected risk which may be, in some cases, too loose for an optimal tuning of the SVR parameters. Hereafter, attention will be focused on the so-called in-sample prediction error, that is the expected error associated with a new set of data

$$y_i^{new} = f^0(x_i) + v_i^{new},$$

where v^{new} has the same statistics as v but is independent of it. The in-sample prediction error, see for example Hastie et al. (2001), is defined as

$$Err_{in} := E\left[\frac{1}{\ell}\|y^{new} - \hat{y}\|^2\right],$$

where the x_i are fixed and the expected value is taken with respect to both the distribution of y_i and y_i^{new}. A major motivation for using the in-sample prediction error is that, as shown below, it can be assessed with good accuracy, without introducing undue assumptions on $f^0(x)$.

Remark 3 *Recalling the expression of the cost function (1) it would be tempting to use*

$$Err_{in}^V := E\left[\frac{1}{\ell}\sum_{i=1}^{\ell} V(y_i^{new}, \hat{y}_i)\right]$$

as a measure of generalization performance. In particular, the parameters (C, ε) would be tuned so as to minimize Err_{in}^V. However, it is immediate to see that $Err_{in}^V = 0$ for sufficiently large ε so that a joint tuning of the two parameters is not possible. Conversely, as observed by Hastie et al. (2001), the in-sample prediction error proves useful for model comparison and selection because, although it underestimates the expected risk, in this context the relative size of the error is what matters, see also Efron (1986). Note also that the use of SVR is not necessarily in contrast with square loss minimization, insofar sparsity of the solution is an important feature. On these premises, in the present paper the minimization of the quadratic in-sample prediction error Err_{in} is pursued. A similar choice has been made by Gunter and Zhu (2007) who derive a quadratic-type GCV criterion for SVR.

The empirical and the in-sample prediction error are linked as stated in the following proposition, see for example Hastie et al. (2001). Note that the expectations are taken over the training set.

Proposition 3 *Define the 'optimism' as*

$$op := \frac{2}{\ell} E[\hat{y}^T v].$$

Then,

$$Err_{in} = E[\overline{err}] + op.$$

The index Err_{in} can be approximated by

$$\widehat{Err}_{in} = \overline{err} + \widehat{op},$$

where $\widehat{op}$ is an estimate of op. If $\hat{y}$ is a linear function of y, and $\sigma_i^2 = \sigma^2, \forall i$, then the optimism can be expressed as

$$op = \frac{2q\sigma^2}{\ell} \tag{16}$$

(note that in the linear case the degrees of freedom q do not depend on the training data y). In this linear case, $\widehat{Err}_{in}$ is better known as C_p statistic

$$C_p = \overline{err} + \frac{2q\sigma^2}{\ell}. \tag{17}$$

The purpose of the present section is to extend the C_p statistic to Support Vector Regression. The following theorem highlights the relationship between the optimism and the sensitivities $\partial h_i/\partial y_i$ of a generic estimator $\hat{y} = h(y)$. Let us define $y^0 = [f^0(x_1) \dots f^0(x_\ell)]^T$.

Theorem 2 *Assume that*

(i) eq. (15) holds,

(ii) the errors v_i are independent of each other,

(iii) the variances $\sigma_i^2 = Var[v_i]$ are finite,

(iv) the estimator $h(y)$ is such that for $i = 1, \dots, \ell$

$$\lim_{|y_i| \to \infty} \frac{|h_i(y)|}{|y_i|} = 0.$$

Then,

$$op = \frac{2}{\ell} \sum_{i=1}^{\ell} \sigma_i^2 \int_{\mathbb{R}^\ell} \frac{\partial h_i}{\partial y_i}(y) \prod_{j \neq i}^{\ell} p_j(v_j) \phi_i(v_i) dv,$$

where

$$\phi_i(v_i) = \frac{1}{\sigma_i^2} \int_{v_i}^{+\infty} s p_i(s) ds,$$

and $p_i(v_i)$ denotes the probability density function of v_i.
Moreover, if the errors v_i are Gaussian,

$$op = \frac{2}{\ell} \sum_{i=1}^{\ell} \sigma_i^2 E\left[\frac{\partial h_i}{\partial y_i}(y)\right]. \tag{18}$$

Proof. By definition,

$$\begin{aligned} op &= \frac{2}{\ell} E[\hat{y}^T v] = \frac{2}{\ell} \sum_{i=1}^{\ell} E[h_i(y) v_i] \\ &= \frac{2}{\ell} \sum_{i=1}^{\ell} \int_{\mathbb{R}^{\ell-1}} \prod_{j \neq i}^{\ell} p_j(v_j) \left(\int_{\mathbb{R}} h_i(y^0 + v) v_i p_i(v_i) dv_i \right) dv^{[-i]}, \end{aligned}$$

where

$$dv^{[-i]} = \prod_{j \neq i}^{\ell} dv_j.$$

Integration by parts of the inner integral yields

$$\int_{\mathbb{R}} h_i(y^0+v)v_i p_i(v_i)dv_i = \sigma_i^2 \left(\int_{\mathbb{R}} \frac{\partial h_i}{\partial v_i}(y^0+v)\phi_i(v_i)dv_i - [h_i(y^0+v)\phi_i(v_i)]|_{-\infty}^{+\infty} \right).$$

Now, we can show that the last term on the right hand side is zero. In fact, for positive v_i we have

$$\phi_i(v_i) = \frac{1}{\sigma_i^2}\int_{v_i}^{+\infty} s p_i(s)ds = \frac{1}{\sigma_i^2}\int_{v_i}^{+\infty} \frac{s^2 p_i(s)}{s}ds \leq \frac{1}{\sigma_i^2 v_i}\int_{v_i}^{+\infty} s^2 p_i(s)ds = \frac{1}{v_i}.$$

For negative v_i, observing that $\int_{v_i}^{+\infty} s p_i(s)ds = -\int_{-\infty}^{v_i} s p_i(s)ds$ (recall that $E[v_i]=0$), a similar argument yields $|\phi_i(v_i)| \leq \frac{1}{|v_i|}$. In conclusions, we have that $|\phi_i(v_i)| \leq \frac{1}{|v_i|}$. Now, the sublinear growth of the estimator (iv) gives

$$\lim_{|v_i|\to\infty} |h_i(y^0+v)\phi_i(v_i)| \leq \lim_{|v_i|\to\infty} \frac{|h_i(y^0+v)|}{|v_i|} = 0.$$

Now, we have

$$\frac{\partial h_i}{\partial v_i}(y^0+v) = \frac{\partial h_i}{\partial y_i}(y^0+v),$$

so that

$$\int_{\mathbb{R}} h_i(y^0+v)v_i p_i(v_i)dv_i = \sigma_i^2 \int_{\mathbb{R}} \frac{\partial h_i}{\partial y_i}(y^0+v)\phi_i(v_i)dv_i.$$

Then,

$$\begin{aligned} op &= \frac{2}{\ell}\sum_{i=1}^{\ell} \sigma_i^2 \int_{\mathbb{R}^{\ell-1}} \prod_{j\neq i}^{\ell} p_j(v_j) \left(\int_{\mathbb{R}} \frac{\partial h_i}{\partial y_i}(y^0+v)\phi_i(v_i)dv_i \right) dv^{[-i]} \\ &= \frac{2}{\ell}\sum_{i=1}^{\ell} \sigma_i^2 \int_{\mathbb{R}^{\ell}} \frac{\partial h_i}{\partial y_i}(y) \prod_{j\neq i}^{\ell} p_j(v_j)\phi_i(v_i)dv. \end{aligned}$$

Finally, if the errors v_i are Gaussian,

$$\phi_i(v_i) = \frac{1}{\sigma_i^2}\int_{v_i}^{+\infty} \frac{s}{\sqrt{2\pi}\sigma_i} e^{-\frac{s^2}{2\sigma_i^2}} ds = -\frac{1}{\sqrt{2\pi}\sigma_i} e^{-\frac{s^2}{2\sigma_i^2}} \Big|_{v_i}^{+\infty} = p_i(v_i).$$

Therefore,

$$op = \frac{2}{\ell}\sum_{i=1}^{\ell} \sigma_i^2 \int_{\mathbb{R}^{\ell}} \frac{\partial h_i}{\partial y_i}(y) \prod_{j\neq i}^{\ell} p_j(v_j)\phi_i(v_i)dv = \frac{2}{\ell}\sum_{i=1}^{\ell} \sigma_i^2 E\left[\frac{\partial h_i}{\partial y_i}(y)\right]$$

thus proving the thesis.

Remark 4 *Although linear estimators do not fulfill assumption (iv), the thesis still holds. In fact, recalling that for a linear estimator the degrees of freedom do not depend on y, expression (16) is eventually recovered.*

Note that (18) was already known in the context of Stein's unbiased risk estimators (Stein, 1981). The next theorem derives a simple expression for the optimism of the SVR estimator in a somehow ideal case (see assumption *(v)* below).

Theorem 3 *Assume that*

(i) eq. (15) holds,

(ii) the errors v_i are independent of each other,

(iii) the variances $\sigma_i^2 = Var[v_i]$ are finite,

(iv) $C < +\infty$,

(v) the set I_M of the marginal vectors does not depend on v.

Then, the optimism of the SVR is

$$op^{SVR} = \frac{2}{\ell} \sum_{i \in I_M} \sigma_i^2.$$

Proof. The proof is based on Theorem 2, whose assumptions *(i)-(iii)* are obviously satisfied. Concerning assumption *(iv)*, consider a vector y and fix all its entries but the i-th one y_i. Then, there exists $\kappa_i > 0$ such that $i \in I_{out}$ whenever $|y_i| > \kappa_i$. Hence, for $|y_i|$ large enough, $|h_i(y)|$ is a finite constant so that assumption *(iv)* of Theorem 2 is satisfied.

Now, observe that, for SVR, the derivatives $\frac{\partial h_i}{\partial y_i}$ are all equal to either 0 or 1, see (14). In particular, $\frac{\partial h_i}{\partial y_i}$ is different from zero if and only if $i \in I_M$. Then, in view of assumption *(v)*,

$$op^{SVR} = \frac{2}{\ell} \sum_{i \in I_M} \sigma_i^2 \int_{\mathbb{R}^\ell} \prod_{\substack{j \in I \\ j \neq i}} p_j(v_j) \phi_i(v_i) dv = \frac{2}{\ell} \sum_{i \in I_M} \sigma_i^2 \int_{\mathbb{R}} \phi_i(v_i) dv_i$$

The thesis is proven by showing that the last integral equals one:

$$\begin{aligned} \int_{\mathbb{R}} \phi_i(v_i) dv_i &= \frac{1}{\sigma_i^2} \int_{-\infty}^{+\infty} \int_{v_i}^{+\infty} s p_i(s) ds dv_i = \frac{1}{\sigma_i^2} \int_{-\infty}^{+\infty} \int_{1}^{+\infty} v_i^2 z p_i(z v_i) dz dv_i \\ &= \frac{1}{\sigma_i^2} \int_{1}^{+\infty} z \int_{-\infty}^{+\infty} v_i^2 p_i(z v_i) dv_i dz = \frac{1}{\sigma_i^2} \int_{1}^{+\infty} \frac{1}{z^2} \int_{-\infty}^{+\infty} w^2 p_i(w) dw dz \\ &= \int_{1}^{+\infty} \frac{dz}{z^2} = 1. \end{aligned}$$

■

In practice, it is difficult to guarantee that assumption (v) is satisfied and, in general, it will not. Nevertheless, if the noise variances σ_i^2 are not too large, the result of Theorem 3 could still be used to approximate the true optimism, as shown in the simulated experiment of Section 5.3. For the sake of simplicity, let us consider the homoskedastic case $\sigma_i^2 = \sigma^2, \forall i$ and define:

$$\widehat{op}^{SVR} = \frac{2m\sigma^2}{\ell}.$$

This approximated optimism can be used to assess the in-sample error:

$$C_p^{SVR} = \overline{err} + \widehat{op}^{SVR}. \tag{19}$$

This last expression is in very close analogy with the linear case (17), provided that the model order q is replaced by the number m of marginal vectors. Formula (19) provides a further justification for the definition of approximate degrees of freedom given in Proposition 2.
Note that, in the Gaussian case, from Theorem 2 it follows that

$$op^{SVR} = \frac{2\sigma^2}{\ell} \sum_{i=1}^{\ell} E\left[\frac{\partial h_i}{\partial y_i}(y)\right] = \frac{2\sigma^2}{\ell} E\left[\#I_M\right].$$

Therefore, $\widehat{op}^{SVR}$ is an unbiased estimate of the true optimism op^{SVR}.

5. Numerical Examples

In this section the use of the C_p statistic for tuning the SVR parameters (ε, C) is illustrated by means of two numerical examples. Finally, a simulated experiment is used to assess the precision of the optimism estimate $\widehat{op}^{SVR}$ as a function of the noise variance. The SVR solution was obtained by a Finite Newton algorithm implemented in MatLab.

5.1 Simulated Data

The true function to be reconstructed is

$$f^0(x) = e^{\sin(8x)}, \qquad 0 \le x \le 1.$$

The training data (x_i, y_i), $i = 1, \ldots, \ell$, are generated as

$$y_i = y_i^0 + v_i,$$

$$y_i^0 = f^0(x_i),$$

where the errors $v_i \sim N(0, \sigma^2)$, $\sigma^2 = 0.09$, are independently distributed and

$$x_i = \frac{i-1}{\ell - 1},$$

with $\ell = 64$. In order to obtain a statistical assessment of the tuning procedure, $n = 100$ independent data sets were generated according to the above model. A cubic B-spline kernel was adopted:

$$K(x, x') = B_3(x - x').$$

The tuning of the parameters (ε, C) was carried out on a 30×30 equally spaced rectangular grid in the region

$$0.05 \le \varepsilon \le 0.5,$$

$$1 \le \log_{10} C \le 3.$$

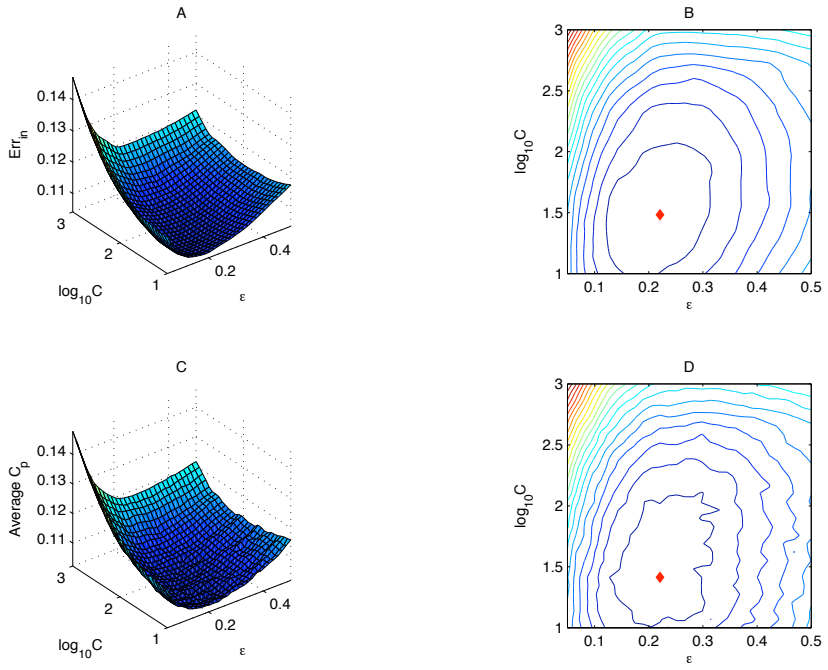


Figure 2: Estimated Err_{in} for the numerical example (Panels A and B) and average C_p over the 100 data sets (Panels C and D).

The choice of a logarithmically spaced C is in agreement with a common practice in Gaussian Processes and Tikhonov regularization methods, see for example De Nicolao et al. (1997) and De Nicolao et al. (2000).

First of all, the in-sample error $Err_{in}(\varepsilon, C)$ was computed as

$$Err_{in}(\varepsilon, C) \simeq \sigma^2 + \frac{1}{n\ell} \sum_{i=1}^{n} \|\hat{y}^{(i)}(\varepsilon, C) - y^0\|^2,$$

where $\hat{y}^{(i)}(\varepsilon, C)$ is the estimate of the vector y^0 obtained from the i-th data set. The function $Err_{in}(\varepsilon, C)$ is shown in Fig. 2. The optimal pair (ε^*, C^*) minimizing $Err_{in}(\varepsilon, C)$ is given by $\varepsilon^* = 0.22069, C^* = 30.392$, yielding $Err_{in}(\varepsilon^*, C^*) = 0.10413$.

In order to asses the average performance of the C_p statistic as an estimate of Err_{in}, the SVR estimate was calculated for each pair (ε, C) on the grid and for all the 100 data sets. In Fig. 2C the average C_p over the data sets is plotted against C and ε. The corresponding contour plot is shown in Fig. 2D. The minimal $C_p' = 0.10220$ is obtained in correspondence with $\varepsilon' = 0.22069$, $C' = 25.929$. From Fig. 2, it appears that, on the average, C_p provides a good estimate of Err_{in}. Moreover, $Err_{in}(\varepsilon', C') = 0.10436$ is reasonably close to the optimal $Err_{in}(\varepsilon^*, C^*) = 0.10413$.

In Fig. 3, C_p, Err_{in} and $E[\overline{err}]$ are plotted against C for $\varepsilon = 0.25$. The expected empirical risk $E[\overline{err}]$ was estimated by averaging over the 100 data sets. Then, the optimism op was estimated as the

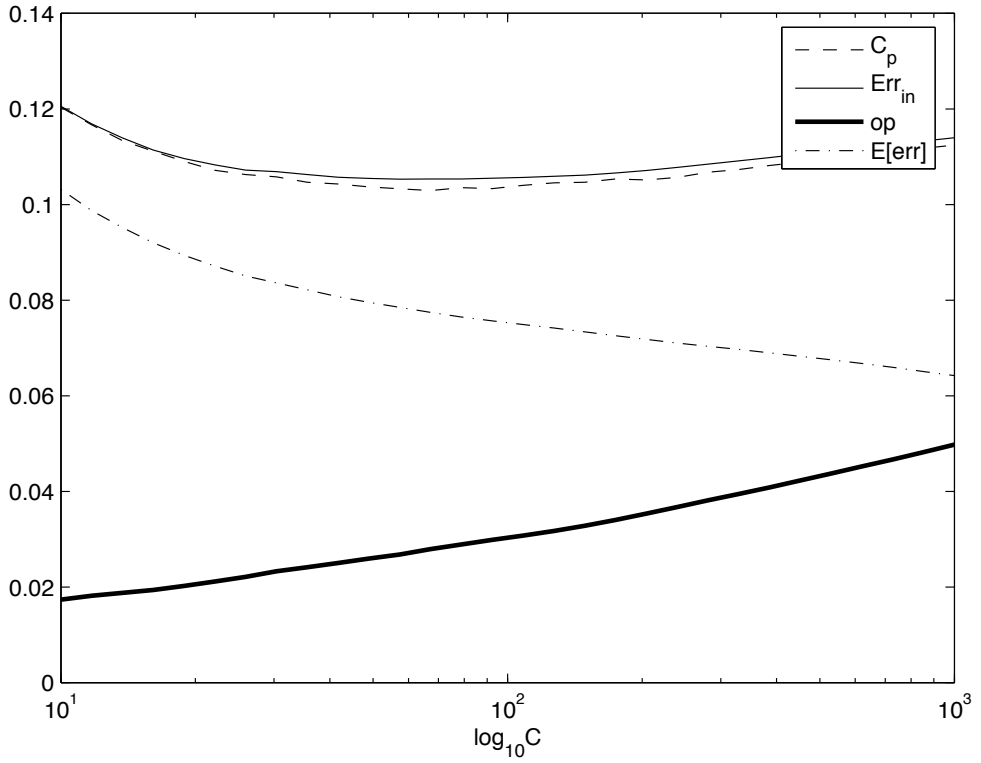


Figure 3: Decomposition of Err_{in} into the sum of $E[\overline{err}]$ and op as a function of C for $\varepsilon = 0.25$. Note that Err_{in} is well approximated by C_p.

difference between the estimates of Err_{in} and $E[\overline{err}]$. Also from this plot it is seen that C_p provides an accurate approximation of Err_{in}.

The real goal of a tuning procedure is obtaining a faithful reconstruction of the true function. A quantitative measure of the predictive performance on a single data set is given by the *RMSE* (Root Mean Square Error) defined as:

$$RMSE^{(i)}(\varepsilon, C) = \frac{1}{\sqrt{n}} \|\hat{y}^{(i)}(\varepsilon, C) - y^0\|.$$

For each of the 100 data sets, the function $f^0(x)$ was estimated using the pair $(\varepsilon^{(i)}, C^{(i)})$ minimizing the C_p statistic for the i-th data set. The average of such estimated functions is plotted in Fig. 4A where also the true function $f^0(x)$ is reported for comparison. In order to visualize the variability of the estimates, pointwise ± 2 standard deviations bands are plotted.

For the i-th data set the best possible tuning is

$$(\bar{\varepsilon}^{(i)}, \bar{C}^{(i)}) = \arg\min_{\varepsilon, C} RMSE^{(i)}(\varepsilon, C).$$

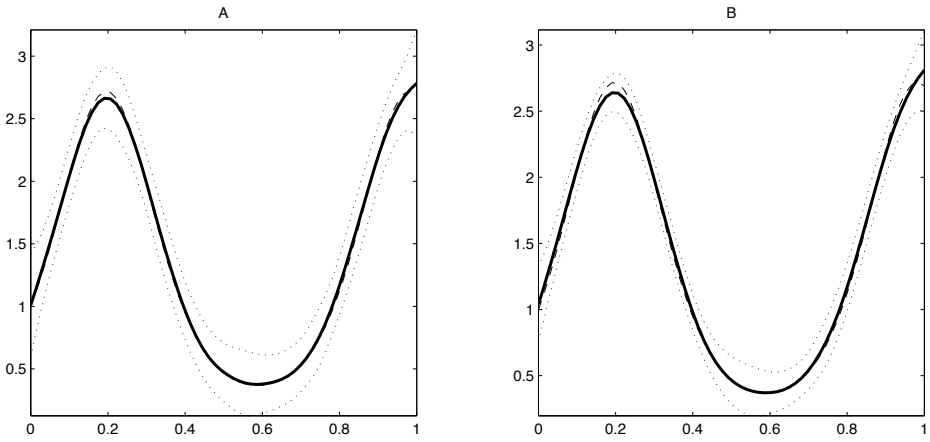


Figure 4: Average of the estimated functions over the 100 data sets: average (thick continuous) and true function (dashed). The results have been obtained by SVR with C_p tuning (Panel A) and SVR with best possible tuning (Panel B). In both cases, the $\pm$ 2 standard deviation bands are reported (dotted black).

Obviously, this cannot be used in practice because y^0 is unknown. Nevertheless, this ideal tuning is interesting because it gives a lower bound on the best achievable performance.

For each of the 100 data sets, the function $f^0(x)$ was estimated using the ideal tuning $(\varepsilon^{(i)},C^{(i)})$. The average of such estimated functions with pointwise ± 2 SD bands is plotted in Fig. 4B. The comparison with Panel A of the same figure demonstrates that the predictive performance of the C_p tuning scheme is very close to the best achievable performance.

In Fig. 5 the histogram of $RMSE^{(i)}(\varepsilon^{(i)},C^{(i)})$ (Panel A) is compared with the histogram of the best achievable errors $RMSE^{(i)}(\bar{\varepsilon}^{(i)},\bar{C}^{(i)})$ (Panel B). Finally, the application of the C_p tuning scheme is illustrated on the first data set. The value of C_p as a function of ε and C is reported in Fig. 6 A-B. On the considered grid, the C_p statistic is minimized by $\varepsilon^{(1)} \simeq 0.28, C^{(1)} \simeq 30.4$, yielding $C_p(\varepsilon^{(1)}, C^{(1)}) \simeq 0.103418$. For the sake of comparison, in Fig. 6 C-D the plot of $RMSE^{(1)}(\varepsilon,C)$ is given. The best possible tuning for data set #1 is $\bar{\varepsilon}^{(1)} \simeq 0.28, \bar{C}^{(1)} \simeq 25.93$, yielding $RMSE^{(1)}(\bar{\varepsilon}^{(1)},\bar{C}^{(1)}) \simeq 0.095357$. Using the C_p tuning scheme, a very similar value is obtained: $RMSE^{(1)}(\varepsilon^{(1)},C^{(1)}) \simeq 0.095727$.

The SVR estimate corresponding to $(\varepsilon^{(1)},C^{(1)})$ is plotted in Fig. 7 together with the true function $f^0(x)$. The SVR estimate corresponding to the best possible tuning $(\bar{\varepsilon}^{(1)},\bar{C}^{(1)})$ is plotted for

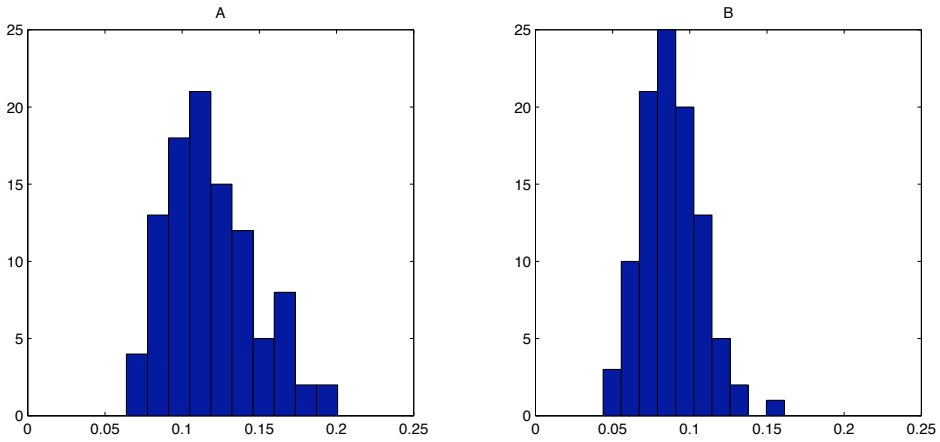


Figure 5: Distribution of the *RMSE* over the 100 data sets using C_p tuning (Panel A) and the best possible tuning (Panel B).

comparison. Taking into account the signal-to-noise ratio of the data it can be concluded that the C_p tuning scheme performs more than satisfactorily.

5.2 Boston Housing Data

To show the effectiveness on real-world data of the tuning procedure based on the C_p statistic, we applied it to the Boston Housing data set from the UCI Repository. The data set consists of 516 instances with 12 input variables (including a binary one) and an output variable representing the median housing values in suburbs of Boston.

The input variables were shifted and scaled to the unit hypercube, while the output variable was first shifted to have zero mean and then scaled to fit into the interval $[-1,1]$. More precisely, letting $m_j = \min_i x_{i,j}$ and $M_j = \max_i x_{i,j}$, the inputs $x_{i,j}$ were transformed into $(x_{i,j} - m_j)/(M_j - m_j)$, while the outputs y_i were transformed into $(y_i - \bar{y}) / \max_i |y_i - \bar{y}|$, where $\bar{y}$ denotes the sample mean.

The data set was randomly split into two parts: 450 instances to be used for training and 56 for testing. Pairs (ε, C) over a 20×20 uniform grid were considered with

$$0 \leq \varepsilon \leq 0.3, \qquad 0 \leq \log_{10} C \leq 4.$$

For each pair (ε, C), the SVR fit solving (1)-(2) was evaluated using a Gaussian RBF kernel with fixed bandwidth ($2\sigma^2_{kernel} = 3.9$). An estimate of the noise variance σ^2 was obtained from

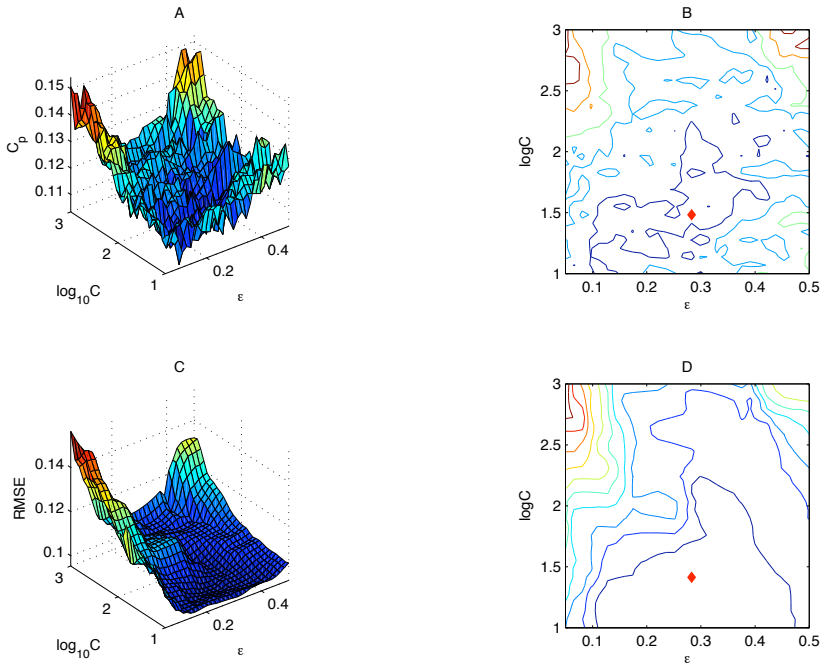


Figure 6: Data set #1: the statistic C_p as a function of ε and C (Panels A and B) and the *RMSE* between the estimate and the true function (Panels C and D).

the residuals generated from a low-bias linear regression. For details on this procedure, see for example Hastie and Tibshirani (1990), page 48, and Loader (1999), page 160. In particular, we used regularized least squares with polynomial kernel of degree 2. The noise variance of the data set was estimated as $\hat{\sigma}^2 = 0.01$ using the estimator

$$\hat{\sigma}^2 = \frac{SSR^L}{\ell - 2\nu_1 + \nu_2},$$

where SSR^L is the sum of squared residuals using the linear estimator, $\nu_1 = tr(H)$, $\nu_2 = tr(H^T H)$, and H is the "hat matrix" of the linear estimator (that is the matrix such that $\hat{\mathbf{y}} = H\mathbf{y}$). Then, the following quantities were evaluated:

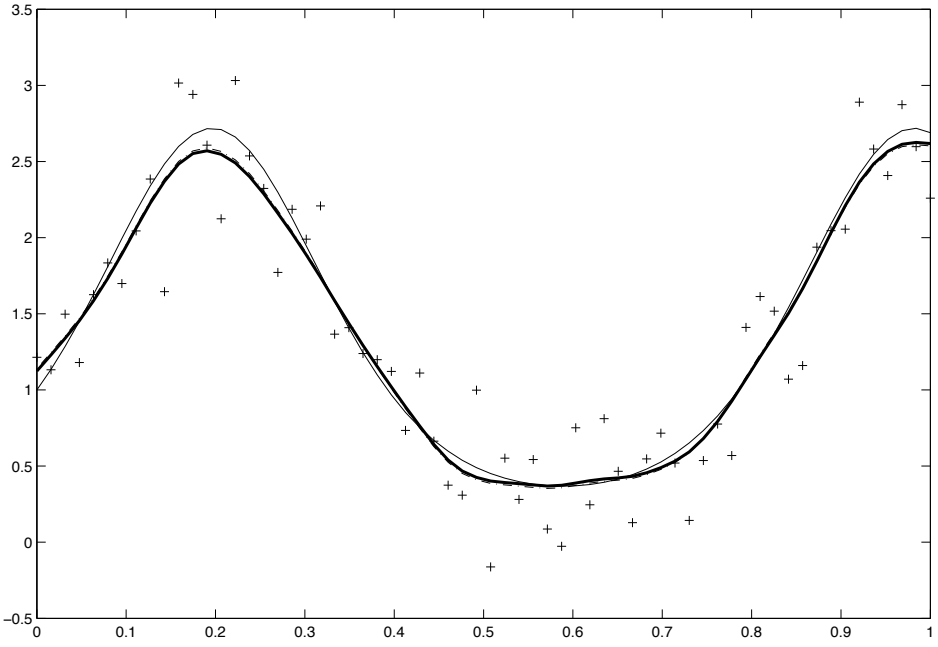


Figure 7: Data set #1: True function (continuous), data (crosses), SVR estimate with C_p tuning (thick continuous) and SVR with best possible tuning (dash-dot).

$$
\begin{aligned}
\overline{err} &= \frac{1}{\ell}\sum_{i=1}^{\ell}\left(y_i-\hat{f}(x_i)\right)^2,\\
\widehat{op}^{SVR} &= \frac{2\hat{\sigma}^2 m}{\ell},\\
C_p^{SVR} &= \overline{err}+\widehat{op}^{SVR},\\
GCV^{SVR} &= \frac{\ell^2\overline{err}}{(\ell-m)^2}.
\end{aligned}
$$

The score GCV^{SVR} was recently proposed as a tuning criterion by Gunter and Zhu (2007). These quantities are plotted in Fig. 8 and Fig. 9 together with the 5-fold cross-validation score whose computation is much heavier and the (quadratic) test error. In the contour plots of Fig. 9, the position of the minimizers are also showed. It can be seen that C_p and GCV pick the same value of ε and C. On the considered grid, the minimum value of the test error is 0.01628. The model selected by C_p and GCV achieves a test error equal to 0.01670, while the model selected by 5-fold cross-validation achieves 0.01742.

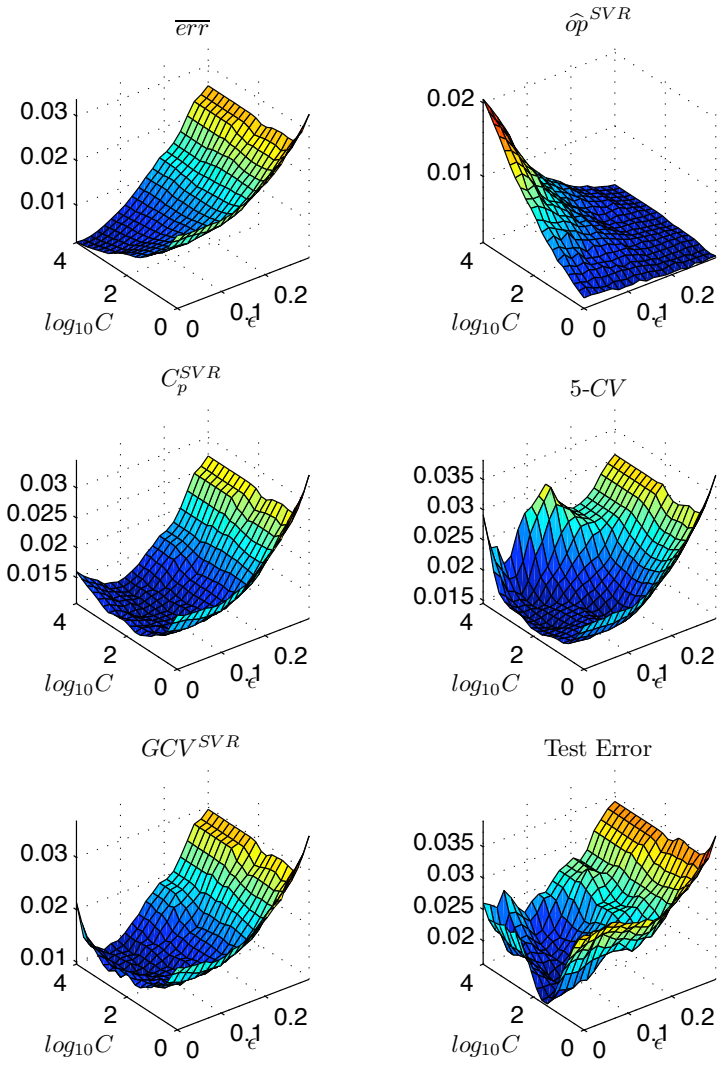


Figure 8: Boston Housing data: empirical risk ($\overline{err}$), optimism estimate ($\widehat{op}^{SVR}$), C_p statistic (C_p^{SVR}), 5-fold cross-validation score (5-CV), Generalized Cross Validation score (GCV^{SVR}), and mean square error on test data (Test Error).

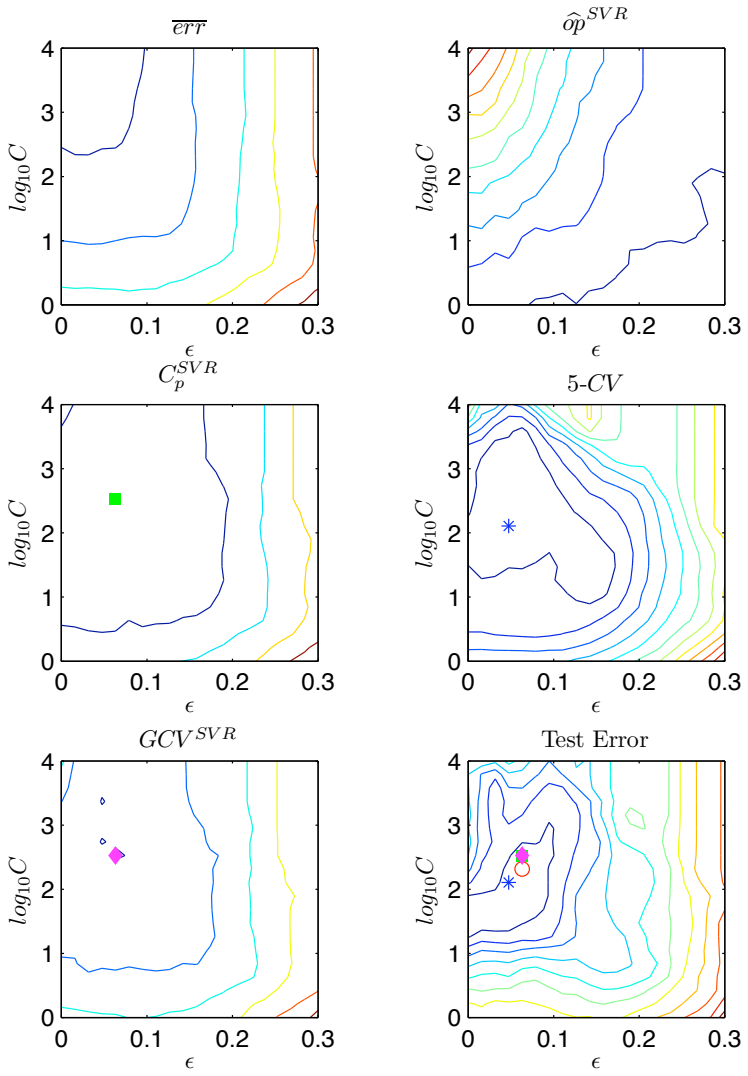


Figure 9: Boston Housing data: contour plots of empirical risk ($\overline{err}$), optimism estimate ($\widehat{op}^{SVR}$), C_p statistic (C_p^{SVR}), 5-fold cross-validation score (5-CV), Generalized Cross Validation score (GCV^{SVR}), and mean square error on test data (Test Error). In the plots of C_p^{SVR}, 5-CV and GCV^{SVR} the minimizer position is marked. In the test error plot, the marks of all minimizers are reported.

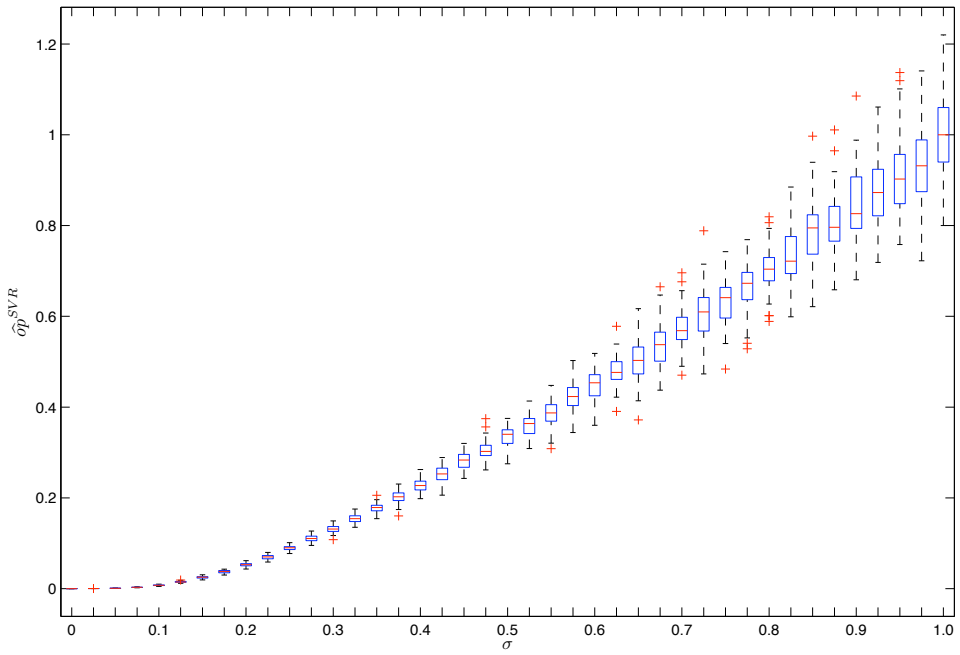


Figure 10: Simulated experiment: boxplots of the estimated optimism $\widehat{op}^{SVR}$ against different values of the noise standard deviation.

5.3 Dependence of $\widehat{op}^{SVR}$ on the Noise Variance

In order to investigate the dependence of the variability of $\widehat{op}^{SVR}$ on the noise variance, we ran a simulated experiment. Specifically, we considered 41 standard deviations in the interval $[0,1]$:

$$\sigma_j = \frac{j-1}{40}, \qquad j = 1,\ldots,41.$$

Next, for each σ_j, 100 independent data sets were generated according to the model

$$\begin{aligned} y_i &= \operatorname{sinc}(3x_i) + v_i, \qquad v_i \sim N(0,\sigma_j^2), \\ x_i &= \frac{2i-101}{99}, \qquad i = 1,\ldots,100. \end{aligned}$$

For each data set, the SVR was computed using the kernel

$$K(x,x') = e^{-\frac{|x-x'|}{4}}$$

with the values of C and ε fixed to $C = 100$, $\varepsilon = 0.1$. For each data set, we evaluated $\widehat{op}^{SVR} = 2\hat{\sigma}^2 m/\ell$. In Fig. 10 the boxplots of $\widehat{op}^{SVR}$ are reported against the considered noise standard devi-

ations (the "+" marks denote the outliers). As expected, both the mean and the variance of $\widehat{op}^{SVR}$ increase with the noise variance.
Since, under Gaussian noise, $\widehat{op}^{SVR}$ is an unbiased estimator, the true optimism op^{SVR}, which is not reported in the plot, coincides with the expected value of $\widehat{op}^{SVR}$. From Fig. 10 it appears that there is a whole range of signal-to-noise ratios such that $\widehat{op}^{SVR}$ estimates op^{SVR} with good precision.

6. Concluding Remarks

In this paper, a novel formulation of the quantitative representer theorem is derived for convex loss functions. More precisely, using the newly introduced notion of pseudoresidual the inclusions appearing in the previous formulations are replaced by equations. This result is exploited in order to study the sensitivity of both the SVR coefficients and predictions with respect to the data. In view of the sensitivity analysis, the degrees of freedom of SVR are defined as the number of marginal support vectors. Such a definition is further justified by the role that the degrees of freedom play in the assessment of the optimism, that is the difference between the in-sample prediction error and the expected empirical risk. A C_p statistic for SVR is defined and proposed as a criterion for tuning both the parameters ε and C. The performance observed on both a simulated benchmark and a real world problem appears more than satisfactory. Among the future developments one may mention the extension of the results of the present paper to kernel based classifiers.

Acknowledgments

This research has been partially supported by the Italian Ministry of University and Research through the FIRB Project "Learning theory and application" and the PRIN Project "New methods and algorithms for identification and adaptive control of technological systems".

Appendix A.

In this appendix, a Fourier series demonstration of the representer theorem is provided. The rationale is inspired by Evgeniou et al. (2000) who prove the representer theorem for differentiable loss functions. Let us assume that the function $K(\mathbf{x},\mathbf{t})$ is such that the bilinear formula holds:

$$K(\mathbf{x},\mathbf{t}) = \sum_{n=1}^{+\infty} \lambda_n \phi_n(\mathbf{x})\phi_n(\mathbf{t}),$$

where $\forall n, \lambda_n > 0$, and ϕ_n denote the n-th eigenvalue and eigenfunction of the operator

$$Tf = (f, K(\mathbf{x},\mathbf{t}))_{\mathcal{L}^2}.$$

Then, K is positive definite and a generic function $f \in \mathcal{L}^2$ admits the Fourier expansion

$$f(\mathbf{x}) = \sum_{n=1}^{+\infty} c_n \phi_n(\mathbf{x}). \tag{20}$$

Now, we can build the RKHS $\mathcal{H}$ taking all the functions f such that $\sum_{n=1}^{+\infty} \frac{c_n^2}{\lambda_n}$ is finite and defining the inner product between the two functions $u, v \in \mathcal{H}$, $u = \sum_{n=1}^{+\infty} a_n \phi_n$, $v = \sum_{n=1}^{+\infty} b_n \phi_n$ as

$$(u,v)_{\mathcal{H}} = \sum_{n=1}^{+\infty} \frac{a_n b_n}{\lambda_n},$$

so that the norm is

$$\|f\|_{\mathcal{H}}^2 = \sum_{n=1}^{+\infty} \frac{c_n^2}{\lambda_n}.$$

It is easy to check that the reproducing property holds

$$f(\mathbf{x}) = (f(\mathbf{t}), K(\mathbf{x},\mathbf{t}))_{\mathcal{H}},$$

so that $K(\mathbf{x},\mathbf{y})$ is indeed the reproducing kernel of $\mathcal{H}$. In particular, the reproducing property implies that the series (20) is, in fact, pointwise convergent.
In view of this, solving (1) is equivalent to minimizing the following functional with respect to the coefficient sequence:

$$F[\{c_n\}] = C \sum_{i=1}^{\ell} V\left(y_i, \sum_{n=1}^{+\infty} c_n \phi_n(\mathbf{x}_i)\right) + \frac{1}{2} \sum_{n=1}^{+\infty} \frac{c_n^2}{\lambda_n}.$$

Noting that the sequence $\{c_n\}$ belongs to $\ell^2(\mathbb{R})$, we can see that the functional F to be minimized maps a subset of ℓ^2 into $\mathbb{R}$. From the necessary condition for optimality, we have $0 \in \partial F$, where ∂F denotes the subdifferential. Exploiting the linearity of the subdifferential with respect to sums of convex functions and the fact that the second term is Gâteaux-differentiable, we obtain:

$$\partial F = \left\{ C \sum_{i=1}^{\ell} \partial V\left(y_i, \sum_{n=1}^{+\infty} c_n \phi_n(\mathbf{x}_i)\right) + \frac{c_n}{\lambda_n} \right\}.$$

Now, let us recall the following result (see Prop. 5.7 of Ekeland and Temam (1974), where it is given for the more general case of topological vector spaces):

Proposition 4 *Let $\mathcal{H}$, $\mathcal{H}'$ two Banach spaces, V a convex function from $\mathcal{H}$ into $\mathbb{R} \cup \{+\infty\}$, and J a continuous linear operator from $\mathcal{H}'$ into $\mathcal{H}$. Assume that there is $v_0' \in \mathcal{H}'$ such that V is continuous and finite at Jv_0'. Then, for all $v' \in \mathcal{H}'$*

$$(\partial V \circ J)(v') = J^*(\partial V)(Jv'),$$

where $J^* : \mathcal{H} \to \mathcal{H}'$ is the adjoint defined by

$$\left\langle v', J^* v \right\rangle_{\mathcal{H}'} = \left\langle Jv', v \right\rangle_{\mathcal{H}}$$

for all $v \in \mathcal{H}$ and $v' \in \mathcal{H}'$, where $< \cdot, \cdot >_{\mathcal{H}}$ stands for the duality pairing in the Banach space $\mathcal{H}$. ■

Introducing the linear operators $J_i : \ell^2 \to \mathbb{R}$

$$J_i(\{c_n\}) = \sum_{n=1}^{+\infty} c_n \phi_n(\mathbf{x}_i),$$

we can write

$$\partial V\left(y_i, \sum_{n=1}^{+\infty} c_n \phi_n(\mathbf{x}_i)\right) = \partial V\left(y_i, J_i(\{c_n\})\right).$$

Notice that the adjoint $J_i^* : \mathbb{R} \to \ell^2(\mathbb{R})$ is given by $J_i^*(t) = \{t\phi_n(\mathbf{x}_i)\}$. In fact,

$$\langle \{c_n\}, J_i^*(t)\rangle_{\ell^2} = \sum_{n=1}^{+\infty} c_n (J_i^*(t))_n = t \sum_{n=1}^{+\infty} c_n \phi_n(\mathbf{x}_i) = \langle J_i(\{c_n\}), t\rangle_{\mathbb{R}}.$$

In view of Proposition 4,

$$\partial F = \left\{ C \sum_{i=1}^{\ell} \phi_n(\mathbf{x}_i) \partial_2 V\left(y_i, \sum_{n=1}^{+\infty} c_n \phi_n(\mathbf{x}_i)\right) + \frac{c_n}{\lambda_n} \right\}$$

so that the condition $0 \in \partial F$ implies that the optimal sequence $\{\hat{c}_n\}$ must satisfy

$$\hat{c}_n \in -\sum_{i=1}^{\ell} C \partial_2 V\left(y_i, \hat{f}(\mathbf{x}_i)\right) \lambda_n \phi_n(\mathbf{x}_i).$$

It is then possible to write

$$\hat{c}_n = \sum_{i=1}^{\ell} a_i \lambda_n \phi_n(\mathbf{x}_i),$$

where

$$a_i \in -C \partial_2 V(y_i, \hat{f}(\mathbf{x}_i)).$$

Finally, exploiting the bilinear formula for the reproducing kernel, we obtain

$$\hat{f}(\mathbf{x}) = \sum_{n=1}^{+\infty} \hat{c}_n \phi_n(\mathbf{x}) = \sum_{i=1}^{\ell} a_i \sum_{n=1}^{+\infty} \lambda_n \phi_n(\mathbf{x}_i) \phi_n(\mathbf{x}) = \sum_{i=1}^{\ell} a_i K(\mathbf{x}_i, \mathbf{x}).$$

References

J. M. Borwein and A. J. Lewis. *Convex Analisys and Nonlinear Optimization*. Springer, 2000.

M. W. Chang and C. J. Lin. Leave-one-out bounds for support vector regression model selection. *Neural Computation*, 17:1188–1222, 2005.

V. Cherkassky and Y. Ma. Comparison of model selection for regression. *Neural Computation*, 15: 1691–1714, 2003.

D. Cox and F. O' Sullivan. Asymptotic analysis of penalized likelihood and related estimators. *Ann.Stat.*, 18:1676–1695, 1990.

F. Cucker and S. Smale. On the mathematical foundations of learning. *Bulletin of AMS*, 39:1–49, 2001.

G. De Nicolao, G. Sparacino, and C. Cobelli. Nonparametric input estimation in physiological systems: Problems, methods, and case studies. *Automatica*, 33:851–870, 1997.

G. De Nicolao, G. Ferrari Trecate, and G. Sparacino. Fast spline smoothing via spectral factorization concepts. *Automatica*, 36:1733–1739, 2000.

E. De Vito, L. Rosasco, A. Caponnetto, M. Piana, and A. Verri. Some properties of regularized kernel methods. *Journal of Machine Learning Research*, 5:1363–1390, 2004.

B. Efron. How biased is the apparent error rate of a prediction rule? *Journal of the American Statistical Association*, 81(394):461–470, 1986.

I. Ekeland and R. Temam. *Analyse Convexe et Problémes Variationnels*. Gauthier-Villards, Paris, 1974.

T. Evgeniou, M. Pontil, and T. Poggio. Regularization networks and support vector machines. *Advances in Computational Mathematics*, 13:1–150, 2000.

J. B. Gao, S. R. Gunn, C. J. Harris, and M. Brown. A probabilistic framework for SVM regression and error bar estimation. *Machine Learning*, 46:71–89, 2002.

L. Gunter and J. Zhu. Efficient computation and model selection for the support vector regression. *Neural Computation*, 19:1633–1655, 2007.

T. J. Hastie and R. J. Tibshirani. Generalized additive models. In *Monographs on Statistics and Applied Probability*, volume 43. Chapman and Hall, London, UK, 1990.

T. J. Hastie, R. J. Tibshirani, and J. Friedman. *The Elements of Statistical Learning. Data Mining, Inference and Prediction*. Springer, Canada, 2001.

T. J. Hastie, R. J. Tibshirani, and J. Friedman. Note on "Comparison of Model Selection for Regression" by Vladimir Cherkassky and Yunqian Ma. *Neural Computation*, 15:1477–1480, 2003.

T. J. Hastie, S. Rosset, R. J. Tibshirani, and J. Zhu. The entire regularization path for the support vector machine. *Journal of Machine Learning Research*, 5:1391–1415, 2004.

G. Kimeldorf and G. Wahba. A correspondence between Bayesian estimation of stochastic processes and smoothing by splines. *Ann. Math. Stat.*, 41:495–502, 1979.

J. T. Kwok and I. W. Tsang. Linear dependency between ε and the input noise in ε-support vector regression. *IEEE Transactions On Neural Networks*, XX, 2003.

C. Loader. *Local Regression and Likehood*. Statistics and Computing. Springer, 1999.

T. Poggio and F. Girosi. A theory of networks for approximation and learning. *Foundation of Neural Networks*, page 91–106, 1992.

M. Pontil and A. Verri. Properties of support vector machines. *Neural Computation*, 10:955–974, 1998.

B. Schölkopf, A. J. Smola, R. C. Williamson, and P. L. Bartlett. New support vector algorithms. *Neural Computation*, 12:1207–1245, 2000.

B. Schölkopf, R. Herbrich, and A. J. Smola. A generalized representer theorem. *Neural Networks and Computational Learning Theory*, 81:416–426, 2001.

A. J. Smola, N. Murata, B. Schölkopf, and K. Muller. Asymptotically optimal choice of ε-loss for support vector machines. In L. Niklasson, M. Boden, and T. Ziemke, editors, *Proceedings of the 8th International Conference on Artificial Neural Networks, Perspectives in Neural Computing*, pages 105–110, Berlin, 1998. Springer.

C. Stein. Estimation of the mean of a multivariate normal distribution. *Annals of Statistics*, 9: 1135–1151, 1981.

I. Steinwart. Sparseness of support vector machines. *Journal of Machine Learning Research*, 4: 1071–1105, 2003.

A. N. Tikhonov and V. Y. Arsenin. *Solutions of Ill Posed Problems*. W. H. Winston, Washington, D. C., 1977.

V. Vapnik. *The Nature of Statistical Learning Theory*. Springer, New York, NY, USA, 1995.

G. Wahba. *Spline Models for Observational Data*. SIAM, Philadelphia, USA, 1990.

G. Wahba. Support vector machines, reproducing kernel Hilbert spaces and randomized GACV. Technical Report 984, Department of Statistics, University of Wisconsin, 1998.

Journal of Machine Learning Research 8 (2007) 2497-2532 Submitted 5/06; Revised 10/06; Published 10/07

Nonlinear Estimators and Tail Bounds for Dimension Reduction in l_1 Using Cauchy Random Projections

Ping Li PINGLI@CORNELL.EDU
Department of Statistical Science
Faculty of Computing and Information Science
Cornell University
Ithaca, NY 14853, USA

Trevor J. Hastie HASTIE@STANFORD.EDU
Department of Statistics
Stanford University
Stanford, CA 94305, USA

Kenneth W. Church CHURCH@MICROSOFT.COM
Microsoft Research
Microsoft Corporation
Redmond, WA 98052, USA

Editor: Sam Roweis

Abstract

For[1] dimension reduction in the l_1 norm, the method of *Cauchy random projections* multiplies the original data matrix $\mathbf{A} \in \mathbb{R}^{n \times D}$ with a random matrix $\mathbf{R} \in \mathbb{R}^{D \times k}$ ($k \ll D$) whose entries are i.i.d. samples of the standard Cauchy $C(0,1)$. Because of the impossibility result, one can not hope to recover the pairwise l_1 distances in $\mathbf{A}$ from $\mathbf{B} = \mathbf{A} \times \mathbf{R} \in \mathbb{R}^{n \times k}$, using linear estimators without incurring large errors. However, nonlinear estimators are still useful for certain applications in data stream computations, information retrieval, learning, and data mining.

We study three types of nonlinear estimators: the *sample median* estimators, the *geometric mean* estimators, and the *maximum likelihood* estimators (MLE). We derive tail bounds for the *geometric mean* estimators and establish that $k = O\left(\frac{\log n}{\varepsilon^2}\right)$ suffices with the constants explicitly given. Asymptotically (as $k \to \infty$), both the *sample median* and the *geometric mean* estimators are about 80% efficient compared to the MLE. We analyze the moments of the MLE and propose approximating its distribution of by an inverse Gaussian.

Keywords: dimension reduction, l_1 norm, Johnson-Lindenstrauss (JL) lemma, Cauchy random projections

1. Introduction

There has been considerable interest in the l_1 norm in statistics and machine learning, as it is now well-known that the l_1 distance is far more robust than the l_2 distance against "outliers" (Huber, 1981). It is sometimes a good practice to replace the l_2 norm minimization with the l_1 norm minimization, for example, the Least Absolute Deviation (LAD) Boost (Friedman, 2001). Chapelle et al. (1999) demonstrated that using the l_1 (Laplacian) radial basis kernel produced better classification

1. A preliminary version appeared in COLT 2007 (Li et al., 2007b).

results than the usual l_2 (Gaussian) radial basis kernel, in their histogram-based image classification project using support vector machines (SVM). Recently, it also becomes popular to use the l_1 norm for variable (feature) selection; success stories include LASSO (Tibshirani, 1996), LARS (Efron et al., 2004) and 1-norm SVM (Zhu et al., 2003).

This paper focuses on dimension reduction in the l_1 norm, in particular, on the method based on *Cauchy random projections*, which is a special case of *linear (stable) random projections* (Johnson and Schechtman, 1982; Indyk, 2000, 2006; Li, 2008).

The idea of *linear random projections* is to multiply the original data matrix $\mathbf{A} \in \mathbb{R}^{n \times D}$ with a random projection matrix $\mathbf{R} \in \mathbb{R}^{D \times k}$, resulting in a projected matrix $\mathbf{B} = \mathbf{AR} \in \mathbb{R}^{n \times k}$. We would like k to be as small as possible. If $k \ll D$, then it should be much more efficient to compute certain summary statistics (e.g., pairwise distances) from $\mathbf{B}$ as opposed to $\mathbf{A}$. Moreover, $\mathbf{B}$ may be small enough to reside in physical memory while $\mathbf{A}$ is often too large to fit in the main memory.

The choice of the random projection matrix $\mathbf{R}$ depends on which norm we would like to work with. For dimension reduction in l_p ($0 < p \leq 2$), it is common practice to construct $\mathbf{R}$ from i.i.d. samples of p-stable distributions (Johnson and Schechtman, 1982; Indyk, 2000, 2006; Li, 2008). In the stable distribution family (Zolotarev, 1986), normal is 2-stable and Cauchy is 1-stable. Thus, we will call random projections for l_2 and l_1, *normal random projections* and *Cauchy random projections*, respectively.

In *normal random projections* (Vempala, 2004), we can estimate the original pairwise l_2 distances in $\mathbf{A}$ directly using the corresponding l_2 distances in $\mathbf{B}$ (up to a normalizing constant). Furthermore, the Johnson-Lindenstrauss (JL) Lemma (Johnson and Lindenstrauss, 1984) provides the performance guarantee. We will review *normal random projections* in more detail in Section 2.

For *Cauchy random projections*, however, one shall not use the l_1 distance in $\mathbf{B}$ to approximate the original l_1 distance in $\mathbf{A}$, as the Cauchy distribution does not even have a finite first moment. The impossibility results (Brinkman and Charikar, 2003; Lee and Naor, 2004; Brinkman and Charikar, 2005) have proved that one can not hope to recover the l_1 distance using linear projections and linear estimators (e.g., sample mean), without incurring large errors. Fortunately, the impossibility results do not rule out nonlinear estimators, which may be still useful in certain applications in data stream computations, information retrieval, learning, and data mining.

In this paper, we study three types of nonlinear estimators: the *sample median* estimators, the *geometric mean* estimators, and the *maximum likelihood* estimators (MLE). The *sample median* and the *geometric mean* estimators are asymptotically (as $k \to \infty$) equivalent (i.e., both are about 80% efficient as the MLE), but the latter is more accurate at small sample size k. Furthermore, we derive explicit tail bounds for the *geometric mean* estimators and establish an analog of the JL Lemma for dimension reduction in l_1.

This analog of the JL Lemma for l_1 is weaker than the classical JL Lemma for l_2, as the geometric mean is not convex and hence is not a metric. Many efficient algorithms, such as some sub-linear time (using super-linear memory) nearest neighbor algorithms (Shakhnarovich et al., 2005), rely on metric properties (e.g., the triangle inequality). Nevertheless, nonlinear estimators may be still useful in important scenarios.

- *Estimating l_1 distances online*
  The original data matrix $\mathbf{A} \in \mathbb{R}^{n \times D}$ requires $O(nD)$ storage space; and hence it is often too large for physical memory. The storage cost of materializing all pairwise distances is $O(n^2)$, which may be also too large for the memory. For example, in information retrieval, n could be the total number of word types or documents at Web scale. To avoid page faults, it may

be more efficient to estimate the distances *on the fly* from the projected data matrix $\mathbf{B}$ in the memory.

- *Computing all pairwise l_1 distances*
  In distance-based clustering, classification, and kernels (e.g., for SVM), we need to compute all pairwise distances in $\mathbf{A}$, at the cost of time $O(n^2D)$, which can be prohibitive, especially when $\mathbf{A}$ does not fit in the memory. Using *Cauchy random projections*, the cost is reduced to $O(nDk+n^2k)$.

- *Linear scan nearest neighbor searching*
  Nearest neighbor searching is notorious for being inefficient, especially when the data matrix $\mathbf{A}$ is too large for the memory. Searching for the nearest neighbors from the projected data matrix $\mathbf{B}$ (which is in the memory) becomes much more efficient, even by linear scans. The cost of searching for the nearest neighbor for one data point is reduced from $O(nD)$ to $O(nk)$.

- *Data stream computations*.
  Massive data streams come from Internet routers, phone switches, atmospheric observations, sensor networks, highway traffic, finance data, and more (Henzinger et al., 1999; Feigenbaum et al., 1999; Indyk, 2000; Babcock et al., 2002; Cormode et al., 2002). Unlike in the traditional databases, it is not common to store massive data streams; and hence the processing is often done *on the fly*. In data stream computations, Cauchy random projections can be used for (A): approximating the l_1 frequency moments for individual streams; (B): approximating the l_1 differences between a pair of streams.

We briefly comment on *random coordinate sampling*, another strategy for dimension reduction. One can randomly sample k columns from $\mathbf{A} \in \mathbb{R}^{n\times D}$ and estimate the summary statistics (including l_1 and l_2 distances). Despite its simplicity, this strategy has two major drawbacks. First, in heavy-tailed data, one may have to choose k very large in order to achieve a sufficient accuracy. Second, large data sets are often highly sparse, for example, text data (Dhillon and Modha, 2001) and market-basket data (Aggarwal and Wolf, 1999; Strehl and Ghosh, 2000). For sparse data, Li and Church (2005, 2007); Li et al. (2007a) provided an alternative coordinate sampling strategy, called *Conditional Random Sampling (CRS)*. For non-sparse data, however, methods based on *linear (stable) random projections* are superior.

The rest of the paper is organized as follows. Section 2 reviews *linear random projections*. Section 3 summarizes the main results for three types of nonlinear estimators. Section 4 presents the *sample median* estimators. Section 5 concerns the *geometric mean* estimators. Section 6 is devoted to the *maximum likelihood* estimators. Section 7 concludes the paper.

2. Introduction to Linear (Stable) Random Projections

We give a review on *linear random projections*, including *normal* and *Cauchy random projections*.

Denote the original data matrix by $\mathbf{A} \in \mathbb{R}^{n\times D}$, that is, n data points in D dimensions. Let $\{u_i^{\mathrm{T}}\}_{i=1}^n \in \mathbb{R}^D$ be the ith row of $\mathbf{A}$. Let $\mathbf{R} \in \mathbb{R}^{D\times k}$ be a projection matrix and denote the entries of $\mathbf{R}$ by $\{r_{ij}\}_{i=1}^{D}{}_{j=1}^{k}$. The projected data matrix $\mathbf{B} = \mathbf{AR} \in \mathbb{R}^{n\times k}$. Let $\{v_i^{\mathrm{T}}\}_{i=1}^n \in \mathbb{R}^k$ be the ith row of $\mathbf{B}$, that is, $v_i = \mathbf{R}^{\mathrm{T}}u_i$.

For simplicity, we focus on the leading two rows, u_1 and u_2, in $\mathbf{A}$, and the leading two rows, v_1 and v_2, in $\mathbf{B}$. Define $\{x_j\}_{j=1}^k$ to be

$$x_j = v_{1,j} - v_{2,j} = \sum_{i=1}^{D} r_{ij}\left(u_{1,i} - u_{2,i}\right), \qquad j = 1,2,...,k.$$

If we sample r_{ij} i.i.d. from a *p-stable distribution* (Zolotarev, 1986), then x_j's are also i.i.d. samples of a p-stable distribution with a different scale parameter. In the family of stable distributions, normal $(p = 2)$ and Cauchy $(p = 1)$ are two important special cases.

2.1 Normal Random Projections

When r_{ij} is sampled from the standard normal, that is, $r_{ij} \sim N(0,1)$, i.i.d., then

$$x_j = v_{1,j} - v_{2,j} = \sum_{i=1}^{D} r_{ij}\left(u_{1,i} - u_{2,i}\right) \sim N\left(0, \sum_{i=1}^{D} |u_{1,i} - u_{2,i}|^2\right), \quad j = 1,2,...,k,$$

because a weighted sum of normals is also normal.

Denote the squared l_2 distance between u_1 and u_2 by

$$d_{l_2} = \|u_1 - u_2\|_2^2 = \sum_{i=1}^{D} |u_{1,i} - u_{2,i}|^2.$$

We can estimate d_{l_2} from the sample squared l_2 distance (i.e., sample mean):

$$\hat{d}_{l_2} = \frac{1}{k}\sum_{j=1}^{k} x_j^2.$$

Note that $k\hat{d}_{l_2}/d_{l_2}$ follows a Chi-square distribution with k degrees of freedom, χ_k^2. Therefore, it is easy to prove the following Lemma about the tail bounds:

Lemma 1

$$\mathbf{Pr}(\hat{d}_{l_2} - d_{l_2} \geq \varepsilon d_{l_2}) \leq \exp\left(-\frac{k}{2}\left(\varepsilon - \log(1+\varepsilon)\right)\right) = \exp\left(-k\frac{\varepsilon^2}{G_R}\right), \qquad \varepsilon > 0,$$

$$\mathbf{Pr}(\hat{d}_{l_2} - d_{l_2} \leq -\varepsilon d_{l_2}) \leq \exp\left(-\frac{k}{2}\left(-\varepsilon - \log(1-\varepsilon)\right)\right) = \exp\left(-k\frac{\varepsilon^2}{G_L}\right), \qquad 0 < \varepsilon < 1,$$

where the constants

$$G_R = \frac{2\varepsilon^2}{\varepsilon - \log(1+\varepsilon)} \leq \frac{4}{1-\frac{2}{3}\varepsilon},$$

$$G_L = \frac{2\varepsilon^2}{-\varepsilon - \log(1-\varepsilon)} \leq \frac{4}{1+\frac{2}{3}\varepsilon} \leq \frac{4}{1-\frac{2}{3}\varepsilon}.$$

Proof *Using the standard Chernoff inequality (Chernoff, 1952),*

$$\begin{aligned}\mathbf{Pr}(\hat{d}_{l_2} - d_{l_2} \geq \varepsilon d_{l_2}) =& \mathbf{Pr}\left(k\hat{d}_{l_2}/d_{l_2} \geq k(1+\varepsilon)\right) \\ \leq& \frac{E\left(\exp(k\hat{d}_{l_2}/d_{l_2}t)\right)}{\exp((1+\varepsilon)kt)} \qquad (t>0) \\ =& \exp\left(-\frac{k}{2}\left(\log(1-2t)+2(1+\varepsilon)t\right)\right),\end{aligned}$$

which is minimized at $t = \frac{\varepsilon}{2(1+\varepsilon)}$. *Thus, for any* $\varepsilon > 0$

$$\mathbf{Pr}(\hat{d}_{l_2} - d_{l_2} > \varepsilon d_{l_2}) \leq \exp\left(-\frac{k}{2}\left(\varepsilon - \log(1+\varepsilon)\right)\right).$$

We can similarly prove the other tail bound for $\mathbf{Pr}(\hat{d}_{l_2} - d_{l_2} \leq -\varepsilon d_{l_2})$. ■

For convenience, sometimes we would like to write the tail bounds in a symmetric form

$$\mathbf{Pr}\left(\left|\hat{d}_{l_2} - d_{l_2}\right| \geq \varepsilon d_{l_2}\right) \leq 2\exp\left(-k\frac{\varepsilon^2}{G}\right), \quad 0 < \varepsilon < 1,$$

and we know that it suffices to let $G = \max\{G_R, G_L\} \leq \frac{4}{1-\frac{2}{3}\varepsilon}$.

Since there are in total $\frac{n(n-1)}{2} < \frac{n^2}{2}$ pairs among n data points, we would like to bound the tail probabilities simultaneously for all pairs. By the Bonferroni union bound, it suffices if

$$\frac{n^2}{2}\mathbf{Pr}\left(\left|\hat{d}_{l_2} - d_{l_2}\right| \geq \varepsilon d_{l_2}\right) \leq \delta,$$

that is, it suffices if

$$\frac{n^2}{2}2\exp\left(-k\frac{\varepsilon^2}{G}\right) \leq \delta \Longrightarrow k \geq G\frac{2\log n - \log\delta}{\varepsilon^2}.$$

Therefore, we obtain one version of the Johnson-Lindenstrauss (JL) Lemma:

Lemma 2 *If* $k \geq G\frac{2\log n - \log\delta}{\varepsilon^2}$, *where* $G = \frac{4}{1-\frac{2}{3}\varepsilon}$, *then with probability at least* $1-\delta$, *the squared* l_2 *distance between any pair of data points (among n data points) can be approximated within a* $1 \pm \varepsilon$ *factor* $(0 < \varepsilon < 1)$, *using the squared* l_2 *distance of the projected data after normal random projections*.

Many versions of the JL Lemma have been proved (Johnson and Lindenstrauss, 1984; Frankl and Maehara, 1987; Indyk and Motwani, 1998; Arriaga and Vempala, 1999; Dasgupta and Gupta, 2003; Indyk, 2000, 2001; Achlioptas, 2003; Arriaga and Vempala, 2006; Ailon and Chazelle, 2006).

Note that we do not have to use $r_{ij} \sim N(0,1)$ for dimension reduction in l_2. For example, we can sample r_{ij} from the following *sparse projection distribution*:

$$r_{ij} = \sqrt{s} \times \begin{cases} 1 & \text{with prob. } \frac{1}{2s} \\ 0 & \text{with prob. } 1 - \frac{1}{s} \\ -1 & \text{with prob. } \frac{1}{2s} \end{cases}. \tag{1}$$

When $1 \leq s \leq 3$, Achlioptas (2001, 2003) proved the JL Lemma for the above sparse projection. Recently, Li et al. (2006b) proposed *very sparse random projections* using $s \gg 3$ in (1), based on two practical considerations:

- D should be very large, otherwise there would be no need for dimension reduction.
- The original l_2 distance should make engineering sense, in that the second (or higher) moments should be bounded (otherwise various *term-weighting* schemes will be applied).

Based on these two practical assumptions, the projected data are asymptotically normal at a fast rate of convergence when $s = \sqrt{D}$ and the data have bounded third moments. Of course, *very sparse random projections* do not have worst case performance guarantees.

2.2 Cauchy Random Projections

In *Cauchy random projections*, we sample r_{ij} i.i.d. from the standard Cauchy distribution, that is, $r_{ij} \sim C(0,1)$. By the 1-stability of Cauchy (Zolotarev, 1986), we know that

$$x_j = v_{1,j} - v_{2,j} \sim C\left(0, \sum_{i=1}^{D} |u_{1,i} - u_{2,i}|\right).$$

That is, the projected differences $x_j = v_{1,j} - v_{2,j}$ are also Cauchy random variables with the scale parameter being the l_1 distance, $d = |u_1 - u_2| = \sum_{i=1}^{D} |u_{1,i} - u_{2,i}|$, in the original space.

Recall that a Cauchy random variable $z \sim C(0,\gamma)$ has the density

$$f(z) = \frac{\gamma}{\pi} \frac{1}{z^2 + \gamma^2}, \qquad \gamma > 0, \quad -\infty < z < \infty.$$

The easiest way to see the 1-stability is via the characteristic function,

$$\mathrm{E}\left(\exp(\sqrt{-1} z_1 t)\right) = \exp\left(-\gamma |t|\right),$$
$$\mathrm{E}\left(\exp\left(\sqrt{-1} t \sum_{i=1}^{D} c_i z_i\right)\right) = \exp\left(-\gamma \sum_{i=1}^{D} |c_i| t\right),$$

for $z_1, z_2, ..., z_D$, i.i.d. $C(0,\gamma)$, and any constants $c_1, c_2, ..., c_D$.

Therefore, in *Cauchy random projections*, the problem boils down to estimating the Cauchy scale parameter of $C(0,d)$ from k i.i.d. samples $x_j \sim C(0,d)$. Unlike in *normal random projections*, we can no longer estimate d from the sample mean (i.e., $\frac{1}{k}\sum_{j=1}^{k} |x_j|$) because $\mathrm{E}(x_j) = \infty$.

3. Main Results

Although the impossibility results (Lee and Naor, 2004; Brinkman and Charikar, 2005) have ruled out accurate estimators that are also metrics, there is enough information to recover d from k samples $\{x_j\}_{j=1}^{k}$, with high accuracy.

We analyze three types of nonlinear estimators: the *sample median* estimators, the *geometric mean* estimators, and the *maximum likelihood* estimators.

3.1 The Sample Median Estimators

The *sample median* estimator,

$$\hat{d}_{me} = \text{median}(|x_j|, j = 1, 2, ..., k),$$

is simple and computationally convenient. We recommend the bias-corrected version:

$$\hat{d}_{me,c} = \frac{\hat{d}_{me}}{b_{me}},$$

where

$$b_{me} = \int_0^1 \frac{(2m+1)!}{(m!)^2} \tan\left(\frac{\pi}{2}t\right)\left(t-t^2\right)^m dt, \quad k = 2m+1.$$

Here, for convenience, we only consider $k = 2m+1$, $m = 1, 2, 3, ...$
Some properties of $\hat{d}_{me,c}$:

- $\mathrm{E}\left(\hat{d}_{me,c}\right) = d$, that is, $\hat{d}_{me,c}$ is unbiased.
- When $k \geq 5$, the variance of $\hat{d}_{me,c}$ is

$$\begin{aligned}\mathrm{Var}\left(\hat{d}_{me,c}\right) =& d^2\left(\frac{(m!)^2}{(2m+1)!}\frac{\int_0^1 \tan^2\left(\frac{\pi}{2}t\right)\left(t-t^2\right)^m dt}{\left(\int_0^1 \tan\left(\frac{\pi}{2}t\right)\left(t-t^2\right)^m dt\right)^2} - 1\right), \quad k \geq 5\\ =& \frac{\pi^2}{4k}d^2 + O\left(\frac{1}{k^2}\right).\end{aligned}$$

- $b_{me} \geq 1$ and $b_{me} \to 1$ monotonically with increasing k.

3.2 The Geometric Mean Estimators

The *geometric mean* estimator

$$\hat{d}_{gm} = \prod_{j=1}^{k} |x_j|^{1/k}$$

has tail bounds

$$\begin{aligned}&\mathbf{Pr}\left(\hat{d}_{gm} \geq (1+\varepsilon)d\right) \leq U_{R,gm} = \exp\left(-k\frac{\varepsilon^2}{G_{R,gm}}\right), \quad \varepsilon > 0\\ &\mathbf{Pr}\left(\hat{d}_{gm} \leq (1-\varepsilon)d\right) \leq U_{L,gm} = \exp\left(-k\frac{\varepsilon^2}{G_{L,gm}}\right), \quad 0 < \varepsilon < 1\end{aligned}$$

where

$$\begin{aligned}G_{R,gm} &= \frac{\varepsilon^2}{\left(-\frac{1}{2}\log\left(1+\left(\frac{2}{\pi}\log(1+\varepsilon)\right)^2\right) + \frac{2}{\pi}\tan^{-1}\left(\frac{2}{\pi}\log(1+\varepsilon)\right)\log(1+\varepsilon)\right)},\\ G_{L,gm} &= \frac{\varepsilon^2}{\left(-\frac{1}{2}\log\left(1+\left(\frac{2}{\pi}\log(1-\varepsilon)\right)^2\right) + \frac{2}{\pi}\tan^{-1}\left(\frac{2}{\pi}\log(1-\varepsilon)\right)\log(1-\varepsilon)\right)}.\end{aligned}$$

Moreover, for small ε, we obtain the following convenient approximations:

$$G_{R,gm} = \frac{\pi^2}{2}\left(1+\varepsilon+\left(\frac{1}{12}+\frac{2}{3\pi^2}\right)\varepsilon^2+...\right),$$
$$G_{L,gm} = \frac{\pi^2}{2}\left(1-\varepsilon+\left(\frac{1}{12}+\frac{2}{3\pi^2}\right)\varepsilon^2+...\right).$$

Consequently, we establish an analog of the Johnson-Lindenstrauss (JL) Lemma for dimension reduction in l_1:

If $k \geq G_{gm}\frac{(2\log n - \log\delta)}{\varepsilon^2}$, then with probability at least $1-\delta$, one can recover the original l_1 distance between any pair of data points (among all n data points) within a $1\pm\varepsilon$ factor ($0<\varepsilon<1$), using $\hat{d}_{gm}$. The constant G_{gm} can be specified from $G_{R,gm}$ and $G_{L,gm}$: $G_{gm} = \max\{G_{R,gm}, G_{L,gm}\}$.

To remove the bias and also reduce the variance, we recommend the bias-corrected *geometric mean* estimator:

$$\hat{d}_{gm,c} = \cos^k\left(\frac{\pi}{2k}\right)\prod_{j=1}^{k}|x_j|^{1/k},$$

which is unbiased and has variance

$$\mathrm{Var}\left(\hat{d}_{gm,c}\right) = d^2\left(\frac{\cos^{2k}\left(\frac{\pi}{2k}\right)}{\cos^k\left(\frac{\pi}{k}\right)}-1\right) = \frac{\pi^2}{4}\frac{d^2}{k}+\frac{\pi^4}{32}\frac{d^2}{k^2}+O\left(\frac{1}{k^3}\right).$$

We also derive tail bounds for $\hat{d}_{gm,c}$:

$$\mathbf{Pr}\left(\hat{d}_{gm,c} \geq (1+\varepsilon)d\right) \leq U_{R,gm,c}, \qquad \varepsilon > 0$$
$$\mathbf{Pr}\left(\hat{d}_{gm,c} \leq (1-\varepsilon)d\right) \leq U_{L,gm,c}, \qquad 0<\varepsilon<1,$$

and show that, compared with $\hat{d}_{gm}$, the ratios of the tail bounds

$$\rho_{R,k} = \frac{U_{R,gm,c}}{U_{R,gm}} \to \rho_{R,\infty} = \frac{1}{(1+\varepsilon)^{C_1}}\exp\left(-\frac{\pi^2}{8}A_1+\frac{\pi}{2}C_1\tan\left(\frac{\pi^2}{2}A_1\right)\right),$$
$$\rho_{L,k} = \frac{U_{L,gm,c}}{U_{L,gm}} \to \rho_{L,\infty} = \frac{1}{(1-\varepsilon)^{C_2}}\exp\left(\frac{\pi^2}{8}A_2-\frac{\pi}{2}C_2\tan\left(\frac{\pi^2}{2}A_2\right)\right),$$

as $k\to\infty$, where A_1, C_1, A_2, and C_2 are only functions of ε.

3.3 The Maximum Likelihood Estimators

Denoted by $\hat{d}_{MLE,c}$, the bias-corrected maximum likelihood estimator (MLE) is

$$\hat{d}_{MLE,c} = \hat{d}_{MLE}\left(1-\frac{1}{k}\right),$$

where $\hat{d}_{MLE}$ solves a nonlinear MLE equation

$$-\frac{k}{\hat{d}_{MLE}}+\sum_{j=1}^{k}\frac{2\hat{d}_{MLE}}{x_j^2+\hat{d}_{MLE}^2} = 0.$$

Some properties of $\hat{d}_{MLE,c}$:

- It is nearly unbiased, $\mathrm{E}\left(\hat{d}_{MLE,c}\right) = d + O\left(\frac{1}{k^2}\right)$.
- Its asymptotic variance is

$$\mathrm{Var}\left(\hat{d}_{MLE,c}\right) = \frac{2d^2}{k} + \frac{3d^2}{k^2} + O\left(\frac{1}{k^3}\right),$$

that is, $\frac{\mathrm{Var}\left(\hat{d}_{MLE,c}\right)}{\mathrm{Var}\left(\hat{d}_{me,c}\right)} \to \frac{8}{\pi^2}$, $\frac{\mathrm{Var}\left(\hat{d}_{MLE,c}\right)}{\mathrm{Var}\left(\hat{d}_{gm,c}\right)} \to \frac{8}{\pi^2}$, as $k \to \infty$. ($\frac{8}{\pi^2} \approx 80\%$)

- Its distribution can be accurately approximated by an inverse Gaussian, at least in the small deviation range, which suggests the following approximate tail bound

$$\mathbf{Pr}\left(|\hat{d}_{MLE,c} - d| \geq \varepsilon d\right) \overset{\sim}{\leq} 2\exp\left(-\frac{\varepsilon^2/(1+\varepsilon)}{2\left(\frac{2}{k} + \frac{3}{k^2}\right)}\right), \quad 0 < \varepsilon < 1,$$

which is verified by simulations for the tail probability $\geq 10^{-10}$ range.

4. The Sample Median Estimators

Recall in Cauchy random projections, $\mathbf{B} = \mathbf{AR}$, we denote the leading two rows in $\mathbf{A}$ by $u_1, u_2 \in \mathbb{R}^D$, and the leading two rows in $\mathbf{B}$ by $v_1, v_2 \in \mathbb{R}^k$. Our goal is to estimate the l_1 distance $d = |u_1 - u_2| = \sum_{i=1}^D |u_{1,i} - u_{2,i}|$ from $\{x_j\}_{j=1}^k$, $x_j = v_{1,j} - v_{2,j} \sim C(0,d)$, i.i.d.

A widely-used estimator in statistics is based on the sample inter-quantiles (Fama and Roll, 1968, 1971; McCulloch, 1986). For the symmetric Cauchy, the (absolute) *sample median* estimator

$$\hat{d}_{me} = \mathrm{median}\{|x_j|, j = 1, 2, ..., k\}$$

is convenient because the population median of absolute Cauchy is exactly d (Indyk, 2006).

It is well-known in statistics that $\hat{d}_{me}$, is asymptotically unbiased and normal; see Lemma 3. For small samples (e.g., $k \leq 20$), however, $\hat{d}_{me}$ is severely biased.

Lemma 3 *The* sample median *estimator, $\hat{d}_{me}$, is asymptotically unbiased and normal*

$$\sqrt{k}\left(\hat{d}_{me} - d\right) \overset{D}{\Longrightarrow} N\left(0, \frac{\pi^2}{4}d^2\right).$$

When $k = 2m+1$, m = 1, 2, 3, ..., the r^{th} moment of $\hat{d}_{me}$ can be represented as

$$E\left(\hat{d}_{me}\right)^r = d^r\left(\int_0^1 \frac{(2m+1)!}{(m!)^2}\tan^r\left(\frac{\pi}{2}t\right)\left(t - t^2\right)^m dt\right), \quad m \geq r \tag{2}$$

If $m < r$, then $E\left(\hat{d}_{me}\right)^r = \infty$.

Proof *Let $f(z;d)$ and $F(z;d)$ be the probability density and cumulative density respectively for $|C(0,d)|$:*

$$f(z;d) = \frac{2d}{\pi}\frac{1}{z^2 + d^2}, \quad F(z;d) = \frac{2}{\pi}\tan^{-1}\left(\frac{z}{d}\right), \quad z \geq 0.$$

The inverse of $F(z;d)$ *is* $F^{-1}(q;d) = d\tan\left(\frac{\pi}{2}q\right)$. *Here, we take* $q = 0.5$, *to consider the sample median. By the asymptotic normality of sample quantiles (David, 1981, Theorem 9.2), we know that*

$$\sqrt{k}\left(\hat{d}_{me} - d\right) \xrightarrow{D} N\left(0, \frac{\frac{1}{2}\frac{1}{2}}{\left(f\left(d\tan\left(\frac{\pi}{2}\frac{1}{2}\right);d\right)\times\tan\left(\frac{\pi}{2}\frac{1}{2}\right)\right)^2} = \frac{\pi^2}{4}d^2\right),$$

that is, $\hat{d}_{me}$ *is asymptotically unbiased and normal with the variance* $Var\left(\hat{d}_{me}\right) = \frac{\pi^2}{4k}d^2 + O\left(\frac{1}{k^2}\right)$.

For convenience, we assume $k = 2m+1$. *Again, by properties of sample quantile (David, 1981, Chapter 2.1), the probability density of* $\hat{d}_{me}$ *is*

$$f_{\hat{d}_{me}}(z) = \frac{(2m+1)!}{(m!)^2}\left(F(z;d)(1-F(z;d))\right)^m f(z;d),$$

from which we can write down the r^{th} *moment of* $\hat{d}_{me}$ *in (2), after some change of variables.* ■

Once we know $\mathrm{E}\left(\hat{d}_{me}\right)$, we can design an unbiased estimator as described in Lemma 4.

Lemma 4 *The estimator,*

$$\hat{d}_{me,c} = \frac{\hat{d}_{me}}{b_{me}},$$

is unbiased, that is, $E\left(\hat{d}_{me,c}\right) = d$, *where the bias-correction factor* b_{me} *is*

$$b_{me} = \frac{E\left(\hat{d}_{me}\right)}{d} = \int_0^1 \frac{(2m+1)!}{(m!)^2}\tan\left(\frac{\pi}{2}t\right)\left(t-t^2\right)^m dt, \qquad (k = 2m+1). \tag{3}$$

The variance of $\hat{d}_{me,c}$ *is*

$$Var\left(\hat{d}_{me,c}\right) = d^2\left(\frac{(m!)^2}{(2m+1)!}\frac{\int_0^1\tan^2\left(\frac{\pi}{2}t\right)\left(t-t^2\right)^m dt}{\left(\int_0^1\tan\left(\frac{\pi}{2}t\right)\left(t-t^2\right)^m dt\right)^2} - 1\right), \quad k = 2m+1 \geq 5.$$

$\hat{d}_{gm,c}$ *and* $\hat{d}_{gm}$ *are asymptotically equivalent, that is,*

$$\sqrt{k}\left(\hat{d}_{me,c} - d\right) \xrightarrow{D} N\left(0, \frac{\pi^2}{4}d^2\right).$$

The bias-correction factor b_{me} *is monotonically decreasing with increasing* m, *and*

$$b_{me} \geq 1, \qquad \lim_{m\to\infty} b_{me} = 1.$$

Proof *Most of the results follow directly from Lemma 3. Here we only show* b_{me} *decreases monotonically and* $b_{me} \to 1$ *as* $m \to \infty$.

Note that $\frac{(2m+1)!}{(m!)^2}\left(t-t^2\right)^m, 0 \leq t \leq 1$, *is the probability density of a Beta distribution* $Beta(m+1, m+1)$, *whose rth moment is* $E(z^r) = \frac{(2m+1)!(m+r)!}{(2m+1+r)!m!}$.

By Taylor expansions (Gradshteyn and Ryzhik, 1994, 1.411.6),

$$\tan\left(\frac{\pi}{2}t\right) = \sum_{j=1}^{\infty} \frac{2^{2j}\left(2^{2j}-1\right)}{(2j)!}|B_{2j}|\left(\frac{\pi}{2}\right)^{2j-1} t^{2j-1},$$

where B_{2j} is the Bernoulli number *(Gradshteyn and Ryzhik, 1994, 9.61).*

Therefore,

$$b_{me} = \sum_{j=1}^{\infty} \frac{2^{2j}\left(2^{2j}-1\right)}{(2j)!}|B_{2j}|\left(\frac{\pi}{2}\right)^{2j-1} \frac{(2m+1)!(m+2j-1)!}{(2m+2j)!m!}.$$

It is easy to show that $\frac{(2m+1)!(m+2j-1)!}{(2m+2j)!m!}$ decreases monotonically with increasing m and it converges to $\left(\frac{1}{2}\right)^{2j-1}$. Thus, b_{me} also decreases monotonically with increasing m.

From the Taylor expansion of $\tan(t)$*, we know that*

$$b_{me} \to \sum_{j=1}^{\infty} \frac{2^{2j}\left(2^{2j}-1\right)}{(2j)!}|B_{2j}|\left(\frac{\pi}{2}\right)^{2j-1}\left(\frac{1}{2}\right)^{2j-1} = \tan\left(\frac{\pi}{2}\frac{1}{2}\right) = 1.$$

■

It is well-known that bias-corrections are not always beneficial because of the bias-variance trade-off phenomenon. In our case, because the correction factor $b_{me} \geq 1$ always, the bias-correction not only removes the bias of $\hat{d}_{me}$ but also reduces the variance of $\hat{d}_{me}$.

The bias-correction factor b_{me} can be numerically evaluated and tabulated, at least for small k. Figure 1 plots b_{me} as a function of k, indicating that $\hat{d}_{me}$ is severely biased when $k \leq 20$. When $k > 50$, the bias becomes negligible.

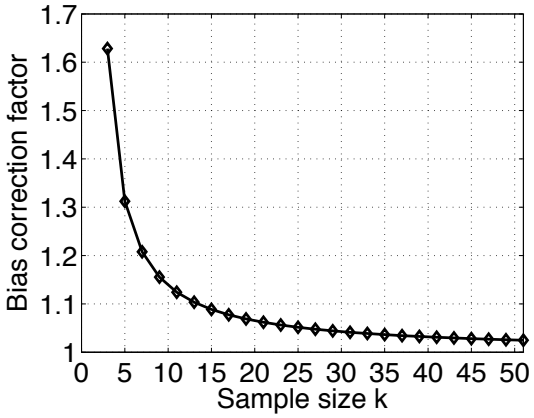


Figure 1: The bias correction factor, b_{me} in (3), as a function of $k = 2m+1$. After $k > 50$, the bias is negligible. Note that $b_{me} = \infty$ when $k = 1$.

5. The Geometric Mean Estimators

This section derives estimators based on the *geometric mean*, which are more accurate than the *sample median* estimators. The *geometric mean* estimators allow us to derive tail bounds in explicit forms and (consequently) establish an analog of the Johnson-Lindenstrauss (JL) Lemma for dimension reduction in the l_1 norm.

Lemma 5 *Assume $x \sim C(0,d)$. Then*

$$E\left(|x|^\lambda\right) = \frac{d^\lambda}{\cos(\lambda\pi/2)}, \qquad |\lambda| < 1.$$

Proof *Using the integral table (Gradshteyn and Ryzhik, 1994, 3.221.1, page 337),*

$$E\left(|x|^\lambda\right) = \frac{2d}{\pi}\int_0^\infty \frac{y^\lambda}{y^2+d^2}dy = \frac{d^\lambda}{\pi}\int_0^\infty \frac{y^{\frac{\lambda-1}{2}}}{y+1}dy = \frac{d^\lambda}{\cos(\lambda\pi/2)}.$$

■

From Lemma 5, by taking $\lambda = \frac{1}{k}$, we obtain an unbiased estimator, $\hat{d}_{gm,c}$, based on the bias-corrected *geometric mean* in the next lemma, which is proved in Appendix A.

Lemma 6

$$\hat{d}_{gm,c} = \cos^k\left(\frac{\pi}{2k}\right)\prod_{j=1}^k |x_j|^{1/k}, \quad k > 1 \tag{4}$$

is unbiased, with the variance (valid when $k > 2$)

$$Var\left(\hat{d}_{gm,c}\right) = d^2\left(\frac{\cos^{2k}\left(\frac{\pi}{2k}\right)}{\cos^k\left(\frac{\pi}{k}\right)} - 1\right) = \frac{d^2}{k}\frac{\pi^2}{4} + \frac{\pi^4}{32}\frac{d^2}{k^2} + O\left(\frac{1}{k^3}\right).$$

The third and fourth central moments are

$$E\left(\hat{d}_{gm,c} - E\left(\hat{d}_{gm,c}\right)\right)^3 = \frac{3\pi^4}{16}\frac{d^3}{k^2} + O\left(\frac{1}{k^3}\right),$$
$$E\left(\hat{d}_{gm,c} - E\left(\hat{d}_{gm,c}\right)\right)^4 = \frac{3\pi^4}{16}\frac{d^4}{k^2} + O\left(\frac{1}{k^3}\right).$$

The higher (third or fourth) moments may be useful for approximating the distribution of $\hat{d}_{gm,c}$. In Section 6, we will show how to approximate the distribution of the maximum likelihood estimator by matching the first four moments (in the leading terms). We could apply the similar technique to approximate $\hat{d}_{gm,c}$. Fortunately, we do not have to do so because we are able to derive the exact tail bounds for $\hat{d}_{gm,c}$ in Lemma 9.

Note that in (4), as $k \to \infty$, the bias-correction term converges to 1 quickly:

$$\cos^k\left(\frac{\pi}{2k}\right) = \left(1 - \frac{1}{2}\left(\frac{\pi}{2k}\right)^2 + ...\right)^k = 1 - \frac{k}{2}\left(\frac{\pi}{2k}\right)^2 + ... = 1 - \frac{\pi^2}{8}\frac{1}{k} + ... \to 1.$$

When k is not too small (e.g., $k > 50$), the *geometric mean* estimator without bias-correction,

$$\hat{d}_{gm} = \prod_{j=1}^{k} |x_j|^{1/k}, \quad k > 1,$$

gives similar results as $\hat{d}_{gm,c}$. As shown in Figure 2, the ratios of the mean square errors (MSE)

$$\frac{\text{MSE}\left(\hat{d}_{gm}\right)}{\text{MSE}\left(\hat{d}_{gm,c}\right)} = \frac{\frac{1}{\cos^k\left(\frac{\pi}{k}\right)} - \frac{2}{\cos^k\left(\frac{\pi}{2k}\right)} + 1}{\frac{\cos^{2k}\left(\frac{\pi}{2k}\right)}{\cos^k\left(\frac{\pi}{k}\right)} - 1} \tag{5}$$

demonstrate that the two *geometric mean* estimators are similar when $k > 50$, in terms of the MSE.

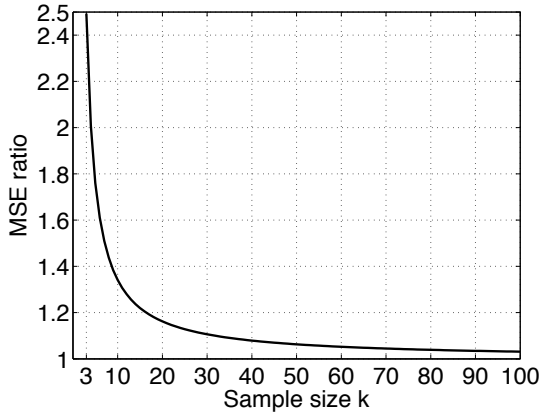


Figure 2: The ratios of the mean square errors (MSE) $\frac{\text{MSE}\left(\hat{d}_{gm}\right)}{\text{MSE}\left(\hat{d}_{gm,c}\right)}$ in (5) indicate that the difference between $\hat{d}_{gm}$ and $\hat{d}_{gm,c}$ becomes negligible when $k > 50$.

One advantage of $\hat{d}_{gm}$ is the convenience for deriving tail bounds. Thus, before presenting Lemma 9 for $\hat{d}_{gm,c}$, we prove tail bounds for $\hat{d}_{gm}$ in Lemma 7 (proved in Appendix B).

Lemma 7

$$\mathbf{Pr}\left(\hat{d}_{gm} \geq (1+\varepsilon)d\right) \leq U_{R,gm} = \exp\left(-k\frac{\varepsilon^2}{G_{R,gm}}\right), \quad \varepsilon > 0$$

$$\mathbf{Pr}\left(\hat{d}_{gm} \leq (1-\varepsilon)d\right) \leq U_{L,gm} = \exp\left(-k\frac{\varepsilon^2}{G_{L,gm}}\right), \quad 0 < \varepsilon < 1$$

where

$$G_{R,gm} = \frac{\varepsilon^2}{\left(-\frac{1}{2}\log\left(1+\left(\frac{2}{\pi}\log(1+\varepsilon)\right)^2\right) + \frac{2}{\pi}\tan^{-1}\left(\frac{2}{\pi}\log(1+\varepsilon)\right)\log(1+\varepsilon)\right)}, \tag{6}$$

$$G_{L,gm} = \frac{\varepsilon^2}{\left(-\frac{1}{2}\log\left(1+\left(\frac{2}{\pi}\log(1-\varepsilon)\right)^2\right) + \frac{2}{\pi}\tan^{-1}\left(\frac{2}{\pi}\log(1-\varepsilon)\right)\log(1-\varepsilon)\right)}. \tag{7}$$

Moreover, for small ϵ, *we have the following convenient approximations:*

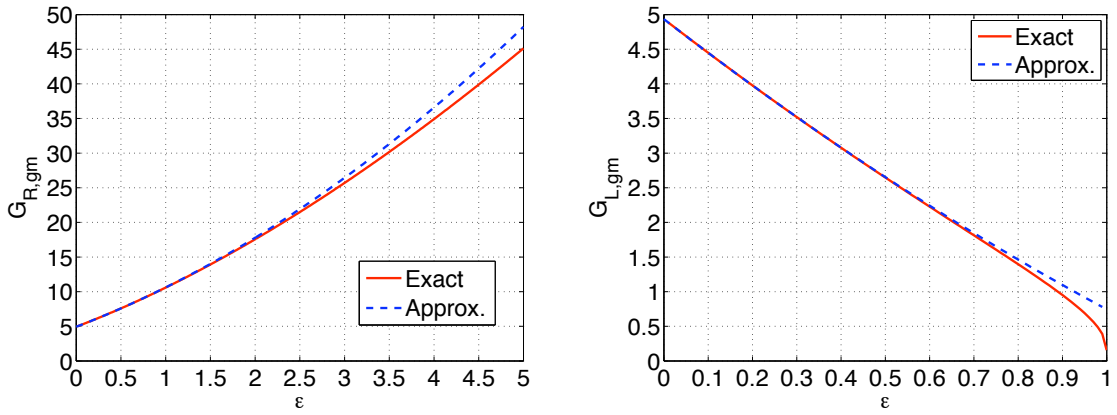
$$G_{R,gm} = \frac{\pi^2}{2}\left(1+\epsilon+\left(\frac{1}{12}+\frac{2}{3\pi^2}\right)\epsilon^2+...\right), \tag{8}$$

$$G_{L,gm} = \frac{\pi^2}{2}\left(1-\epsilon+\left(\frac{1}{12}+\frac{2}{3\pi^2}\right)\epsilon^2+...\right). \tag{9}$$

Consequently, as $\epsilon \to 0+$, *we know*

$$G_{R,gm} \to \frac{\pi^2}{2}, \qquad G_{L,gm} \to \frac{\pi^2}{2}.$$

Figure 3 plots the constants $G_{R,gm}$ and $G_{L,gm}$ in (6) and (7), along with their convenient approximations (8) and (9). For $G_{R,gm}$, the exact and approximate expressions are indistinguishable when $\epsilon < 2$. For $G_{L,gm}$, the exact and approximate expressions are indistinguishable when $\epsilon < 0.7$. The plots also suggest that the approximations, (8) and (9), are upper bounds of the exact constants, (6) and (7), respectively.



(a) $G_{R,gm}$ (b) $G_{L,gm}$

Figure 3: We plot the constants $G_{R,gm}$ and $G_{L,gm}$ in (6) and (7), along with their convenient approximations (8) and (9).

Consequently, Lemma 7 establishes an analog of the Johnson-Lindenstrauss (JL) Lemma for dimension reduction in l_1:

Lemma 8 *If* $k \geq G_{gm}\frac{(2\log n - \log\delta)}{\epsilon^2}$, *then with probability at least* $1-\delta$, *one can recover the original* l_1 *distance between any pair of data points (among all* n *data points) within a* $1\pm\epsilon$ *factor* ($0<\epsilon<1$), *using* $\hat{d}_{gm}$. *It suffices to specify the constant* $G_{gm} = \max\{G_{R,gm}, G_{L,gm}\}$.

Similarly, we derive tail bounds for the unbiased *geometric mean* estimator $\hat{d}_{gm,c}$, in Lemma 9, which is proved in Appendix C.

Lemma 9

$$\mathbf{Pr}\left(\hat{d}_{gm,c} \geq (1+\varepsilon)d\right) \leq U_{R,gm,c} = \frac{\cos^{kt_1^*}\left(\frac{\pi}{2k}\right)}{\cos^k\left(\frac{\pi t_1^*}{2k}\right)(1+\varepsilon)^{t_1^*}}, \qquad \varepsilon > 0$$

where

$$t_1^* = \frac{2k}{\pi}\tan^{-1}\left(\left(\log(1+\varepsilon) - k\log\cos\left(\frac{\pi}{2k}\right)\right)\frac{2}{\pi}\right).$$

$$\mathbf{Pr}\left(\hat{d}_{gm,c} \leq (1-\varepsilon)d\right) \leq U_{L,gm,c} = \frac{(1-\varepsilon)^{t_2^*}}{\cos^k\left(\frac{\pi t_2^*}{2k}\right)\cos^{kt_2^*}\left(\frac{\pi}{2k}\right)}, \qquad 0<\varepsilon<1, \quad k \geq \frac{\pi^2}{8\varepsilon}$$

where

$$t_2^* = \frac{2k}{\pi}\tan^{-1}\left(\left(-\log(1-\varepsilon) + k\log\cos\left(\frac{\pi}{2k}\right)\right)\frac{2}{\pi}\right).$$

As $k \to \infty$, for any fixed ε, we have

$$\rho_{R,k} = \frac{U_{R,gm,c}}{U_{R,gm}} \to \rho_{R,\infty} = \frac{1}{(1+\varepsilon)^{C_1}}\exp\left(-\frac{\pi^2}{8}A_1 + \frac{\pi}{2}C_1\tan\left(\frac{\pi^2}{2}A_1\right)\right), \tag{10}$$

$$\rho_{L,k} = \frac{U_{L,gm,c}}{U_{L,gm}} \to \rho_{L,\infty} = \frac{1}{(1-\varepsilon)^{C_2}}\exp\left(\frac{\pi^2}{8}A_2 - \frac{\pi}{2}C_2\tan\left(\frac{\pi^2}{2}A_2\right)\right), \tag{11}$$

where $U_{R,gm}$ and $U_{L,gm}$ are upper bounds for $\hat{d}_{gm}$ as derived in Lemma 7, and

$$A_1 = \frac{2}{\pi}\left(\tan^{-1}\left(\log(1+\varepsilon)\frac{2}{\pi}\right)\right), \quad C_1 = \frac{1/2}{1+\left(\log(1+\varepsilon)\frac{2}{\pi}\right)^2},$$

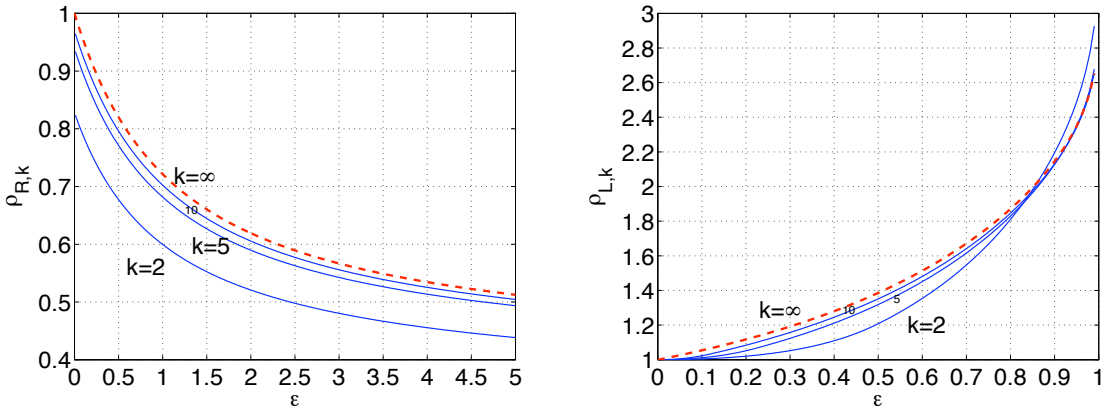
$$A_2 = \frac{2}{\pi}\left(\tan^{-1}\left(-\log(1-\varepsilon)\frac{2}{\pi}\right)\right), \quad C_2 = \frac{1/2}{1+\left(\log(1-\varepsilon)\frac{2}{\pi}\right)^2}.$$

Figure 4 plots the tail bound ratios $\rho_{R,k}$ and $\rho_{L,k}$ as defined in Lemma 9, indicating that the asymptotic expressions $\rho_{R,\infty}$ and $\rho_{L,\infty}$ are in fact very accurate even for small k (e.g., $k = 10$).

Figure 4 illustrates that introducing the bias-correction term in $\hat{d}_{gm,c}$ reduces the right tail bound but amplifies the left tail bound. Because the left tail bound is usually much smaller than the right tail bound, we expect that overall the bias-correction should be beneficial, as shown in Figure 5, which plots the overall ratio of tail bounds:

$$\rho_k = \frac{U_{R,gm,c} + U_{L,gm,c}}{U_{R,gm} + U_{L,gm}} = \frac{\frac{\cos^{kt_1^*}\left(\frac{\pi}{2k}\right)}{\cos^k\left(\frac{\pi t_1^*}{2k}\right)(1+\varepsilon)^{t_1^*}} + \frac{(1-\varepsilon)^{t_2^*}}{\cos^k\left(\frac{\pi t_2^*}{2k}\right)\cos^{kt_2^*}\left(\frac{\pi}{2k}\right)}}{\exp\left(-k\frac{\varepsilon^2}{G_{R,gm}}\right) + \exp\left(-k\frac{\varepsilon^2}{G_{L,gm}}\right)}. \tag{12}$$

Finally, Figure 6 compares $\hat{d}_{gm,c}$ with the sample median estimators $\hat{d}_{me}$ and $\hat{d}_{me,c}$, in terms of the mean square errors. $\hat{d}_{gm,c}$ is considerably more accurate than $\hat{d}_{me}$ at small k. The bias correction significantly reduces the mean square errors of $\hat{d}_{me}$.



(a) Right bound ratios $\rho_{R,k}$ (b) Left bound ratios $\rho_{L,k}$

Figure 4: Tail bound ratios $\rho_{R,k}$ and $\rho_{L,k}$ as defined in (10) and (11) for $k = 2, 5, 10$, along with the asymptotic expressions $\rho_{R,\infty}$ and $\rho_{L,\infty}$. The dashed curves correspond to $k = \infty$.

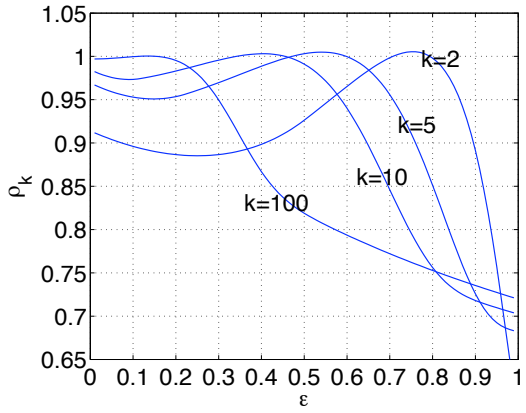


Figure 5: The overall ratios of tail bounds, ρ_k as defined in (12) are almost always below one, demonstrating that the bias-corrected estimator $\hat{d}_{gm,c}$ may exhibit better overall tail behavior than the biased estimator $\hat{d}_{gm}$.

6. The Maximum Likelihood Estimators

This section analyzes the maximum likelihood estimators (MLE), which are asymptotically optimum (in terms of the variance). In comparisons, the *sample median* and *geometric mean* estimators are not optimum. Our contribution in this section includes the higher-order analysis for the bias and moments and accurate closed-from approximations to the distribution of the MLE.

The method of maximum likelihood is widely used. For example, Li et al. (2006a) applied the maximum likelihood method to *normal random projections* and provided an improved estimator of the l_2 distance by taking advantage of the marginal information.

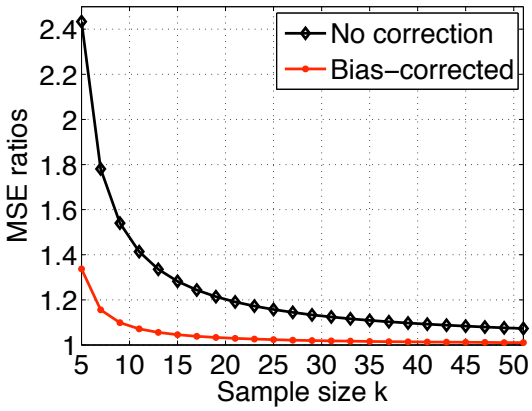


Figure 6: The ratios of the mean square errors (MSE), $\frac{\text{MSE}(\hat{d}_{me})}{\text{MSE}(\hat{d}_{gm,c})}$ and $\frac{\text{MSE}(\hat{d}_{me,c})}{\text{MSE}(\hat{d}_{gm,c})}$, demonstrate that the bias-corrected geometric mean estimator $\hat{d}_{gm,c}$ is considerably more accurate than the sample median estimator $\hat{d}_{me}$. The bias correction on $\hat{d}_{me}$ considerably reduces the MSE. Note that when $k = 3$, the ratios are ∞.

Recall our goal is to estimate d from k i.i.d. samples $x_j \sim C(0,d), j = 1,2,...,k$. The log joint likelihood of $\{x_j\}_{j=1}^k$ is

$$L(x_1, x_2, ...x_k; d) = k\log(d) - k\log(\pi) - \sum_{j=1}^{k} \log(x_j^2 + d^2),$$

whose first and second derivatives (w.r.t. d) are

$$L'(d) = \frac{k}{d} - \sum_{j=1}^{k} \frac{2d}{x_j^2 + d^2},$$

$$L''(d) = -\frac{k}{d^2} - \sum_{j=1}^{k} \frac{2x_j^2 - 2d^2}{(x_j^2 + d^2)^2} = -\frac{L'(d)}{d} - 4\sum_{j=1}^{k} \frac{x_j^2}{(x_j^2 + d^2)^2}.$$

The maximum likelihood estimator of d, denoted by $\hat{d}_{MLE}$, is the solution to $L'(d) = 0$, that is,

$$-\frac{k}{\hat{d}_{MLE}} + \sum_{j=1}^{k} \frac{2\hat{d}_{MLE}}{x_j^2 + \hat{d}_{MLE}^2} = 0. \tag{13}$$

Because $L''(\hat{d}_{MLE}) \leq 0$, $\hat{d}_{MLE}$ indeed maximizes the joint likelihood and is the only solution to the MLE equation (13). Solving (13) numerically is not difficult (e.g., a few iterations using the Newton's method). For a better accuracy, we recommend the following bias-corrected estimator:

$$\hat{d}_{MLE,c} = \hat{d}_{MLE}\left(1 - \frac{1}{k}\right).$$

Lemma 10 concerns the asymptotic moments of $\hat{d}_{MLE}$ and $\hat{d}_{MLE,c}$, proved in Appendix D.

Lemma 10 *Both $\hat{d}_{MLE}$ and $\hat{d}_{MLE,c}$ are asymptotically unbiased and normal. The first four moments of $\hat{d}_{MLE}$ are*

$$E\left(\hat{d}_{MLE}-d\right)=\frac{d}{k}+O\left(\frac{1}{k^2}\right)$$
$$Var\left(\hat{d}_{MLE}\right)=\frac{2d^2}{k}+\frac{7d^2}{k^2}+O\left(\frac{1}{k^3}\right)$$
$$E\left(\hat{d}_{MLE}-E(\hat{d}_{MLE})\right)^3=\frac{12d^3}{k^2}+O\left(\frac{1}{k^3}\right)$$
$$E\left(\hat{d}_{MLE}-E(\hat{d}_{MLE})\right)^4=\frac{12d^4}{k^2}+\frac{222d^4}{k^3}+O\left(\frac{1}{k^4}\right).$$

The first four moments of $\hat{d}_{MLE,c}$ are

$$E\left(\hat{d}_{MLE,c}-d\right)=O\left(\frac{1}{k^2}\right)$$
$$Var\left(\hat{d}_{MLE,c}\right)=\frac{2d^2}{k}+\frac{3d^2}{k^2}+O\left(\frac{1}{k^3}\right)$$
$$E\left(\hat{d}_{MLE,c}-E(\hat{d}_{MLE,c})\right)^3=\frac{12d^3}{k^2}+O\left(\frac{1}{k^3}\right)$$
$$E\left(\hat{d}_{MLE,c}-E(\hat{d}_{MLE,c})\right)^4=\frac{12d^4}{k^2}+\frac{186d^4}{k^3}+O\left(\frac{1}{k^4}\right).$$

The order $O\left(\frac{1}{k}\right)$ term of the variance, that is, $\frac{2d^2}{k}$, is well-known (Haas et al., 1970). We derive the bias-corrected estimator, $\hat{d}_{MLE,c}$, and the higher order moments using stochastic Taylor expansions (Bartlett, 1953; Shenton and Bowman, 1963; Ferrari et al., 1996; Cysneiros et al., 2001).

We will propose an inverse Gaussian distribution to approximate the distribution of $\hat{d}_{MLE,c}$, by matching the first four moments (at least in the leading terms).

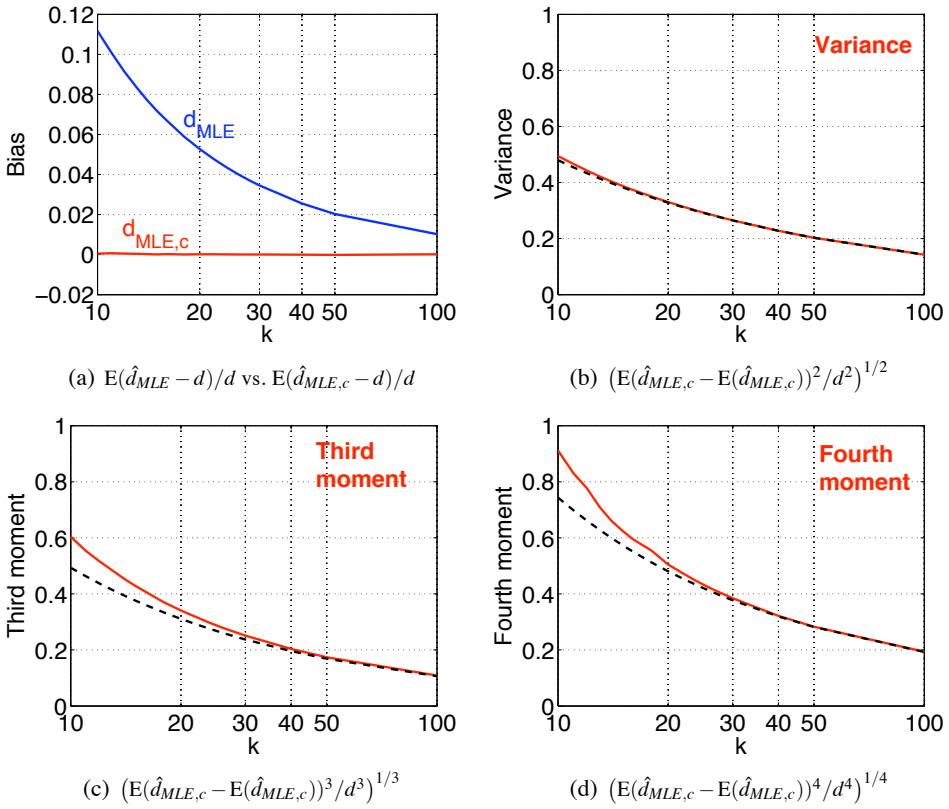
6.1 A Numerical Example

The maximum likelihood estimators are tested on some Microsoft Web crawl data, a term-by-document matrix with $D=2^{16}$ Web pages. We conduct Cauchy random projections and estimate the l_1 distances between words. In this experiment, we compare the empirical and (asymptotic) theoretical moments, using one pair of words. Figure 7 illustrates that the bias correction is effective and these (asymptotic) formulas for the first four moments of $\hat{d}_{MLE,c}$ in Lemma 10 are accurate, especially when $k \geq 20$.

6.2 Approximate Distributions

Theoretical analysis on the exact distribution of a maximum likelihood estimator is difficult. It is common practice to assume normality, which, however, is inaccurate.[2] The *Edgeworth expansion*

2. The simple normal approximation can be improved by taking advantage of the conditional density on the ancillary configuration statistic, based on the observations $x_1, x_2, ..., x_k$ (Fisher, 1934; Lawless, 1972; Hinkley, 1978).



(a) $\mathrm{E}(\hat{d}_{MLE}-d)/d$ vs. $\mathrm{E}(\hat{d}_{MLE,c}-d)/d$

(b) $\left(\mathrm{E}(\hat{d}_{MLE,c}-\mathrm{E}(\hat{d}_{MLE,c}))^2/d^2\right)^{1/2}$

(c) $\left(\mathrm{E}(\hat{d}_{MLE,c}-\mathrm{E}(\hat{d}_{MLE,c}))^3/d^3\right)^{1/3}$

(d) $\left(\mathrm{E}(\hat{d}_{MLE,c}-\mathrm{E}(\hat{d}_{MLE,c}))^4/d^4\right)^{1/4}$

Figure 7: One pair of words are selected from a Microsoft term-by-document matrix with $D = 2^{16}$ Web pages. We conduct Cauchy random projections and estimate the l_1 distance between one pair of words using the maximum likelihood estimator $\hat{d}_{MLE}$ and the bias-corrected version $\hat{d}_{MLE,c}$. Panel (a) plots the biases of $\hat{d}_{MLE}$ and $\hat{d}_{MLE,c}$, indicating that the bias correction is effective. Panels (b), (c), and (d) plot the variance, third moment, and fourth moment of $\hat{d}_{MLE,c}$, respectively. The dashed curves are the theoretical asymptotic moments. When $k \geq 20$, the theoretical asymptotic formulas for moments are accurate.

improves the normal approximation by matching higher moments (Feller, 1971; Bhattacharya and Ghosh, 1978; Severini, 2000), which however, has some well-known drawbacks. The resultant expressions are quite sophisticated and are not accurate at the tails. It is possible that the approximate probability has values below zero. Also, Edgeworth expansions consider the support to be $(-\infty, \infty)$, while $\hat{d}_{MLE,c}$ is non-negative.

The *saddle-point approximation* (Jensen, 1995) in general improves Edgeworth expansions, often considerably. Unfortunately, we can not apply the saddle-point approximation in our case (at least not directly), because it requires a bounded moment generating function.

We propose approximating the distributions of $\hat{d}_{MLE,c}$ directly using some well-studied common distributions. We will first consider a gamma distribution with the same first two (asymptotic) moments of $\hat{d}_{MLE,c}$. That is, the gamma distribution will be asymptotically equivalent to the normal approximation. While a normal has zero third central moment, a gamma has nonzero third central

moment. This, to an extent, speeds up the rate of convergence. Another important reason why a gamma is more accurate is because it has the same support as $\hat{d}_{MLE,c}$, that is, $[0,\infty)$.

We will furthermore consider a *generalized gamma* distribution, which allows us to match the first three (asymptotic) moments of $\hat{d}_{MLE,c}$. Interestingly, in this case, the generalized gamma approximation turns out to be an inverse Gaussian distribution, which has a closed-form probability density. More interestingly, this inverse Gaussian distribution also matches the fourth central moment of $\hat{d}_{MLE,c}$ in the $O\left(\frac{1}{k^2}\right)$ term and almost in the $O\left(\frac{1}{k^3}\right)$ term. By simulations, the inverse Gaussian approximation is highly accurate.

Note that, since we are interested in the very small (e.g., 10^{-10}) tail probability range, $O\left(k^{-3/2}\right)$ is not too meaningful. For example, $k^{-3/2}=10^{-3}$ if $k=100$. Therefore, we will have to rely on simulations to assess the accuracy of the approximations. On the other hand, an upper bound may hold exactly (verified by simulations) even if it is based on an approximate distribution.

As the related work, Li et al. (2006c) applied gamma and generalized gamma approximations to model the performance measure distribution in some wireless communication channels using random matrix theory and produced accurate results in evaluating the error probabilities.

6.2.1 THE GAMMA APPROXIMATION

The gamma approximation is an obvious improvement over the normal approximation.[3] A gamma distribution, $G(\alpha,\beta)$, has two parameters, α and β, which can be determined by matching the first two (asymptotic) moments of $\hat{d}_{MLE,c}$. That is, we assume that $\hat{d}_{MLE,c}\sim G(\alpha,\beta)$, with

$$\alpha\beta=d,\qquad \alpha\beta^2=\frac{2d^2}{k}+\frac{3d^2}{k^2},\quad\Longrightarrow\quad \alpha=\frac{1}{\frac{2}{k}+\frac{3}{k^2}},\qquad \beta=\frac{2d}{k}+\frac{3d}{k^2}.$$

Assuming a gamma distribution, it is easy to obtain the following Chernoff bounds:

$$\mathbf{Pr}\left(\hat{d}_{MLE,c}\geq(1+\varepsilon)d\right)\overset{\sim}{\leq}\exp\left(-\alpha\left(\varepsilon-\log(1+\varepsilon)\right)\right),\qquad \varepsilon>0 \tag{14}$$

$$\mathbf{Pr}\left(\hat{d}_{MLE,c}\leq(1-\varepsilon)d\right)\overset{\sim}{\leq}\exp\left(-\alpha\left(-\varepsilon-\log(1-\varepsilon)\right)\right),\qquad 0<\varepsilon<1, \tag{15}$$

where we use $\overset{\sim}{\leq}$ to indicate that these inequalities are based on an approximate distribution.

Note that the distribution of $\hat{d}_{MLE}/d$ (and hence $\hat{d}_{MLE,c}/d$) is only a function of k (Antle and Bain, 1969; Haas et al., 1970). Therefore, we can evaluate the accuracy of the gamma approximation by simulations with $d=1$, as presented in Figure 8.

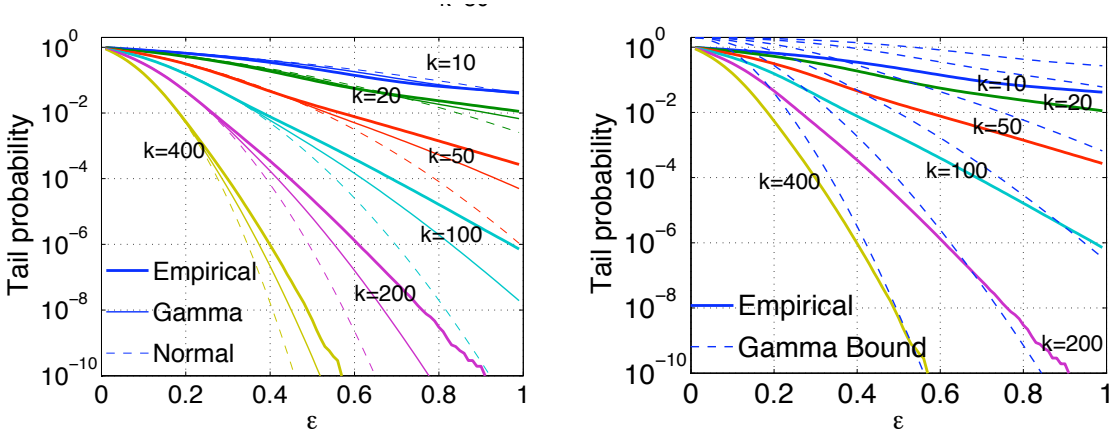
Figure 8(a) shows that both the gamma and normal approximations are fairly accurate when the tail probability $\geq 10^{-2}\sim 10^{-3}$; and the gamma approximation is obviously better.

Figure 8(b) compares the empirical tail probabilities with the gamma Chernoff upper bound (14)+(15), indicating that these bounds are reliable, when the tail probability $\geq 10^{-5}\sim 10^{-6}$.

6.2.2 THE INVERSE GAUSSIAN (GENERALIZED GAMMA) APPROXIMATION

The distribution of $\hat{d}_{MLE,c}$ can be well approximated by an inverse Gaussian distribution, which is a special case of the three-parameter generalized gamma distribution (Hougaard, 1986; Gerber, 1991;

3. Recall that, in *normal random projections* for dimension reduction in l_2 (see Lemma 1), the resultant estimator of the squared l_2 distance has a Chi-squared distribution, which is a special case of gamma.



(a) (b)

Figure 8: We consider $k = 10, 20, 50, 100, 200$, and 400. For each k, we simulate standard Cauchy samples, from which we estimate the Cauchy parameter by the MLE $\hat{d}_{MLE,c}$ and compute the tail probabilities. Panel (a) compares the empirical tail probabilities (thick solid) with the gamma tail probabilities (thin solid), indicating that the gamma distribution is better than the normal (dashed) for approximating the distribution of $\hat{d}_{MLE,c}$. Panel (b) compares the empirical tail probabilities with the gamma upper bound (14)+(15).

Li et al., 2006c), denoted by $GG(\alpha,\beta,\eta)$. Note that the usual gamma distribution is a special case with $\eta = 1$.

If $z \sim GG(\alpha,\beta,\eta)$, then the first three moments are

$$\mathrm{E}(z) = \alpha\beta, \quad \mathrm{Var}(z) = \alpha\beta^2, \quad \mathrm{E}\left(z - \mathrm{E}(z)\right)^3 = \alpha\beta^3(1+\eta).$$

We can approximate the distribution of $\hat{d}_{MLE,c}$ by matching the first three moments, that is,

$$\alpha\beta = d, \quad \alpha\beta^2 = \frac{2d^2}{k} + \frac{3d^2}{k^2}, \quad \alpha\beta^3(1+\eta) = \frac{12d^3}{k^2},$$

from which we obtain

$$\alpha = \frac{1}{\frac{2}{k} + \frac{3}{k^2}}, \quad \beta = \frac{2d}{k} + \frac{3d}{k^2}, \quad \eta = 2 + O\left(\frac{1}{k}\right). \tag{16}$$

Taking only the leading term for η, the generalized gamma approximation of $\hat{d}_{MLE,c}$ would be

$$GG\left(\frac{1}{\frac{2}{k} + \frac{3}{k^2}}, \frac{2d}{k} + \frac{3d}{k^2}, 2\right). \tag{17}$$

In general, a generalized gamma distribution does not have a closed-form density function although it always has a closed-from moment generating function. In our case, (17) is actually an inverse Gaussian (IG) distribution, which has a closed-form density function. Assuming $\hat{d}_{MLE,c} \sim IG(\alpha,\beta)$, with parameters α and β defined in (16), the moment generating function (MGF),

the probability density function (PDF), and cumulative density function (CDF) would be (Seshadri, 1993, Chapter 2) (Tweedie, 1957a,b)[4]

$$\begin{aligned}
&\mathrm{E}\left(\exp(\hat{d}_{MLE,c}t)\right) \cong \exp\left(\alpha\left(1-(1-2\beta t)^{1/2}\right)\right),\\
&f_{\hat{d}_{MLE,c}}(y) \cong \frac{\alpha\sqrt{\beta}}{\sqrt{2\pi}}y^{-\frac{3}{2}}\exp\left(-\frac{(y/\beta-\alpha)^2}{2y/\beta}\right) = \sqrt{\frac{\alpha d}{2\pi}}y^{-\frac{3}{2}}\exp\left(-\frac{(y-d)^2}{2y\beta}\right),\\
&\mathbf{Pr}\left(\hat{d}_{MLE,c}\le y\right) \cong \Phi\left(\sqrt{\frac{\alpha^2\beta}{y}}\left(\frac{y}{\alpha\beta}-1\right)\right)+e^{2\alpha}\Phi\left(-\sqrt{\frac{\alpha^2\beta}{y}}\left(\frac{y}{\alpha\beta}+1\right)\right)\\
&\qquad = \Phi\left(\sqrt{\frac{\alpha d}{y}}\left(\frac{y}{d}-1\right)\right)+e^{2\alpha}\Phi\left(-\sqrt{\frac{\alpha d}{y}}\left(\frac{y}{d}+1\right)\right),
\end{aligned}$$

where $\Phi(.)$ is the standard normal CDF, that is, $\Phi(z)=\int_{-\infty}^{z}\frac{1}{\sqrt{2\pi}}e^{-\frac{t^2}{2}}dt$. Here we use $\cong$ to indicate that these equalities are based on an approximate distribution.

Assuming $\hat{d}_{MLE,c}\sim IG(\alpha,\beta)$, the fourth central moment should be

$$\begin{aligned}
\mathrm{E}\left(\hat{d}_{MLE,c}-\mathrm{E}\left(\hat{d}_{MLE,c}\right)\right)^4 &\cong 15\alpha\beta^4+3\left(\alpha\beta^2\right)^2\\
&= 15d\left(\frac{2d}{k}+\frac{3d}{k^2}\right)^3+3\left(\frac{2d^2}{k}+\frac{3d^2}{k^2}\right)^2\\
&= \frac{12d^4}{k^2}+\frac{156d^4}{k^3}+O\left(\frac{1}{k^4}\right).
\end{aligned}$$

Lemma 10 has shown the true asymptotic fourth central moment:

$$\mathrm{E}\left(\hat{d}_{MLE,c}-\mathrm{E}\left(\hat{d}_{MLE,c}\right)\right)^4 = \frac{12d^4}{k^2}+\frac{186d^4}{k^3}+O\left(\frac{1}{k^4}\right).$$

That is, the inverse Gaussian approximation matches not only the leading term, $\frac{12d^4}{k^2}$, but also almost the higher order term, $\frac{186d^4}{k^3}$, of the true asymptotic fourth moment of $\hat{d}_{MLE,c}$.

Assuming $\hat{d}_{MLE,c}\sim IG(\alpha,\beta)$, the tail probability of $\hat{d}_{MLE,c}$ can be expressed as

$$\begin{aligned}
&\mathbf{Pr}\left(\hat{d}_{MLE,c}\ge(1+\varepsilon)d\right) \cong \Phi\left(-\varepsilon\sqrt{\frac{\alpha}{1+\varepsilon}}\right)-e^{2\alpha}\Phi\left(-(2+\varepsilon)\sqrt{\frac{\alpha}{1+\varepsilon}}\right),\quad \varepsilon>0\\
&\mathbf{Pr}\left(\hat{d}_{MLE,c}\le(1-\varepsilon)d\right) \cong \Phi\left(-\varepsilon\sqrt{\frac{\alpha}{1-\varepsilon}}\right)+e^{2\alpha}\Phi\left(-(2-\varepsilon)\sqrt{\frac{\alpha}{1-\varepsilon}}\right),\quad 0<\varepsilon<1.
\end{aligned}$$

Assuming $\hat{d}_{MLE,c}\sim IG(\alpha,\beta)$, it is easy to show the following Chernoff bounds:

$$\mathbf{Pr}\left(\hat{d}_{MLE,c}\ge(1+\varepsilon)d\right) \lesssim \exp\left(-\frac{\alpha\varepsilon^2}{2(1+\varepsilon)}\right),\quad \varepsilon>0 \tag{18}$$

$$\mathbf{Pr}\left(\hat{d}_{MLE,c}\le(1-\varepsilon)d\right) \lesssim \exp\left(-\frac{\alpha\varepsilon^2}{2(1-\varepsilon)}\right),\quad 0<\varepsilon<1. \tag{19}$$

4. The inverse Gaussian distribution was first noted as the distribution of the first passage time of the Brownian motion with a positive drift. It has many interesting properties such as infinitely divisibility. Two monographs (Chhikara and Folks, 1989; Seshadri, 1993) are devoted entirely to the inverse Gaussian distributions. For a quick reference, one can check *http://mathworld.wolfram.com/InverseGaussianDistribution.html*.

To see (18), assume $z \sim IG(\alpha, \beta)$. Then, using the Chernoff inequality:

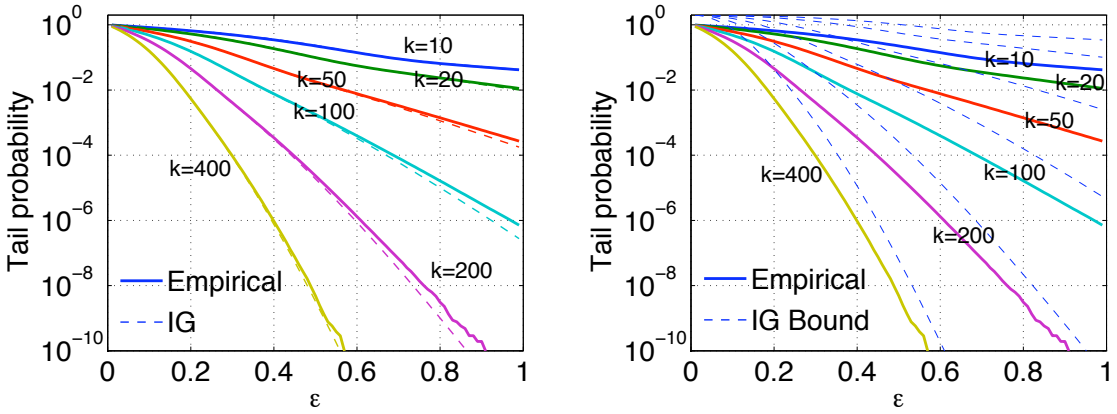
$$\begin{aligned}\Pr(z \geq (1+\varepsilon)d) \leq & \mathrm{E}(zt)\exp(-(1+\varepsilon)dt) \\ = & \exp\left(\alpha\left(1-(1-2\beta t)^{1/2}\right)-(1+\varepsilon)dt\right),\end{aligned}$$

whose minimum is $\exp\left(-\frac{\alpha\varepsilon^2}{2(1+\varepsilon)}\right)$, attained at $t=\left(1-\frac{1}{(1+\varepsilon)^2}\right)\frac{1}{2\beta}$. We can similarly show (19).

Combining (18) and (19) yields a symmetric approximate bound

$$\Pr\left(\left|\hat{d}_{MLE,c}-d\right| \geq \varepsilon d\right) \overset{\sim}{\leq} 2\exp\left(-\frac{\varepsilon^2/(1+\varepsilon)}{2\left(\frac{2}{k}+\frac{3}{k^2}\right)}\right), \quad 0<\varepsilon<1.$$

Figure 9 compares the inverse Gaussian approximation with the same simulations as presented in Figure 8, indicating that the inverse Gaussian approximation is highly accurate. When the tail probability $\geq 10^{-4} \sim 10^{-6}$, we can treat the inverse Gaussian as the exact distribution of $\hat{d}_{MLE,c}$. The Chernoff upper bounds for the inverse Gaussian are always reliable in our simulation range (the tail probability $\geq 10^{-10}$).



(a) (b)

Figure 9: We compare the inverse Gaussian approximation with the same simulations as presented in Figure 8. Panel (a) compares the empirical tail probabilities with the inverse Gaussian tail probabilities, indicating that the approximation is highly accurate. Panel (b) compares the empirical tail probabilities with the inverse Gaussian upper bound (18)+(19). The upper bounds are all above the corresponding empirical curves, indicating that our proposed bounds are reliable at least in our simulation range.

7. Conclusion

In machine learning, it is well-known that the l_1 distance is far more robust than the l_2 distance against "outliers." Dimension reduction in the l_1 norm, however, has been proved *impossible* if we use *linear random projections* and *linear estimators*. In this study, we analyze three types of

nonlinear estimators for *Cauchy random projections*: the *sample median* estimators, the *geometric mean* estimators, and the *maximum likelihood* estimators. Our theoretical analysis has shown that these nonlinear estimators can accurately recover the original l_1 distance, even though none of them can be a metric.

The *sample median* estimators and the *geometric mean* estimators are asymptotically equivalent but the latter are more accurate at small sample size. We have derived explicit tail bounds for the *geometric mean* estimators in exponential forms. Using these tail bounds, we have established an analog of the Johnson-Lindenstrauss (JL) Lemma for dimension reduction in l_1, which is weaker than the classical JL Lemma for dimension reduction in l_2.

We conduct theoretic analysis on the *maximum likelihood* estimators (MLE), which are "asymptotically optimum." Both the *sample median* and *geometric mean* estimators are about 80% efficient as the MLE. We propose approximating the distribution of the MLE by an inverse Gaussian, which has the same support and matches the leading terms of the first four moments of the MLE. Approximate tail bounds have been provided based on the inverse Gaussian approximation. Verified by simulations, these approximate tail bounds hold at least in the $\geq 10^{-10}$ tail probability range.

Although these nonlinear estimators are not metrics, they are still useful for certain applications in, for example, data stream computations, information retrieval, learning and data mining, whenever the goal is to compute the l_1 distances efficiently using a small storage space in a single pass of the data.

Li (2008) generalized the *geometric mean* estimators to the stable distribution family, for dimension reduction in the l_p norm ($0 < p \leq 2$). Li (2008) also proposed the *harmonic mean* estimator for $p \to 0+$, which is far more accurate than the *geometric mean* estimator.[5] In addition, Li (2007) suggested *very sparse stable random projections* by replacing the stable distribution with a mixture of a symmetric Pareto distribution and point mass at the origin, for considerably simplifying the sampling procedure (to generate the projection matrix) and for achieving a significant cost reduction of matrix multiplication operations.

The general method of *linear (stable) random projections* is an appealing paradigm for applications involving massive, high-dimensional, non-sparse, and heavy-tailed data. If there is prior information that the data are highly sparse (e.g., text data), other alternative dimension reduction methods may be more suitable; for example, the new technique called *Conditional Random Sampling (CRS)* (Li and Church, 2005, 2007; Li et al., 2007a) was particularly designed for approximating distances (and other summary statistics) in highly sparse data.

Acknowledgments

We thank Piotr Indyk, Assaf Naor, Art Owen, and Anand Vidyashankar. Ping Li was partially supported by NSF Grant DMS-0505676 and Cornell University junior faculty startup fund. Trevor Hastie was partially supported by NSF Grant DMS-0505676 and NIH Grant 2R01 CA 72028-07.

5. Stable random projections with very small p ($p \to 0+$) have been applied to approximating the Hamming distances (Cormode et al., 2002, 2003) and the max-dominance norm (Cormode and Muthukrishnan, 2003).

Appendix A. Proof of Lemma 6

Assume that $x_1, x_2, ..., x_k$, are i.i.d. $C(0,d)$. The estimator, $\hat{d}_{gm,c}$, expressed as

$$\hat{d}_{gm,c} = \cos^k\left(\frac{\pi}{2k}\right)\prod_{j=1}^{k}|x_j|^{1/k},$$

is unbiased, because, from Lemma 5,

$$\mathrm{E}\left(\hat{d}_{gm,c}\right) = \cos^k\left(\frac{\pi}{2k}\right)\prod_{j=1}^{k}\mathrm{E}\left(|x_j|^{1/k}\right) = \cos^k\left(\frac{\pi}{2k}\right)\prod_{j=1}^{k}\left(\frac{d^{1/k}}{\cos\left(\frac{\pi}{2k}\right)}\right) = d.$$

The variance is

$$\begin{aligned}\mathrm{Var}\left(\hat{d}_{gm,c}\right) &= \cos^{2k}\left(\frac{\pi}{2k}\right)\prod_{j=1}^{k}\mathrm{E}\left(|x_j|^{2/k}\right) - d^2\\ &= d^2\left(\frac{\cos^{2k}\left(\frac{\pi}{2k}\right)}{\cos^k\left(\frac{\pi}{k}\right)} - 1\right) = \frac{\pi^2}{4}\frac{d^2}{k} + \frac{\pi^4}{32}\frac{d^2}{k^2} + O\left(\frac{1}{k^3}\right),\end{aligned}$$

because

$$\begin{aligned}\frac{\cos^{2k}\left(\frac{\pi}{2k}\right)}{\cos^k\left(\frac{\pi}{k}\right)} &= \left(\frac{1}{2} + \frac{1}{2}\left(\frac{1}{\cos(\pi/k)}\right)\right)^k\\ &= \left(1 + \frac{1}{4}\frac{\pi^2}{k^2} + \frac{5}{48}\frac{\pi^4}{k^4} + O\left(\frac{1}{k^6}\right)\right)^k\\ &= 1 + k\left(\frac{1}{4}\frac{\pi^2}{k^2} + \frac{5}{48}\frac{\pi^4}{k^4}\right) + \frac{k(k-1)}{2}\left(\frac{1}{4}\frac{\pi^2}{k^2} + \frac{5}{48}\frac{\pi^4}{k^4}\right)^2 + ...\\ &= 1 + \frac{\pi^2}{4}\frac{1}{k} + \frac{\pi^4}{32}\frac{1}{k^2} + O\left(\frac{1}{k^3}\right).\end{aligned}$$

Some more algebra can similarly show the third and fourth central moments:

$$\begin{aligned}\mathrm{E}\left(\hat{d}_{gm,c} - \mathrm{E}\left(\hat{d}_{gm,c}\right)\right)^3 &= \frac{3\pi^4}{16}\frac{d^3}{k^2} + O\left(\frac{1}{k^3}\right),\\ \mathrm{E}\left(\hat{d}_{gm,c} - \mathrm{E}\left(\hat{d}_{gm,c}\right)\right)^4 &= \frac{3\pi^4}{16}\frac{d^4}{k^2} + O\left(\frac{1}{k^3}\right).\end{aligned}$$

Appendix B. Proof of Lemma 7

We will use the Markov moment bound, because $\hat{d}_{gm,c}$ does not have a moment generating function ($\mathrm{E}\left(\hat{d}_{gm,c}\right)^t = \infty$ if $t \geq k$). In fact, even when the Chernoff bound is applicable, for any positive random variable, the Markov moment bound is always sharper than the Chernoff bound (Philips and Nelson, 1995; Lugosi, 2004).

By the Markov moment bound, for any $\varepsilon > 0$ and $0 < t < k$,

$$\mathbf{Pr}\left(\hat{d}_{gm} \geq (1+\varepsilon)d\right) \leq \frac{\mathrm{E}\left(\hat{d}_{gm}\right)^t}{((1+\varepsilon)d)^t} = \frac{1}{\cos^k\left(\frac{\pi t}{2k}\right)(1+\varepsilon)^t},$$

whose minimum is attained at $t = k\frac{2}{\pi}\tan^{-1}\left(\frac{2}{\pi}\log(1+\varepsilon)\right)$. Thus

$$\begin{aligned}
&\mathbf{Pr}\left(\hat{d}_{gm} \geq (1+\varepsilon)d\right) \\
\leq &\exp\left(-k\left(\log\left(\cos\left(\tan^{-1}\left(\frac{2}{\pi}\log(1+\varepsilon)\right)\right)\right) + \frac{2}{\pi}\tan^{-1}\left(\frac{2}{\pi}\log(1+\varepsilon)\right)\log(1+\varepsilon)\right)\right) \\
= &\exp\left(-k\left(-\frac{1}{2}\log\left(1+\left(\frac{2}{\pi}\log(1+\varepsilon)\right)^2\right) + \frac{2}{\pi}\tan^{-1}\left(\frac{2}{\pi}\log(1+\varepsilon)\right)\log(1+\varepsilon)\right)\right) \\
= &\exp\left(-k\frac{\varepsilon^2}{G_{R,gm}}\right),
\end{aligned}$$

where

$$G_{R,gm} = \frac{\varepsilon^2}{\left(-\frac{1}{2}\log\left(1+\left(\frac{2}{\pi}\log(1+\varepsilon)\right)^2\right) + \frac{2}{\pi}\tan^{-1}\left(\frac{2}{\pi}\log(1+\varepsilon)\right)\log(1+\varepsilon)\right)}.$$

Again, by the Markov moment bound, for any $0 < \varepsilon < 1$,

$$\mathbf{Pr}\left(\hat{d}_{gm} \leq (1-\varepsilon)d\right) = \mathbf{Pr}\left(\frac{1}{\hat{d}_{gm}} \geq \frac{1}{(1-\varepsilon)d}\right) \leq \frac{\mathrm{E}\left(\hat{d}_{gm}\right)^{-t}}{((1-\varepsilon)d)^{-t}} = \frac{(1-\varepsilon)^t}{\cos^k\left(\frac{\pi t}{2k}\right)},$$

whose minimum is attained at $t = -k\frac{2}{\pi}\tan^{-1}\left(\frac{2}{\pi}\log(1-\varepsilon)\right)$. Thus

$$\begin{aligned}
&\mathbf{Pr}\left(\hat{d}_{gm} \leq (1-\varepsilon)d\right) \\
\leq &\exp\left(-k\left(\log\left(\cos\left(\tan^{-1}\left(\frac{2}{\pi}\log(1-\varepsilon)\right)\right)\right) + \frac{2}{\pi}\tan^{-1}\left(\frac{2}{\pi}\log(1-\varepsilon)\right)\log(1-\varepsilon)\right)\right) \\
= &\exp\left(-k\left(-\frac{1}{2}\log\left(1+\left(\frac{2}{\pi}\log(1-\varepsilon)\right)^2\right) + \frac{2}{\pi}\tan^{-1}\left(\frac{2}{\pi}\log(1-\varepsilon)\right)\log(1-\varepsilon)\right)\right) \\
= &\exp\left(-k\frac{\varepsilon^2}{G_{L,gm}}\right),
\end{aligned}$$

where

$$G_{L,gm} = \frac{\varepsilon^2}{\left(-\frac{1}{2}\log\left(1+\left(\frac{2}{\pi}\log(1-\varepsilon)\right)^2\right) + \frac{2}{\pi}\tan^{-1}\left(\frac{2}{\pi}\log(1-\varepsilon)\right)\log(1-\varepsilon)\right)}.$$

Finally, we derive convenient approximations for $G_{G,gm}$ and $G_{L,gm}$, for small ε (e.g., $\varepsilon < 1$). Recall that, for $|x| < 1$, we have

$$\begin{aligned}
\log(1+x) &= x - \frac{x^2}{2} + \frac{x^3}{3} - \frac{x^4}{4} + ... \\
\tan^{-1}(x) &= x - \frac{x^3}{3} + ...
\end{aligned}$$

Thus for small ε, we have

$$
\begin{aligned}
&G_{R,gm} \\
=&\frac{\varepsilon^2}{\left(-\frac{1}{2}\log\left(1+\left(\frac{2}{\pi}\log(1+\varepsilon)\right)^2\right)+\frac{2}{\pi}\tan^{-1}\left(\frac{2}{\pi}\log(1+\varepsilon)\right)\log(1+\varepsilon)\right)} \\
=&\frac{\varepsilon^2}{-\frac{1}{2}\left(\left(\frac{2}{\pi}\log(1+\varepsilon)\right)^2-\frac{1}{2}\left(\frac{2}{\pi}\log(1+\varepsilon)\right)^4+...\right)+\frac{2}{\pi}\left(\frac{2}{\pi}\log(1+\varepsilon)-\frac{1}{3}\left(\frac{2}{\pi}\log(1+\varepsilon)\right)^3+...\right)\log(1+\varepsilon)} \\
=&\frac{\frac{\pi^2}{2}\varepsilon^2}{\log^2(1+\varepsilon)\left(1-\frac{2}{3\pi^2}\log^2(1+\varepsilon)+...\right)}=\frac{\frac{\pi^2}{2}\varepsilon^2}{\left(\varepsilon-\frac{\varepsilon^2}{2}+\frac{\varepsilon^3}{3}+...\right)^2\left(1-\frac{2}{3\pi^2}\varepsilon^2+...\right)} \\
=&\frac{\pi^2}{2}\left(1-\frac{\varepsilon}{2}+\frac{\varepsilon^2}{3}+...\right)^{-2}\left(1-\frac{2}{3\pi^2}\varepsilon^2+...\right)^{-1} \\
=&\frac{\pi^2}{2}\left(1+\varepsilon-\frac{2}{3}\varepsilon^2+\frac{(-2)(-3)}{2}\left(-\frac{\varepsilon}{2}+\frac{\varepsilon^2}{3}+...\right)^2+...\right)\left(1+\frac{2}{3\pi^2}\varepsilon^2+...\right) \\
=&\frac{\pi^2}{2}\left(1+\varepsilon+\left(\frac{1}{12}+\frac{2}{3\pi^2}\right)\varepsilon^2+...\right).
\end{aligned}
$$

Similarly, for small ε, we have

$$
\begin{aligned}
&G_{L,gm} \\
=&\frac{\varepsilon^2}{\left(-\frac{1}{2}\log\left(1+\left(\frac{2}{\pi}\log(1-\varepsilon)\right)^2\right)+\frac{2}{\pi}\tan^{-1}\left(\frac{2}{\pi}\log(1-\varepsilon)\right)\log(1-\varepsilon)\right)} \\
=&\frac{\varepsilon^2}{-\frac{1}{2}\left(\left(\frac{2}{\pi}\log(1-\varepsilon)\right)^2-\frac{1}{2}\left(\frac{2}{\pi}\log(1-\varepsilon)\right)^4+...\right)+\frac{2}{\pi}\left(\frac{2}{\pi}\log(1-\varepsilon)-\frac{1}{3}\left(\frac{2}{\pi}\log(1-\varepsilon)\right)^3+...\right)\log(1-\varepsilon)} \\
=&\frac{\frac{\pi^2}{2}\varepsilon^2}{\log^2(1-\varepsilon)\left(1-\frac{2}{3\pi^2}\log^2(1-\varepsilon)+...\right)}=\frac{\frac{\pi^2}{2}\varepsilon^2}{\left(-\varepsilon-\frac{\varepsilon^2}{2}-\frac{\varepsilon^3}{3}+...\right)^2\left(1-\frac{2}{3\pi^2}\varepsilon^2+...\right)} \\
=&\frac{\pi^2}{2}\left(1+\frac{\varepsilon}{2}+\frac{\varepsilon^2}{3}+...\right)^{-2}\left(1-\frac{2}{3\pi^2}\varepsilon^2+...\right)^{-1} \\
=&\frac{\pi^2}{2}\left(1-\varepsilon-\frac{2}{3}\varepsilon^2+\frac{(-2)(-3)}{2}\left(\frac{\varepsilon}{2}+\frac{\varepsilon^2}{3}+...\right)^2+...\right)\left(1+\frac{2}{3\pi^2}\varepsilon^2+...\right) \\
=&\frac{\pi^2}{2}\left(1-\varepsilon+\left(\frac{1}{12}+\frac{2}{3\pi^2}\right)\varepsilon^2+...\right).
\end{aligned}
$$

Appendix C. Proof of Lemma 9

For any $\varepsilon > 0$ and $0 < t < k$, the Markov inequality says

$$
\mathbf{Pr}\left(\hat{d}_{gm,c} \geq (1+\varepsilon)d\right) \leq \frac{\mathrm{E}\left(\hat{d}_{gm,c}\right)^t}{(1+\varepsilon)^t d^t} = \frac{\cos^{kt}\left(\frac{\pi}{2k}\right)}{\cos^k\left(\frac{\pi t}{2k}\right)(1+\varepsilon)^t},
$$

which can be minimized by choosing the optimum $t = t_1^*$, where

$$t_1^* = \frac{2k}{\pi} \tan^{-1}\left(\left(\log(1+\varepsilon) - k\log\cos\left(\frac{\pi}{2k}\right)\right)\frac{2}{\pi}\right).$$

We need to make sure that $0 \le t_1^* < k$. $t_1^* \ge 0$ because $\log\cos(.) \le 0$; and $t_1^* < k$ because $\tan^{-1}(.) \le \frac{\pi}{2}$, with equality holding only when $k \to \infty$.

Now we show the other tail bound $\mathbf{Pr}\left(\hat{d}_{gm,c} \le (1-\varepsilon)d\right)$. Let $0 < t < k$.

$$\begin{aligned}\mathbf{Pr}\left(\hat{d}_{gm,c} \le (1-\varepsilon)d\right) =& \mathbf{Pr}\left(\cos\left(\frac{\pi}{2k}\right)^k \prod_{j=1}^k |x_j|^{1/k} \le (1-\varepsilon)d\right) \\ =& \mathbf{Pr}\left(\prod_{j=1}^k |x_j|^{-t/k} \ge \left(\frac{(1-\varepsilon)d}{\cos^k\left(\frac{\pi}{2k}\right)}\right)^{-t}\right) \le \left(\frac{(1-\varepsilon)}{\cos^k\left(\frac{\pi}{2k}\right)}\right)^t \frac{1}{\cos^k\left(\frac{\pi t}{2k}\right)},\end{aligned}$$

which is minimized at $t = t_2^*$

$$t_2^* = \frac{2k}{\pi} \tan^{-1}\left(\left(-\log(1-\varepsilon) + k\log\cos\left(\frac{\pi}{2k}\right)\right)\frac{2}{\pi}\right),$$

provided $k \ge \frac{\pi^2}{8\varepsilon}$, otherwise t_2^* may be less than 0. To see this, in order for $t_2^* \ge 0$, we must have

$$\log(1-\varepsilon) \le k\log\cos\left(\frac{\pi}{2k}\right), \text{ i.e., } 1-\varepsilon \le \cos^k\left(\frac{\pi}{2k}\right).$$

Because

$$\cos^k\left(\frac{\pi}{2k}\right) \ge \left(1 - \frac{1}{2}\left(\frac{\pi}{2k}\right)^2\right)^k \ge 1 - \frac{\pi^2}{8k},$$

it suffices if $1-\varepsilon \le 1 - \frac{\pi^2}{8k}$, that is, $k \ge \frac{\pi^2}{8\varepsilon}$.

Now we prove the asymptotic (as $k \to \infty$) expressions for the ratios of tail bounds, another way to compare $\hat{d}_{gm}$ and $\hat{d}_{gm,c}$.

First, we consider the right tail bounds. For large k, the optimal $t = t_1^*$ can be approximated as

$$\begin{aligned}t_1^* =& \frac{2k}{\pi}\tan^{-1}\left(\left(\log(1+\varepsilon) - k\log\cos\left(\frac{\pi}{2k}\right)\right)\frac{2}{\pi}\right) \\ \sim& \frac{2k}{\pi}\tan^{-1}\left(\log(1+\varepsilon)\frac{2}{\pi} - k\frac{2}{\pi}\log\left(1 - \frac{\pi^2}{8k^2}\right)\right) \\ \sim& \frac{2k}{\pi}\tan^{-1}\left(\log(1+\varepsilon)\frac{2}{\pi} + \frac{\pi}{4k}\right) \\ \sim& \frac{2k}{\pi}\left(\tan^{-1}\left(\log(1+\varepsilon)\frac{2}{\pi}\right) + \frac{\pi}{4k}\frac{1}{1+\left(\log(1+\varepsilon)\frac{2}{\pi}\right)^2}\right) \quad \text{(Taylor expansion)} \\ \sim& \frac{2k}{\pi}\left(\tan^{-1}\left(\log(1+\varepsilon)\frac{2}{\pi}\right)\right) + \frac{1/2}{1+\left(\log(1+\varepsilon)\frac{2}{\pi}\right)^2} \\ =& kA_1 + C_1,\end{aligned}$$

where

$$A_1 = \frac{2}{\pi}\left(\tan^{-1}\left(\log(1+\varepsilon)\frac{2}{\pi}\right)\right), \quad C_1 = \frac{1/2}{1+\left(\log(1+\varepsilon)\frac{2}{\pi}\right)^2}.$$

Note that, in the asymptotic decomposition, $t_1^* \sim kA_1 + C_1$, the term kA_1 is the optimal "t" in proving the right tail bound for $\hat{d}_{gm}$. Thus to study the asymptotic ratio of the right tail bounds, we only need to keep track of the additional terms in

$$\frac{\cos^{kt_1^*}\left(\frac{\pi}{2k}\right)}{\cos^k\left(\frac{\pi t_1^*}{2k}\right)(1+\varepsilon)^{t_1^*}}.$$

Because

$$\cos^{kt_1^*}\left(\frac{\pi}{2k}\right) \sim \left(1-\frac{\pi^2}{8k^2}\right)^{k^2A_1+kC_1} \sim \exp\left(-\frac{\pi^2}{8}A_1\right),$$

and

$$\begin{aligned}
\cos^k\left(\frac{\pi t_1^*}{2k}\right) \sim & \cos^k\left(\frac{\pi A_1}{2}+\frac{\pi C_1}{2k}\right) \\
\sim & \cos^k\left(\frac{\pi A_1}{2}\right)\left(1-\frac{\pi C_1}{2k}\tan\left(\frac{\pi}{2}A_1\right)\right)^k \quad \text{(Taylor expansion)} \\
\sim & \cos^k\left(\frac{\pi A_1}{2}\right)\exp\left(-\frac{\pi C_1}{2}\tan\left(\frac{\pi}{2}A_1\right)\right),
\end{aligned}$$

we know the ratio of the right tail bounds

$$\rho_{R,k} = \frac{\frac{\cos^{kt_1^*}\left(\frac{\pi}{2k}\right)}{\cos^k\left(\frac{\pi t_1^*}{2k}\right)(1+\varepsilon)^{t_1^*}}}{\exp\left(-k\frac{\varepsilon^2}{G_{R,gm}}\right)} \to \rho_{R,\infty} = \frac{1}{(1+\varepsilon)^{C_1}}\exp\left(-\frac{\pi^2}{8}A_1+\frac{\pi}{2}C_1\tan\left(\frac{\pi^2}{2}A_1\right)\right).$$

Next, we consider the left tail bound. First, we obtain an asymptotic decomposition of t_2^*:

$$\begin{aligned}
t_2^* = & \frac{2k}{\pi}\tan^{-1}\left(\left(-\log(1-\varepsilon)+k\log\cos\left(\frac{\pi}{2k}\right)\right)\frac{2}{\pi}\right) \\
\sim & \frac{2k}{\pi}\tan^{-1}\left(-\log(1-\varepsilon)\frac{2}{\pi}-\frac{\pi}{4k}\right) \\
\sim & \frac{2k}{\pi}\left(\tan^{-1}\left(-\log(1-\varepsilon)\frac{2}{\pi}\right)\right)-\frac{1/2}{1+\left(\log(1-\varepsilon)\frac{2}{\pi}\right)^2} \quad \text{(Taylor expansion)} \\
= & kA_2 - C_2,
\end{aligned}$$

where

$$A_2 = \frac{2}{\pi}\left(\tan^{-1}\left(-\log(1-\varepsilon)\frac{2}{\pi}\right)\right), \quad C_2 = \frac{1/2}{1+\left(\log(1-\varepsilon)\frac{2}{\pi}\right)^2}.$$

Again, in the above asymptotic decomposition, $t_1^* \sim kA_2 - C_2$, the term kA_2 is the optimal "t" in proving the left tail bound for $\hat{d}_{gm}$. Thus to study the asymptotic ratio of the left tail bounds, we only need to keep track of the additional terms in

$$\frac{(1-\varepsilon)^{t_2^*}}{\cos^{kt_2^*}\left(\frac{\pi}{2k}\right)} \frac{1}{\cos^k\left(\frac{\pi t_2^*}{2k}\right)}.$$

Because

$$\cos^{kt_2^*}\left(\frac{\pi}{2k}\right) \sim \left(1 - \frac{\pi^2}{8k^2}\right)^{k^2A_2 - kC_2} \sim \exp\left(-\frac{\pi^2}{8}A_2\right),$$

and

$$\begin{aligned}
\cos^k\left(\frac{\pi t_2^*}{2k}\right) \sim & \cos^k\left(\frac{\pi A_2}{2} - \frac{\pi C_2}{2k}\right) \\
\sim & \cos^k\left(\frac{\pi A_2}{2}\right)\left(1 + \frac{\pi C_2}{2k}\tan\left(\frac{\pi}{2}A_2\right)\right)^k \\
\sim & \cos^k\left(\frac{\pi A_2}{2}\right)\exp\left(\frac{\pi C_2}{2}\tan\left(\frac{\pi}{2}A_2\right)\right),
\end{aligned}$$

we know the ratio of the left tail bounds

$$\rho_{L,k} = \frac{\frac{(1-\varepsilon)^{t_2^*}}{\cos^{kt_2^*}\left(\frac{\pi}{2k}\right)} \frac{1}{\cos^k\left(\frac{\pi t_2^*}{2k}\right)}}{\exp\left(-k\frac{\varepsilon^2}{G_{L,gm}}\right)} \to \rho_{L,\infty} = \frac{1}{(1-\varepsilon)^{C_2}} \exp\left(\frac{\pi^2}{8}A_2 - \frac{\pi}{2}C_2\tan\left(\frac{\pi^2}{2}A_2\right)\right).$$

Appendix D. Proof of Lemma 10

Assume $x \sim C(0,d)$. The log likelihood $l(x;d)$ and its first three derivatives are

$$\begin{aligned}
l(x;d) &= \log(d) - \log(\pi) - \log(x^2+d^2), \\
l'(d) &= \frac{1}{d} - \frac{2d}{x^2+d^2}, \\
l''(d) &= -\frac{1}{d^2} - \frac{2x^2-2d^2}{(x^2+d^2)^2}, \\
l'''(d) &= \frac{2}{d^3} + \frac{4d}{(x^2+d^2)^2} + \frac{8d(x^2-d^2)}{(x^2+d^2)^3}.
\end{aligned}$$

The MLE $\hat{d}_{MLE}$ is asymptotically normal with mean d and variance $\frac{1}{k\mathrm{I}(d)}$, where $\mathrm{I}(d)$, the expected Fisher Information, is

$$\mathrm{I} = \mathrm{I}(d) = \mathrm{E}\left(-l''(d)\right) = \frac{1}{d^2} + 2\mathrm{E}\left(\frac{x^2-d^2}{(x^2+d^2)^2}\right) = \frac{1}{2d^2},$$

because

$$\begin{aligned}
\mathrm{E}\left(\frac{x^2-d^2}{(x^2+d^2)^2}\right) &= \frac{d}{\pi}\int_{-\infty}^{\infty}\frac{x^2-d^2}{(x^2+d^2)^3}dx \\
&= \frac{d}{\pi}\int_{-\pi/2}^{\pi/2}\frac{d^2(\tan^2(t)-1)}{d^6/\cos^6(t)}\frac{d}{\cos^2(t)}dt \\
&= \frac{1}{d^2\pi}\int_{-\pi/2}^{\pi/2}\cos^2(t)-2\cos^4(t)dt \\
&= \frac{1}{d^2\pi}\left(\frac{\pi}{2}-2\frac{3}{8}\pi\right) = -\frac{1}{4d^2}.
\end{aligned}$$

Therefore, we obtain

$$\mathrm{Var}\left(\hat{d}_{MLE}\right) = \frac{2d^2}{k}+O\left(\frac{1}{k^2}\right).$$

General formulas for the bias and higher moments of the MLE are available in Bartlett (1953) and Shenton and Bowman (1963). We need to evaluate the expressions in (Shenton and Bowman, 1963, 16a-16d), involving tedious algebra:

$$\begin{aligned}
\mathrm{E}\left(\hat{d}_{MLE}\right) &= d-\frac{[12]}{2k\mathrm{I}^2}+O\left(\frac{1}{k^2}\right) \\
\mathrm{Var}\left(\hat{d}_{MLE}\right) &= \frac{1}{k\mathrm{I}}+\frac{1}{k^2}\left(-\frac{1}{\mathrm{I}}+\frac{[1^4]-[1^22]-[13]}{\mathrm{I}^3}+\frac{3.5[12]^2-[1^3]^2}{\mathrm{I}^4}\right)+O\left(\frac{1}{k^3}\right) \\
\mathrm{E}\left(\hat{d}_{MLE}-\mathrm{E}\left(\hat{d}_{MLE}\right)\right)^3 &= \frac{[1^3]-3[12]}{k^2\mathrm{I}^3}+O\left(\frac{1}{k^3}\right) \\
\mathrm{E}\left(\hat{d}_{MLE}-\mathrm{E}\left(\hat{d}_{MLE}\right)\right)^4 &= \frac{3}{k^2\mathrm{I}^2}+\frac{1}{k^3}\left(-\frac{9}{\mathrm{I}^2}+\frac{7[1^4]-6[1^22]-10[13]}{\mathrm{I}^4}\right) \\
&\quad +\frac{1}{k^3}\left(\frac{-6[1^3]^2-12[1^3][12]+45[12]^2}{\mathrm{I}^5}\right)+O\left(\frac{1}{k^4}\right),
\end{aligned}$$

where, after re-formatting,

$$\begin{aligned}
&[12]=\mathrm{E}(l')^3+\mathrm{E}(l'l''), \qquad [1^4]=\mathrm{E}(l')^4, \qquad [1^22]=\mathrm{E}(l''(l')^2)+\mathrm{E}(l')^4, \\
&[13]=\mathrm{E}(l')^4+3\mathrm{E}(l''(l')^2)+\mathrm{E}(l'l'''), \qquad [1^3]=\mathrm{E}(l')^3.
\end{aligned}$$

We will neglect most of the algebra. To help readers verifying the results, the following formula we derive may be useful:

$$\mathrm{E}\left(\frac{1}{x^2+d^2}\right)^m = \frac{1\times 3\times 5\times ...\times(2m-1)}{2\times 4\times 6\times ...\times(2m)}\frac{1}{d^{2m}}, \quad m=1,2,3,...$$

Without giving the detail, we report

$$\begin{aligned}
&\mathrm{E}\left(l'\right)^3=0, \qquad \mathrm{E}\left(l'l''\right)=-\frac{1}{2}\frac{1}{d^3}, \qquad \mathrm{E}\left(l'\right)^4=\frac{3}{8}\frac{1}{d^4}, \\
&\mathrm{E}(l''(l')^2)=-\frac{1}{8}\frac{1}{d^4}, \qquad \mathrm{E}\left(l'l'''\right)=\frac{3}{4}\frac{1}{d^4}.
\end{aligned}$$

Hence

$$[12] = -\frac{1}{2}\frac{1}{d^3}, \qquad [1^4] = \frac{3}{8}\frac{1}{d^4}, \qquad [1^2 2] = \frac{1}{4}\frac{1}{d^4}, \qquad [13] = \frac{3}{4}\frac{1}{d^4}, \qquad [1^3] = 0.$$

Thus, we obtain

$$\begin{aligned}
&\mathrm{E}\left(\hat{d}_{MLE}\right) = d + \frac{d}{k} + O\left(\frac{1}{k^2}\right) \\
&\mathrm{Var}\left(\hat{d}_{MLE}\right) = \frac{2d^2}{k} + \frac{7d^2}{k^2} + O\left(\frac{1}{k^3}\right) \\
&\mathrm{E}\left(\hat{d}_{MLE} - \mathrm{E}\left(\hat{d}_{MLE}\right)\right)^3 = \frac{12d^3}{k^2} + O\left(\frac{1}{k^3}\right) \\
&\mathrm{E}\left(\hat{d}_{MLE} - \mathrm{E}\left(\hat{d}_{MLE}\right)\right)^4 = \frac{12d^4}{k^2} + \frac{222d^4}{k^3} + O\left(\frac{1}{k^4}\right).
\end{aligned}$$

Because $\hat{d}_{MLE}$ has $O\left(\frac{1}{k}\right)$ bias, we recommend the bias-corrected estimator

$$\hat{d}_{MLE,c} = \hat{d}_{MLE}\left(1 - \frac{1}{k}\right),$$

whose first four moments are, after some algebra,

$$\begin{aligned}
&\mathrm{E}\left(\hat{d}_{MLE,c}\right) = d + O\left(\frac{1}{k^2}\right) \\
&\mathrm{Var}\left(\hat{d}_{MLE,c}\right) = \frac{2d^2}{k} + \frac{3d^2}{k^2} + O\left(\frac{1}{k^3}\right) \\
&\mathrm{E}\left(\hat{d}_{MLE,c} - \mathrm{E}\left(\hat{d}_{MLE,c}\right)\right)^3 = \frac{12d^3}{k^2} + O\left(\frac{1}{k^3}\right) \\
&\mathrm{E}\left(\hat{d}_{MLE,c} - \mathrm{E}\left(\hat{d}_{MLE,c}\right)\right)^4 = \frac{12d^4}{k^2} + \frac{186d^4}{k^3} + O\left(\frac{1}{k^4}\right).
\end{aligned}$$

References

Dimitris Achlioptas. Database-friendly random projections: Johnson-Lindenstrauss with binary coins. *Journal of Computer and System Sciences*, 66(4):671–687, 2003.

Dimitris Achlioptas. Database-friendly random projections. In *PODS*, pages 274–281, Santa Barbara, CA, 2001.

Charu C. Aggarwal and Joel L. Wolf. A new method for similarity indexing of market basket data. In *SIGMOD*, pages 407–418, Philadelphia, PA, 1999.

Nir Ailon and Bernard Chazelle. Approximate nearest neighbors and the fast Johnson-Lindenstrauss transform. In *STOC*, pages 557–563, Seattle, WA, 2006.

Charles Antle and Lee Bain. A property of maximum likelihood estimators of location and scale parameters. *SIAM Review*, 11(2):251–253, 1969.

Rosa Arriaga and Santosh Vempala. An algorithmic theory of learning: Robust concepts and random projection. *Machine Learning*, 63(2):161–182, 2006.

Rosa Arriaga and Santosh Vempala. An algorithmic theory of learning: Robust concepts and random projection. In *FOCS*, pages 616–623, New York, 1999.

Brian Babcock, Shivnath Babu, Mayur Datar, Rajeev Motwani, and Jennifer Widom. Models and issues in data stream systems. In *PODS*, pages 1–16, Madison, WI, 2002.

Maurice S. Bartlett. Approximate confidence intervals, II. *Biometrika*, 40(3/4):306–317, 1953.

Rabi N. Bhattacharya and Jayanta K. Ghosh. On the validity of the formal edgeworth expansion. *The Annals of Statistics*, 6(2):434–451, 1978.

Bo Brinkman and Mose Charikar. On the impossibility of dimension reduction in l_1. *Journal of ACM*, 52(2):766–788, 2005.

Bo Brinkman and Mose Charikar. On the impossibility of dimension reduction in l_1. In *FOCS*, pages 514–523, Cambridge, MA, 2003.

Olivier Chapelle, Patrick Haffner, and Vladimir N. Vapnik. Support vector machines for histogram-based image classification. *IEEE Trans. Neural Networks*, 10(5):1055–1064, 1999.

Herman Chernoff. A measure of asymptotic efficiency for tests of a hypothesis based on the sum of observations. *The Annals of Mathematical Statistics*, 23(4):493–507, 1952.

Raj S. Chhikara and J. Leroy Folks. *The Inverse Gaussian Distribution: Theory, Methodology, and Applications*. Marcel Dekker, Inc, New York, 1989.

Graham Cormode and S. Muthukrishnan. Estimating dominance norms of multiple data streams. In *ESA*, pages 148–160, 2003.

Graham Cormode, Mayur Datar, Piotr Indyk, and S. Muthukrishnan. Comparing data streams using hamming norms (how to zero in). In *VLDB*, pages 335–345, Hong Kong, China, 2002.

Graham Cormode, Mayur Datar, Piotr Indyk, and S. Muthukrishnan. Comparing data streams using hamming norms (how to zero in). *IEEE Transactions on Knowledge and Data Engineering*, 15 (3):529–540, 2003.

Francisco Jose De. A. Cysneiros, Sylvio Jose P. dos Santos, and Gass M. Cordeiro. Skewness and kurtosis for maximum likelihood estimator in one-parameter exponential family models. *Brazilian Journal of Probability and Statistics*, 15(1):85–105, 2001.

Sanjoy Dasgupta and Anupam Gupta. An elementary proof of a theorem of Johnson and Lindenstrauss. *Random Structures and Algorithms*, 22(1):60 – 65, 2003.

Herbert A. David. *Order Statistics*. John Wiley & Sons, Inc., New York, NY, second edition, 1981.

Inderjit S. Dhillon and Dharmendra S. Modha. Concept decompositions for large sparse text data using clustering. *Machine Learning*, 42(1-2):143–175, 2001.

Bradley Efron, Trevor Hastie, Iain Johnstone, and Robert Tibshirani. Least angle regression. *The Annals of Statistics*, 32(2):407–499, 2004.

Eugene F. Fama and Richard Roll. Some properties of symmetric stable distributions. *Journal of the American Statistical Association*, 63(323):817–836, 1968.

Eugene F. Fama and Richard Roll. Parameter estimates for symmetric stable distributions. *Journal of the American Statistical Association*, 66(334):331–338, 1971.

Joan Feigenbaum, Sampath Kannan, Martin Strauss, and Mahesh Viswanathan. An approximate l_1-difference algorithm for massive data streams. In *FOCS*, pages 501–511, New York, 1999.

William Feller. *An Introduction to Probability Theory and Its Applications (Volume II)*. John Wiley & Sons, New York, NY, second edition, 1971.

Silvia L. P. Ferrari, Denise A. Botter, Gauss M. Cordeiro, and Francisco Cribari-Neto. Second and third order bias reduction for one-parameter family models. *Stat. and Prob. Letters*, 30:339–345, 1996.

Ronald A. Fisher. Two new properties of mathematical likelihood. *Proceedings of the Royal Society of London*, 144(852):285–307, 1934.

Peter Frankl and Hiroshi Maehara. The Johnson-Lindenstrauss lemma and the sphericity of some graphs. *Journal of Combinatorial Theory A*, 44(3):355–362, 1987.

Jerome H. Friedman. Greedy function approximation: A gradient boosting machine. *The Annals of Statistics*, 29(5):1189–1232, 2001.

Hans U. Gerber. From the generalized gamma to the generalized negative binomial distribution. *Insurance:Mathematics and Economics*, 10(4):303–309, 1991.

Izrail S. Gradshteyn and Iosif M. Ryzhik. *Table of Integrals, Series, and Products*. Academic Press, New York, fifth edition, 1994.

Gerald Haas, Lee Bain, and Charles Antle. Inferences for the Cauchy distribution based on maximum likelihood estimation. *Biometrika*, 57(2):403–408, 1970.

Monika R. Henzinger, Prabhakar Raghavan, and Sridhar Rajagopalan. *Computing on Data Streams*. American Mathematical Society, Boston, MA, USA, 1999.

David V. Hinkley. Likelihood inference about location and scale parameters. *Biometrika*, 65(2): 253–261, 1978.

Philip Hougaard. Survival models for heterogeneous populations derived from stable distributions. *Biometrika*, 73(2):387–396, 1986.

Peter J. Huber. *Robust Statistics*. Wiley, New York, NY, 1981.

Piotr Indyk. Stable distributions, pseudorandom generators, embeddings, and data stream computation. *Journal of ACM*, 53(3):307–323, 2006.

Piotr Indyk. Stable distributions, pseudorandom generators, embeddings and data stream computation. In *FOCS*, pages 189–197, Redondo Beach, CA, 2000.

Piotr Indyk. Algorithmic applications of low-distortion geometric embeddings. In *FOCS*, pages 10–33, Las Vegas, NV, 2001.

Piotr Indyk and Rajeev Motwani. Approximate nearest neighbors: Towards removing the curse of dimensionality. In *STOC*, pages 604–613, Dallas, TX, 1998.

Jens Ledet Jensen. *Saddlepoint Approximations*. Oxford University Press, New York, 1995.

William B. Johnson and Joram Lindenstrauss. Extensions of Lipschitz mapping into Hilbert space. *Contemporary Mathematics*, 26:189–206, 1984.

William B. Johnson and Gideon Schechtman. Embedding l_p into l_1. *Acta. Math.*, 149:71–85, 1982.

Jerry F. Lawless. Conditional confidence interval procedures for the location and scale parameters of the Cauchy and logistic distributions. *Biometrika*, 59(2):377–386, 1972.

James R. Lee and Assaf Naor. Embedding the diamond graph in l_p and dimension reduction in l_1. *Geometric And Functional Analysis*, 14(4):745–747, 2004.

Ping Li. Very sparse stable random projections for dimension reduction in l_α $(0 < \alpha \leq 2)$ norm. In *KDD*, San Jose, CA, 2007.

Ping Li. Estimators and tail bounds for dimension reduction in l_α $(0 < \alpha \leq 2)$ using stable random projections. In *SODA*, 2008.

Ping Li and Kenneth W. Church. A sketch algorithm for estimating two-way and multi-way associations. *Computational Linguistics*, 33(3):305–354, 2007.

Ping Li and Kenneth W. Church. Using sketches to estimate associations. In *HLT/EMNLP*, pages 708–715, Vancouver, BC, Canada, 2005.

Ping Li, Trevor J. Hastie, and Kenneth W. Church. Improving random projections using marginal information. In *COLT*, pages 635–649, Pittsburgh, PA, 2006a.

Ping Li, Trevor J. Hastie, and Kenneth W. Church. Very sparse random projections. In *KDD*, pages 287–296, Philadelphia, PA, 2006b.

Ping Li, Debashis Paul, Ravi Narasimhan, and John Cioffi. On the distribution of SINR for the MMSE MIMO receiver and performance analysis. *IEEE Trans. Inform. Theory*, 52(1):271–286, 2006c.

Ping Li, Kenneth W. Church, and Trevor J. Hastie. Conditional random sampling: A sketch-based sampling technique for sparse data. In *NIPS*, pages 873–880, Vancouver, BC, Canada, 2007a.

Ping Li, Trevor J. Hastie, and Kenneth W. Church. Nonlinear estimators and tail bounds for dimensional reduction in l_1 using Cauchy random projections. In *COLT*, 2007b.

Gabor Lugosi. Concentration-of-measure inequalities. *Lecture Notes*, 2004.

J. Huston McCulloch. Simple consistent estimators of stable distribution parameters. *Communications on Statistics-Simulation*, 15(4):1109–1136, 1986.

Thomas K. Philips and Randolph Nelson. The moment bound is tighter than Chernoff's bound for positive tail probabilities. *The American Statistician*, 49(2):175–178, 1995.

V. Seshadri. *The Inverse Gaussian Distribution: A Case Study in Exponential Families*. Oxford University Press Inc., New York, 1993.

Thomas A. Severini. *Likelihood Methods in Statistics*. Oxford University Press, New York, 2000.

Gregory Shakhnarovich, Trevor Darrell, and Piotr Indyk, editors. *Nearest-Neighbor Methods in Learning and Vision, Theory and Practice*. The MIT Press, Cambridge, MA, 2005.

Leonard. R. Shenton and Kimiko O. Bowman. Higher moments of a maximum-likelihood estimate. *Journal of Royal Statistical Society B*, 25(2):305–317, 1963.

Alexander Strehl and Joydeep Ghosh. A scalable approach to balanced, high-dimensional clustering of market-baskets. In *HiPC*, pages 525–536, Bangalore, India, 2000.

Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of Royal Statistical Society B*, 58(1):267–288, 1996.

Maurice C. K. Tweedie. Statistical properties of inverse Gaussian distributions. I. *The Annals of Mathematical Statistics*, 28(2):362–377, 1957a.

Maurice C. K. Tweedie. Statistical properties of inverse Gaussian distributions. II. *The Annals of Mathematical Statistics*, 28(3):696–705, 1957b.

Santosh Vempala. *The Random Projection Method*. American Mathematical Society, Providence, RI, 2004.

Ji Zhu, Saharon Rosset, Trevor Hastie, and Robert Tibshirani. 1-norm support vector machines. In *NIPS*, Vancouver, BC, Canada, 2003.

Vladimir M. Zolotarev. *One-dimensional Stable Distributions*. American Mathematical Society, Providence, RI, 1986.

Journal of Machine Learning Research 8 (2007) 2533-2549 Submitted 3/06; Revised 5/07; Published 11/07

Revised Loss Bounds for the Set Covering Machine and Sample-Compression Loss Bounds for Imbalanced Data

Zakria Hussain Z.HUSSAIN@CS.UCL.AC.UK
Centre for Computational Statistics and Machine Learning
University College London
London, UK, WC1E 6BT

François Laviolette FRANCOIS.LAVIOLETTE@IFT.ULAVAL.CA
Mario Marchand MARIO.MARCHAND@IFT.ULAVAL.CA
Département IFT-GLO
Université Laval
Québec, Canada, G1V 0A6

John Shawe-Taylor JST@CS.UCL.AC.UK
Centre for Computational Statistics and Machine Learning
University College London
London, UK, WC1E 6BT

Spencer Charles Brubaker BRUBAKER@CC.GATECH.EDU
Matthew D. Mullin MDMULLIN@CC.GATECH.EDU
College of Computing
Georgia Institute of Technology
Atlanta, Georgia, 30332

Editor: Gábor Lugosi

Abstract

Marchand and Shawe-Taylor (2002) have proposed a loss bound for the set covering machine that has the property to depend on the observed fraction of positive examples and on what the classifier achieves on the positive training examples. We show that this loss bound is incorrect. We then propose a loss bound, valid for any sample-compression learning algorithm (including the set covering machine), that depends on the observed fraction of positive examples and on what the classifier achieves on them. We also compare numerically the loss bound proposed in this paper with the incorrect bound, the original SCM bound and a recently proposed loss bound of Marchand and Sokolova (2005) (which does not depend on the observed fraction of positive examples) and show that the latter loss bounds can be substantially larger than the new bound in the presence of imbalanced misclassifications.

Keywords: set covering machines, sample-compression, loss bounds

1. Introduction

One of the key objectives of learning theory is to identify classes of functions and associated learning algorithms that deliver hypotheses with good guarantees of test set performance for a range of practical applications. Support vector machines (SVMs) have achieved this objective for problems for which classifiers with large margins can be identified. An alternative guiding principle for

the selection of classifiers with guarantees of good generalization is to require some type of parsimony in the form of the functions. Typically seeking parsimonious solutions reduces to an NP-hard optimization, but an algorithm that delivers a good approximation to the optimal solution using a greedy approach is the so-called *set covering machine (SCM)*. This approach for producing very sparse classifiers having good generalization was proposed by Marchand and Shawe-Taylor (2001).

A *generalization error bound* is an upper bound on the expected test set performance that holds with high probability over the (random) choice of the training set. There are three ways in which such a bound can assist a user of adaptive systems technology. Firstly, the existence of such a bound justifying the form of a learning algorithm gives confidence in its reliability. This is the primary role of SVM bounds in most applications. The second is to guide model selection through setting regularization and hyperparameters to optimize the bound. The third is to give to users as a measure of performance. Experiments with recent SVM bounds have begun to make progress with the second goal, but it is fair to say that the third goal is still to be realized.

The situation with SCM bounds is that they are typically tighter and more reliable than those derived for SVMs. The classifier output by the SCM is described by a small subset of the training data called the *compression set*. By adapting the pioneering work of Littlestone and Warmuth (1986) on sample-compression schemes, Marchand and Shawe-Taylor (2001) have been able to obtain a loss bound that depends on the size (i.e., the number of examples) of the compression set of the SCM classifier. More recently, Marchand and Sokolova (2005) have been able to obtain a tighter bound, which applies to any sample-compression learning algorithm (including the SCM), by making use of sample-compression-dependent sets of messages. None of these loss bounds, however, depend on the observed fraction of positive examples in the training set and on the fraction of positive examples used for the compression set of the final classifier. Consequently, these loss bounds are not appropriate for identifying classifiers that perform well under frequently encountered distributions where the examples of one class are much more abundant than the examples of the other class (the class imbalance case) or when the loss suffered by misclassifying a positive example differs greatly from the loss suffered by misclassifying a negative example (the asymmetrical loss case).

To obtain a loss bound that reflects more accurately the performance of classifiers trained on imbalanced data sets, Marchand and Shawe-Taylor (2002) have proposed a SCM loss bound that depends on the observed fraction of positive examples in the training set and on the fraction of positive examples used for the compression set of the final classifier. However, we will show in Section 3 that this bound is *incorrect*. We then propose, in Section 4, a loss bound which is valid for any sample-compression learning algorithm (including the SCM) and that depends on the observed fraction of positive examples and on what the classifier achieves on the positive training examples. The proof of this new loss bound turns out to be much more involved than all other sample-compression loss bounds that do not depend on the observed fraction of positive examples (as in Marchand and Sokolova, 2005). Finally, for the SCM case, we compare numerically the loss bound of Section 4 with the recently proposed loss bound of Marchand and Sokolova (2005) (which does not depend on the observed fraction of positive examples) and show that the latter loss bound can be substantially larger than the former in the presence of imbalanced misclassifications.

The novelty of this paper is that we correct a bound that was found to be wrong, but in doing so, we derive a more general form that allows any learning algorithm, relying on sample compression schemes, to be upper bounded. Separately bounding the positive and negative errors also gives rise to a natural extension—namely asymmetric loss of sample compression risk bounds. In these cases, we give a higher weight for misclassification of one class over the other, typically because it is far

less frequent than the dominant class. An example is classification of news articles by topic, where classification of all documents as *not* relevant will give good performance if we do not impose a greater cost on the misclassification of a relevant document.

We begin our discussion with preliminary definitions and terminology that will be used throughout the remainder of this paper.

2. Preliminary Definitions

Let the input space X be a set of n-dimensional vectors of $\mathbb{R}^n$ and let $\mathbf{x}$ be a member of X. We define a *feature* as an arbitrary Boolean-valued function that maps X onto $\{0,1\}$.

Consider any set $\mathcal{H} = \{h_i\}_{i=1}^{|\mathcal{H}|}$ of Boolean-valued features h_i. We will consider learning algorithms that are given any such set $\mathcal{H}$ and return a small subset $\mathcal{R} \subset \mathcal{H}$ of features. Given that subset $\mathcal{R}$, and an arbitrary input vector $\mathbf{x}$, the output $f(\mathbf{x})$ of the Set Covering Machine (SCM) is given by the conjunction

$$f(\mathbf{x}) \;=\; \bigwedge_{i \in \mathcal{R}} h_i(\mathbf{x})\,.$$

The function f contains a conjunction of features h_i that individually give outputs $h_i(\mathbf{x})$ of 0 or 1 to denote whether the input vector $\mathbf{x}$ belongs to class 0 or class 1, respectively. Therefore, the function f outputs 0 or 1 according to a conjunction of features h_i. A positive example will be referred to as a $\mathcal{P}$-example and a negative example as a $\mathcal{N}$-example. Given a training set $S = S_\mathcal{P} \cup S_\mathcal{N}$ of examples, the set of positive training examples will be denoted by $S_\mathcal{P}$ and the set of negative training examples by $S_\mathcal{N}$.

Any learning algorithm that constructs a conjunction (such as the one above) can be transformed into an algorithm constructing a disjunction just by exchanging the role of the positive and negative examples. Hence, simply by reassigning the set of negative training examples to the set of positive training examples (i.e., $S_\mathcal{P} \leftarrow S_\mathcal{N}$) and the set of positive training examples to the set of negative training examples (i.e., $S_\mathcal{N} \leftarrow S_\mathcal{P}$) we can transform the algorithm into one that constructs a disjunction. However, for the remainder of the paper we will assume, without loss of generality, that the SCM always produces a conjunction.

In this paper, we consider the case where $\mathcal{H}$ is the set of *data-dependent balls*.

Definition 1 *For each training example $\mathbf{x}_i$ with label $y_i \in \{0,1\}$ and (real-valued) radius ρ, we define feature $h_{i,\rho}$ to be the following* data-dependent ball *centered on $\mathbf{x}_i$:*

$$h_{i,\rho}(\mathbf{x}) \stackrel{\text{def}}{=} h_\rho(\mathbf{x},\mathbf{x}_i) = \begin{cases} y_i & \text{if } d(\mathbf{x},\mathbf{x}_i) < \rho \\ \bar{y}_i & \text{otherwise}\,, \end{cases}$$

where $\bar{y}_i$ denotes the Boolean complement of y_i and $d(\mathbf{x},\mathbf{x}')$ denotes the distance between $\mathbf{x}$ and $\mathbf{x}'$. Training example $\mathbf{x}_i$ will be called the ball center *of $h_{i,\rho}$.*

To determine the radius ρ of ball $h_{i,\rho}$, we will use another training example $\mathbf{x}_j$, called the ball border *of $h_{i,\rho}$, such that $\rho = d(\mathbf{x}_j,\mathbf{x}_i)$.*

Hence, the set $\mathcal{R} \subset \mathcal{H}$ of features used by the SCM gives us a set of ball centers and a set of ball borders. The union of these two sets gives the *compression set* of the SCM. Following Littlestone and Warmuth (1986) and Floyd and Warmuth (1995), the compression set is a small subset of the

training set which identifies a classifier (here a SCM). The function that maps arbitrary compression sets to classifiers is called the *reconstruction function*. We refine further these notions in Section 4.

We adopt the PAC model where it is assumed that each example $(\mathbf{x}, y)$ is drawn independently at random according to a fixed (but unknown) distribution. In this paper, we consider the probabilities of events taken separately over the $\mathcal{P}$-examples and the $\mathcal{N}$-examples. We will therefore denote by $P\{a(\mathbf{x},y)\big|(\mathbf{x},y) \in \mathcal{P}\}$ the probability that predicate a is true on a random draw of an example $(\mathbf{x},y)$, given that this example is positive. Hence, the error probability of classifier f on $\mathcal{P}$-examples and on $\mathcal{N}$-examples, that we call respectively the *expected $\mathcal{P}$-loss* and the *expected $\mathcal{N}$-loss*, are given by

$$\begin{aligned} \mathrm{er}_{\mathcal{P}}(f) &\stackrel{\text{def}}{=} P\{f(\mathbf{x}) \neq y\big|(\mathbf{x},y) \in \mathcal{P}\}, \\ \mathrm{er}_{\mathcal{N}}(f) &\stackrel{\text{def}}{=} P\{f(\mathbf{x}) \neq y\big|(\mathbf{x},y) \in \mathcal{N}\}. \end{aligned}$$

Similarly, let $\hat{\mathrm{er}}_{\mathcal{P}}(f,S)$ denote the number of examples in $S_{\mathcal{P}}$ misclassified by f and let $\hat{\mathrm{er}}_{\mathcal{N}}(f,S))$ denote the number of examples in $S_{\mathcal{N}}$ misclassified by f. Hence

$$\begin{aligned} \hat{\mathrm{er}}_{\mathcal{P}}(f,S) &\stackrel{\text{def}}{=} |\{(\mathbf{x},y) \in S_{\mathcal{P}} \colon f(\mathbf{x}) \neq y\}|, \\ \hat{\mathrm{er}}_{\mathcal{N}}(f,S) &\stackrel{\text{def}}{=} \left|\{(\mathbf{x},y) \in S_{\mathcal{N}} \colon f(\mathbf{x}) \neq y\}\right|. \end{aligned}$$

Throughout this paper, the probability of occurrence of a positive example will be denoted by $p_{\mathcal{P}}$. Similarly, $p_{\mathcal{N}}$ will denote the probability of occurrence of a negative example. We will consider the general case where the loss $l_{\mathcal{P}}$ of misclassifying a positive example can differ from the loss $l_{\mathcal{N}}$ of misclassifying a negative example. We will denote by $A(S)$ the classifier returned by the learning algorithm A trained on a set S of examples. In this case, the expected loss $\mathbb{E}[l(A(S))]$ of classifier $A(S)$ is defined as

$$\mathbb{E}[l(A(S))] \stackrel{\text{def}}{=} l_{\mathcal{P}} \cdot p_{\mathcal{P}} \cdot \mathrm{er}_{\mathcal{P}}[A(S)] + l_{\mathcal{N}} \cdot p_{\mathcal{N}} \cdot \mathrm{er}_{\mathcal{N}}[A(S)]. \tag{1}$$

3. Incorrect Bound

The Theorem 5 of Marchand and Shawe-Taylor (2002) gives the following loss bound for the SCM with the symmetric loss case of $l_{\mathcal{P}} = l_{\mathcal{N}} = 1$.

Given the above definitions, let A be any learning algorithm that builds a SCM with data-dependent balls with the constraint that the returned function $A(S)$ always correctly classifies every example in the compression set. Then, with probability $1-\delta$ over all training sets S of m examples,

$$\mathbb{E}[l(A(S))] \leq 1 - \exp\left\{ -\frac{1}{m - c_p - b - c_n - k_p - k_n} \left(\ln B + \ln \frac{1}{\delta_0} \right) \right\},$$

where

$$\begin{aligned} \delta_0 &\stackrel{\text{def}}{=} \left(\frac{\pi^2}{6}\right)^{-5} \cdot ((c_p+1)(c_n+1)(b+1)(k_p+1)(k_n+1))^{-2} \cdot \delta, \\ B &\stackrel{\text{def}}{=} \binom{m_p}{c_p} \binom{m_p - c_p}{b} \binom{m_n}{c_n} \binom{m_p - c_p - b}{k_p} \binom{m_n - c_n}{k_n}, \end{aligned}$$

and where k_p and k_n are the number of misclassified positive and negative training examples by classifier $A(S)$. Similarly, c_p and c_n are the number of positive and negative ball centers contained in classifier $A(S)$ whereas b denotes the number of ball borders[1] *in classifier $A(S)$. Finally m_p and m_n denote the number of positive and negative examples in training set S.*

Let us take the B expression only and look more closely at the number of ways of choosing the errors on $S_{\mathcal{P}}$ and $S_{\mathcal{N}}$:

$$\binom{m_p - c_p - b}{k_p}\binom{m_n - c_n}{k_n}.$$

The bound on the expected loss given above will be small only if each factor is small. However, each factor can be small for a small number of training errors (desirable) or a large number of training errors (undesirable). In particular, the product of these two factors will be small for a small value of k_n (say, $k_n = 0$) and a large value of k_p (say, $k_p = m_p - c_p - b$). In this case, the denominator of the bound given above will become

$$m - c_p - b - c_n - k_p - k_n = m_n - c_n\,,$$

and will be large whenever $m_n \gg c_n$. Consequently, the bound given by Theorem 5 of Marchand and Shawe-Taylor (2002) will be small for classifiers having a small compression set and making a large number of errors on $S_{\mathcal{P}}$ and a small number of errors on $S_{\mathcal{N}}$. Clearly, this is incorrect as it implies a classifier with good generalization ability and so exposes an error in the proof. In order to derive a loss bound where the issue of imbalanced misclassifications can be handled, the errors for positive and negative examples must be bounded separately.

The error in the proof of Theorem 5 of Marchand and Shawe-Taylor (2002) occurs at the first equality used in their Equation 3. This equality is tantamount to writing that for any fixed classifier f:

$$P\left\{S \in \mathcal{X}:\ \hat{\mathrm{er}}(f,S) = 0 \;\middle|\; |S_{\mathcal{P}}| = m_p\right\}$$
$$= (1 - \mathrm{er}_{\mathcal{P}}(f))^{m_p}(1 - \mathrm{er}_{\mathcal{N}}(f))^{m_n} \times \binom{m}{m_p} p_{\mathcal{P}}^{m_p}(1 - p_{\mathcal{P}})^{m_n} \quad \text{(false)},$$

where $p_{\mathcal{P}}$ denotes the probability of occurrence of a $\mathcal{P}$-example. However this last equation is *false* since the probability on the left hand side is conditioned on the fact the $|S_{\mathcal{P}}| = m_p$. Hence, we have instead

$$P\left\{S \in \mathcal{X}:\ \hat{\mathrm{er}}(f,S) = 0 \;\middle|\; |S_{\mathcal{P}}| = m_p\right\} = (1 - \mathrm{er}_{\mathcal{P}})^{m_p}(1 - \mathrm{er}_{\mathcal{N}})^{m_n}.$$

4. Sample-Compression Loss Bounds for Imbalanced Data

Recall that X denotes the input space. Let $\mathcal{X} = (X \times \{0,1\})^m$ be the set of training sets of size m with inputs from X. We consider any learning algorithm A having the property that, when trained on a training set $S \in \mathcal{X}$, A produces a classifier $A(S)$ which can be identified solely by a subset $\Lambda = \{\Lambda_{\mathcal{P}} \cup \Lambda_{\mathcal{N}}\} \subset S$, called the *compression set*, and a *message string* σ that represents some additional information required to obtain a classifier. Here $\Lambda_{\mathcal{P}}$ represents a subset of positive examples and $\Lambda_{\mathcal{N}}$

1. As explained in Marchand and Shawe-Taylor (2002), the ball borders are always positive examples.

a subset of negative examples. More formally, this means that there exists a *reconstruction function* Φ that produces a classifier $f = \Phi(\Lambda, \sigma)$ when given an arbitrary compression set Λ and message string σ. We can thus consider that the learning algorithm A, trained on S, returns a compression set $\Lambda(S)$ and a message string $\sigma(S)$. The classifier is then given by $\Phi(\Lambda(S), \sigma(S))$.

For any training sample S and compression set Λ, consisting of a subset $\Lambda_{\mathcal{P}}$ of positive examples and a subset $\Lambda_{\mathcal{N}}$ of negative examples, we use the notation $\Lambda(S) = (\Lambda_{\mathcal{P}}(S), \Lambda_{\mathcal{N}}(S))$. Any further partitioning of the compression set Λ can be performed by the message string σ. For example, in the set covering machine, σ specifies for each point in $\Lambda_{\mathcal{P}}$, whether it is a ball center or a ball border (not already used as a center). As explained by Marchand and Shawe-Taylor (2002), this is the only additional information required to obtain a SCM consistent with the compression set.

We will use d_p to denote the number of examples present in $\Lambda_{\mathcal{P}}$. Similarly, d_n will denote the number of examples present in $\Lambda_{\mathcal{N}}$. To simplify the notation, we will use the $\mathbf{m}_{\mathcal{P}}$ and $\mathbf{m}_{\mathcal{N}}$ vectors defined as

$$\begin{aligned} \mathbf{m}_{\mathcal{P}} &\stackrel{\text{def}}{=} (m, m_p, m_n, d_p, d_n, k_p)\,, \\ \mathbf{m}_{\mathcal{N}} &\stackrel{\text{def}}{=} (m, m_p, m_n, d_p, d_n, k_n)\,, \end{aligned} \tag{2}$$

and

$$\mathbf{m}_{\mathcal{P}}(S, A(S)) \stackrel{\text{def}}{=} \left(|S|, |S_{\mathcal{P}}|, |S_{\mathcal{N}}|, |\Lambda_{\mathcal{P}}(S)|, |\Lambda_{\mathcal{N}}(S)|, \hat{\text{er}}_{\mathcal{P}}(A(S), S)\right)\,, \tag{3}$$

$$\mathbf{m}_{\mathcal{N}}(S, A(S)) \stackrel{\text{def}}{=} \left(|S|, |S_{\mathcal{P}}|, |S_{\mathcal{N}}|, |\Lambda_{\mathcal{P}}(S)|, |\Lambda_{\mathcal{N}}(S)|, \hat{\text{er}}_{\mathcal{N}}(A(S), S)\right)\,. \tag{4}$$

Hence, the predicate $\mathbf{m}_{\mathcal{P}}(S, A(S)) = \mathbf{m}_{\mathcal{P}}$ means that $|S| = m$, $|S_{\mathcal{P}}| = m_p$, $|S_{\mathcal{N}}| = m_n$, $|\Lambda_{\mathcal{P}}(S)| = d_p$, $|\Lambda_{\mathcal{N}}(S)| = d_n$, $\hat{\text{er}}_{\mathcal{P}}(A(S), S) = k_p$. We use a similar definition for predicate $\mathbf{m}_{\mathcal{N}}(S, A(S)) = \mathbf{m}_{\mathcal{N}}$. We will also use $B_{\mathcal{P}}(\mathbf{m}_{\mathcal{P}})$ and $B_{\mathcal{N}}(\mathbf{m}_{\mathcal{N}})$ defined as

$$\begin{aligned} B_{\mathcal{P}}(\mathbf{m}_{\mathcal{P}}) &\stackrel{\text{def}}{=} \binom{m_p}{d_p}\binom{m_n}{d_n}\binom{m_p - d_p}{k_p}\,, \\ B_{\mathcal{N}}(\mathbf{m}_{\mathcal{N}}) &\stackrel{\text{def}}{=} \binom{m_p}{d_p}\binom{m_n}{d_n}\binom{m_n - d_n}{k_n}\,. \end{aligned}$$

The proposed loss bound will hold uniformly for all possible messages that can be chosen by A. It will thus loosen as we increase the set $\mathcal{M}$ of possible messages that can be used. To obtain a smaller loss bound, we will therefore permit $\mathcal{M}$ to be *dependent* on the compression set chosen by A. In fact, the loss bound will depend on a *prior* distribution $P_\Lambda(\sigma)$ of message strings over the set $\mathcal{M}_\Lambda$ of possible messages that can be used with a compression set Λ. We will see that the only condition that P_Λ needs to satisfy is

$$\sum_{\sigma \in \mathcal{M}_\Lambda} P_\Lambda(\sigma) \leq 1\,.$$

Consider, for example, the case of a SCM conjunction of balls. Given a compression set $\Lambda = (\Lambda_{\mathcal{P}}, \Lambda_{\mathcal{N}})$ of size $(|\Lambda_{\mathcal{P}}|, |\Lambda_{\mathcal{N}}|) = (d_p, d_n)$, recall that each example in $\Lambda_{\mathcal{N}}$ is a ball center whereas each example in $\Lambda_{\mathcal{P}}$ can either be a ball border or a ball center. Hence, to specify a classifier given Λ, we only need to specify the examples in $\Lambda_{\mathcal{P}}$ that are ball borders.[2] This specification can be used

2. For a SCM making no error with Λ, we can pair each center with its border in the following way. For each negative center, we choose the closest border. For each positive center, we choose the furthest border.

with a message string containing two parts. The first part specifies the number $b \in \{0,\ldots,d_p\}$ of ball borders in $\Lambda_{\mathcal{P}}$. The second part specifies which subset, among the set of $\binom{d_p}{b}$ possible subsets, is used for the set of ball borders. Consequently, if $b(\sigma)$ denotes the number of ball borders specified by message string σ, we can choose

$$P_\Lambda(\sigma) \;=\; \zeta(b(\sigma)) \cdot \binom{d_p}{b(\sigma)}^{-1} \quad \text{(SCM case)}, \tag{5}$$

where, for any non-negative integer b, we define

$$\zeta(b) \stackrel{\text{def}}{=} \frac{6}{\pi^2}(b+1)^{-2}. \tag{6}$$

Indeed, in this case, we clearly satisfy

$$\sum_{\sigma \in \mathcal{M}_\Lambda} P_\Lambda(\sigma) \;=\; \sum_{b=0}^{d_p} \zeta(b) \sum_{\sigma : b(\sigma)=b} \binom{d_p}{b(\sigma)}^{-1} \;\leq\; 1\,.$$

The proposed loss bound will make use of the following functions:

$$\varepsilon_{\mathcal{P}}(\mathbf{m}_{\mathcal{P}},\delta) \;\stackrel{\text{def}}{=}\; 1-\exp\left(-\frac{1}{m_p-d_p-k_p}\left[\ln\left(B_{\mathcal{P}}(\mathbf{m}_{\mathcal{P}})\right)+\ln\frac{1}{\delta}\right]\right), \tag{7}$$

$$\varepsilon_{\mathcal{N}}(\mathbf{m}_{\mathcal{N}},\delta) \;\stackrel{\text{def}}{=}\; 1-\exp\left(-\frac{1}{m_n-d_n-k_n}\left[\ln\left(B_{\mathcal{N}}(\mathbf{m}_{\mathcal{N}})\right)+\ln\frac{1}{\delta}\right]\right). \tag{8}$$

Theorem 2 *Given the above definitions, let A be any learning algorithm having a reconstruction function that maps compression sets and message strings to classifiers. For any prior distribution P_Λ of messages and for any $\delta \in (0,1]$:*

$$P\Big\{S \in \mathcal{X} :\; \mathrm{er}_{\mathcal{P}}[A(S)] \leq \varepsilon_{\mathcal{P}}\Big(\mathbf{m}_{\mathcal{P}}(S,A(S)), g_{\mathcal{P}}(S)\delta\Big)\Big\} \;\;\geq\;\; 1-\delta\,,$$

$$P\Big\{S \in \mathcal{X} :\; \mathrm{er}_{\mathcal{N}}[A(S)] \leq \varepsilon_{\mathcal{N}}\Big(\mathbf{m}_{\mathcal{N}}(S,A(S)), g_{\mathcal{N}}(S)\delta\Big)\Big\} \;\;\geq\;\; 1-\delta\,,$$

where $\mathbf{m}_{\mathcal{P}}(S,A(S))$ and $\mathbf{m}_{\mathcal{N}}(S,A(S))$ are defined by Equation 3 and Equation 4, and

$$g_{\mathcal{P}}(S) \;\stackrel{\text{def}}{=}\; \zeta(d_p(S))\cdot\zeta(d_n(S))\cdot\zeta(k_p(S))\cdot P_{\Lambda(S)}(\sigma(S))\,, \tag{9}$$

$$g_{\mathcal{N}}(S) \;\stackrel{\text{def}}{=}\; \zeta(d_p(S))\cdot\zeta(d_n(S))\cdot\zeta(k_n(S))\cdot P_{\Lambda(S)}(\sigma(S))\,. \tag{10}$$

Note that Theorem 2 directly applies to the SCM when we use the distribution of messages given by Equation 5.

Proof To prove Theorem 2, it suffice to upper bound by δ the following probability

$$\begin{aligned} P \;&\stackrel{\text{def}}{=}\; P\Big\{S \in \mathcal{X} :\; \mathrm{er}_{\mathcal{P}}[A(S)] \geq \varepsilon\Big(\mathbf{m}_{\mathcal{P}}(S,A(S)),\Lambda(S),\sigma(S)\Big)\Big\} \\ &=\; \sum_{\mathbf{m}_{\mathcal{P}}} P\Big\{S \in \mathcal{X} :\; \mathrm{er}_{\mathcal{P}}[A(S)] \geq \varepsilon\Big(\mathbf{m}_{\mathcal{P}},\Lambda(S),\sigma(S)\Big), \mathbf{m}_{\mathcal{P}}(S,A(S)) = \mathbf{m}_{\mathcal{P}}\Big\}, \end{aligned}$$

where $\varepsilon(\mathbf{m}_{\mathcal{P}}, \Lambda(S), \sigma(S))$ denotes a risk bound on $\mathrm{er}_{\mathcal{P}}(A(S))$ that depends (partly) on the compression set $\Lambda(S)$ and the message string $\sigma(S)$ returned by $A(S)$. The summation over $\mathbf{m}_{\mathcal{P}}$ stands for

$$\sum_{\mathbf{m}_{\mathcal{P}}}(\cdot) \stackrel{\text{def}}{=} \sum_{m_p=0}^{m} \sum_{d_p=0}^{m_p} \sum_{d_n=0}^{m-m_p} \sum_{k_p=0}^{m_p-d_p} (\cdot)\,.$$

Note that the summation over k_p stops at $m_p - d_p$ because, as we will see later in the proof, we can upper bound the risk of a sample-compressed classifier only from the training errors it makes on the examples that are not used for the compression set.

We will now use the notation $\mathbf{i} = (i_1, \ldots, i_d)$ for a sequence (or a vector) of strictly increasing indices, $0 < i_1 < i_2 < \cdots < i_d \leq m$. Hence there are 2^m distinct sequences $\mathbf{i}$. We will also use $|\mathbf{i}|$ to denote the length d of a sequence $\mathbf{i}$. Such sequences (or vectors) of indices will be used to identify subsets of S. For $S \in \mathcal{X}$, we define $S_{\mathbf{i}}$ as

$$S_{\mathbf{i}} \stackrel{\text{def}}{=} ((x_{i_1}, y_{i_1}), \ldots, (x_{i_d}, y_{i_d}))\,.$$

Under the constraint that $\mathbf{m}(S, A(S)) = \mathbf{m}$, we will denote by $\mathbf{i}_p$ any sequence (or vector) of indices where each index points to an example of $S_{\mathcal{P}}$. We also use an equivalent definition for $\mathbf{i}_n$. If, for example, $\mathbf{i}_n = (2,3,6,9)$, then $S_{\mathbf{i}_n}$ will denote the set of examples consisting of the second, third, sixth, and ninth $\mathcal{N}$-example of S. Therefore, given a training set S and vectors $\mathbf{i}_p$ and $\mathbf{i}_n$, the subset $S_{\mathbf{i}_p,\mathbf{i}_n}$ will denote a compression set. We will also denote by I_{m_p} the set of all the 2^{m_p} possible vectors $\mathbf{i}_p$ under the constraint that $|S_{\mathcal{P}}| = m_p$. We also use an equivalent definition for I_{m_n}. Using these definitions, we will now upper bound P uniformly over all possible realizations of $\mathbf{i}_p$ and $\mathbf{i}_n$ under the constraint $\mathbf{m}_{\mathcal{P}}(S, A(S)) = \mathbf{m}_{\mathcal{P}}$. Thus

$$\begin{aligned}
P \;\leq\; & \sum_{\mathbf{m}_{\mathcal{P}}} P\Big\{ S \in \mathcal{X} \colon \exists \mathbf{i}_p \in I_{m_p}, \exists \mathbf{i}_n \in I_{m_n}, \exists \sigma \in \mathcal{M}_{S_{\mathbf{i}_p,\mathbf{i}_n}} : \\
& \qquad \mathrm{er}_{\mathcal{P}}[\Phi(S_{\mathbf{i}_p,\mathbf{i}_n}, \sigma)] \geq \varepsilon\Big(\mathbf{m}_{\mathcal{P}}, S_{\mathbf{i}_p,\mathbf{i}_n}, \sigma\Big), \mathbf{m}_{\mathcal{P}}(S, A(S)) = \mathbf{m}_{\mathcal{P}} \Big\} \\
\leq \; & \sum_{\mathbf{m}_{\mathcal{P}}} \sum_{\mathbf{i}_p \in I_{m_p}} \sum_{\mathbf{i}_n \in I_{m_n}} P\Big\{ S \in \mathcal{X} \;:\; \exists \sigma \in \mathcal{M}_{S_{\mathbf{i}_p,\mathbf{i}_n}} : \\
& \qquad \mathrm{er}_{\mathcal{P}}[\Phi(S_{\mathbf{i}_p,\mathbf{i}_n}, \sigma)] \geq \varepsilon\Big(\mathbf{m}_{\mathcal{P}}, S_{\mathbf{i}_p,\mathbf{i}_n}, \sigma\Big), \mathbf{m}_{\mathcal{P}}(S, A(S)) = \mathbf{m}_{\mathcal{P}} \Big\},
\end{aligned}$$

where $\Phi(S_{\mathbf{i}_p,\mathbf{i}_n}, \sigma)$ denotes the classifier obtained once, $S, \mathbf{i}_p, \mathbf{i}_n$, and σ have been fixed. The last inequality comes from the union bound over all the possible choices of $\mathbf{i}_p \in I_{m_p}$ and $\mathbf{i}_n \in I_{m_n}$. Let

$$P' \stackrel{\text{def}}{=} P\Big\{ S \in \mathcal{X} \colon \exists \sigma \in \mathcal{M}_{S_{\mathbf{i}_p,\mathbf{i}_n}} : \mathrm{er}_{\mathcal{P}}[\Phi(S_{\mathbf{i}_p,\mathbf{i}_n}, \sigma)] \geq \varepsilon\Big(\mathbf{m}_{\mathcal{P}}, S_{\mathbf{i}_p,\mathbf{i}_n}, \sigma\Big), \mathbf{m}_{\mathcal{P}}(S, A(S)) = \mathbf{m}_{\mathcal{P}} \Big\}.$$

We now make explicit how the positive and negative examples are interleaved in the training sequence S by introducing a new variable $\mathbf{b}$, which is a bit-string of length m such that S_i is a positive example if and only if $\mathbf{b}_i = 1$. Let B_{m_p} denote the set of possible $\mathbf{b}$ vectors that we can have

under the constraint that $|S_{\mathcal{P}}| = m_p$. We then have

$$\begin{aligned} P' &= \sum_{\mathbf{b}\in B_{m_p}} P\Big\{ S\in\mathcal{X} :\ \exists \sigma\in\mathcal{M}_{S_{\mathbf{i}_p,\mathbf{i}_n}} : \mathrm{er}_{\mathcal{P}}[\Phi(S_{\mathbf{i}_p,\mathbf{i}_n},\sigma)] \geq \varepsilon\Big(\mathbf{m}_{\mathcal{P}}, S_{\mathbf{i}_p,\mathbf{i}_n},\sigma\Big), \\ &\qquad\qquad \mathbf{m}_{\mathcal{P}}(S,A(S)) = \mathbf{m}_{\mathcal{P}} \mid\ \mathbf{b}(S)=\mathbf{b}\Big\} P\Big\{ S\in\mathcal{X} :\ \mathbf{b}(S)=\mathbf{b}\Big\} \\ &= \sum_{\mathbf{b}\in B_{m_p}} P\Big\{ S\in\mathcal{X} :\ \exists \sigma\in\mathcal{M}_{S_{\mathbf{i}_p,\mathbf{i}_n}} : \mathrm{er}_{\mathcal{P}}[\Phi(S_{\mathbf{i}_p,\mathbf{i}_n},\sigma)] \geq \varepsilon\Big(\mathbf{m}_{\mathcal{P}}, S_{\mathbf{i}_p,\mathbf{i}_n},\sigma\Big), \\ &\qquad\qquad \mathbf{m}_{\mathcal{P}}(S,A(S)) = \mathbf{m}_{\mathcal{P}} \mid\ \mathbf{b}(S)=\mathbf{b}\Big\} p_{\mathcal{P}}^{m_p}(1-p_{\mathcal{P}})^{m-m_p}. \end{aligned}$$

$$\begin{aligned} P' \leq \binom{m}{m_p} p_{\mathcal{P}}^{m_p}(1-p_{\mathcal{P}})^{m-m_p} \sup_{\mathbf{b}\in B_{m_p}} P\Big\{ & S\in\mathcal{X} :\ \exists \sigma\in\mathcal{M}_{S_{\mathbf{i}_p,\mathbf{i}_n}} : \\ & \mathrm{er}_{\mathcal{P}}[\Phi(S_{\mathbf{i}_p,\mathbf{i}_n},\sigma)] \geq \varepsilon\Big(\mathbf{m}_{\mathcal{P}}, S_{\mathbf{i}_p,\mathbf{i}_n},\sigma\Big), \mathbf{m}_{\mathcal{P}}(S,A(S)) = \mathbf{m}_{\mathcal{P}} \mid\ \mathbf{b}(S)=\mathbf{b}\Big\}. \end{aligned}$$

Under the condition $\mathbf{b}(S)=\mathbf{b}$, index vectors $\mathbf{i}_p$ and $\mathbf{i}_n$ are now pointing to specific examples in S. Consequently, under this condition, we can compute the above probability by first conditioning on the compression set $S_{\mathbf{i}_p,\mathbf{i}_n}$ and then performing the expectation over $S_{\mathbf{i}_p,\mathbf{i}_n}$. Hence

$$P\Big\{ S\in\mathcal{X}\colon (\cdot) \Big| \mathbf{b}(S)=\mathbf{b}\Big\} \ =\ E_{S_{\mathbf{i}_p,\mathbf{i}_n}|\mathbf{b}} P\Big\{ S\in\mathcal{X}\colon (\cdot) \Big| \mathbf{b}(S)=\mathbf{b}, S_{\mathbf{i}_p,\mathbf{i}_n}\Big\}.$$

By applying the union bound over $\sigma\in\mathcal{M}_{S_{\mathbf{i}_p,\mathbf{i}_n}}$, we obtain

$$\begin{aligned} P\Big\{ S\in\mathcal{X}\colon \exists\sigma\in\mathcal{M}_{S_{\mathbf{i}_p,\mathbf{i}_n}} : \mathrm{er}_{\mathcal{P}}[\Phi(S_{\mathbf{i}_p,\mathbf{i}_n},\sigma)] \geq \varepsilon\Big(\mathbf{m}_{\mathcal{P}}, S_{\mathbf{i}_p,\mathbf{i}_n},\sigma\Big), & \\ \mathbf{m}_{\mathcal{P}}(S,A(S)) = \mathbf{m}_{\mathcal{P}} \mid\ \mathbf{b}(S)=\mathbf{b}, S_{\mathbf{i}_p,\mathbf{i}_n}\Big\} & \\ \leq \sum_{\sigma\in\mathcal{M}_{S_{\mathbf{i}_p,\mathbf{i}_n}}} P\Big\{ S\in\mathcal{X}\colon \mathrm{er}_{\mathcal{P}}[\Phi(S_{\mathbf{i}_p,\mathbf{i}_n},\sigma)] \geq \varepsilon\Big(\mathbf{m}_{\mathcal{P}}, S_{\mathbf{i}_p,\mathbf{i}_n},\sigma\Big), & \\ \mathbf{m}_{\mathcal{P}}(S,A(S)) = \mathbf{m}_{\mathcal{P}} \mid\ \mathbf{b}(S)=\mathbf{b}, S_{\mathbf{i}_p,\mathbf{i}_n}\Big\}. & \end{aligned}$$

We will now stratify this last probability by the set of possible errors that classifier $\Phi(S_{\mathbf{i}_p,\mathbf{i}_n},\sigma)$ can perform on the training examples that are not in the compression set $S_{\mathbf{i}_p,\mathbf{i}_n}$. Note that we do not force here the learner to produce a classifier that does not make errors on $S_{\mathbf{i}_p,\mathbf{i}_n}$. However, the set of message strings needed by Φ to identify a classifier h might be larger when h can err on $S_{\mathbf{i}_p,\mathbf{i}_n}$. To perform this stratification, let $\hat{\mathbf{er}}(f,S_{\mathcal{P}})$ be the vector of indices pointing to the examples of $S_{\mathcal{P}}$ that are misclassified by f. Moreover, let $I_{m_p}(\mathbf{i}_p)$ denote the set of all vectors $\mathbf{j}_p \in I_{m_p}$ for which no index $i\in\mathbf{j}_p$ is also in $\mathbf{i}_p$. In other words, for all $\mathbf{i}_p\in I_{m_p}$ and all $\mathbf{j}_p\in I_{m_p}(\mathbf{i}_p)$, we have $\mathbf{j}_p\cap\mathbf{i}_p=\emptyset$.

Therefore

$$P\Big\{ S\in\mathcal{X}\colon \mathrm{er}_{\mathcal{P}}[\Phi(S_{\mathbf{i}_p,\mathbf{i}_n},\sigma)]\geq \varepsilon\Big(\mathbf{m}_{\mathcal{P}},S_{\mathbf{i}_p,\mathbf{i}_n},\sigma\Big),\mathbf{m}_{\mathcal{P}}(S,A(S))=\mathbf{m}_{\mathcal{P}}\mid\ \mathbf{b}(S)=\mathbf{b},S_{\mathbf{i}_p,\mathbf{i}_n}\Big\}$$
$$=\sum_{\mathbf{j}_p\in I_{m_p}(\mathbf{i}_p)} P\Big\{ S\in\mathcal{X}\colon \mathrm{er}_{\mathcal{P}}[\Phi(S_{\mathbf{i}_p,\mathbf{i}_n},\sigma)]\geq \varepsilon\Big(\mathbf{m}_{\mathcal{P}},S_{\mathbf{i}_p,\mathbf{i}_n},\sigma\Big),$$
$$\hat{\mathbf{er}}[\Phi(S_{\mathbf{i}_p,\mathbf{i}_n},\sigma),S_{\mathcal{P}}]=\mathbf{j}_p,\mathbf{m}_{\mathcal{P}}(S,A(S))=\mathbf{m}_{\mathcal{P}}\mid\ \mathbf{b}(S)=\mathbf{b},S_{\mathbf{i}_p,\mathbf{i}_n}\Big\}$$
$$=\sum_{\mathbf{j}_p\in I_{m_p}(\mathbf{i}_p)} P\Big\{ S\in\mathcal{X}\colon \mathrm{er}_{\mathcal{P}}[\Phi(S_{\mathbf{i}_p,\mathbf{i}_n},\sigma)]\geq \varepsilon\Big(\mathbf{m}_{\mathcal{P}},S_{\mathbf{i}_p,\mathbf{i}_n},\sigma\Big),$$
$$\hat{\mathbf{er}}[\Phi(S_{\mathbf{i}_p,\mathbf{i}_n},\sigma),S_{\mathcal{P}}]=\mathbf{j}_p\mid\ \mathbf{b}(S)=\mathbf{b},S_{\mathbf{i}_p,\mathbf{i}_n}\Big\},$$

where the last equality comes from the fact that the condition $\mathbf{m}_{\mathcal{P}}(S,A(S))=\mathbf{m}_{\mathcal{P}}$ is obsolete when $\mathbf{b}(S)=\mathbf{b}$ with fixed vectors $\mathbf{i}_p,\mathbf{i}_n,\mathbf{j}_p$. Now, under the condition $\mathbf{b}(S)=\mathbf{b}$ with a fixed compression set $S_{\mathbf{i}_p,\mathbf{i}_n}$, this last probability is obtained for the random draws of the training examples that are not in $S_{\mathbf{i}_p,\mathbf{i}_n}$. Consequently, this last probability is at most equal to the probability that a fixed classifier, having $\mathrm{er}_{\mathcal{P}}\geq\varepsilon(\mathbf{m}_{\mathcal{P}},S_{\mathbf{i}_p,\mathbf{i}_n},\sigma)$, makes no errors on $m_p-d_p-k_p$ positive examples that are not in the compression set $S_{\mathbf{i}_p,\mathbf{i}_n}$. Note that the probability space created by the conditioning specifies only the positions of the positive examples but places no further restrictions on them. They can therefore be viewed as independent draws from the distribution of positive examples. This makes it possible to bound the probability of the event by the probability that $m_p-d_p-k_p$ independent draws are all correctly classified. Hence, we have

$$P\Big\{ S\in\mathcal{X}\colon \mathrm{er}_{\mathcal{P}}[\Phi(S_{\mathbf{i}_p,\mathbf{i}_n},\sigma)]\geq \varepsilon\Big(\mathbf{m}_{\mathcal{P}},S_{\mathbf{i}_p,\mathbf{i}_n},\sigma\Big),$$
$$\hat{\mathbf{er}}[\Phi(S_{\mathbf{i}_p,\mathbf{i}_n},\sigma),S_{\mathcal{P}}]=\mathbf{j}_p\mid\ \mathbf{b}(S)=\mathbf{b},S_{\mathbf{i}_p,\mathbf{i}_n}\Big\}$$
$$\leq\ \Big(1-\varepsilon(\mathbf{m}_{\mathcal{P}},S_{\mathbf{i}_p,\mathbf{i}_n},\sigma)\Big)^{m_p-d_p-k_p}.$$

By regrouping the previous results, we get

$$\begin{aligned}
P\quad\leq\quad & \sum_{\mathbf{m}_{\mathcal{P}}}\binom{m}{m_p}p_{\mathcal{P}}^{m_p}(1-p_{\mathcal{P}})^{m-m_p}\sum_{\mathbf{i}_p\in I_{m_p}}\sum_{\mathbf{i}_n\in I_{m_n}}\\
&\sup_{\mathbf{b}\in B_{m_p}} E_{S_{\mathbf{i}_p,\mathbf{i}_n}|\mathbf{b}}\sum_{\sigma\in\mathcal{M}_{S_{\mathbf{i}_p,\mathbf{i}_n}}}\sum_{\mathbf{j}_p\in I_{m_p}(\mathbf{i}_p)}\Big(1-\varepsilon(\mathbf{m}_{\mathcal{P}},S_{\mathbf{i}_p,\mathbf{i}_n},\sigma)\Big)^{m_p-d_p-k_p}\\
=\quad & \sum_{m_p=0}^{m}\binom{m}{m_p}p_{\mathcal{P}}^{m_p}(1-p_{\mathcal{P}})^{m-m_p}\sum_{d_p=0}^{m_p}\binom{m_p}{d_p}\sum_{d_n=0}^{m-m_p}\binom{m_n}{d_n}\sum_{k_p=0}^{m_p-d_p}\binom{m_p-d_p}{k_p}\\
&\sup_{\mathbf{b}\in B_{m_p}} E_{S_{\mathbf{i}_p,\mathbf{i}_n}|\mathbf{b}}\sum_{\sigma\in\mathcal{M}_{S_{\mathbf{i}_p,\mathbf{i}_n}}}\Big(1-\varepsilon(\mathbf{m}_{\mathcal{P}},S_{\mathbf{i}_p,\mathbf{i}_n},\sigma)\Big)^{m_p-d_p-k_p}.
\end{aligned}$$

By using

$$\Big(1-\varepsilon(\mathbf{m}_{\mathcal{P}},S_{\mathbf{i}_p,\mathbf{i}_n},\sigma)\Big)^{m_p-d_p-k_p} \;=\; P_{S_{\mathbf{i}_p,\mathbf{i}_n}}(\sigma)\cdot\frac{1}{B_{\mathcal{P}}(\mathbf{m}_{\mathcal{P}})}\cdot\zeta(k_p)\cdot\zeta(d_n)\cdot\zeta(d_p)\cdot\delta\,,$$

we get $P\leq\delta$ as desired. Similarly, we have

$$P\Big\{S\in\mathcal{X}\,:\;\mathrm{er}_{\mathcal{N}}[A(S)]\geq\varepsilon_{\mathcal{N}}\Big(\mathbf{m}_{\mathcal{N}}(S,A(S)),g_{\mathcal{N}}(S)\delta\Big)\Big\}\;\leq\;\delta\,,$$

which completes the proof. ■

Remark 3 *This theorem can be viewed in a standard asymptotic form by using the inequality* $1-\exp(-x)\leq x$*, for* $x\geq 0$*. To see this, we simply need to substitute Equations 7 and 8 into each probability given in Theorem 2 and weaken them with the above inequality. Doing so yields the following bounds:*

$$P\Big\{S\in\mathcal{X}\,:\;\mathrm{er}_{\mathcal{P}}[A(S)]\leq\frac{1}{m_p-d_p-k_p}\left[\ln\left(B_{\mathcal{P}}(\mathbf{m}_{\mathcal{P}})\right)+\ln\frac{1}{g_{\mathcal{P}}(S)\delta}\right]\Big\}\;\;\geq\;\;1-\delta,$$

$$P\Big\{S\in\mathcal{X}\,:\;\mathrm{er}_{\mathcal{N}}[A(S)]\leq\frac{1}{m_n-d_n-k_n}\left[\ln\left(B_{\mathcal{N}}(\mathbf{m}_{\mathcal{N}})\right)+\ln\frac{1}{g_{\mathcal{N}}(S)\delta}\right]\Big\}\;\;\geq\;\;1-\delta\,.$$

However, each probability is separately bounding the error on the positive and negative examples and so will not (in the final bound) hold with probability $1-\delta$ *but with probability* $1-\delta/4$ *(to be shown) as the expected loss will rely on four bounds simultaneously holding true (i.e., from Equation 1 we would like to upper bound* $\mathrm{er}_{\mathcal{P}}[A(S)]$*,* $\mathrm{er}_{\mathcal{N}}[A(S)]$*,* $p_{\mathcal{P}}$ *and* $p_{\mathcal{N}}$*).*

Now that we have a bound on both $\mathrm{er}_{\mathcal{P}}[A(S)]$ and $\mathrm{er}_{\mathcal{N}}[A(S)]$, to bound the expected loss $\mathbb{E}[l(A(S))]$ of Equation 1 we now need to upper bound the probabilities $p_{\mathcal{P}}$ and $p_{\mathcal{N}}$. For this task, we could use a well-known approximation of the binomial tail such as the additive Hoeffding bound or the multiplicative Chernoff bound. However, the Hoeffding bound is known to be very loose when the the probability of interest (here $p_{\mathcal{P}}$ and $p_{\mathcal{N}}$) is close to zero. Conversely, the multiplicative Chernoff bound is known to be loose when the probability of interest is close to $1/2$. In order to obtain a tight loss bound for both balanced and imbalanced data sets, we have decided to use the binomial distribution without any approximation.

Recall that the probability $\mathrm{Bin}(m,k,p)$ of having at most k successes among m Bernoulli trials, each having probability of success p, is given by the binomial tail

$$\mathrm{Bin}(m,k,p)\stackrel{\text{def}}{=}\sum_{i=0}^{k}\binom{m}{i}p^i(1-p)^{m-i}\,.$$

Following Langford (2005), we now define the *binomial tail inversion* $\overline{\mathrm{Bin}}(m,k,\delta)$ as the largest value of probability of success such that we still have a probability of at least δ of observing at most k successes out of m Bernoulli trials. In other words,

$$\overline{\mathrm{Bin}}(m,k,\delta)\stackrel{\text{def}}{=}\sup\Big\{p:\mathrm{Bin}(m,k,p)\geq\delta\Big\}\,. \tag{11}$$

From this definition, it follows that $\overline{\text{Bin}}(m, m_n, \delta)$ is the *smallest* upper bound on $p_{\mathcal{N}}$, which holds with probability at least $1-\delta$, over the random draws of m examples. Hence

$$P\left\{ S \in \mathcal{X} \colon p_{\mathcal{N}} \leq \overline{\text{Bin}}\Big(m, m_n, \delta\Big) \right\} \geq 1-\delta\,.$$

From this bound (applied to both $p_{\mathcal{P}}$ and $p_{\mathcal{N}}$), and from the previous theorem, the following predicates hold simultaneously with probability $1-\delta$ over the random draws of S:

$$\begin{aligned}
\text{er}_{\mathcal{P}}[A(S)] &\leq \varepsilon_{\mathcal{P}}\Big(\mathbf{m}_{\mathcal{P}}, g_{\mathcal{P}}(S)\frac{\delta}{4}\Big)\,,\\
\text{er}_{\mathcal{N}}[A(S)] &\leq \varepsilon_{\mathcal{N}}\Big(\mathbf{m}_{\mathcal{N}}, g_{\mathcal{N}}(S)\frac{\delta}{4}\Big)\,,\\
p_{\mathcal{N}} &\leq \overline{\text{Bin}}\Big(m, m_n, \frac{\delta}{4}\Big)\,,\\
p_{\mathcal{P}} &\leq \overline{\text{Bin}}\Big(m, m_p, \frac{\delta}{4}\Big)\,,
\end{aligned}$$

where $\mathbf{m}_{\mathcal{P}} = \mathbf{m}_{\mathcal{P}}(S, A(S))$ and $\mathbf{m}_{\mathcal{N}} = \mathbf{m}_{\mathcal{N}}(S, A(S))$. Consequently, we have the next theorem.

Theorem 4 *Given the above definitions, let A be any learning algorithm having a reconstruction function that maps compression sets and message strings to classifiers. With probability* $1-\delta$ *over the random draws of a training set S, we have*

$$\begin{aligned}
\mathbb{E}[l(A(S))] \;\leq\; l_{\mathcal{P}} \cdot \overline{\text{Bin}}\Big(m, m_p, \frac{\delta}{4}\Big) \cdot \varepsilon_{\mathcal{P}}\Big(\mathbf{m}_{\mathcal{P}}, g_{\mathcal{P}}(S)\frac{\delta}{4}\Big) \\
+ l_{\mathcal{N}} \cdot \overline{\text{Bin}}\Big(m, m_n, \frac{\delta}{4}\Big) \cdot \varepsilon_{\mathcal{N}}\Big(\mathbf{m}_{\mathcal{N}}, g_{\mathcal{N}}(S)\frac{\delta}{4}\Big)\,,
\end{aligned}$$

where $\mathbf{m}_{\mathcal{P}} = \mathbf{m}_{\mathcal{P}}(S, A(S))$ *and* $\mathbf{m}_{\mathcal{N}} = \mathbf{m}_{\mathcal{N}}(S, A(S))$ *are defined by Equations 3 and 4.*

We can now improve the loss bound given by Theorem 4 in the following way. Consider the frequencies $\hat{p}_{\mathcal{P}} \stackrel{\text{def}}{=} m_p/m$ and $\hat{p}_{\mathcal{N}} \stackrel{\text{def}}{=} m_n/m$. Let us simply denote by $\varepsilon_{\mathcal{P}}$ and $\varepsilon_{\mathcal{N}}$ some upper bounds on $\text{er}_{\mathcal{P}}[A(S)]$ and $\text{er}_{\mathcal{P}}[A(S)]$. Let us also denote by $\overline{p}_{\mathcal{P}}$ and $\overline{p}_{\mathcal{N}}$ some upper bounds on $p_{\mathcal{P}}$ and $p_{\mathcal{N}}$. Let us first assume that $l_{\mathcal{N}}\varepsilon_{\mathcal{N}} \geq l_{\mathcal{P}}\varepsilon_{\mathcal{P}}$. Then we have

$$\begin{aligned}
\mathbb{E}[l(A(S))] &\leq p_{\mathcal{P}} l_{\mathcal{P}} \varepsilon_{\mathcal{P}} + p_{\mathcal{N}} l_{\mathcal{N}} \varepsilon_{\mathcal{N}}\\
&= l_{\mathcal{P}}\varepsilon_{\mathcal{P}} + p_{\mathcal{N}}(l_{\mathcal{N}}\varepsilon_{\mathcal{N}} - l_{\mathcal{P}}\varepsilon_{\mathcal{P}})\\
&\leq l_{\mathcal{P}}\varepsilon_{\mathcal{P}} + \overline{p}_{\mathcal{N}}(l_{\mathcal{N}}\varepsilon_{\mathcal{N}} - l_{\mathcal{P}}\varepsilon_{\mathcal{P}})\\
&= \hat{p}_{\mathcal{P}} l_{\mathcal{P}}\varepsilon_{\mathcal{P}} + \hat{p}_{\mathcal{N}} l_{\mathcal{N}}\varepsilon_{\mathcal{N}} + (\overline{p}_{\mathcal{N}} - \hat{p}_{\mathcal{N}})(l_{\mathcal{N}}\varepsilon_{\mathcal{N}} - l_{\mathcal{P}}\varepsilon_{\mathcal{P}})\,.
\end{aligned}$$

Likewise, if $l_{\mathcal{P}}\varepsilon_{\mathcal{P}} \geq l_{\mathcal{N}}\varepsilon_{\mathcal{N}}$, we have

$$\mathbb{E}[l(A(S))] \leq \hat{p}_{\mathcal{P}} l_{\mathcal{P}}\varepsilon_{\mathcal{P}} + \hat{p}_{\mathcal{N}} l_{\mathcal{N}}\varepsilon_{\mathcal{N}} + (\overline{p}_{\mathcal{P}} - \hat{p}_{\mathcal{P}})(l_{\mathcal{P}}\varepsilon_{\mathcal{P}} - l_{\mathcal{N}}\varepsilon_{\mathcal{N}})\,.$$

Consequently, we have the following theorem.

Theorem 5 *Given the above definitions, let A be any learning algorithm having a reconstruction function that maps compression sets and message strings to classifiers. For any real numbers a, b, c, let*

$$\Psi(a;b;c) \stackrel{\text{def}}{=} \begin{cases} a \cdot |c| & \text{if} \quad c \geq 0 \\ b \cdot |c| & \text{if} \quad c \leq 0\,. \end{cases}$$

Then, with probability $1-\delta$ over the random draws of a training set S, we have

$$\begin{aligned} \mathbb{E}[l(A(S))] \;\leq\; & \frac{m_p}{m} \cdot l_{\mathcal{P}} \cdot \varepsilon_{\mathcal{P}}\Big(\mathbf{m}_{\mathcal{P}}, g_{\mathcal{P}}(S)\frac{\delta}{4}\Big) + \frac{m_n}{m} \cdot l_{\mathcal{N}} \cdot \varepsilon_{\mathcal{N}}\Big(\mathbf{m}_{\mathcal{N}}, g_{\mathcal{N}}(S)\frac{\delta}{4}\Big) \\ & + \Psi\Bigg(\overline{\mathrm{Bin}}\Big(m, m_p, \frac{\delta}{4}\Big) - \frac{m_p}{m}\;;\; \overline{\mathrm{Bin}}\Big(m, m_n, \frac{\delta}{4}\Big) - \frac{m_n}{m}\;; \\ & \qquad l_{\mathcal{P}}\varepsilon_{\mathcal{P}}\Big(\mathbf{m}_{\mathcal{P}}, g_{\mathcal{P}}(S)\frac{\delta}{4}\Big) - l_{\mathcal{N}}\varepsilon_{\mathcal{N}}\Big(\mathbf{m}_{\mathcal{N}}, g_{\mathcal{N}}(S)\frac{\delta}{4}\Big)\Bigg)\,, \end{aligned}$$

where $\mathbf{m}_{\mathcal{P}} = \mathbf{m}_{\mathcal{P}}(S, A(S))$ and $\mathbf{m}_{\mathcal{N}} = \mathbf{m}_{\mathcal{N}}(S, A(S))$ are defined by Equations 3 and 4.

To compare the bound given by Theorem 5 with the bound given by Theorem 4, let us assume that $l_{\mathcal{N}}\varepsilon_{\mathcal{N}} \geq l_{\mathcal{P}}\varepsilon_{\mathcal{P}}$. Using our shorthand notation, the bound of Theorem 5 is given by

$$l_{\mathcal{P}}\hat{p}_{\mathcal{P}}\varepsilon_{\mathcal{P}} + l_{\mathcal{N}}\hat{p}_{\mathcal{N}}\varepsilon_{\mathcal{N}} + (\overline{p}_{\mathcal{N}} - \hat{p}_{\mathcal{N}})(l_{\mathcal{N}}\varepsilon_{\mathcal{N}} - l_{\mathcal{P}}\varepsilon_{\mathcal{P}})\,.$$

Whereas the bound of Theorem 4 is given by

$$l_{\mathcal{P}}\overline{p}_{\mathcal{P}}\varepsilon_{\mathcal{P}} + l_{\mathcal{N}}\overline{p}_{\mathcal{N}}\varepsilon_{\mathcal{N}}\,.$$

The bound of Theorem 4 minus the bound of Theorem 5 then gives

$$\begin{aligned} & (l_{\mathcal{P}}\overline{p}_{\mathcal{P}}\varepsilon_{\mathcal{P}} + l_{\mathcal{N}}\overline{p}_{\mathcal{N}}\varepsilon_{\mathcal{N}}) - (l_{\mathcal{P}}\hat{p}_{\mathcal{P}}\varepsilon_{\mathcal{P}} + l_{\mathcal{N}}\hat{p}_{\mathcal{N}}\varepsilon_{\mathcal{N}} + (\overline{p}_{\mathcal{N}} - \hat{p}_{\mathcal{N}})(l_{\mathcal{N}}\varepsilon_{\mathcal{N}} - l_{\mathcal{P}}\varepsilon_{\mathcal{P}})) \\ & \quad = \quad (\overline{p}_{\mathcal{P}} - \hat{p}_{\mathcal{P}})l_{\mathcal{P}}\varepsilon_{\mathcal{P}} + (\overline{p}_{\mathcal{N}} - \hat{p}_{\mathcal{N}})l_{\mathcal{N}}\varepsilon_{\mathcal{N}} - (\overline{p}_{\mathcal{N}} - \hat{p}_{\mathcal{N}})(l_{\mathcal{N}}\varepsilon_{\mathcal{N}} - l_{\mathcal{P}}\varepsilon_{\mathcal{P}}) \\ & \quad = \quad (\overline{p}_{\mathcal{P}} - \hat{p}_{\mathcal{P}} + \overline{p}_{\mathcal{N}} - \hat{p}_{\mathcal{N}})l_{\mathcal{P}}\varepsilon_{\mathcal{P}} \\ & \quad = \quad (\overline{p}_{\mathcal{P}} + \overline{p}_{\mathcal{N}} - 1)l_{\mathcal{P}}\varepsilon_{\mathcal{P}}\,. \end{aligned}$$

Since $l_{\mathcal{P}}\varepsilon_{\mathcal{P}} > 0$ and $\overline{p}_{\mathcal{P}} + \overline{p}_{\mathcal{N}} > 1$, we have an improvement using Theorem 5.

Example 1 *If $l_{\mathcal{P}} = l_{\mathcal{N}} = 1, \delta = 0.05, m = 100, m_p = 40, m_n = 60, \varepsilon_{\mathcal{P}} = 0.3, \varepsilon_{\mathcal{N}} = 0.4$, we get 0.439 for the bound of Theorem 4 and only 0.371 for the bound of Theorem 5. Hence, the bound of Theorem 5 can be significantly better than the the bound of Theorem 4.*

5. Discussion and Numerical Comparisons with Other Bounds

Let us first discuss the bounds that we have proposed and make explicit some of the details and consequences. In general, risk bounds are simply upper bounds of the true error calculated from the (overall) error achieved during training. There is no distinction made between the positive and negative class. The results of the current paper are bounds on the error achieved separately on the positive and negative examples. Hence, making the distinction between the two classes explicit. Furthermore, the risk bound on one class depends on what the classifier achieves on the training

examples of that class. Thus, making the bound more data-dependent then the usual bounds on the true error. This strong data-dependence also allows the user to take into account the observed number of positive and negative examples in the training sample as well as the flexibility of specifying different losses for each class. This is known as asymmetric loss and is not possible with the current crop of sample-compression loss bounds.

Note also that the proposed bounds are data dependent bounds for which there are no corresponding lower bounds. A small compression scheme is evidence of simplicity in the structure of the classifier, but one that is related to the training distribution rather than a priori determined.

Any algorithm that uses a compression scheme can use the bounds that we have proposed and take advantage of asymmetrical loss and cases of imbalanced data sets. However, the tightness of the bound relies on the sparsity of the classifiers (e.g., the size of the compression set). Hence, it may not be advantageous to use algorithms that do not possess levels of sparsity similar (or comparable) to the SCM. This is one reason why we will provide a numerical comparison of various sample-compression bounds for the case of the SCM.

In order to show the merits of our bound we must now compare numerically against more common sample compression bounds and the bound found to be incorrect. In doing so we point out when our bound can be smaller and when it can become larger. All the compared bounds are specialized to the set covering machine compression scheme that uses data-dependent balls. Here each ball is constructed from two data points—one that defines the center of the ball and another that helps define the radius of the ball (known as the border point). Hence to build a classifier from the compression set, we also need an informative message string to discriminate between the border points and the centers.

Let us now discuss the experimental setup, including a list of all the bounds compared, and then conclude with a review of the results.

5.1 Setup

From Example 1 of Section 4, it is clear that using Theorem 5 is more advantageous than Theorem 4. Hence, all experiments will be conducted with the bound of Theorem 5. The first bound we compare against is taken from the original set covering machine paper by Marchand and Shawe-Taylor (2001) and is similar to the Littlestone and Warmuth (1986) bound but with more specialization for the SCM compression set defined from the set of data-dependent balls. The second generalization error bound is adapted from Marchand and Sokolova (2005) and is a slight modification of the Marchand and Shawe-Taylor (2001) result. All these bounds will also be compared against the incorrect bound given in Marchand and Shawe-Taylor (2002).

Please note that traditional sample compression bounds, such as that given by Theorem 6.1 of Langford (2005), cannot be used with the set covering machine as it does *not* allow the inclusion of any side information in the reconstruction of the classifier. The SCM, however, stores both the center and border points in order to construct its hypotheses. This implies the need for side information to discriminate between centers and border points, something that traditional sample compression bounds do not cater for. Therefore, we cannot give numerical comparisons against these types of bounds.

All generalization error bounds detailed below will make use of the following definitions: $d_n = c_n$, $d_p = c_p + b$, $d = d_p + d_n$ and $k = k_p + k_n$. For completeness, we give the definitions of all risk

bounds not already stated and, to avoid repetition, we only give references to the bounds described earlier.

- **new bound** (Theorem 5). When applied to the SCM, the new bound uses the distribution of messages given by Equation 5 and Equations 6, 7, 8, 9, 10, and 11.
- **incorrect bound** (Theorem 5 of Marchand and Shawe-Taylor, 2002). This bound can also be found in Section 3 of the current paper.
- **MS01 bound** (Theorem 5.2 of Marchand and Shawe-Taylor, 2001):

$$\varepsilon(m,d,c_p,k,\delta) = 1-\exp\left(\frac{-1}{m-2d-k}\left[\ln\binom{m}{2d}+\ln\binom{2d}{c_p}+\ln\binom{m-2d}{k}+\ln\left(\frac{2m^2d}{\delta}\right)\right]\right).$$

- **MS05 bound** (Equation 10 of Marchand and Sokolova, 2005):

$$\varepsilon(m,d,d_p,b,k,\delta) = 1-\exp\left(\frac{-1}{m-d-k}\left[\ln\binom{m}{d}+\ln\binom{m-d}{k}+\ln\binom{d_p}{b}+\ln\left(\frac{1}{\zeta(d)\zeta(k)\zeta(b)\delta}\right)\right]\right),$$

  where $\zeta(a)$ is given by Equation 6.

5.2 Discussion of Results

The numerical comparisons of these four bounds (*new bound, incorrect bound, MS01 bound and MS05 bound*) are shown in Figure 1 and Figure 2. Each plot contains the number of positive examples m_p, the number of negative examples m_n, the number of positive centers c_p, the number of negative centers c_n and the number of borders b. The number of negative misclassifications k_n was fixed for all plots and these values can be found in the x-axis label (either 0 or 500). The number of positive examples was varied and its quantity was set to those values given by the x-axis of the plot. For example, in the left hand side plot of Figure 1, the number of negative misclassifications k_n was 0 and the number of positive misclassifications k_p varied from 1 to 2000. The y-axis give the bound values achieved. Finally, the empirical error was also included in each plot—which is simply the number of examples misclassified divided by the number of examples, that is, $(k_p+k_n)/(m_p+m_n)$.

Figure 1 shows the case where the number of positive and negative examples is approximately the same. We clearly see that the incorrect bound becomes erroneous when the number k_p of errors on the positive training examples approaches the total number m_p of positive training examples. We also see that the new bound is tighter than the MS01 and MS05 bounds when the k_p differs greatly from k_n. However, the latter bound is slightly tighter than the new bound when $k_p = k_n$.

Figure 2 depicts the case where there is an imbalance in the data set ($m_n \gg m_p$), implying greater possibility of imbalance in misclassifications. However, the behavior is similar as the one found in Figure 1. Indeed, the MS01 and MS05 loss bounds are slightly smaller than the new bound when k_p/m_p is similar to k_n/m_n, but the new bound becomes smaller when these two quantities greatly differ. This is where the new bound is most advantageous—in the case when there is an imbalance in misclassifications. As we would expect, the new bound is smaller when one class of examples is more abundant than the other.

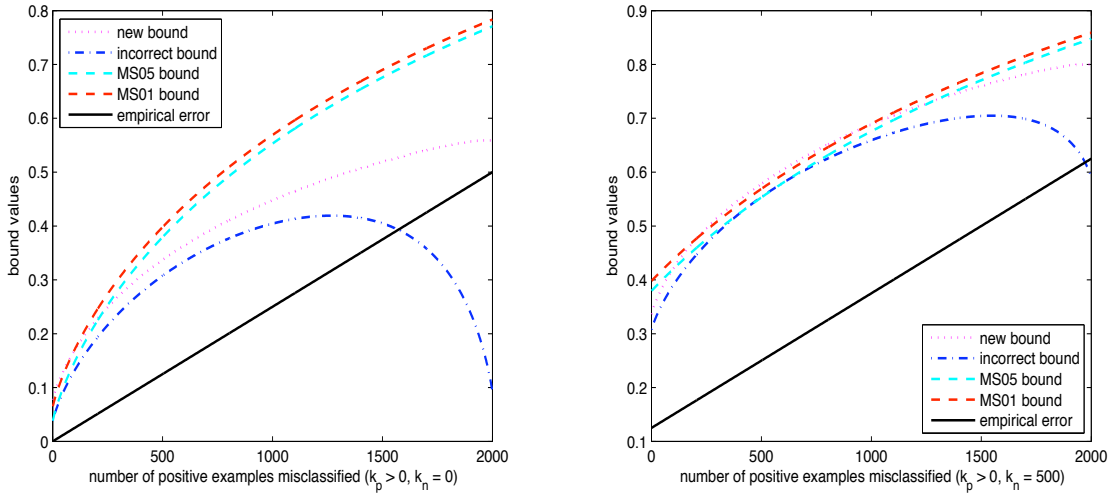


Figure 1: Bound values for the SCM when $m_p = 2020, m_n = 1980, c_p = 5, c_n = 5, b = 10$.

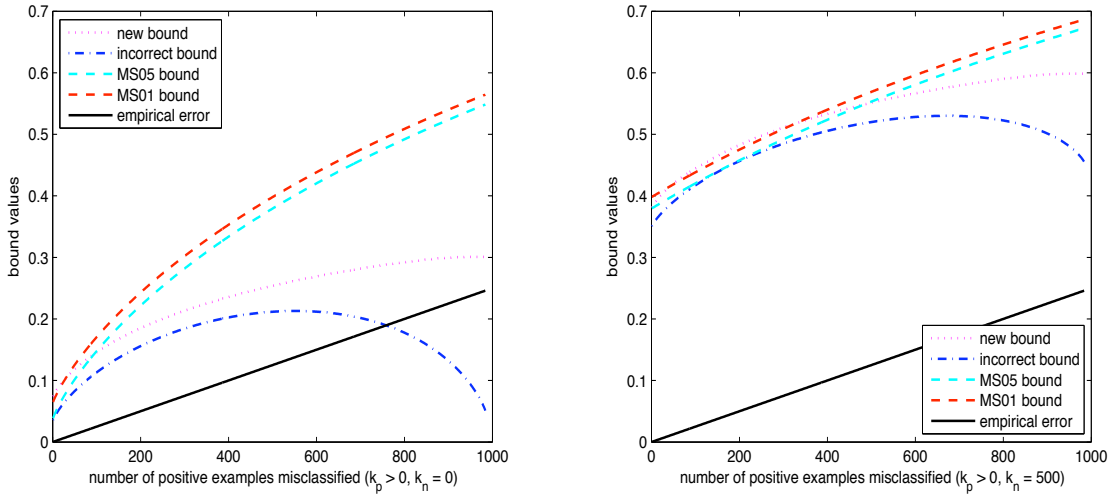


Figure 2: Bound values for the SCM when $m_p = 1000, m_n = 3000, c_p = 5, c_n = 5, b = 10$.

6. Conclusion

We have observed that the SCM loss bound proposed by Marchand and Shawe-Taylor (2002) is incorrect and, in fact, becomes erroneous in the limit where the number of errors on the positive training examples approaches the total number of positive training examples. We have then proposed a new loss bound, valid for any sample-compression learning algorithm (including the SCM), that depends on the observed fraction of positive examples and on what the classifier achieves on them. This new bound captures the spirit of Marchand and Shawe-Taylor (2002) with very similar tightness in the regimes in which the bound could hold. This is shown in numerical comparisons of the loss bound proposed in this paper with all of the earlier bounds that can be applied to the SCM.

As mentioned above, an advantage of the bound is its ability to take into account the observed number of positive examples in the training set in order to arrive at tighter estimates. It also has the advantage of being applicable in cases where the loss function is asymmetrical for type I and type II errors, a situation that is not uncommon in practical applications.

The tightness of the bounds derived for the set covering machine make it tempting to use them to perform model selection as well as to consider integrating them more closely into the workings of the algorithm. Both of these directions are the subject of ongoing research.

Acknowledgments

We would like to thank the anonymous reviewers for their helpful comments. This work was supported by NSERC Discovery grants 262067 and 122405 and, in part, by the IST Programme of the European Community, under the PASCAL Network of Excellence, IST-2002-506778. This publication only reflects the authors' views.

References

Sally Floyd and Manfred Warmuth. Sample compression, learnability, and the Vapnik-Chervonenkis dimension. *Machine Learning*, 21(3):269–304, 1995.

John Langford. Tutorial on practical prediction theory for classification. *Journal of Machine Learning Research*, 6:273–306, 2005.

Nick Littlestone and Manfred Warmuth. Relating data compression and learnability. Technical report, University of California Santa Cruz, Santa Cruz, CA, 1986.

Mario Marchand and John Shawe-Taylor. Learning with the set covering machine. *Proceedings of the Eighteenth International Conference on Machine Learning (ICML 2001)*, pages 345–352, 2001.

Mario Marchand and John Shawe-Taylor. The set covering machine. *Journal of Machine Learning Reasearch*, 3:723–746, 2002.

Mario Marchand and Marina Sokolova. Learning with decision lists of data-dependent features. *Journal of Machine Learning Reasearch*, 6:427–451, 2005.

Journal of Machine Learning Research 8 (2007) 2551-2594 Submitted 1/07; Revised 8/07; Published 11/07

VC Theory of Large Margin Multi-Category Classifiers

Yann Guermeur YANN.GUERMEUR@LORIA.FR
LORIA-CNRS
Campus Scientifique, BP 239
54506 Vandœuvre-lès-Nancy cedex, France

Editors: Isabelle Guyon and Amir Saffari

Abstract

In the context of discriminant analysis, Vapnik's statistical learning theory has mainly been developed in three directions: the computation of dichotomies with binary-valued functions, the computation of dichotomies with real-valued functions, and the computation of polytomies with functions taking their values in finite sets, typically the set of categories itself. The case of classes of vector-valued functions used to compute polytomies has seldom been considered independently, which is unsatisfactory, for three main reasons. First, this case encompasses the other ones. Second, it cannot be treated appropriately through a naïve extension of the results devoted to the computation of dichotomies. Third, most of the classification problems met in practice involve multiple categories.

In this paper, a VC theory of large margin multi-category classifiers is introduced. Central in this theory are generalized VC dimensions called the γ-Ψ-dimensions. First, a uniform convergence bound on the risk of the classifiers of interest is derived. The capacity measure involved in this bound is a covering number. This covering number can be upper bounded in terms of the γ-Ψ-dimensions thanks to generalizations of Sauer's lemma, as is illustrated in the specific case of the scale-sensitive Natarajan dimension. A bound on this latter dimension is then computed for the class of functions on which multi-class SVMs are based. This makes it possible to apply the structural risk minimization inductive principle to those machines.

Keywords: multi-class discriminant analysis, large margin classifiers, uniform strong laws of large numbers, generalized VC dimensions, multi-class SVMs, structural risk minimization inductive principle, model selection

1. Introduction

One of the central domains of Vapnik's statistical learning theory (Vapnik, 1998) is the theory of bounds, which is at the origin of the structural risk minimization (SRM) inductive principle (Vapnik, 1982; Shawe-Taylor et al., 1998) and, as such, has not only a theoretical interest, but also a practical one. This theory has been developed for pattern recognition, regression estimation and density estimation. The first results in the field of discrimination, exposed in Vapnik and Chervonenkis (1971), were dealing with the computation of dichotomies with binary-valued functions. Later on, several studies were devoted to the case of multi-class $[\![1, Q]\!]$-valued classifiers (Ben-David et al., 1995), and large margin classifiers computing dichotomies (Alon et al., 1997; Bartlett, 1998; Bartlett and Shawe-Taylor, 1999) (see also Bartlett et al., 1996, for the case of regression). However, the case of large margin classifiers computing polytomies (models taking their values in $\mathbb{R}^Q$) has seldom been tackled independently, although it cannot be considered as a trivial extension of the three former ones (Guermeur et al., 1999).

In this paper, we unify two complementary and well established theories, the theory of large margin (bi-class) classifiers and the theory of multi-class $[\![1,Q]\!]$-valued classifiers, to lay the bases of a simple theory of large margin multi-class classifiers. Central in the process is the specification of a new class of generalized Vapnik-Chervonenkis (VC) dimensions, the γ-Ψ-dimensions. They can be seen either as scale-sensitive extensions of the Ψ-dimensions (Ben-David et al., 1995), or multivariate extensions of the fat-shattering dimension (Kearns and Schapire, 1994). An application to the class of functions on which multi-class SVMs (M-SVMs) are based is provided. This makes it possible to justify a posteriori the choice of their training criteria, which appear as implementations of the SRM inductive principle. This also gives birth to a model selection procedure of low computational cost. The main stages of our study are summarized in Figure 1.

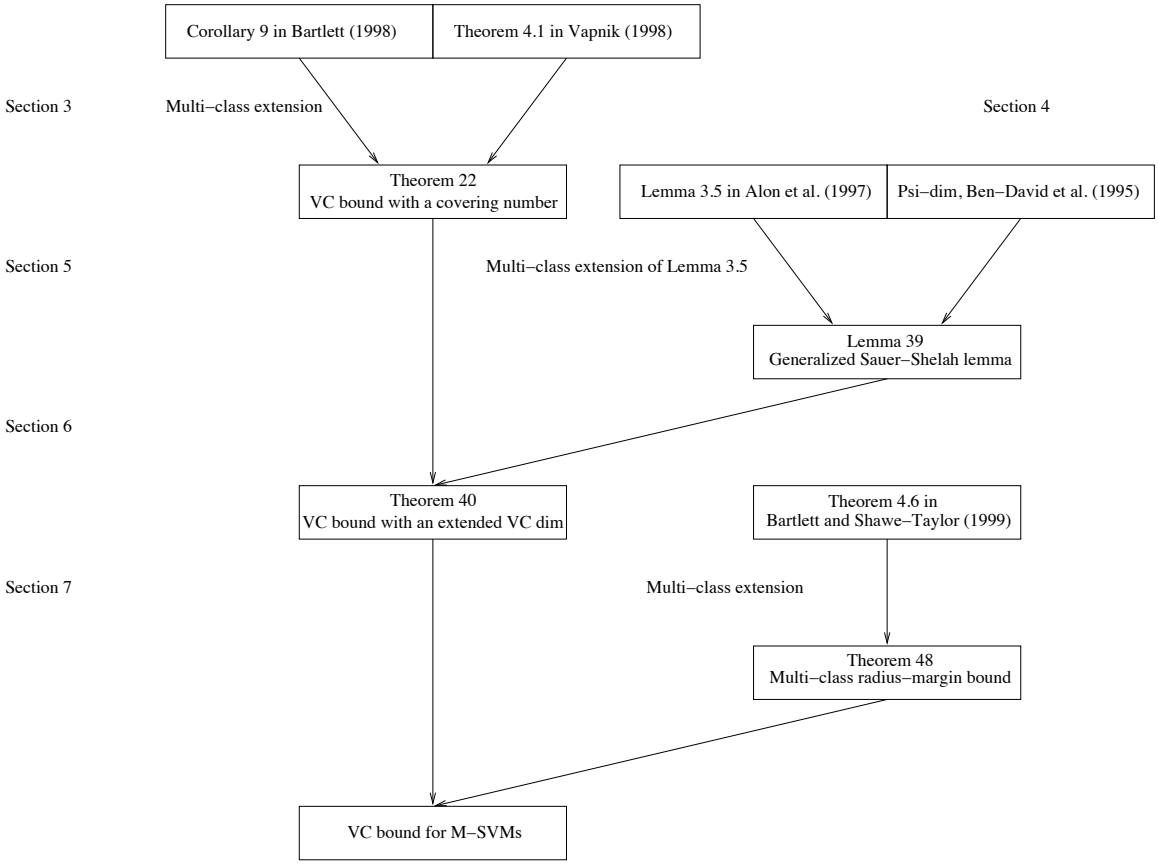


Figure 1: Organigram of the results of the paper.

Although the theorems of this theoretical contribution take the form of guaranteed risks, our aim is not to derive a tight bound on the risk, but rather to highlight the instructive features of the unification, and some specificities of the multi-class case. In that sense, our work is similar in spirit to the one exposed in Tewari and Bartlett (2007). We are all interested in the way a convergence can happen in the multi-class case. They consider the problem from the point of view of the training algorithm, whereas we focus on the capacity of the class of functions. In short, the new theory can be derived by extending concepts and results from only three famous papers: Ben-David et al. (1995), Alon et al. (1997) and Bartlett (1998), plus a fourth reference, Bartlett and Shawe-Taylor (1999), to

treat specifically the case of M-SVMs. This derivation appears rather straightforward once one has understood that different descriptors of the behaviour of the class of functions of interest are to be taken into account at the different steps of the reasoning, and this calls for the application of two different "margin operators" to this class. This phenomenon, a specificity of the multi-class case, is most noticeable at the level of the generalized Sauer-Shelah lemma, where the transition between the two operators is performed.

The organization of the paper is as follows. Section 2 introduces the notion of multi-class margin and margin risk for multi-class discriminant models, as well as the capacity measure that will appear in the confidence interval of the basic guaranteed risk, a covering number. Section 3 is then devoted to the formulation of this risk and its discussion. The γ-Ψ-dimensions are introduced in Section 4. The extension of Sauer's lemma relating the covering number of interest to one of the γ-Ψ-dimensions, the margin Natarajan dimension, is established in Section 5. Our master theorem, a combination of the basic convergence result and the aforementioned lemma, is then exposed in Section 6. Section 7 is devoted to the computation of a bound on the margin Natarajan dimension of the architecture shared by all the M-SVMs. In Section 8, the synthesis of the results derived in the preceding sections is performed, underlining the specificities of the multi-class case. This section also highlights the usefulness of our uniform convergence result for model selection. At last, we draw conclusions and outline our ongoing research in Section 9.

2. Margin Risk for Multi-Category Discriminant Models

In this section, the theoretical framework of the study is introduced. It is based on a notion of margin generalizing to an arbitrary (but finite) number of categories the standard (bi-class) one.

2.1 Formalization of the Learning Problem

We consider the case of a Q-category pattern recognition problem, with $3 \leq Q < \infty$ (so that the degenerate case of dichotomies is a priori excluded). A pattern is represented by its description $x \in \mathcal{X}$ and the set of categories $\mathcal{Y}$ is identified with the set of indexes of the categories, $[\![1,Q]\!]$. The link between patterns and categories is supposed to be of probabilistic nature. We make the assumption that $\mathcal{X}$, $\mathcal{Y}$ and the product space $\mathcal{X} \times \mathcal{Y}$ are probability spaces, and $\mathcal{X} \times \mathcal{Y}$ is endowed with a probability measure P, fixed but unknown. The measure P completely characterizes the problem of interest. In the PAC framework, this standard setting is known as *probabilistic concept learning* (Kearns and Schapire, 1994). Hereafter, $\mathcal{Z}$ will designate the product space $\mathcal{X} \times \mathcal{Y}$, and $z = (x,y)$ its elements. Our goal is to find, in a given set $\mathcal{G}$ of functions $g = (g_k)_{1 \leq k \leq Q}$ from $\mathcal{X}$ into $\mathbb{R}^Q$, a function classifying data in an optimal way. Let (X,Y) be a random pair distributed according to P. The function selection procedure, or training, makes use of a m-sample $D_m = ((X_i,Y_i))_{1 \leq i \leq m}$ of independent copies of (X,Y). It consists in trying to optimize over $\mathcal{G}$ a criterion, called the *(expected) risk*, which is the expectation with respect to P of a given *loss function*. At this point, the properties of the functions in $\mathcal{G}$ and the way they perform classification must be specified. They are supposed to satisfy some measurability conditions that will appear implicitly in the sequel (see Dudley, 1984, Chap. 10 for a detailed study of the question in a similar context), plus the constraint $\sum_{k=1}^{Q} g_k = 0$ (the purpose of this constraint will appear later). g assigns $x \in \mathcal{X}$ to the category l if and only if $g_l(x) > \max_{k \neq l} g_k(x)$. In case of ex æquo, x is assigned to a dummy category denoted by $*$. Let f be the decision function (from $\mathcal{X}$ into $\mathcal{Y} \bigcup \{*\}$) associated with g. The criterion to be

optimized is the probability of error $P(f(X) \neq Y)$. This calls for the choice of the following loss function.

Definition 1 (Multi-Class loss) *Let ℓ, the* multi-class loss function, *be defined on $\mathcal{Y} \times \mathbb{R}^Q$ by:*

$$\forall (y,v) \in \mathcal{Y} \times \mathbb{R}^Q,\ \ell(y,v) = \mathbb{1}_{\left\{v_y \leq \max_{k \neq y} v_k\right\}}$$

where $\mathbb{1}$ is the indicator function, which takes the value 1 if its argument is true, and 0 otherwise.

ℓ is simply the 0-1 loss in the multi-class setting. The expected risk of a function g is consequently defined as follows.

Definition 2 (Expected risk) *The* expected risk *of a function $g \in \mathcal{G}$, $R(g)$, is given by:*

$$R(g) = \mathbb{E}\left[\ell(Y, g(X))\right] = \int_{\mathcal{Z}} \mathbb{1}_{\left\{g_y(x) \leq \max_{k \neq y} g_k(x)\right\}} dP(z).$$

The empirical risk is simply the estimate of the risk computed on the training sample.

Definition 3 (Empirical risk) *The empirical risk of $g \in \mathcal{G}$ measured on a m-sample, $R_m(g)$, is the random variable given by:*

$$R_m(g) = \frac{1}{m} \sum_{i=1}^{m} \mathbb{1}_{\left\{g_{Y_i}(X_i) \leq \max_{k \neq Y_i} g_k(X_i)\right\}}.$$

When needed, the m-sample used will be specified, by writing for instance $R_{D_m}(g)$ in place of $R_m(g)$. Let $n \in \mathbb{N}^* = \mathbb{N} \setminus \{0\}$ and let $z^n = ((x_i, y_i))_{1 \leq i \leq n} \in \mathcal{Z}^n$. In the sequel, $R_{z^n}(g)$ will designate the frequency of errors $\frac{1}{n} \sum_{i=1}^{n} \mathbb{1}_{\left\{g_{y_i}(x_i) \leq \max_{k \neq y_i} g_k(x_i)\right\}}$.

2.2 Multi-Class Margin and Multi-Class Margin Risk

For the classes of vector-valued functions we are interested in, the two elements which are the most important to assign a pattern to a category and to derive a level of confidence in this assignment are the index of the highest output and the difference between this output and the second highest one. This calls for the use of a measure different from the standard indicator function ℓ to assess the quality of a discrimination. This measure can be built around a notion of multi-class margin which has been studied independently by different groups of authors (see for instance Elisseeff et al., 1999; Allwein et al., 2000). To define it, we first define an auxiliary function.

Definition 4 (Function M) *Let M be the function from $\mathbb{R}^Q \times [\![1, Q]\!]$ to $\mathbb{R}$ defined as:*

$$\forall (v,k) \in \mathbb{R}^Q \times [\![1, Q]\!],\ M(v,k) = \frac{1}{2}\left(v_k - \max_{l \neq k} v_l\right).$$

Let $M(v,.) = \max_{1 \leq k \leq Q} M(v,k)$.

Definition 5 (Multi-Class margin) *Let g be a function of a class $\mathcal{G}$. Its* margin *on $(x,y) \in \mathcal{X} \times \mathcal{Y}$ is defined to be $M(g(x), y)$.*

To take this margin into account, the following operators are introduced:

Definition 6 (Δ operator) *Define Δ as an operator on $\mathcal{G}$ such that:*

$$\begin{aligned}\Delta: \quad & \mathcal{G} \longrightarrow \Delta\mathcal{G}, \\ & g \mapsto \Delta g = (\Delta g_k)_{1\leq k\leq Q},\end{aligned}$$

$$\forall x \in \mathcal{X},\ \Delta g(x) = (M(g(x),k))_{1\leq k\leq Q}.$$

For the sake of simplicity, we write Δg_k in place of $(\Delta g)_k$. In the sequel, similar simplifications will be performed implicitly with other operators.

Definition 7 (Δ^* operator) *Define Δ^* as an operator on $\mathcal{G}$ such that:*

$$\begin{aligned}\Delta^*: \quad & \mathcal{G} \longrightarrow \Delta^*\mathcal{G} \\ & g \mapsto \Delta^* g = (\Delta^* g_k)_{1\leq k\leq Q}\end{aligned}$$

$$\forall x \in \mathcal{X},\ \Delta^* g(x) = (\max(\Delta g_k(x), -M(g(x),.)))_{1\leq k\leq Q}.$$

Remark 8 *If $M(g(x),.) > 0$, $\Delta g(x)$ has a unique (strictly) positive component, otherwise it has none. Let us consider the first case, and let $k^* = \operatorname{argmax}_{1\leq k\leq Q} \Delta g_k(x) = \operatorname{argmax}_{1\leq k\leq Q} g_k(x)$ $(\Delta g_{k^*}(x) = M(g(x),.))$.*

$$\forall x \in \mathcal{X},\ \begin{cases} \text{if } M(g(x),.) > 0, & \Delta^* g(x) = ((2\delta_{k,k^*} - 1)\,\Delta g_{k^*}(x))_{1\leq k\leq Q} \\ \text{if } M(g(x),.) = 0, & \Delta^* g(x) = 0 \end{cases}.$$

where δ is the Kronecker symbol.

Example 1 *Suppose that $g(x) = (-0.1, 0.6, -0.3, -0.2)$. Then*

$$\begin{cases} \Delta g(x) = (-0.35, 0.35, -0.45, -0.4) \\ \Delta^* g(x) = (-0.35, 0.35, -0.35, -0.35) \end{cases}.$$

Before proceeding, it is useful to highlight the way those definitions relate to the bi-class case. Let $\tilde{\mathcal{G}}$ denote the class of real-valued functions implemented by a large margin bi-class classifier. There is a one-to-one map from this class onto a class $\mathcal{G}$ as defined above. To each function $\tilde{g}$ in $\tilde{\mathcal{G}}$, a function $g = (g_1, g_2)$ in $\mathcal{G}$ can be associated such that $g_1 = \tilde{g} = -g_2$. This is precisely to ensure the existence of this one-to-one map that the constraint $\sum_{k=1}^{Q} g_k = 0$ has been introduced. Then, $\Delta g = \Delta^* g = g = (\tilde{g}, -\tilde{g})$. As a consequence, one can consider that when implementing a large margin bi-class classifier, the functions effectively handled are the component functions Δg_1 (or equivalently the component functions $\Delta^* g_1$). In the sequel, $\Delta^\#$ is used in place of Δ and Δ^* in the formulas that hold true for both operators. Obviously, the first of these formulas is the one connecting the risk of g with the behaviour of $\Delta^\# g$.

Proposition 9 *The risk of a function g of $\mathcal{G}$ can be expressed as:*

$$R(g) = \mathbb{E}\left[\mathbb{1}_{\{\Delta^\# g_Y(X)\leq 0\}}\right].$$

With these definitions at hand, the margin risk is defined as follows.

Definition 10 (Margin risk) *Let* $\gamma \in \mathbb{R}_+^* = (0,\infty)$. *The* risk with margin γ *of a function* g *of* $\mathcal{G}$, $R_\gamma(g)$, *is defined as:*

$$R_\gamma(g) = \mathbb{E}\left[\mathbb{1}_{\{\Delta^\# g_{Y}(X)<\gamma\}}\right].$$

The empirical risk with margin γ of g, $R_{\gamma,m}(g)$ (or $R_{\gamma,D_m}(g)$ if the sample needs to be specified), and the frequency of errors with margin γ, $R_{\gamma,z^n}(g)$, are defined accordingly.

A consequence of the definition of the margin risk is the fact that knowing the exact behaviour of the component functions $\Delta^\# g_k$ below $-\gamma$ and over γ is useless. On the contrary, one can take benefit from working with classes of functions taking values in $[-\gamma,\gamma]^Q$, which is compact, rather than in $\mathbb{R}^Q$. This advantage will appear in the first place in Section 3, and then more clearly in Section 5. Such a transform is achieved by application of the following piecewise-linear squashing operator.

Definition 11 (π_γ operator, Bartlett, 1998) *For* $\gamma \in \mathbb{R}_+^*$, *define* π_γ *as an operator on* $\mathcal{G}$ *such that:*

$$\begin{aligned}\pi_\gamma : \quad & \mathcal{G} \longrightarrow \pi_\gamma \mathcal{G}, \\ & g \mapsto \pi_\gamma g = \left(\pi_\gamma g_k\right)_{1\le k\le Q},\end{aligned}$$

$$\forall x \in \mathcal{X},\ \pi_\gamma g(x) = \left(sign\left(g_k(x)\right)\cdot \min\left(|g_k(x)|,\gamma\right)\right)_{1\le k\le Q}$$

where the sign function is defined by $sign(t) = 1$ *if* $t \ge 0$, *and* $sign(t) = -1$ *otherwise*.

For $\gamma \in \mathbb{R}_+^*$, let $\Delta_\gamma^\#$ denote $\pi_\gamma \circ \Delta^\#$ and $\Delta_\gamma^\# \mathcal{G} = \left\{\Delta_\gamma^\# g : g \in \mathcal{G}\right\}$.

The capacity measure that will appear in the basic guaranteed risk stated in Section 3 is a covering number. Its definition, and the definition of related concepts, is the subject of the following section. Introductions to the basic notions of functional analysis used in this article can be found in Carl and Stephani (1990), Devroye et al. (1996) and van der Vaart and Wellner (1996).

2.3 Capacity Measures: Covering and Packing Numbers

The notion of covering number is based on the notions of ε-cover and ε-net.

Definition 12 (ε-cover and ε-net, Kolmogorov and Tihomirov, 1961) *Let* (E,ρ) *be a pseudo-metric space. For* $e \in E$ *and* $r \in \mathbb{R}_+^*$, *let* $B(e,r)$ *be the open ball of center* e *and radius* r *in* E. *Let* E' *be a subset of* E. *For* $\varepsilon \in \mathbb{R}_+^*$, *an* ε-net *of* E' *is a subset* $\overline{E'}$ *of* E *such that:*

$$E' \subset \bigcup_{e\in\overline{E'}} B(e,\varepsilon).$$

$\bigcup_{e\in\overline{E'}} B(e,\varepsilon)$ *is then an* ε-cover *of* E'. $\overline{E'}$ *is a* proper ε-net *of* E' *if it is included in* E'.

Definition 13 (Covering number, Kolmogorov and Tihomirov, 1961) *Let* (E,ρ) *be a pseudo-metric space. For* $\varepsilon \in \mathbb{R}_+^*$, *if* $E' \subset E$ *has an* ε*-net of finite cardinality, then its* covering number $\mathcal{N}(\varepsilon,E',\rho)$ *is the smallest cardinality of its* ε*-nets. If there is no such finite net, then the covering number is defined to be* ∞. *We denote* $\mathcal{N}^{(p)}(\varepsilon,E',\rho)$ *the covering number obtained by considering proper* ε*-nets only.*

Hereafter, the pseudo-metric that will be used on the families of functions considered is the following one:

Definition 14 (d_{x^n} pseudo-metric) *Let $n \in \mathbb{N}^*$. For a sequence $x^n = (x_i)_{1\leq i\leq n} \in \mathcal{X}^n$, define the pseudo-metric d_{x^n} on $\mathcal{G}$ as:*

$$\forall (g,g') \in \mathcal{G}^2,\ d_{x^n}(g,g') = \max_{1\leq i\leq n} \left\| g(x_i) - g'(x_i) \right\|_\infty .$$

Definition 15 $\forall n \in \mathbb{N}^*, \forall \varepsilon \in \mathbb{R}_+^*$,

$$\mathcal{N}^{(p)}(\varepsilon, \mathcal{G}, n) = \max_{x^n \in \mathcal{X}^n} \mathcal{N}^{(p)}(\varepsilon, \mathcal{G}, d_{x^n}),$$

the maximum being used in place of a supremum to highlight the fact that we implicitly make the assumption that all the ε-nets considered are of finite cardinality.

There is a close connection between covering and packing properties of bounded subsets in pseudo-metric spaces.

Definition 16 (ε-separation and packing number, Kolmogorov and Tihomirov, 1961) *Let (E,ρ) be a pseudo-metric space and $\varepsilon \in \mathbb{R}_+^*$. A set $E' \subset E$ is* ε-separated *if, for any distinct points e_1 and e_2 in E', $\rho(e_1,e_2) \geq \varepsilon$. The* ε-packing number *of $E'' \subset E$, $\mathcal{M}(\varepsilon, E'', \rho)$, is the maximal size of an ε-separated subset of E''.*

Definition 17 (Separation) *For $n \in \mathbb{N}^*$, let $\mathcal{F}$ be a class of functions on $\mathcal{X}$ taking their values in $[\![-n,n]\!]^Q$ and $\mathcal{F}|_{\mathcal{D}}$ its restriction to a subset $\mathcal{D}$ of $\mathcal{X}$ of finite cardinality. Two functions f and f' in the class $\mathcal{F}|_{\mathcal{D}}$ are* separated *if they are 2-separated in the pseudo-metric $d_{\mathcal{D}}$, that is, if*

$$\max_{x\in\mathcal{D}} \left\| f(x) - f'(x) \right\|_\infty \geq 2.$$

Definition 18 (Pairwise separated set of functions) *Let $\mathcal{F}$, $\mathcal{D}$ and $\mathcal{F}|_{\mathcal{D}}$ be defined as above. $\mathcal{F}|_{\mathcal{D}}$ is* pairwise separated *if any two distinct functions of $\mathcal{F}|_{\mathcal{D}}$ are separated.*

2.4 Additional Definitions

This section gathers definitions which will be used in the proof of our basic uniform convergence result.

Definition 19 ($\overline{\mathcal{G}}(\gamma,x^n)$ and $\overline{\mathcal{G}}(\gamma,D_n)$) *Let $n \in \mathbb{N}^*$ and $\gamma \in \mathbb{R}_+^*$. Let $x^n = (x_i)_{1\leq i\leq n} \in \mathcal{X}^n$. Let us consider any function (deterministic algorithm) f_{net} that takes as input γ, $\mathcal{G}$, and x^n, and returns a subset of $\mathcal{G}$ such that its image by the operator $\Delta_\gamma^\#$ is a proper $\gamma/2$-net of the set $\Delta_\gamma^\#\mathcal{G}$ (in the pseudo-metric d_{x^n}), and this net is of minimal cardinality, that is, of cardinality $\mathcal{N}^{(p)}\left(\gamma/2, \Delta_\gamma^\#\mathcal{G}, d_{x^n}\right)$.*

$$\overline{\mathcal{G}}(\gamma,x^n) = f_{net}(\gamma, \mathcal{G}, x^n).$$

The random variable $\overline{\mathcal{G}}(\gamma,D_n)$ is defined accordingly, by replacing in the definition of $\overline{\mathcal{G}}(\gamma,x^n)$ the sequence x^n with $(X_i)_{1\leq i\leq n}$.

Note that for the sake of simplicity, we use $\overline{\mathcal{G}}(\gamma, D_n)$ in place of $\overline{\mathcal{G}}\left(\gamma, (X_i)_{1\leq i\leq n}\right)$, although the latter formulation is more precise.

Definition 20 (Swapping group $\mathfrak{T}_{2n}$) *For $n \in \mathbb{N}^*$, let $\mathfrak{T}_{2n}$ be the "swapping" subgroup of $\mathfrak{S}_{2n}$, the symmetric group of degree $2n$. $\mathfrak{T}_{2n}$ is the set of all permutations σ over $[\![1,2n]\!]$ that swap i and $n+i$ for all i in some subset of $[\![1,n]\!]$. Precisely, for all i in $[\![1,n]\!]$, $(\sigma(i),\sigma(i+n))$ is either equal to $(i,i+n)$ or to $(i+n,i)$. The permutations σ are regarded as acting on coordinates. For $z^{2n} \in \mathcal{Z}^{2n}$ and $\sigma \in \mathfrak{T}_{2n}$, let $\sigma\left(z^{2n}\right) = \left(\left(x_{\sigma(i)}, y_{\sigma(i)}\right)\right)_{1\leq i\leq 2n}$. $\mathfrak{T}_{2n}$ is endowed with a uniform probability distribution.*

Definition 21 (Bernoulli/Rademacher sequence) *For $n \in \mathbb{N}^*$, a* Bernoulli or Rademacher sequence *is a sequence $\alpha = (\alpha_i)_{1\leq i\leq n}$ of independent real random variables with $\mathbb{P}(\alpha_i = -1) = \mathbb{P}(\alpha_i = 1) = \frac{1}{2}$ for all i.*

3. Uniform Convergence of the Empirical Margin Risk

With the hypotheses and definitions of the previous section at hand, we prove the following uniform convergence result.

3.1 Basic Uniform Convergence Result

Theorem 22 *Let $\mathcal{G}$ be the class of functions that a large margin Q-category classifier on a domain $\mathcal{X}$ can implement. Let $\Gamma \in \mathbb{R}_+^*$ and $\delta \in (0,1)$. With probability at least $1-\delta$, for every value of γ in $(0,\Gamma]$, the risk of any function g in $\mathcal{G}$ is bounded from above by:*

$$R(g) \leq R_{\gamma,m}(g) + \sqrt{\frac{2}{m}\left(\ln\left(2\mathcal{N}^{(p)}\left(\gamma/4, \Delta_\gamma^{\#}\mathcal{G}, 2m\right)\right) + \ln\left(\frac{2\Gamma}{\gamma\delta}\right)\right)} + \frac{1}{m}.$$

The proof is given in Appendix B. This theorem can be seen as a multi-class extension of Corollary 9 in Bartlett (1998). Indeed, setting $Q = 2$ (and $\Gamma = 1$), we get a slightly improved version of this corollary. The difference rests on the fact that in the first symmetrization, we took advantage of an idea which is implicitly at the basis of Formula (4.28) in Vapnik (1998). This idea consists in making use of Lemma 49. As a consequence, Theorem 22 can also be seen as a specification for the case of large margin multi-category classifiers of Theorem 4.1 in Vapnik (1998).

3.2 Choice of the Margin Operator

Theorem 22 has been derived for both margin operators, Δ and Δ^*. The choice between them should thus rest on the use which is done of the bound, that is, on the nature of the pathway followed to bound from above the covering number of interest. This question, the nature of which is primarily technical, will turn out to be of central importance in the following sections. At this point, we can already notice that the Δ^* operator provides less information on the behaviour of the function on which it is applied than the Δ operator. Such a difference would appear as an advantage to derive a generalization of Sauer's lemma, and a drawback to compute an upper bound on the corresponding generalized VC dimension. This suggests to implement a hybrid strategy, mixing results involving Δ^* with results involving Δ. This is precisely what will be done here.

4. γ-Ψ-dimensions: the Generalized VC Dimensions of Large Margin Multi-Category Classifiers

Several approaches can be applied to bound from above the covering number of interest for a given class of functions $\mathcal{G}$. The standard one, introduced in Vapnik and Chervonenkis (1971), consists in involving in the process the VC dimension, or one of its extensions. If VC dimensions appear useful in practice, their interest is primarily of theoretical nature. Indeed, they characterize learnability in different settings (see for instance Alon et al., 1997). In this section, the γ-Ψ-dimensions are introduced as the generalized VC dimensions suited for large margin multi-category classifiers. They appear as syntheses of the Ψ-dimensions and the fat-shattering dimension (also known as the γ-dimension). The pertinence of this specification will be established in Section 5.

The basic result relating a covering number (precisely the growth function) to the VC dimension is the Sauer-Shelah lemma (Vapnik and Chervonenkis, 1971; Sauer, 1972; Shelah, 1972). As stated in the introduction, extensions of the standard VC theory, which only deals with the computation of dichotomies with indicator functions, have mainly been proposed for large margin bi-class discriminant models and multi-class discriminant models taking their values in finite sets. In both cases, generalized Sauer-Shelah lemmas have been derived (see for instance Haussler and Long, 1995; Alon et al., 1997), which involve extended notions of VC dimension. For large margin bi-class discriminant models, the generalization of the VC dimension which has given birth to the richest set of theoretical results is a scale-sensitive variant called the fat-shattering dimension (Kearns and Schapire, 1994). In the multi-class case, several alternative solutions were proposed by different authors, such as the graph dimension (Dudley, 1987; Natarajan, 1989), or the Natarajan dimension (Natarajan, 1989). It was proved in Ben-David et al. (1995) that most of these extensions could be gathered in a general scheme, which makes it possible to derive necessary and sufficient conditions for PAC learning (Valiant, 1984). In this scheme, they appear as special cases of Ψ-dimensions.

We introduce scale-sensitive extensions of the Ψ-dimensions. The underlying idea is simple: in the same way as scale-sensitive extensions of the VC dimension, such as the fat-shattering dimension, make it possible to study the generalization capabilities of bi-class discriminant models taking their values in $\mathbb{R}$, scale-sensitive extensions of the Ψ-dimensions should make it possible to study the generalization capabilities of Q-class discriminant models taking their values in $\mathbb{R}^Q$.

4.1 Ψ-dimensions

Definition 23 (Ψ-dimensions, Ben-David et al., 1995) *Let $\mathcal{F}$ be a class of functions on a set $\mathcal{X}$ taking their values in the finite set $[\![1,Q]\!]$. Let Ψ be a family of mappings ψ from $[\![1,Q]\!]$ into $\{-1,1,*\}$, where $*$ is thought of as a null element. A subset $s_{\mathcal{X}^n} = \{x_i : 1 \leq i \leq n\}$ of $\mathcal{X}$ is said to be* Ψ-shattered *by $\mathcal{F}$ if there is a mapping $\psi^n = \left(\psi^{(i)}\right)_{1\leq i\leq n}$ in Ψ^n such that for each vector v_y in $\{-1,1\}^n$, there is a function f_y in $\mathcal{F}$ satisfying*

$$\left(\psi^{(i)} \circ f_y(x_i)\right)_{1\leq i\leq n} = v_y.$$

The Ψ-dimension *of $\mathcal{F}$, denoted by Ψ-dim$(\mathcal{F})$, is the maximal cardinality of a subset of $\mathcal{X}$ Ψ-shattered by $\mathcal{F}$, if this cardinality is finite. If no such maximum exists, $\mathcal{F}$ is said to have infinite Ψ-dimension.*

Remark 24 *Let $\mathcal{F}$ and Ψ be defined as above. Extending the definition of the standard VC dimension, VC-dim, so that it applies to classes of functions taking values in $\{-1,1,*\}$, which has no incidence in practice, the following proposition holds true:*

$$\Psi\text{-}dim(\mathcal{F}) = VC\text{-}dim\left(\{(x,\psi) \mapsto \psi \circ f(x) : f \in \mathcal{F}, \psi \in \Psi\}\right).$$

In words, the idea common to all these dimensions is to introduce adequately chosen mappings from $[\![1,Q]\!]$ into $\{-1,1,*\}$ so that the problem of the computation of the capacity measure boils down to the computation of several standard VC dimensions. In that context, the motivation for the choice of one particular dimension (set Ψ) utterly rests on the possibility to derive two tight bounds: a generalized Sauer-Shelah lemma and a bound on the dimension itself. The most frequently used Ψ-dimension is the graph dimension.

Definition 25 (Graph dimension, Natarajan, 1989) *Let $\mathcal{F}$ be a class of functions on a set $\mathcal{X}$ taking their values in $[\![1,Q]\!]$. The* graph dimension *of $\mathcal{F}$, G-dim$(\mathcal{F})$, is the Ψ-dimension of $\mathcal{F}$ in the specific case where $\Psi = \{\psi_k : 1 \leq k \leq Q\}$, such that ψ_k takes the value 1 if its argument is equal to k, and the value -1 otherwise. Reformulated in the context of multi-class discriminant analysis, the functions ψ_k are the indicator functions of the categories.*

Obviously, this notion of Ψ-dimension is connected with one of the standard decomposition schemes implemented to tackle multi-class problems with bi-class classifiers: the *one-against-all* method. Another popular decomposition scheme is the *one-against-one* method. The corresponding Ψ-dimension has been proposed by Natarajan.

Definition 26 (Natarajan dimension, Natarajan, 1989) *Let $\mathcal{F}$ be a class of functions on a set $\mathcal{X}$ taking their values in $[\![1,Q]\!]\}$. The* Natarajan dimension *of $\mathcal{F}$, N-dim$(\mathcal{F})$, is the Ψ-dimension of $\mathcal{F}$ in the specific case where $\Psi = \{\psi_{k,l} : 1 \leq k \neq l \leq Q\}$, such that $\psi_{k,l}$ takes the value 1 if its argument is equal to k, the value -1 if its argument is equal to l, and $*$ otherwise.*

4.2 Margin Ψ-dimensions

Our scale-sensitive version of the concept of Ψ-dimension is devised so that the corresponding dimensions can alternatively be seen as multivariate extensions of the fat-shattering dimension.

Definition 27 (Fat-shattering dimension, Kearns and Schapire, 1994) *Let $\mathcal{G}$ be a class of real-valued functions on a set $\mathcal{X}$. For $\gamma \in \mathbb{R}_+^*$, a subset $s_{\mathcal{X}^n} = \{x_i : 1 \leq i \leq n\}$ of $\mathcal{X}$ is said to be* γ-shattered *by $\mathcal{G}$ if there is a vector $v_b = (b_i) \in \mathbb{R}^n$ such that, for each vector $v_y = (y_i)$ in $\{-1,1\}^n$, there is a function g_y in $\mathcal{G}$ satisfying*

$$\forall i \in [\![1,n]\!],\ y_i\left(g_y(x_i) - b_i\right) \geq \gamma. \tag{1}$$

The fat-shattering dimension with margin γ, *or P_γ* dimension, *of the class $\mathcal{G}$, P_γ-dim$(\mathcal{G})$, is the maximal cardinality of a subset of $\mathcal{X}$ γ-shattered by $\mathcal{G}$, if this cardinality is finite. If no such maximum exists, $\mathcal{G}$ is said to have infinite P_γ dimension.*

Let $\wedge$ denote the conjunction of two events. With these definitions at hand, the Ψ-dimensions with margin γ, or γ-Ψ-dimensions, are defined as follows:

Definition 28 (γ-Ψ-dimensions) *Let $\mathcal{G}$ be a class of functions on a set $\mathcal{X}$ taking their values in $\mathbb{R}^Q$. Let Ψ be a family of mappings ψ from $[\![1,Q]\!]$ into $\{-1,1,*\}$. For $\gamma \in \mathbb{R}_+^*$, a subset $s_{\mathcal{X}^n} = \{x_i : 1 \leq i \leq n\}$ of $\mathcal{X}$ is said to be* γ-Ψ-shattered *(*Ψ-shattered with margin γ*) by $\Delta^\# \mathcal{G}$ if there is a mapping $\psi^n = \left(\psi^{(i)}\right)_{1 \leq i \leq n}$ in Ψ^n and a vector $v_b = (b_i)$ in $\mathbb{R}^n$ such that, for each vector $v_y = (y_i)$ in $\{-1,1\}^n$, there is a function g_y in $\mathcal{G}$ satisfying*

$$\forall i \in [\![1,n]\!], \begin{cases} \text{if } y_i = \quad 1, & \exists k : \psi^{(i)}(k) = \quad 1 \quad \wedge \ \Delta^\# g_{y,k}(x_i) - b_i \geq \gamma \\ \text{if } y_i = -1, & \exists l : \psi^{(i)}(l) = -1 \quad \wedge \ \Delta^\# g_{y,l}(x_i) + b_i \geq \gamma \end{cases} . \tag{2}$$

The γ-Ψ-dimension*, or* Ψ-dimension with margin γ*, of $\Delta^\# \mathcal{G}$, denoted by $\Psi\text{-}dim\left(\Delta^\# \mathcal{G}, \gamma\right)$, is the maximal cardinality of a subset of $\mathcal{X}$ γ-Ψ-shattered by $\Delta^\# \mathcal{G}$, if this cardinality is finite. If no such maximum exists, $\Delta^\# \mathcal{G}$ is said to have infinite γ-Ψ-dimension.*

From a theoretical point of view, the one-against-one decomposition method exhibits an advantage over the one-against-all decomposition method: its use makes it easier to extend to the multi-class case bi-class theorems, by application of the pigeonhole principle. Thus, the scale-sensitive Ψ-dimension which will be involved in our generalized Sauer-Shelah lemma is the one extending the Natarajan dimension. Given the definitions of the Natarajan dimension and the scale-sensitive Ψ-dimensions, it can be formulated as:

Definition 29 (Natarajan dimension with margin γ) *Let $\mathcal{G}$ be a class of functions on a set $\mathcal{X}$ taking their values in $\mathbb{R}^Q$. For $\gamma \in \mathbb{R}_+^*$, a subset $s_{\mathcal{X}^n} = \{x_i : 1 \leq i \leq n\}$ of $\mathcal{X}$ is said to be* γ-N-shattered *(*N-shattered with margin γ*) by $\Delta^\# \mathcal{G}$ if there is a set*

$$I(s_{\mathcal{X}^n}) = \{(i_1(x_i), i_2(x_i)) : 1 \leq i \leq n\}$$

of n couples of distinct indexes in $[\![1,Q]\!]$ and a vector $v_b = (b_i)$ in $\mathbb{R}^n$ such that, for each vector $v_y = (y_i)$ in $\{-1,1\}^n$, there is a function g_y in $\mathcal{G}$ satisfying

$$\forall i \in [\![1,n]\!], \begin{cases} \text{if } y_i = \quad 1, & \Delta^\# g_{y,i_1(x_i)}(x_i) - b_i \geq \gamma \\ \text{if } y_i = -1, & \Delta^\# g_{y,i_2(x_i)}(x_i) + b_i \geq \gamma \end{cases} .$$

The Natarajan dimension with margin γ *of the class $\Delta^\# \mathcal{G}$, N-dim $\left(\Delta^\# \mathcal{G}, \gamma\right)$, is the maximal cardinality of a subset of $\mathcal{X}$ γ-N-shattered by $\Delta^\# \mathcal{G}$, if this cardinality is finite. If no such maximum exists, $\Delta^\# \mathcal{G}$ is said to have infinite Natarajan dimension with margin γ.*

4.3 Discussion

In the preceding section, we have given a formulation of the definition of the Natarajan dimension which is inspired from the one in Ben-David et al. (1995) (the definition in Natarajan, 1989, does not involve the $\psi_{k,l}$ mappings). This formulation can be restricted by considering only the mappings $\psi_{k,l}$ such that $k < l$, instead of $k \neq l$. This is possible due to the symmetrical roles played by the indexes of categories $i_1(x_i)$ and $i_2(x_i)$ in the definition. As a consequence, the cardinality of the set Ψ considered can be divided by 2 (reduced from $Q(Q-1)$ to $\binom{Q}{2}$). This is useful indeed, since many theorems dealing with Ψ-dimensions involve the cardinality of Ψ (see for instance Theorem 7 in Ben-David et al., 1995). An equivalent simplification can be performed in the case of the margin Natarajan dimension.

Proposition 30 *The definition of the Natarajan dimension with margin γ is not affected by the introduction of the additional constraint: $\forall i \in [\![1,n]\!]$, $i_1(x_i) < i_2(x_i)$.*

Proof Let $\mathcal{G}_y$ be a subset of $\mathcal{G}$ of cardinality 2^n such that $\Delta^{\#}\mathcal{G}_y$ γ-N-shatters $s_{\mathcal{X}^n}$ with respect to $I(s_{\mathcal{X}^n})$ and v_b. Let $I'(s_{\mathcal{X}^n})$ be the set of n couples of indexes $(i'_1(x_i), i'_2(x_i))$ deduced from $I(s_{\mathcal{X}^n})$ by reordering its elements, that is,

$$\forall i \in [\![1,n]\!],\ \left(i'_1(x_i), i'_2(x_i)\right) = \left(\min\left(i_1(x_i), i_2(x_i)\right), \max\left(i_1(x_i), i_2(x_i)\right)\right).$$

Let $v_{b'} = (b'_i)$ be the vector of $\mathbb{R}^n$ deduced from v_b as follows: $\forall i \in [\![1,n]\!]$, $b'_i = b_i$ if $(i'_1(x_i), i'_2(x_i)) = (i_1(x_i), i_2(x_i))$, $b'_i = -b_i$ otherwise. We establish that $\Delta^{\#}\mathcal{G}_y$ still γ-N-shatters $s_{\mathcal{X}^n}$ with respect to $I'(s_{\mathcal{X}^n})$ and $v_{b'}$. For any vector $v_y = (y_i)$ of $\{-1,1\}^n$, let $g_{y'}$ be the function in $\mathcal{G}_y$ such that $\Delta^{\#}g_{y'}$ "contributes" to the γ-N-shattering of $s_{\mathcal{X}^n}$ with respect to $I(s_{\mathcal{X}^n})$ and v_b for a value of the binary vector equal to $v_{y'} = (y'_i)$, where $y'_i = y_i$ if $(i'_1(x_i), i'_2(x_i)) = (i_1(x_i), i_2(x_i))$, $y'_i = -y_i$ otherwise. According to Definition 29,

$$\forall i \in [\![1,n]\!],\ \left\{ \begin{array}{lll} \text{if } y'_i = & 1, & \Delta^{\#}g_{y',i_1(x_i)}(x_i) - b_i \geq \gamma \\ \text{if } y'_i = & -1, & \Delta^{\#}g_{y',i_2(x_i)}(x_i) + b_i \geq \gamma \end{array} \right. .$$

As a consequence, for the set of indexes i such that $(i'_1(x_i), i'_2(x_i)) = (i_1(x_i), i_2(x_i))$,

$$\left\{ \begin{array}{lll} \text{if } y_i = & 1, & \Delta^{\#}g_{y',i'_1(x_i)}(x_i) - b'_i \geq \gamma \\ \text{if } y_i = & -1, & \Delta^{\#}g_{y',i'_2(x_i)}(x_i) + b'_i \geq \gamma \end{array} \right. . \tag{3}$$

Furthermore, for the set of indexes i such that $(i'_1(x_i), i'_2(x_i)) = (i_2(x_i), i_1(x_i))$,

$$\left\{ \begin{array}{lll} \text{if } y_i = & -1, & \Delta^{\#}g_{y',i'_2(x_i)}(x_i) + b'_i \geq \gamma \\ \text{if } y_i = & 1, & \Delta^{\#}g_{y',i'_1(x_i)}(x_i) - b'_i \geq \gamma \end{array} \right. .$$

This is exactly (3), which thus holds true for all values of i in $[\![1,n]\!]$ (whether the couple $(i'_1(x_i), i'_2(x_i))$ is equal to $(i_1(x_i), i_2(x_i))$ or equal to $(i_2(x_i), i_1(x_i))$). According to Definition 29, the function $\Delta^{\#}g_{y'}$ thus contributes to the γ-N-shattering of $s_{\mathcal{X}^n}$ with respect to $I'(s_{\mathcal{X}^n})$ and $v_{b'}$ for a value of the binary vector equal to v_y. But since the vector v_y has been chosen arbitrarily in $\{-1,1\}^n$, this implies that $\Delta^{\#}\mathcal{G}_y$ γ-N-shatters $s_{\mathcal{X}^n}$ with respect to $I'(s_{\mathcal{X}^n})$ and $v_{b'}$, which, by construction of $I'(s_{\mathcal{X}^n})$, concludes the proof. ■

In the sequel, we will sometimes make use of Proposition 30 implicitly. We now establish that the γ-Ψ-dimensions are actually multivariate extensions of the fat-shattering dimension.

Proposition 31 *Let $\tilde{\mathcal{G}}$ be a class of real-valued functions on a set $\mathcal{X}$. Let $\mathcal{G}$ be the corresponding class of functions from $\mathcal{X}$ into $\mathbb{R}^2$. Then, for all positive value of γ,*

$$P_\gamma\text{-}dim(\tilde{\mathcal{G}}) = \Psi\text{-}dim(\Delta^{\#}\mathcal{G}, \gamma).$$

Proof When $Q = 2$, one can consider that the set Ψ contains only two mappings, ψ_+ and ψ_-, with $\psi_+(1) = 1$, $\psi_+(2) = -1$ and $\psi_-(1) = -1$, $\psi_-(2) = 1$ (adding other mappings, for instance mappings taking the value $*$, would be useless since such mappings either do not take the value 1, or do not take the value -1). Using the same line of reasoning as in the proof of Proposition 30, one

can establish that Ψ can be restricted further to the singleton $\{\psi_+\}$. As a consequence, (2) simplifies into:

$$\forall i \in [\![1,n]\!], \left\{ \begin{array}{ll} \text{if } y_i = \quad 1, & \Delta^{\#} g_{y,1}(x_i) - b_i \geq \gamma \\ \text{if } y_i = -1, & \Delta^{\#} g_{y,2}(x_i) + b_i \geq \gamma \end{array} \right. . \tag{4}$$

Since we have seen in Section 2.2 that $\Delta^{\#} g = (\tilde{g}, -\tilde{g})$, (4) simplifies further into (1) (with $\tilde{g}$ in place of g), which concludes the proof. ■

In both cases (fat-shattering dimension and margin Ψ-dimensions) the introduction of the vector of "biases" v_b could be seen as a simple computational trick, useful to derive the generalized Sauer-Shelah lemma (establish a connection between the property of separation and the capacity to shatter a set of points) at the expense of a more complex computation for the bound on the margin dimension itself. This is partly the case indeed. However, in Section 7, we will see that these extra degrees of freedom can be handled pretty easily.

5. Relating the Covering Number and the Margin Natarajan Dimension

This section is devoted to the formulation of an upper bound on the covering number of interest in terms of the margin Natarajan dimension. Its main result is a generalization of the Sauer-Shelah lemma given by Lemmas 38 and 39. Our basic uniform convergence result, Theorem 22, involves the class of functions $\Delta_\gamma^{\#}\mathcal{G}$. However, in the preceding section, the scale-sensitive Ψ-dimensions have been defined for $\Delta^{\#}\mathcal{G}$ (although the extension to $\Delta_\gamma^{\#}\mathcal{G}$ is straightforward). The reason for this change, and the way it can be handled, is the subject of the following subsection.

5.1 Switching from $\Delta_\gamma^{\#}\mathcal{G}$ to $\Delta^{\#}\mathcal{G}$

As stated in Section 2.2, the advantage of working with the class $\Delta_\gamma^{\#}\mathcal{G}$ is obvious: the range of its functions, $[-\gamma,\gamma]^Q$, is optimal (the smallest range that does not affect the value of the margin risk). The seamy side of things is that the nonlinearity introduced by the π_γ operator is difficult to handle when bounding a generalized VC dimension. Furthermore, there is no direct connection between $\Psi\text{-dim}\left(\Delta_\gamma^{\#}\mathcal{G},\varepsilon\right)$ and $\Psi\text{-dim}\left(\Delta^{\#}\mathcal{G},\varepsilon\right)$. On the contrary, the transition can be performed very easily at the level of the covering number, thanks to the following lemma.

Lemma 32 *Let $\mathcal{G}$ be a class of functions from a domain $\mathcal{X}$ into $\mathbb{R}^Q$, let γ and ε be two positive real numbers and let $n \in \mathbb{N}^*$. Then,*

$$\mathcal{N}^{(p)}(\varepsilon,\Delta_\gamma^{\#}\mathcal{G},n) \leq \mathcal{N}^{(p)}(\varepsilon,\Delta^{\#}\mathcal{G},n).$$

Proof This property directly springs from the fact that π_γ satisfies the Lipschitz condition with constant 1. Thus, $\forall (g,g') \in \mathcal{G}^2, \forall x \in \mathcal{X}, \forall (\gamma,\varepsilon) \in \left(\mathbb{R}_+^*\right)^2$,

$$\left\|\Delta^{\#} g(x) - \Delta^{\#} g'(x)\right\|_\infty < \varepsilon \Longrightarrow \left\|\Delta_\gamma^{\#} g(x) - \Delta_\gamma^{\#} g'(x)\right\|_\infty < \varepsilon.$$

■

Since the computations leading to our generalized Sauer-Shelah lemma will require the functions in $\Delta^{\#}\mathcal{G}$ to have a bounded range, to compensate for the elimination of the π_γ operator, from now on, we make the hypothesis that there exists a positive real number M such that the functions g, and by

way of consequence the functions $\Delta^{\#}g$, take their values in $[-M,M]^Q$. Given this hypothesis, the only values of the margin parameter γ corresponding to a nontrivial situation are those inferior or equal to M. As a consequence, we also assume that the parameter Γ of Theorem 22 is set equal to M. To formulate the main combinatorial result of this section, new concepts are to be defined. They correspond to extensions of concepts introduced in Alon et al. (1997).

5.2 Definitions

Definition 33 (η-discretization operator) *Let $\mathcal{G}$ be a class of functions from $\mathcal{X}$ into $[-M,M]^Q$. For $\eta \in \mathbb{R}_+^*$, define the* η-discretization *as an operator on $\Delta^{\#}\mathcal{G}$ such that:*

$$(.)^{(\eta)}: \quad \begin{array}{l} \Delta^{\#}\mathcal{G} \longrightarrow \left(\Delta^{\#}\mathcal{G}\right)^{(\eta)}, \\ \Delta^{\#}g \mapsto \left(\Delta^{\#}g\right)^{(\eta)} = \left(\left(\Delta^{\#}g_k\right)^{(\eta)}\right)_{1\leq k\leq Q}, \end{array}$$

$$\forall x \in \mathcal{X},\ \left(\Delta^{\#}g\right)^{(\eta)}(x) = \left(sign\left(\Delta^{\#}g_k(x)\right)\cdot\left\lfloor\frac{\left|\Delta^{\#}g_k(x)\right|}{\eta}\right\rfloor\right)_{1\leq k\leq Q}$$

where the function $\lfloor.\rfloor$ is defined by $\forall t \in \mathbb{R}_+$, $\lfloor t\rfloor = \max\{j\in\mathbb{N}:\ j\leq t\}$.

Note that this definition is not a straightforward extension of the original one to the case of vector-valued functions, since we had to relax the hypothesis of nonnegativity. Fortunately, this generalization does not raise any difficulty.

Definition 34 (Strong Natarajan dimension) *Let $\mathcal{G}$ be a class of functions from $\mathcal{X}$ into $[-M,M]^Q$ and let $\eta \in (0,M]$. A subset $s_{\mathcal{X}^n} = \{x_i : 1\leq i\leq n\}$ of $\mathcal{X}$ is said to be* strongly N-shattered *by $\left(\Delta^{\#}\mathcal{G}\right)^{(\eta)}$ if there is a set*

$$I(s_{\mathcal{X}^n}) = \{(i_1(x_i), i_2(x_i)) : 1\leq i\leq n\}$$

of n couples of distinct indexes in $[\![1,Q]\!]$ and a vector $v_b = (b_i)$ in $\left[-\left\lfloor\frac{M}{\eta}\right\rfloor+1, \left\lfloor\frac{M}{\eta}\right\rfloor-1\right]^n$ such that, for each vector $v_y = (y_i)$ in $\{-1,1\}^n$, there is a function g_y in $\mathcal{G}$ satisfying

$$\forall i\in[\![1,n]\!],\ \left\{\begin{array}{lll} \text{if } y_i = & 1, & \left(\Delta^{\#}g_{y,i_1(x_i)}\right)^{(\eta)}(x_i) - b_i \geq 1 \\ \text{if } y_i = & -1, & \left(\Delta^{\#}g_{y,i_2(x_i)}\right)^{(\eta)}(x_i) + b_i \geq 1 \end{array}\right. .$$

The strong Natarajan dimension *of the class $\left(\Delta^{\#}\mathcal{G}\right)^{(\eta)}$, SN-dim$\left(\left(\Delta^{\#}\mathcal{G}\right)^{(\eta)}\right)$, is the maximal cardinality of a subset of $\mathcal{X}$ strongly N-shattered by $\left(\Delta^{\#}\mathcal{G}\right)^{(\eta)}$, if this cardinality is finite. If no such maximum exists, $\left(\Delta^{\#}\mathcal{G}\right)^{(\eta)}$ is said to have infinite strong Natarajan dimension.*

Obviously, as in the case of the margin Natarajan dimension, the definition remains unchanged if the additional constraint: $\forall i\in[\![1,n]\!], i_1(x_i) < i_2(x_i)$ is introduced.

5.3 Relating Separation and Strong N-shattering

The Sauer-Shelah lemma and its generalizations rest on a simple idea: to establish a connection between the property of separation of two functions and their capacity to "shatter" a singleton. This connection is obvious in the case of binary-valued functions (more precisely functions taking values in $\{-1,1\}$). Then,

$$\left|f(x)-f'(x)\right| \geq 2 \Longleftrightarrow f(x) = -f'(x)$$

and thus f and f' classify x in different categories. Things are more complicated is the case of classifiers taking values in $\mathbb{R}^Q$. The corresponding result in that latter context is the following.

Lemma 35 *Let $\mathcal{G}$ be a class of functions from $\mathcal{X}$ into $[-M,M]^Q$ and let η be a real number belonging to $(0,M]$. Let $\mathcal{D}$ be a subset of $\mathcal{X}$ of finite cardinality and let $\mathcal{F}$ and $\mathcal{F}^*$ be respectively the restrictions of $(\Delta\mathcal{G})^{(\eta)}$ and $(\Delta^*\mathcal{G})^{(\eta)}$ to $\mathcal{D}$. $\mathcal{F}$ and $\mathcal{F}^*$ are endowed with the pseudo-metric $d_{\mathcal{D}}$. If two functions g and g' in $\mathcal{G}$ are such that $f^* = (\Delta^* g)^{(\eta)}\Big|_{\mathcal{D}}$ and $f^{*'} = (\Delta^* g')^{(\eta)}\Big|_{\mathcal{D}}$ are separated, then there exists x in $\mathcal{D}$ such that $\left\{f = (\Delta g)^{(\eta)}\Big|_{\mathcal{D}}, f' = (\Delta g')^{(\eta)}\Big|_{\mathcal{D}}\right\}$ strongly N-shatters the singleton $\{x\}$. Suppose further, without loss of generality, that $\max_k f_k^*(x) \geq \max_k f_k^{*'}(x)$ and let $k_0 = \operatorname{argmax}_k f_k^*(x)$. Then there is at least one couple $(I(\{x\}), v_b) = (\{(i_1(x), i_2(x))\}, (b_0))$ with $i_1(x) = k_0$ and $b_0 = f_{k_0}^*(x) - 1$ witnessing the strong N-shattering of $\{x\}$ by $\{f, f'\}$.*

Proof We first demonstrate that k_0 is well defined. Indeed, this is the case unless $f^*(x) = 0$. But $f^*(x) = 0$ and $\max_k f_k^*(x) \geq \max_k f_k^{*'}(x)$ implies that $f^{*'}(x) = 0$, which is in contradiction with the hypothesis $\left\|f^*(x) - f^{*'}(x)\right\|_\infty \geq 2$. By definition of the operator Δ^*, there exists an index l_0 different from k_0 such that $f'_{l_0}(x) = f_{l_0}^{*'}(x)$. l_0 is simply the index of a component of $g'(x)$ satisfying $g'_{l_0}(x) = \max_{k \neq k_0} g'_k(x)$. By definition of k_0 and b_0, $f_{k_0}(x) - b_0 = f_{k_0}^*(x) - b_0 = 1$. By construction, $f'_{l_0}(x) + b_0 = f_{l_0}^{*'}(x) + b_0$. Two cases must now be considered. If $f_{l_0}^{*'}(x) = \max_k f_k^{*'}(x)$, then $f_{l_0}^{*'}(x) = 0 \Longrightarrow f^{*'}(x) = 0 \Longrightarrow f_{k_0}^*(x) \geq 2$ (otherwise f^* and $f^{*'}$ would not be separated). As a consequence, $f_{l_0}^{*'}(x) + f_{k_0}^*(x) \geq 2$ and thus $f_{l_0}^{*'}(x) + b_0 \geq 1$, which is equivalent to $f'_{l_0}(x) + b_0 \geq 1$. If $f_{l_0}^{*'}(x) \neq \max_k f_k^{*'}(x)$, then $f_{k_0}^{*'}(x) = \max_k f_k^{*'}(x)$ and $f_{l_0}^{*'}(x) = -f_{k_0}^{*'}(x)$. Necessarily, $f_{k_0}^*(x) - f_{k_0}^{*'}(x) \geq 2$ (otherwise f^* and $f^{*'}$ would not be separated) and finally $f'_{l_0}(x) + b_0 = f_{k_0}^*(x) - f_{k_0}^{*'}(x) - 1 \geq 1$. Thus, the couple $\left(I(\{x\}) = \{(k_0, l_0)\}, v_b = \left(f_{k_0}^*(x) - 1\right)\right)$ witnesses the strong N-shattering of $\{x\}$ by $\{f, f'\}$. ■

Lemma 35 will turn out to be of central importance in the sequel. We consider it as contributing to characterize the specificity of the multi-class case, since it highlights the usefulness of the Δ^* operator (whereas the usefulness of the Δ operator will appear in Section 7).

Remark 36 *Lemma 35 cannot be stated with the operator Δ only.*

Proof To prove this last assertion, it suffices to exhibit a counter example. Let $\mathcal{G}$ be a class of functions from $\mathcal{X}$ into $[-2,2]^4$ and g and g' be two functions in $\mathcal{G}$ such that there exists $\mathcal{D} = \{x\}$ satisfying $g(x) = (1.4, -0.2, -0.2, -1.0)$ and $g'(x) = (1.4, -0.2, -0.6, -0.6)$. Let $\eta = 0.1$. Using the same notations as above, we get $f(x) = (8, -8, -8, -12)$, $f'(x) = (8, -8, -10, -10)$ and $f^*(x) = f^{*'}(x) = (8, -8, -8, -8)$. Although f and f' are separated, they do not strongly N-shatter

$\{x\}$. Indeed, if it were the case, then according to Definition 34, there would be two different indexes k_0 and l_0 in $[\![1,4]\!]$ such that $f_{k_0}(x)+f'_{l_0}(x)\geq 2$, which is not the case. ■

In contrast with this negative result, the hypothesis $\left\|f^*(x)-f^{*'}(x)\right\|_\infty \geq 2$ also implies that $\left\{f^*, f^{*'}\right\}$ strongly N-shatters $\{x\}$ (Lemma 35 could have been stated with the operator Δ^* only). A tricky thing must be borne in mind. If two pairs $\left(g^{(1)}, g^{(2)}\right)$ and $\left(g^{(3)}, g^{(4)}\right)$ of functions in $\mathcal{G}$ are such that $\left(f^{*(1)}(x), f^{*(2)}(x)\right) = \left(f^{*(3)}(x), f^{*(4)}(x)\right)$, then if $\left\|f^{*(1)}(x)-f^{*(2)}(x)\right\|_\infty \geq 2$, $\{x\}$ is strongly N-shattered both by $\left\{f^{(1)}, f^{(2)}\right\}$ and by $\left\{f^{(3)}, f^{(4)}\right\}$. However, those shatterings could require different witnesses $(I(\{x\}), v_b)$. More precisely, using the notations of Definition 34, given the couple $\left(f^{*(1)}, f^{*(2)}\right)$, one can exhibit an index $i_1(x)$ and a bias b_0 contributing to both shatterings (by $\left\{f^{(1)}, f^{(2)}\right\}$ and by $\left\{f^{(3)}, f^{(4)}\right\}$) but the last component of the witness, $i_2(x)$, must be chosen as a function of the values taken by the functions f on x. It is thus a priori different for $\left\{f^{(1)}, f^{(2)}\right\}$ and for $\left\{f^{(3)}, f^{(4)}\right\}$.

We now prove the main combinatorial result at the basis of our generalization of the Sauer-Shelah lemma, an extension of Lemma 3.3 in Alon et al. (1997).

5.4 Main Combinatorial Result

Lemma 37 *Let $\mathcal{G}$ be a class of functions on $\mathcal{X}$ taking their values in $[-M,M]^Q$ and let η be a real number belonging to $(0,M]$. Let $\mathcal{D}$ be a subset of $\mathcal{X}$ of finite cardinality $|\mathcal{D}|$ and let $\mathcal{F}$ and $\mathcal{F}^*$ be respectively the restrictions of $(\Delta\mathcal{G})^{(\eta)}$ and $(\Delta^*\mathcal{G})^{(\eta)}$ to $\mathcal{D}$. $\mathcal{F}$ and $\mathcal{F}^*$ are endowed with the pseudo-metric $d_{\mathcal{D}}$. Setting $d = SN\text{-}dim(\mathcal{F})$ and $q = \left\lfloor \frac{M}{\eta} \right\rfloor$, the following bound holds true:*

$$\mathcal{M}(2,\mathcal{F}^*,d_{\mathcal{D}}) < 2\left(|\mathcal{D}|\ Q^2(Q-1)\,q^2\right)^{\lceil \log_2(\phi(d,|\mathcal{D}|))\rceil} \tag{5}$$

where $\phi(d,|\mathcal{D}|) = \sum_{i=1}^{d}\binom{|\mathcal{D}|}{i}\left(\binom{Q}{2}(2q-1)\right)^i$.

Proof Let us say that the class $\mathcal{F}$ strongly N-shatters a triplet $(s_{\mathcal{D}}, I(s_{\mathcal{D}}), v_b)$ (for a nonempty subset $s_{\mathcal{D}}$ of $\mathcal{D}$, a set of couples of indexes $I(s_{\mathcal{D}})$ and a vector of biases v_b) if $\mathcal{F}$ strongly N-shatters $s_{\mathcal{D}}$ according to $I(s_{\mathcal{D}})$ and v_b. For all integers $l \geq 2$ and $|\mathcal{D}| \geq 1$, let $t(l,|\mathcal{D}|)$ denote the maximum number t such that, for every set $\mathcal{F}_l^*$ of l pairwise separated functions in $\mathcal{F}^*$, $\mathcal{F}_l = \left\{f \in \mathcal{F} : f^* \in \mathcal{F}_l^*\right\}$ strongly N-shatters at least t triplets $(s_{\mathcal{D}}, I(s_{\mathcal{D}}), v_b)$. If there is no subset of $\mathcal{F}^*$ of cardinality l pairwise separated, then $t(l,|\mathcal{D}|)$ is infinite.

The number of triplets $(s_{\mathcal{D}}, I(s_{\mathcal{D}}), v_b)$ that could be shattered and for which the cardinality of $s_{\mathcal{D}}$ does not exceed $d \geq 1$ is less than $\sum_{i=1}^{d}\binom{|\mathcal{D}|}{i}\left(\binom{Q}{2}(2q-1)\right)^i$, since for $s_{\mathcal{D}}$ of size $i>0$, there are strictly less than $\left(\binom{Q}{2}(2q-1)\right)^i$ possibilities to choose the couple $(I(s_{\mathcal{D}}), v_b)$. It follows that $t(l,|\mathcal{D}|) \geq \phi(d,|\mathcal{D}|)$ for some l implies $t(l,|\mathcal{D}|) = \infty$. By definition of $t(l,|\mathcal{D}|)$, this means that there is no subset of $\mathcal{F}^*$ of cardinality l pairwise separated (otherwise $t(l,|\mathcal{D}|)$ would be finite) and finally, by definition of $\mathcal{M}(2,\mathcal{F}^*,d_{\mathcal{D}})$, $\mathcal{M}(2,\mathcal{F}^*,d_{\mathcal{D}}) < l$. Therefore, to finish the proof, it suffices to show that, for all $d \geq 1$ and $|\mathcal{D}| \geq 1$,

$$t\left(2\left(|\mathcal{D}|\ Q^2(Q-1)\,q^2\right)^{\lceil \log_2(\phi(d,|\mathcal{D}|))\rceil}, |\mathcal{D}|\right) \geq \phi(d,|\mathcal{D}|). \tag{6}$$

We claim that

$$t(2,|\mathcal{D}|) \geq 1 \tag{7}$$

for all $|\mathcal{D}| \geq 1$ and

$$t\left(2p\ |\mathcal{D}|\ Q^2(Q-1)\,q^2, |\mathcal{D}|\right) \geq 2t(2p, |\mathcal{D}|-1) \tag{8}$$

for all $p \geq 1$ and $|\mathcal{D}| \geq 2$.

The first part of the claim is a direct consequence of Lemma 35.

For the second part, first note that if no set of $2p\ |\mathcal{D}|\ Q^2(Q-1)\,q^2$ pairwise separated functions in $\mathcal{F}^*$ exists, then by definition $t\left(2p\ |\mathcal{D}|\ Q^2(Q-1)\,q^2, |\mathcal{D}|\right) = \infty$ and hence the claim holds. Assume then that there is a set $\mathcal{F}_0^*$ of $2p\ |\mathcal{D}|\ Q^2(Q-1)\,q^2$ pairwise separated functions in $\mathcal{F}^*$. Split it arbitrarily into $p\ |\mathcal{D}|\ Q^2(Q-1)\,q^2$ pairs. For each pair $\left(f^*, f^{*'}\right)$, there exists a singleton $\{x\} \subset \mathcal{D}$ strongly N-shattered by $\{f, f'\}$. Once more, this is a direct consequence of Lemma 35. By definition, a vector $f^*(x)$ has all components of equal magnitude. As a consequence, the number of different values that it can take is equal to $Qq+1$. The numbers of different sets of the form $\left\{f^*(x), f^{*'}(x)\right\}$ such that $\left\|f^*(x) - f^{*'}(x)\right\|_\infty \geq 2$ is bounded from above by $\frac{1}{2}(Qq+1)(Qq-1) < \frac{1}{2}Q^2\,q^2$. Thus, by the pigeonhole principle, switching the indexes in the couples of functions if needed, for each procedure of this type, there exists $x_0 \in \mathcal{D}$ such that at least $\left(2p\ |\mathcal{D}|\ Q^2(Q-1)\,q^2\right) / \left(|\mathcal{D}|\ Q^2\,q^2\right) = 2p\,(Q-1)$ of the resulting couples of functions take the same value on x_0, value satisfying $\left\|f^*(x_0) - f^{*'}(x_0)\right\|_\infty \geq 2$. For all these pairs, the corresponding sets $\{f, f'\}$ all shatter $\{x_0\}$ (shatter at least one triplet of the form $(\{x_0\}, I(\{x_0\}), v_b)$). If the components of the couples are reordered in such a way that all the couples are identical with $\max_k f_k^*(x_0) \geq \max_k f_k^{*'}(x_0)$, this result still holds if one imposes that the values of $i_1(x_0)$ and b_0 are those considered in Lemma 35 ($i_1(x_0) = \operatorname{argmax}_k f_k^*(x_0)$ and $b_0 = f^*_{i_1(x_0)}(x_0) - 1$). Once $i_1(x_0)$ is set, $i_2(x_0)$ can take at most $Q-1$ different values. Thus, using once more the pigeonhole principle, among those last couples of functions, there are (at least) $2p\,(Q-1)/(Q-1) = 2p$ of them such that the quintuplet $\left(x_0, f^*(x_0), f^{*'}(x_0), I(\{x_0\}), v_b\right)$ can be the same, that is, a single pair $(I(\{x_0\}), v_b)$ can witness the strong N-shattering of $\{x_0\}$ by all the sets $\{f, f'\}$. To sum up, this means that there are two subclasses of $\mathcal{F}_0^*$ of cardinality at least $2p$, call them $\mathcal{F}_+^*$ and $\mathcal{F}_-^*$, and there are $x_0 \in \mathcal{D}$, two vectors $V_{0,+}$ and $V_{0,-}$ in $[\![-q, q]\!]^Q$ such that $\|V_{0,+} - V_{0,-}\|_\infty \geq 2$, $(k_0, l_0) \in [\![1, Q]\!]^2$ with $k_0 \neq l_0$, and a scalar b_0 in $[\![-q+1, q-1]\!]$ such that:

$$\begin{cases} \forall f_+^* \in \mathcal{F}_+^*, & f_+^*(x_0) & = & V_{0,+} \\ \forall f_-^* \in \mathcal{F}_-^*, & f_-^*(x_0) & = & V_{0,-} \\ \forall f_+ \in \mathcal{F}_+, & f_{+,k_0}(x_0) & \geq & 1 + b_0 \\ \forall f_- \in \mathcal{F}_-, & f_{-,l_0}(x_0) & \geq & 1 - b_0 \end{cases}$$

where $\mathcal{F}_+ = \left\{f_+ \in \mathcal{F} : f_+^* \in \mathcal{F}_+^*\right\}$ and $\mathcal{F}_- = \left\{f_- \in \mathcal{F} : f_-^* \in \mathcal{F}_-^*\right\}$. Since the members of $\mathcal{F}_+^*$ are pairwise separated on $\mathcal{D}$ but are all equal on x_0, they are pairwise separated on $\mathcal{D} \setminus \{x_0\}$. The same holds for the members of $\mathcal{F}_-^*$. Hence, by definition of the function t, $\mathcal{F}_+$ strongly N-shatters at least $t\,(2p, |\mathcal{D}|-1)$ triplets $(s_\mathcal{D}, I(s_\mathcal{D}), v_b)$ with $s_\mathcal{D} \subseteq \mathcal{D} \setminus \{x_0\}$, and the same holds for $\mathcal{F}_-$. Clearly, $\mathcal{F}_0 = \{f \in \mathcal{F} : f^* \in \mathcal{F}_0^*\}$ strongly N-shatters all triplets strongly N-shattered either by $\mathcal{F}_+$ or by $\mathcal{F}_-$. Moreover, if the same triplet $(s_\mathcal{D}, I(s_\mathcal{D}), v_b)$ is strongly N-shattered both by $\mathcal{F}_+$ and by $\mathcal{F}_-$, then $\mathcal{F}_0$ also strongly N-shatters the triplet $(\{x_0\} \bigcup s_\mathcal{D}, \{(k_0, l_0)\} \bigcup I(s_\mathcal{D}), \bar{v}_b)$, where $\bar{v}_b$ is deduced from v_b by adding one component corresponding to the point x_0, component taking the value b_0. Indeed, the sets $\mathcal{F}_+$ and $\mathcal{F}_-$ have been built precisely in that purpose. Suffice it to notice what follows. Let $(s_\mathcal{D}, I(s_\mathcal{D}), v_b)$ be a triplet strongly N-shattered both by $\mathcal{F}_+$ and by $\mathcal{F}_-$. For the sake

of simplicity, reordering the points in $\mathcal{D}$ if needed, we suppose that $s_{\mathcal{D}}$ can be written as follows: $s_{\mathcal{D}} = \{x_i : 1 \leq i \leq |s_{\mathcal{D}}|\}$. Then, for any vector $v_y = (y_i)$ in $\{-1,1\}^{|s_{\mathcal{D}}|}$, there exists (at least) one function $f_{+,y}$ in $\mathcal{F}_+$ such that

$$\forall i \in [\![1, |s_{\mathcal{D}}|]\!], \left\{ \begin{array}{ll} \text{if } y_i = \quad 1, & f_{+,y,i_1(x_i)}(x_i) - b_i \geq 1 \\ \text{if } y_i = -1, & f_{+,y,i_2(x_i)}(x_i) + b_i \geq 1 \end{array} \right.$$

and

$$f_{+,y,k_0}(x_0) - b_0 \geq 1$$

and one function $f_{-,y}$ in $\mathcal{F}_-$ such that

$$\forall i \in [\![1, |s_{\mathcal{D}}|]\!], \left\{ \begin{array}{ll} \text{if } y_i = \quad 1, & f_{-,y,i_1(x_i)}(x_i) - b_i \geq 1 \\ \text{if } y_i = -1, & f_{-,y,i_2(x_i)}(x_i) + b_i \geq 1 \end{array} \right.$$

and

$$f_{-,y,l_0}(x_0) + b_0 \geq 1.$$

Since, once more by construction, neither $\mathcal{F}_+$ nor $\mathcal{F}_-$ strongly N-shatters $\{x_0\} \bigcup s_{\mathcal{D}}$ (whatever the pair $(I(\{x_0\} \bigcup s_{\mathcal{D}}), \bar{v}_b)$ may be), it follows that $t\left(2p\ |\mathcal{D}|\ Q^2(Q-1)\, q^2, |\mathcal{D}|\right) \geq 2t(2p, |\mathcal{D}|-1)$, which is precisely (8).

For any integer number r satisfying $1 \leq r < |\mathcal{D}|$, let

$$l = 2\left(Q^2(Q-1)\, q^2\right)^r\ \Pi_{u=0}^{r-1}(|\mathcal{D}| - u).$$

Applying (8) iteratively and eventually (7), it appears that $t(l, |\mathcal{D}|) \geq 2^r$. Since t is clearly nondecreasing in its first argument, and $2\left(|\mathcal{D}|\ Q^2(Q-1)\, q^2\right)^r \geq l$, this implies

$$t\left(2\left(|\mathcal{D}|\ Q^2(Q-1)\, q^2\right)^r, |\mathcal{D}|\right) \geq 2^r.$$

We make use of this bound by considering separately the case where $\lceil \log_2(\phi(d, |\mathcal{D}|)) \rceil < |\mathcal{D}|$ and the case where $\lceil \log_2(\phi(d, |\mathcal{D}|)) \rceil \geq |\mathcal{D}|$. In the first case, one can set $r = \lceil \log_2(\phi(d, |\mathcal{D}|)) \rceil$. We then get

$$t\left(2\left(|\mathcal{D}|\ Q^2(Q-1)\, q^2\right)^{\lceil \log_2(\phi(d,|\mathcal{D}|)) \rceil}, |\mathcal{D}|\right) \geq 2^{\lceil \log_2(\phi(d,|\mathcal{D}|)) \rceil}$$

and consequently

$$t\left(2\left(|\mathcal{D}|\ Q^2(Q-1)\, q^2\right)^{\lceil \log_2(\phi(d,|\mathcal{D}|)) \rceil}, |\mathcal{D}|\right) \geq 2^{\log_2(\phi(d,|\mathcal{D}|))} = \phi(d, |\mathcal{D}|)$$

which is precisely (6). If on the contrary $\lceil \log_2(\phi(d, |\mathcal{D}|)) \rceil \geq |\mathcal{D}|$, then

$$2\left(|\mathcal{D}|\ Q^2(Q-1)\, q^2\right)^{\lceil \log_2(\phi(d,|\mathcal{D}|)) \rceil} > (Qq+1)^{|\mathcal{D}|}.$$

Since the number of distinct functions in $\mathcal{F}^*$ is bounded from above by $(Qq+1)^{|\mathcal{D}|}$, $\mathcal{F}^*$ cannot contain a set of pairwise separated functions of cardinality larger than this number and hence, by definition of t,

$$t\left(2\left(|\mathcal{D}|\ Q^2(Q-1)\, q^2\right)^{\lceil \log_2(\phi(d,|\mathcal{D}|)) \rceil}, |\mathcal{D}|\right) = \infty.$$

$t\left(2\left(|\mathcal{D}|\ Q^2(Q-1)\, q^2\right)^{\lceil \log_2(\phi(d,|\mathcal{D}|)) \rceil}, |\mathcal{D}|\right)$ is consequently once more superior to $\phi(d, |\mathcal{D}|)$, which completes the proof of (6) and thus concludes the proof of the lemma. ■

Note that expressing Lemma 37 in the bi-class case (by setting $Q = 2$), one obtains almost exactly the expression of Lemma 3.3 in Alon et al. (1997), keeping in mind that our functions and theirs do not take their values in the same intervals.

5.5 Generalized Sauer-Shelah Lemma

Our generalized Sauer-Shelah lemma appears as a direct consequence of Lemma 37.

Lemma 38 (Generalized Sauer-Shelah lemma) *Let $\mathcal{G}$ be a class of functions from $\mathcal{X}$ into $[-M,M]^Q$. For every value of ε in $(0,M]$ and every integer value of n satisfying $n \geq N\text{-dim}(\Delta\mathcal{G}, \varepsilon/6)$, the following bound is true:*

$$\mathcal{N}^{(p)}(\varepsilon, \Delta^*\mathcal{G}, n) < 2\left(n\, Q^2(Q-1)\left\lfloor \frac{3M}{\varepsilon}\right\rfloor^2\right)^{\lceil \log_2(\phi(d,n))\rceil} \tag{9}$$

where $d = N\text{-dim}(\Delta\mathcal{G}, \varepsilon/6)$ and $\phi(d,n) = \sum_{i=1}^{d}\binom{n}{i}\left(\binom{Q}{2}\left(2\left\lfloor\frac{3M}{\varepsilon}\right\rfloor - 1\right)\right)^i$.

Proof $\forall x^n \in \mathcal{X}^n$, applying Lemma 56 (right-hand side inequality) to $\Delta^*\mathcal{G}$ gives:

$$\mathcal{N}^{(p)}\left(\varepsilon, \Delta^*\mathcal{G}, d_{x^n}\right) \leq \mathcal{M}\left(\varepsilon, \Delta^*\mathcal{G}, d_{x^n}\right).$$

Setting $\eta = \varepsilon/3$ in Proposition 2 of Lemma 57, one obtains:

$$\mathcal{N}^{(p)}\left(\varepsilon, \Delta^*\mathcal{G}, d_{x^n}\right) \leq \mathcal{M}\left(2, \left(\Delta^*\mathcal{G}\right)^{(\varepsilon/3)}, d_{x^n}\right). \tag{10}$$

Let $\mathcal{D}_n$ denote the smallest subset of $\mathcal{X}$ including all the elements of x^n (its cardinality is inferior or equal to n since x^n can contain multiple copies of some elements of $\mathcal{X}$). We write $\left(\Delta^*\mathcal{G}\right)^{(\varepsilon/3)}\Big|_{\mathcal{D}_n}$ to designate the restriction of $\left(\Delta^*\mathcal{G}\right)^{(\varepsilon/3)}$ to $\mathcal{D}_n$. Since

$$\mathcal{M}\left(2, \left(\Delta^*\mathcal{G}\right)^{(\varepsilon/3)}, d_{x^n}\right) = \mathcal{M}\left(2, \left(\Delta^*\mathcal{G}\right)^{(\varepsilon/3)}\Big|_{\mathcal{D}_n}, d_{x^n}\right),$$

(10) implies:

$$\mathcal{N}^{(p)}\left(\varepsilon, \Delta^*\mathcal{G}, d_{x^n}\right) \leq \mathcal{M}\left(2, \left(\Delta^*\mathcal{G}\right)^{(\varepsilon/3)}\Big|_{\mathcal{D}_n}, d_{x^n}\right).$$

The packing numbers of $\left(\Delta^*\mathcal{G}\right)^{(\varepsilon/3)}\Big|_{\mathcal{D}_n}$ can be bounded thanks to Lemma 37, by setting $\mathcal{D} = \mathcal{D}_n$, using n as an upper bound on $|\mathcal{D}|$ (which is possible since the right-hand side of (5) is an increasing function of $|\mathcal{D}|$), $q = \left\lfloor \frac{M}{\eta}\right\rfloor = \left\lfloor\frac{3M}{\varepsilon}\right\rfloor$ and $d = \text{SN-dim}\left(\left(\Delta\mathcal{G}\right)^{(\varepsilon/3)}\Big|_{\mathcal{D}_n}\right)$. Thus, we get:

$$\mathcal{N}^{(p)}\left(\varepsilon, \Delta^*\mathcal{G}, d_{x^n}\right) < 2\left(n\, Q^2(Q-1)\left\lfloor \frac{3M}{\varepsilon}\right\rfloor^2\right)^{\lceil \log_2(\phi(d,n))\rceil}, \tag{11}$$

with $\phi(d,n) = \sum_{i=1}^{d}\binom{n}{i}\left(\binom{Q}{2}\left(2\left\lfloor\frac{3M}{\varepsilon}\right\rfloor - 1\right)\right)^i$. Since the right-hand side of (11) is a nondecreasing function of d, one can replace d with an upper bound. By definition of $\left(\Delta\mathcal{G}\right)^{(\varepsilon/3)}\Big|_{\mathcal{D}_n}$,

$$\text{SN-dim}\left(\left(\Delta\mathcal{G}\right)^{(\varepsilon/3)}\Big|_{\mathcal{D}_n}\right) \leq \text{SN-dim}\left(\left(\Delta\mathcal{G}\right)^{(\varepsilon/3)}\right).$$

By application of Proposition 1 in Lemma 57,

$$\text{SN-dim}\left((\Delta\mathcal{G})^{(\varepsilon/3)}\right) \leq \text{N-dim}\left(\Delta\mathcal{G}, \varepsilon/6\right).$$

Thus, (11) still holds if d is set equal to $\text{N-dim}\left(\Delta\mathcal{G}, \varepsilon/6\right)$. Taking the maximum of its left-hand side over $\mathcal{X}^n$ then concludes the proof. ■

To find an upper bound on $\phi(d,n)$, and thus derive a generalized Sauer-Shelah lemma easier to handle than Lemma 38, it suffices to make use of Lemma 58 with $K_1 = d$, $K_2 = n$ and $K_3 = \binom{Q}{2}\left(2\left\lfloor\frac{3M}{\varepsilon}\right\rfloor - 1\right)$. This implies that

$$\phi(d,n) < \Phi\left(d, n, \binom{Q}{2}\left(2\left\lfloor\frac{3M}{\varepsilon}\right\rfloor - 1\right)\right) < \left(\frac{en\binom{Q}{2}\left(2\left\lfloor\frac{3M}{\varepsilon}\right\rfloor - 1\right)}{d}\right)^d$$

and consequently

$$\log_2\left(\phi(d,n)\right) < d\log_2\left(\frac{en\binom{Q}{2}\left(2\left\lfloor\frac{3M}{\varepsilon}\right\rfloor - 1\right)}{d}\right).$$

Substituting the right-hand side of this inequality to its left-hand side in (9), we finally get our master lemma.

Lemma 39 (Final formulation of the generalized Sauer-Shelah lemma) *Let $\mathcal{G}$ be a class of functions from $\mathcal{X}$ into $[-M,M]^Q$. For every value of ε in $(0,M]$ and every integer value of n satisfying $n \geq$ N-dim$\left(\Delta\mathcal{G}, \varepsilon/6\right)$, the following bound is true:*

$$\mathcal{N}^{(p)}(\varepsilon, \Delta^*\mathcal{G}, n) < 2\left(n\,Q^2(Q-1)\left\lfloor\frac{3M}{\varepsilon}\right\rfloor^2\right)^{\left\lceil d\log_2\left(en\binom{Q}{2}\left(2\left\lfloor\frac{3M}{\varepsilon}\right\rfloor-1\right)/d\right)\right\rceil}$$

where $d =$ N-dim$\left(\Delta\mathcal{G}, \varepsilon/6\right)$.

5.6 Discussion

To sum up, in this section, we have derived a bound on the covering number of interest in terms of one of the γ-Ψ-dimensions, the margin Natarajan dimension. Obviously, such a generalized Sauer-Shelah lemma can be derived in a similar way for other scale-sensitive extensions of a Ψ-dimension, such as the one corresponding to the graph dimension. The bound, by the way, is slightly easier to establish in the latter case. It involves smaller constants. However, as was already pointed out in Section 4.1, the choice of one particular variant of the VC dimension rests on the search for an optimal compromise between two requirements that can be contradictory: the need for a tight bound on the capacity measure in terms of the VC dimension, and the need for a tight bound on the VC dimension itself. In Section 7, it will appear clearly that the connection of the Natarajan dimension with the one-against-one decomposition method is a major advantage. Deriving a bound on the margin Natarajan dimension of the M-SVMs can be performed very simply, by extending in a straightforward way the reasoning of the proof of the standard bound on the fat-shattering dimension of the perceptron (or pattern recognition SVM).

6. Almost Sure Convergence Result

The combination of Theorem 22 and Lemma 39 (applied with $\varepsilon = \gamma/4$ and $n = 2m$) provides us with our master theorem.

Theorem 40 *Let $\mathcal{G}$ be the class of functions from $\mathcal{X}$ into $[-M,M]^Q$ that a large margin Q-category classifier can implement. Let $\delta \in (0,1)$. With probability at least $1-\delta$, for every value of γ in $(0,M]$, the risk of any function g in $\mathcal{G}$ is bounded from above by:*

$$R(g) \leq R_{\gamma,m}(g) +$$

$$\sqrt{\frac{2}{m}\left(\ln\left(4\left(2m\,Q^2(Q-1)\left\lfloor\frac{12M}{\gamma}\right\rfloor^2\right)^{\left\lceil d\log_2\left(emQ(Q-1)\left(2\left\lfloor\frac{12M}{\gamma}\right\rfloor-1\right)/d\right)\right\rceil}\right)+\ln\left(\frac{2M}{\gamma\delta}\right)\right)+\frac{1}{m}}$$

where $d = N\text{-}dim(\Delta\mathcal{G},\gamma/24)$.

With our notation, which designates by P and $\mathbb{P}_{D_m}$ respectively the probability measure characterizing the classification problem of interest, and a probability over the m-sample D_m, Theorem 40 states a distribution-free bound corresponding to a one-sided convergence in probability of the form:

$$\lim_{m\longrightarrow+\infty}\sup_P \mathbb{P}_{D_m}\left(\sup_{g\in\mathcal{G}}\left(R(g)-R_{\gamma,D_m}(g)\right)>\varepsilon\right)=0.$$

In fact, a stronger result can be obtained, since the convergence holds with probability 1.

Proposition 41 (Almost sure convergence)

$$\lim_{m\longrightarrow+\infty}\sup_P \mathbb{P}\left(\sup_{n\geq m}\sup_{g\in\mathcal{G}}\left(R(g)-R_{\gamma,n}(g)\right)>\varepsilon\right)=0.$$

Proof For a class $\mathcal{G}$ of functions taking values in $\mathbb{R}^Q$ and a given value of γ in $\mathbb{R}_+^*$, we obtained the following bound as a partial result in the proof of Theorem 22:

$$\mathbb{P}_{D_m}\left(\sup_{g\in\mathcal{G}}\left(R(g)-R_{\gamma,D_m}(g)\right)>\varepsilon\right)\leq 2\mathcal{N}^{(p)}\left(\gamma/2,\Delta_\gamma^{\#}\mathcal{G},2m\right)\exp\left(-\frac{m}{2}\left(\varepsilon-\frac{1}{m}\right)^2\right).$$

Under the restrictive assumption that the functions in $\mathcal{G}$ take their values in $[-M,M]^Q$, Lemmas 32 and 39 can be applied to bound from above the covering number, which yields:

$$\mathbb{P}_{D_m}\left(\sup_{g\in\mathcal{G}}\left(R(g)-R_{\gamma,D_m}(g)\right)>\varepsilon\right)\leq$$

$$4\left(2m\,Q^2(Q-1)\left\lfloor\frac{6M}{\gamma}\right\rfloor^2\right)^{\left\lceil d\log_2\left(emQ(Q-1)\left(2\left\lfloor\frac{6M}{\gamma}\right\rfloor-1\right)/d\right)\right\rceil}\exp\left(-\frac{m}{2}\left(\varepsilon-\frac{1}{m}\right)^2\right) \qquad (12)$$

where $d = \text{N-dim}\,(\Delta\mathcal{G}, \gamma/12)$. Let us denote by u_m the right-hand side of (12). Obviously,

$$\forall \varepsilon > 0,\ 4\left(2m\,Q^2(Q-1)\left\lfloor\frac{6M}{\gamma}\right\rfloor^2\right)^{\left\lceil d\log_2\left(emQ(Q-1)\left(2\left\lfloor\frac{6M}{\gamma}\right\rfloor-1\right)/d\right)\right\rceil} = o\left(\exp\left(\frac{m\varepsilon^2}{4}\right)\right).$$

As a consequence, $u_m = o\left(\exp\left(-\frac{m\varepsilon^2}{4}\right)\right)$. Since $\sum_{m=1}^{\infty}\exp\left(-\frac{m\varepsilon^2}{4}\right) < \infty$, by transitivity,

$$\sum_{m=1}^{\infty}\mathbb{P}_{D_m}\left(\sup_{g\in\mathcal{G}}\left(R(g) - R_{\gamma,D_m}(g)\right) > \varepsilon\right) < \infty.$$

One may thus apply the Borel-Cantelli lemma (see for instance Theorem A.22. in Devroye et al., 1996) and strengthen to almost sure convergence the convergence stated in Theorem 40. ■

7. Margin Natarajan Dimension of the Multi-Class SVMs

The theoretical results derived so far were dealing with general classes of functions $\mathcal{G}$, from $\mathcal{X}$ into $\mathbb{R}^Q$ or $[-M,M]^Q$, satisfying the mild conditions exposed in Section 2.1. In short, our aim was to establish that for those classes, the γ-Ψ-dimensions characterize learnability in the same way as the VC dimension, the fat-shattering dimension and the Ψ-dimensions characterize learnability for classes of functions taking values respectively in $\{-1,1\}$, $\mathbb{R}$ and $[\![1,Q]\!]$. From now on, we assess the use of the γ-Ψ-dimensions to characterize and control the generalization capabilities of classes of parametric functions. To that end, we focus on the main models of large margin multi-category classifiers, the multi-class SVMs.

Support vector machines (SVMs) are learning systems which have been introduced by Vapnik and co-workers (Boser et al., 1992; Cortes and Vapnik, 1995) as nonlinear extensions of the maximal margin hyperplane (Vapnik, 1982). Originally, they were designed to perform pattern recognition (compute dichotomies). In this context, the principle on which they are based is very simple. First, the examples are mapped into a high-dimensional Hilbert space called the *feature space* thanks to a nonlinear transform, the *feature map*, usually denoted by Φ. Second, the maximal margin hyperplane is computed in that space, to separate the two categories. The problem of performing multi-class discriminant analysis with SVMs was initially tackled through decomposition schemes involving bi-class machines. Such possibilities as the one-against-all method (Rifkin and Klautau, 2004), the one-against-one method (Fürnkranz, 2002) (a variant of which is the DAGSVM of Platt et al., 2000), or those based on error correcting codes (ECOC) (Allwein et al., 2000; Crammer and Singer, 2002) have thus been studied in depth during the last decade. Globally, the multi-class SVMs have been proposed more recently. They are all obtained by combining a multivariate affine model with the feature map Φ.

7.1 M-SVMs: Model and Function Selection

As in the bi-class case, the central element of a M-SVM is a *symmetric positive semidefinite (Mercer) kernel* (Aronszajn, 1950). Such kernels correspond to *positive type functions* (Berlinet and Thomas-Agnan, 2004). Let κ be a Mercer kernel on $\mathcal{X}$ and $\left(H_\kappa, \langle .,. \rangle_{H_\kappa}\right)$ the corresponding reproducing kernel Hilbert space (RKHS) (Berlinet and Thomas-Agnan, 2004). Let Φ be

any of the mappings on $\mathcal{X}$ satisfying:

$$\forall (x,x') \in \mathcal{X}^2,\ \kappa(x,x') = \langle \Phi(x), \Phi(x') \rangle, \tag{13}$$

where $\langle .,. \rangle$ is the dot product of the ℓ_2 space. Let $\Phi(\mathcal{X}) = \{\Phi(x) : x \in \mathcal{X}\}$ and let $\left(E_{\Phi(\mathcal{X})}, \langle .,. \rangle\right)$ be the Hilbert space spanned by $\Phi(\mathcal{X})$. According to the usual abuse of language, in the sequel, "the" *feature space* will designate any of the spaces $E_{\Phi(\mathcal{X})}$. By definition of a RKHS, $\mathcal{H} = \left(\left(H_\kappa, \langle .,. \rangle_{H_\kappa}\right) + \{1\}\right)^Q$ is the class of functions $h = (h_k)_{1 \le k \le Q}$ of the form:

$$h(.) = \left(\sum_{i=1}^{m_k} \beta_{ik} \kappa(x_{ik}, .) + b_k \right)_{1 \le k \le Q}$$

where the x_{ik} are elements of $\mathcal{X}$ (the β_{ik} and b_k are scalars) as well as the limits of these functions when the sets $\{x_{ik} : 1 \le i \le m_k\}$ become dense in $\mathcal{X}$ in the norm induced by the dot product (see for instance Wahba, 1999). Due to (13), $\mathcal{H}$ can also be seen as a multivariate affine model on $\Phi(\mathcal{X})$. Functions h can then be rewritten as:

$$h(.) = (\langle w_k, . \rangle + b_k)_{1 \le k \le Q}$$

where the vectors w_k are elements of $E_{\Phi(\mathcal{X})}$. They are thus described by the pair $(\mathbf{w}, \mathbf{b})$ with $\mathbf{w} = (w_k)_{1 \le k \le Q} \in E_{\Phi(\mathcal{X})}^Q$ and $\mathbf{b} = (b_k)_{1 \le k \le Q} \in \mathbb{R}^Q$. Let $\bar{\mathcal{H}}$ stand for the product space H_κ^Q whose functions $\bar{h} = (\langle w_k, . \rangle)_{1 \le k \le Q}$ are seen as functions on $\Phi(\mathcal{X})$. Its norm $\|.\|_{\bar{\mathcal{H}}}$ is given by:

$$\forall \bar{h} \in \bar{\mathcal{H}},\ \left\|\bar{h}\right\|_{\bar{\mathcal{H}}} = \sqrt{\sum_{k=1}^{Q} \|w_k\|^2} = \|\mathbf{w}\|,$$

where $\|w_k\| = \sqrt{\langle w_k, w_k \rangle}$. $\bar{\mathcal{H}}$ also represents the restriction of $\mathcal{H}$ to the functions satisfying $\mathbf{b} = 0$. For convenience, $E_{\Phi(\mathcal{X})}^Q$ is endowed with a second norm, $\|.\|_\infty$. It is defined by $\|\mathbf{w}\|_\infty = \max_{1 \le k \le Q} \|w_k\|$. With these definitions at hand, a generic definition of the M-SVMs can be formulated as follows.

Definition 42 (M-SVM) *Let $((x_i, y_i))_{1 \le i \le m} \in (\mathcal{X} \times [\![1, Q]\!])^m$. A Q-*category M-SVM *is a large margin discriminant model obtained by minimizing over the hyperplane $\sum_{k=1}^Q h_k = 0$ of $\mathcal{H}$ an objective function J of the form:*

$$J(h) = \sum_{i=1}^{m} \ell_{\text{M-SVM}}(y_i, h(x_i)) + \lambda \|\mathbf{w}\|^2 \tag{14}$$

where the data fit component, used in place of the empirical (margin) risk, involves a loss function $\ell_{\text{M-SVM}}$ which is convex.

In accordance with the notations of Section 2.2 and Section 4.3, in what follows, $\tilde{\mathcal{H}}$ will designate the (univariate) affine model corresponding to the bi-class SVMs. The different M-SVMs only differ in the nature of the function $\ell_{\text{M-SVM}}$. This one is systematically built around the standard *hinge loss* of bi-class SVMs. This function, from $\tilde{\mathcal{H}} \times \mathcal{X} \times \{-1, 1\}$ into $\mathbb{R}_+$, maps $\left(\tilde{h}, x, y\right)$ to $\left(1 - y\tilde{h}(x)\right)_+$, where $(t)_+ = \max(0, t)$. Three main models of M-SVMs can be found in literature. The first one in chronological order was introduced independently by Weston and Watkins (1998) and by Blanz and Vapnik (Blanz, personal communication). It corresponds to a loss function ℓ_{WW} given by:

$\ell_{\mathrm{WW}}(y,h(x)) = \sum_{k\neq y}(1-h_y(x)+h_k(x))_+$. Then came the model of Crammer and Singer (2001), model built around $\bar{\mathcal{H}}$, one advantage of which consists in the fact that it requires one single slack variable per training example. Its loss function is $\ell_{\mathrm{CS}}\left(y,\bar{h}(x)\right) = \left(1-\bar{h}_y(x)+\max_{k\neq y}\bar{h}_k(x)\right)_+$. The last model to date is the one of Lee et al. (2004), where $\ell_{\mathrm{LLW}}(y,h(x)) = \sum_{k\neq y}\left(h_k(x)+\frac{1}{Q-1}\right)_+$. Its specificity is that asymptotically, it implements the optimal classification rule, that is, Bayes decision rule. Indeed, this property of *infinite-sample consistency* is not shared by the two first M-SVMs, as was shown by Zhang (2004) and Tewari and Bartlett (2007). For all three machines, a representer theorem establishes that the function selected by the training procedure is of the form:

$$h(.) = \left(\sum_{i=1}^{m}\beta_{ik}\kappa(x_i,.)+b_k\right)_{1\leq k\leq Q}. \tag{15}$$

The lines of reasoning highlighting the fact that a bi-class SVM is intrinsically a large margin classifier can be extended easily to the M-SVMs. This requires however to discuss the form taken by the penalty component of the objective function. Indeed, the notion of multi-class margin given by Definition 5 involves differences between outputs, which suggests to use such penalty terms as $\max_{k<l}\|w_k-w_l\|^2$ or $\sum_{k<l}\|w_k-w_l\|^2$. However, this raises the difficulty that the function minimizing (14) is then defined up to an additive constant. The solution is provided by the restriction $\sum_k h_k = 0$. Under this hypothesis, the equation $\sum_{k<l}\|w_k-w_l\|^2 = Q\sum_k\|w_k\|^2 = Q\|\mathbf{w}\|^2$ justifies the use of $\|\mathbf{w}\|^2$ as penalty term.

Our generalized Sauer-Shelah lemma, Lemma 39, holds for classes of functions with bounded range (taking values in $[-M,M]^Q$). We now introduce the standard hypotheses on $\mathcal{X}$ ($\Phi(\mathcal{X})$) and $\mathcal{H}$ which will allow us to formulate the upper bound on the margin Natarajan dimension of interest.

Hypotheses 43 *To upper bound the capacity of a Q-category M-SVM, the following hypotheses and constraints are introduced regarding its domain and its parameters:*

1. *$\Phi(\mathcal{X})$ is included in the ball of radius $\Lambda_{\Phi(\mathcal{X})}$ about the origin in $E_{\Phi(\mathcal{X})}$;*
2. *the vector $\mathbf{w}$ satisfies $\|\mathbf{w}\|_\infty \leq \Lambda_w$;*
3. *the vector $\mathbf{b}$ belongs to $[-\beta,\beta]^Q$.*

With these hypotheses at hand, Lemma 39 can be applied to the corresponding subset of $\mathcal{H}$, by setting $M = \Lambda_w\Lambda_{\Phi(\mathcal{X})}+\beta$.

7.2 Switching from $\Delta^{\#}\mathcal{H}$ to $\Delta^{\#}\bar{\mathcal{H}}$

The computation of an upper bound on the margin Natarajan dimension is easier when the model is linear than when it is affine. Exactly as in the case of the transition from the class $\Delta^{\#}_{\gamma}\mathcal{G}$ to the class $\Delta^{\#}\mathcal{G}$ (see Section 5.1), the corresponding change is easier to perform when working with covering numbers. To that end, one can make use of the following lemma, the proof of which is inspired from the proof of Lemma 2.4 in Alon et al. (1997).

Lemma 44 *Let $\mathcal{H}$ be the class of functions that a Q-category M-SVM can implement under Hypotheses 43. Let $\bar{\mathcal{H}}$ be the subset of $\mathcal{H}$ corresponding to the functions satisfying $\mathbf{b}=0$. Let $\varepsilon \in \mathbb{R}^*_+$*

and $n \in \mathbb{N}^*$. Then

$$\mathcal{N}^{(p)}(\varepsilon, \Delta^{\#}\mathcal{H}, n) \leq \left(2\left\lceil \frac{\beta}{\varepsilon} \right\rceil + 1\right)^{Q} \mathcal{N}^{(p)}(\varepsilon/2, \Delta^{\#}\bar{\mathcal{H}}, n).$$

Proof Let $B =$

$$\left\{-\beta, -\left(\left\lceil \frac{\beta}{\varepsilon} \right\rceil - 1\right)\varepsilon, -\left(\left\lceil \frac{\beta}{\varepsilon} \right\rceil - 2\right)\varepsilon, \ldots, -2\varepsilon, -\varepsilon, 0, \varepsilon, 2\varepsilon, \ldots, \left(\left\lceil \frac{\beta}{\varepsilon} \right\rceil - 2\right)\varepsilon, \left(\left\lceil \frac{\beta}{\varepsilon} \right\rceil - 1\right)\varepsilon, \beta\right\}.$$

By construction, B^Q is a proper $\varepsilon/2$-net of $[-\beta,\beta]^Q$ in the ℓ_∞ norm. For $x^n \in \mathcal{X}^n$, let $\overline{\Delta^{\#}\bar{\mathcal{H}}}(\varepsilon, x^n)$ be a proper $\varepsilon/2$-net of $\Delta^{\#}\bar{\mathcal{H}}$ in the d_{x^n} pseudo-metric. We make the assumption that $\overline{\Delta^{\#}\bar{\mathcal{H}}}(\varepsilon, x^n)$ is of minimal cardinality, that is to say $\left|\overline{\Delta^{\#}\bar{\mathcal{H}}}(\varepsilon, x^n)\right| = \mathcal{N}^{(p)}(\varepsilon/2, \Delta^{\#}\bar{\mathcal{H}}, d_{x^n})$. Then, due to the triangle inequality, $\overline{\Delta^{\#}\bar{\mathcal{H}}}(\varepsilon, x^n) \times B^Q$ is a proper ε-net of $\Delta^{\#}\mathcal{H}$ in the d_{x^n} pseudo-metric. Since the cardinality of B^Q is $\left(2\left\lceil \frac{\beta}{\varepsilon} \right\rceil + 1\right)^Q$, this ε-net is of cardinality $\left(2\left\lceil \frac{\beta}{\varepsilon} \right\rceil + 1\right)^Q \mathcal{N}^{(p)}(\varepsilon/2, \Delta^{\#}\bar{\mathcal{H}}, d_{x^n})$. As a consequence, $\mathcal{N}^{(p)}(\varepsilon, \Delta^{\#}\mathcal{H}, d_{x^n}) \leq \left(2\left\lceil \frac{\beta}{\varepsilon} \right\rceil + 1\right)^Q \mathcal{N}^{(p)}(\varepsilon/2, \Delta^{\#}\bar{\mathcal{H}}, d_{x^n})$. Taking the maximum of both sides of this inequality over all the possible sequences x^n in $\mathcal{X}^n$ thus concludes the proof. ■

Note that, with little additional work, a tighter bound results from exploiting the restriction $\sum_k h_k = 0$.

7.3 Upper Bounding the Margin Natarajan Dimension of $\Delta\bar{\mathcal{H}}$

In this section, we follow the sketch of the proof of Theorem 4.6 in Bartlett and Shawe-Taylor (1999).

Lemma 45 *Let $\bar{\mathcal{H}}$ be the class of functions that a Q-category M-SVM can implement under Hypotheses 43, and the additional constraint $\mathbf{b} = 0$. Let $\varepsilon \in \mathbb{R}_+^*$ and $n \in \mathbb{N}^*$. If a subset $s_{\mathcal{X}^n} = \{x_i : 1 \leq i \leq n\}$ of $\mathcal{X}$ is N-shattered with margin ε by $\Delta\bar{\mathcal{H}}$, then there exists a subset $s_{\mathcal{X}^p}$ of $s_{\mathcal{X}^n}$ of cardinality p equal to $\left\lceil \frac{n}{\binom{Q}{2}} \right\rceil$ such that for every partition of $s_{\mathcal{X}^p}$ into two subsets s_1 and s_2, the following bound holds true:*

$$\left\| \sum_{x_i \in s_1} \Phi(x_i) - \sum_{x_i \in s_2} \Phi(x_i) \right\| \geq \frac{\left\lceil \frac{n}{\binom{Q}{2}} \right\rceil}{\Lambda_w} \varepsilon. \tag{16}$$

Proof Suppose that $s_{\mathcal{X}^n} = \{x_i : 1 \leq i \leq n\}$ is a subset of $\mathcal{X}$ N-shattered with margin ε by $\Delta\bar{\mathcal{H}}$. Let $(I(s_{\mathcal{X}^n}), v_b)$ witness this shattering. Thanks to Proposition 30, without loss of generality, we can assume that $I(s_{\mathcal{X}^n})$ satisfies the constraint: $\forall i \in [\![1, n]\!], i_1(x_i) < i_2(x_i)$. According to the pigeonhole principle, there is at least one couple of indexes (k_0, l_0) with $1 \leq k_0 < l_0 \leq Q$ such that there are at least $p = \left\lceil \frac{n}{\binom{Q}{2}} \right\rceil$ points in $s_{\mathcal{X}^n}$ for which the couple $(i_1(x_i), i_2(x_i))$ is (k_0, l_0). For the sake of simplicity, the points in $s_{\mathcal{X}^n}$ are reordered in such a way that the p first of them exhibit this property. The corresponding subset of $s_{\mathcal{X}^n}$ is denoted $s_{\mathcal{X}^p}$. This means that for all vector $v_y = (y_i)$ in $\{-1, 1\}^n$,

there is a function $\bar{h}_y$ in $\bar{\mathcal{H}}$ characterized by the vector $\mathbf{w}_y = (w_{y,k})_{1\leq k\leq Q}$ such that:

$$\forall i \in [\![1,p]\!], \left\{ \begin{array}{ll} \text{if } y_i = & 1, \quad \Delta\bar{h}_{y,k_0}(x_i) - b_i \geq \varepsilon \\ \text{if } y_i = & -1, \quad \Delta\bar{h}_{y,l_0}(x_i) + b_i \geq \varepsilon \end{array} \right. . \tag{17}$$

By definition of $\bar{\mathcal{H}}$ and the margin operator Δ, this is equivalent to:

$$\forall i \in [\![1,p]\!], \left\{ \begin{array}{ll} \text{if } y_i = & 1, \quad \frac{1}{2}\left(\langle w_{y,k_0}, \Phi(x_i)\rangle - \max_{k\neq k_0}\langle w_{y,k}, \Phi(x_i)\rangle\right) - b_i \geq \varepsilon \\ \text{if } y_i = & -1, \quad \frac{1}{2}\left(\langle w_{y,l_0}, \Phi(x_i)\rangle - \max_{k\neq l_0}\langle w_{y,k}, \Phi(x_i)\rangle\right) + b_i \geq \varepsilon \end{array} \right.$$

and thus implies

$$\forall i \in [\![1,p]\!], \left\{ \begin{array}{ll} \text{if } y_i = & 1, \quad \frac{1}{2}\langle w_{y,k_0} - w_{y,l_0}, \Phi(x_i)\rangle - b_i \geq \varepsilon \\ \text{if } y_i = & -1, \quad \frac{1}{2}\langle w_{y,l_0} - w_{y,k_0}, \Phi(x_i)\rangle + b_i \geq \varepsilon \end{array} \right. . \tag{18}$$

Consider now any partition of $s_{\mathcal{X}^p}$ into two subsets s_1 and s_2. Consider any vector v_y in $\{-1,1\}^n$ such that $y_i = 1$ if $x_i \in s_1$ and $y_i = -1$ if $x_i \in s_2$. It results from (18) that:

$$\frac{1}{2}\langle w_{y,k_0} - w_{y,l_0}, \sum_{x_i\in s_1}\Phi(x_i)\rangle - \sum_{x_i\in s_1} b_i + \frac{1}{2}\langle w_{y,l_0} - w_{y,k_0}, \sum_{x_i\in s_2}\Phi(x_i)\rangle + \sum_{x_i\in s_2} b_i \geq |s_{\mathcal{X}^p}|\,\varepsilon$$

which simplifies into

$$\frac{1}{2}\langle w_{y,k_0} - w_{y,l_0}, \sum_{x_i\in s_1}\Phi(x_i) - \sum_{x_i\in s_2}\Phi(x_i)\rangle - \sum_{x_i\in s_1} b_i + \sum_{x_i\in s_2} b_i \geq p\varepsilon.$$

Conversely, consider any vector v_y such that $y_i = -1$ if $x_i \in s_1$ and $y_i = 1$ if $x_i \in s_2$. We have:

$$\frac{1}{2}\langle w_{y,l_0} - w_{y,k_0}, \sum_{x_i\in s_1}\Phi(x_i) - \sum_{x_i\in s_2}\Phi(x_i)\rangle + \sum_{x_i\in s_1} b_i - \sum_{x_i\in s_2} b_i \geq p\varepsilon.$$

As a consequence, if $\sum_{x_i\in s_1} b_i - \sum_{x_i\in s_2} b_i \geq 0$, there is a function $\bar{h}_y$ in $\bar{\mathcal{H}}$ such that

$$\frac{1}{2}\langle w_{y,k_0} - w_{y,l_0}, \sum_{x_i\in s_1}\Phi(x_i) - \sum_{x_i\in s_2}\Phi(x_i)\rangle \geq \left\lceil \frac{n}{\binom{Q}{2}} \right\rceil \varepsilon \tag{19}$$

whereas if $\sum_{x_i\in s_1} b_i - \sum_{x_i\in s_2} b_i < 0$, there is another function $\bar{h}_y$ in $\bar{\mathcal{H}}$ such that

$$\frac{1}{2}\langle w_{y,l_0} - w_{y,k_0}, \sum_{x_i\in s_1}\Phi(x_i) - \sum_{x_i\in s_2}\Phi(x_i)\rangle \geq \left\lceil \frac{n}{\binom{Q}{2}} \right\rceil \varepsilon. \tag{20}$$

Applying the Cauchy-Schwarz inequality to (19) and (20) yields

$$\frac{1}{2}\|w_{y,k_0} - w_{y,l_0}\| \left\| \sum_{x_i\in s_1}\Phi(x_i) - \sum_{x_i\in s_2}\Phi(x_i) \right\| \geq \left\lceil \frac{n}{\binom{Q}{2}} \right\rceil \varepsilon,$$

which thus holds true irrespective of the value of $\sum_{x_i\in s_1} b_i - \sum_{x_i\in s_2} b_i$. Finally, (16) directly springs from this last bound, as a consequence of fact that the constraint $\|\mathbf{w}\|_\infty \leq \Lambda_w$ implies $1/2\max_{1\leq k<l\leq Q}\|w_k - w_l\| \leq \Lambda_w$. ■

Remark 46 *The proof of Lemma 45 does not hold any more if one uses the Δ^* operator in place of the Δ operator. Indeed, reformulating (17) with Δ^* in place of Δ, one cannot derive (18) any more. This is precisely the reason why it is specifically the Δ operator which appears in the hypotheses of Lemma 45 and, by way of consequence, the final bound on the margin Natarajan dimension (see Theorem 48 below).*

Lemma 47 (Bartlett and Shawe-Taylor, 1999, Lemma 4.3) *If $\Phi(\mathcal{X})$ is included in the ball of radius $\Lambda_{\Phi(\mathcal{X})}$ about the origin in $E_{\Phi(\mathcal{X})}$, then for all $n \in \mathbb{N}^*$, all subset $s_{\mathcal{X}^n} = \{x_i : 1 \leq i \leq n\}$ of $\mathcal{X}$ can be partitioned into two subsets s_1 and s_2 satisfying*

$$\left\| \sum_{x_i \in s_1} \Phi(x_i) - \sum_{x_i \in s_2} \Phi(x_i) \right\| \leq \sqrt{n} \Lambda_{\Phi(\mathcal{X})}. \tag{21}$$

The following theorem is a direct consequence of Lemma 45 and Lemma 47.

Theorem 48 *Let $\bar{\mathcal{H}}$ be the class of functions that a Q-category M-SVM can implement under Hypotheses 43, and the additional constraint $\mathbf{b} = 0$. Then, for any positive real value ε, the following bound holds true:*

$$N\text{-}dim\left(\Delta\bar{\mathcal{H}}, \varepsilon\right) \leq \binom{Q}{2} \left(\frac{\Lambda_w \Lambda_{\Phi(\mathcal{X})}}{\varepsilon}\right)^2. \tag{22}$$

Proof Let $s_{\mathcal{X}^n}$ be a subset of $\mathcal{X}$ of cardinality n N-shattered with margin ε by $\Delta\bar{\mathcal{H}}$. According to Lemma 45, there is at least a subset $s_{\mathcal{X}^p}$ of $s_{\mathcal{X}^n}$ of cardinality $p = \left\lceil \frac{n}{\binom{Q}{2}} \right\rceil$ satisfying (16) for all its partitions into two subsets s_1 and s_2. Since, according to Lemma 47, there is at least one of these partitions for which (21) holds true,

$$\frac{p}{\Lambda_w}\varepsilon \leq \sqrt{p}\Lambda_{\Phi(\mathcal{X})}$$

which implies that

$$p \leq \left(\frac{\Lambda_w \Lambda_{\Phi(\mathcal{X})}}{\varepsilon}\right)^2.$$

Since $n \leq \binom{Q}{2} p$, one finally obtains

$$n \leq \binom{Q}{2} \left(\frac{\Lambda_w \Lambda_{\Phi(\mathcal{X})}}{\varepsilon}\right)^2$$

which concludes the proof. ■

7.4 Discussion

Proposition 31 states that in the bi-class case, there is only one γ-Ψ-dimension, which corresponds to the fat-shattering dimension. Thus, it is satisfactory to notice that for $Q = 2$, (22) becomes

$$P_\varepsilon\text{-dim}\left(H_\kappa\right) \leq \left(\frac{\Lambda_w \Lambda_{\Phi(\mathcal{X})}}{\varepsilon}\right)^2$$

which is precisely the bound provided by Theorem 4.6 in Bartlett and Shawe-Taylor (1999) (see also Remark 1 in Gurvits, 2001), that is, the tightest bound on the fat-shattering dimension of a linear classifier currently available. In the general case, Theorem 48 tells us that the margin Natarajan dimension of a Q-category M-SVM can be bounded from above by a uniform bound on the fat-shattering dimensions of its separating hyperplanes (defined by the equation $\langle w_k - w_l, \Phi(x) \rangle = 0$) times the number of those hyperplanes, $\binom{Q}{2}$. It must be borne in mind that this expression is directly connected with the idea at the basis of the definition of the Ψ-dimensions (see the discussion in Section 4.1), which is to simulate the implementation of a decomposition scheme, and take benefit of this to make use of standard bi-class results. In the case of the Natarajan dimension, this scheme corresponds to the one-against-one method. The terms $\binom{Q}{2}$ and $\|\mathbf{w}\|_\infty$ then appear in (22) as a consequence of the fact that all the pairs of categories are considered independently one from the other and play an utterly symmetrical part (we need a bound on $1/2 \max_{1 \leq k < l \leq Q} \|w_k - w_l\|$). Obviously, a tighter bound should result from taking into account the fact that the $\binom{Q}{2}$ binary classifiers are not independent, since they are based on a common set of Q vectors w_k. Here appears once more the need to derive original solutions for the multi-class case, instead of simple extensions of bi-class results.

Deriving a nontrivial bound on N-dim $\left(\Delta \bar{\mathcal{H}}, \varepsilon\right)$ in terms of $\|\mathbf{w}\|$, that is, a tighter bound than the one resulting from just replacing in the hypotheses of Theorem 48 $\|\mathbf{w}\|_\infty$ with $\|\mathbf{w}\|$, remains an open problem. The fact that the norm used in the penalty term of the objective function (14) and the one appearing in the upper bound on the margin Natarajan dimension are different is unsatisfactory. The point is that, so far, no one has put forward a theoretical argument (guaranteed risk) to justify the use of $\|\mathbf{w}\|$, whereas the use of $\|\mathbf{w}\|_\infty$ as penalty term, considered only in Guermeur (2002), raises significant technical difficulties. Indeed, in that case, the convex programming problem corresponding to the training algorithm cannot be solved by means of Lagrangian duality any more, since one cannot compute the gradient of the Lagrangian function with respect to the vectors w_k. In that sense, there remains a gap to fill between theory and practice.

8. γ-Ψ-dimensions and Implementation of the SRM Inductive Principle

In this section, we discuss the significance of the main results of the paper. We first summarize the specificities of the multi-class case highlighted by their proofs, and then outline an application of our bound on the risk of M-SVMs for model selection.

8.1 Characterization of Relevant Information

The main results of this article involve two distinct margin operators, Δ and Δ^*. Theorem 22, the basic uniform convergence result on which all this study is based, holds true for both of them. However, we pointed out in Remark 36 the reason why the generalized Sauer-Shelah lemma (Lemmas 38 and 39) requires specifically the use of Δ^*. On the contrary, Remark 46 highlights the fact that the proof of the bound on the margin Natarajan dimension of the M-SVMs, Theorem 48, makes use of a specific property of Δ. Fortunately, the connection between the capacities of $\Delta^* \mathcal{G}$ and $\Delta \mathcal{G}$ is provided by Lemmas 35 and 37. These observations highlight the fact that the link between separation and shattering capacity is more complex in the multi-class case than in the bi-class case (for which we simply have $\Delta = \Delta^*$). At different steps of the reasoning, different pieces of information on the behaviour of the functions of interest are needed. One must provide neither too many nor too few of

them. It is a bit disappointing to notice that the computation of the bound on the margin Natarajan dimension requires more information than simply the index of the highest output and the difference between the two highest outputs, that is, what is relevant to determine both the classification performed and the confidence one can have in the accuracy of this classification. This suggests that some improvement could be made to our generalization of the standard bi-class results, regarding for instance the choice of the functional pseudo-metric. However, it is difficult to figure out how these changes could remain compatible with the whole line of reasoning leading to the bound on the risk of the M-SVMs. Indeed, the choices we made to extend the VC theory to the case of large margin multi-category discriminant models and apply it to M-SVMs were primarily governed by one concern: allowing a natural extension of the proof of Lemma 3.3 in Alon et al. (1997) and the proof of Theorem 4.6 in Bartlett and Shawe-Taylor (1999) to the multi-class case. As a consequence, the question could be now: can we develop our theory without making use of those two pillars of the standard theory?

8.2 Application for Model Selection

When working with SVMs, performing model selection amounts to choosing the value of the "soft margin parameter" C, the kernel κ and the values of its parameters. Cross-validation was initially regarded as the method of choice to perform this task, although it exhibits some drawbacks, as was first pointed out by Stone (1977). This strategy has induced many authors to derive upper bounds on the leave-one-out error of SVMs (see Chapelle et al., 2002, for a survey). The most widely used of them is probably the famous "radius-margin bound", for which several multi-class extensions have been proposed independently by Wang et al. (2005); Darcy and Guermeur (2005); Monfrini and Guermeur (2007), as criteria for the choice of the values of the hyperparameters of M-SVMs (or SVMs involved in decomposition schemes). Care was taken to the fact that they could be differentiated with respect to those parameters, in order to make the optimization procedure tractable.

With that difficulty in mind, it appears that model selection for SVMs, either bi-class or multi-class, made a great stride when Hastie et al. (2004) introduced their algorithm fitting the entire path of SVM solutions for every value of C (see also Lee and Cui, 2006, for an algorithm dedicated to the M-SVM of Lee and co-authors). Indeed, with this algorithm at hand, requirements in computational time are drastically reduced, which makes it possible to use new criteria (tighter bounds on the risk) for the selection of C. The idea is simple: starting with a small value of C, it suffices to follow the path, that is, increase progressively the value of C, and assess the bound at each step. Eventually, the value selected is the one corresponding to the smallest value of the bound. This is precisely what was done in Guermeur et al. (2005). In that paper, taking our inspiration from Williamson et al. (2000), we used a bound on the generalization error of M-SVMs obtained as a function of a bound on the entropy numbers of the evaluation operator. The corresponding experimental protocol provides us with an easy way to assess the usefulness of our new bound for model selection. For a given position in the path (a given value of C), all what has to be done is to optimize the guaranteed risk with respect to the margin parameter γ. With the notation introduced in (15), the formula at the basis of the computation of the upper bound on the margin Natarajan dimension is the following one:

$$\forall (k,l) \in [\![1, Q]\!]^2 \,,\ \| w_k - w_l \|^2 = \sum_{i=1}^{m} \sum_{j=1}^{m} \left(\beta_{ik} - \beta_{il} \right) \left(\beta_{jk} - \beta_{jl} \right) \kappa \left(x_i, x_j \right)$$

(obviously, one benefits from using in the computations $\frac{1}{4}\max_{k<l}\|w_k - w_l\|^2$ in place of its upper bound $\|\mathbf{w}\|_\infty^2$).

9. Conclusions and Ongoing Research

In this article, the standard theories of large margin bi-class classifiers and Q-class classifiers taking values in $[\![1, Q]\!]$ have been unified to give birth to a VC theory of large margin multi-class classifiers. This could be done in a straightforward way, by extending concepts and results from only four references: Ben-David et al. (1995), Alon et al. (1997), Bartlett (1998), and Bartlett and Shawe-Taylor (1999). The main difficulty was to identify the need to introduce two margin operators, Δ and Δ^*. The generalized VC dimensions at the center of the new theory are the γ-Ψ-dimensions. They can be seen either as scale-sensitive extensions of the Ψ-dimensions, or multivariate extensions of the fat-shattering dimension. In particular, they characterize learnability for the classes of functions of interest.

It is possible to select the most appropriate of these dimensions as a function of the model studied. In the case of the multi-class SVMs, we have found the margin Natarajan dimension to be the easiest to bound from above making use of standard results derived with the fat-shattering dimension. As a consequence, all the M-SVMs proposed so far can now be evaluated in the unifying framework of the implementation of the SRM inductive principle. Indeed, the main practical interest of guaranteed risks based on γ-Ψ-dimensions should regard the implementation of this learning principle. They make it possible to characterize the variation of the capacity of large margin multi-category discriminant models based on classes of parametric functions with respect to the constraints on their domain and parameters. An obvious application of this study is in model selection, for instance to choose the values of the "soft margin parameter" C and the kernel parameters of M-SVMs.

Readers more interested in computing sample complexities than in the characterization of Glivenko-Cantelli classes, capacity control or model selection, should be aware of the fact that sharper bounds should result from using different sources of inspiration, although even in that case, the lessons drawn from the present study should still prove useful. An obvious possibility is represented by new PAC-Bayes bounds (Ambroladze et al., 2007), or, to remain nearer to the present study, new tools of concentration theory and empirical processes (Talagrand, 1995, 1996; Ledoux, 1996; Massart, 2000; Lugosi, 2004). They make it possible, for instance, to work with data dependent capacity measures such as the empirical VC entropy. A great survey of the recent advances in this field, especially focusing on Rademacher averages, is provided by Boucheron et al. (2005). Regarding more specifically pattern recognition SVMs, the results the extension of which appears most promising are those reported in Bousquet (2002), Steinwart and Scovel (2005), and Blanchard et al. (2007). Performing these multi-class extensions is the subject of an ongoing work.

Acknowledgments

This work was initiated with H. Paugam-Moisy and A. Elisseeff. The author would like to thank the anonymous reviewers for their comments. It is also a pleasure to thank M. Sebag, P. Bartlett, S. Kroon, R. Vert and M. Warmuth for instructive discussions and bibliographical help, as well as E. Monfrini and F. Sur for carefully reading this manuscript.

Appendix A. Technical Lemmas

This appendix is devoted to technical lemmas that are at the basis of the proofs of the main theorems of the paper.

Lemma 49 Jogdeo and Samuels, 1968, Theorem 3.2. *Let T be a random variable described by a binomial distribution with parameters n and p ($T \hookrightarrow \mathcal{B}(n,p)$). Then its median is either $\lfloor np \rfloor$ or $\lfloor np \rfloor + 1$. Moreover, if np is an integer, the median is simply np.*

Lemma 50 *Let $D_{2m} = ((X_i,Y_i))_{1\leq i\leq 2m}$ be a $2m$-sample of independent copie of (X,Y). Let $D_m = ((X_i,Y_i))_{1\leq i\leq m}$ and $\tilde{D}_m = \left(\left(\tilde{X}_i,\tilde{Y}_i\right)\right)_{1\leq i\leq m} = ((X_{m+i},Y_{m+i}))_{1\leq i\leq m}$. $\mathbb{P}_{D_m}$ is a probability over the sample D_m, and $\mathbb{P}_{D_{2m}}$ is a probability over D_{2m}. The distribution of the random variable $\sup_{g\in\mathcal{G}}\left(R(g) - R_{\gamma,D_m}(g)\right)$ is connected with the distribution of the random variable $\sup_{g\in\mathcal{G}}\left(R_{\tilde{D}_m}(g) - R_{\gamma,D_m}(g)\right)$ by the inequality*

$$\mathbb{P}_{D_m}\left(\sup_{g\in\mathcal{G}}\left(R(g) - R_{\gamma,D_m}(g)\right) > \varepsilon\right) \leq 2\mathbb{P}_{D_{2m}}\left(\sup_{g\in\mathcal{G}}\left(R_{\tilde{D}_m}(g) - R_{\gamma,D_m}(g)\right) \geq \varepsilon - \frac{1}{m}\right).$$

Proof The proof of this lemma is inspired from the proof of Vapnik's basic lemma in Vapnik (1998, Section 4.5.1). For $n\in\mathbb{N}^*$, let $z^{2n} = ((x_i,y_i))_{1\leq i\leq 2n}$ be an element of $\mathcal{Z}^{2n}$. In what follows, we will use z^n to designate its "first half", whereas $\tilde{z}^n$, will designate its "second half". $\tilde{z}^n = ((\tilde{x}_i,\tilde{y}_i))_{1\leq i\leq n}$, with $(\tilde{x}_i,\tilde{y}_i) = (x_{n+i},y_{n+i})$. Since D_m and $\tilde{D}_m$ are supposed to be independent, by definition:

$$\mathbb{P}_{D_{2m}}\left(\sup_{g\in\mathcal{G}}\left(R_{\tilde{D}_m}(g) - R_{\gamma,D_m}(g)\right) \geq \varepsilon - \frac{1}{m}\right) =$$

$$\int_{\mathcal{Z}^{2m}} \mathbb{1}\left[\sup_{g\in\mathcal{G}}\left(R_{\tilde{z}^m}(g) - R_{\gamma,z^m}(g)\right) \geq \varepsilon - \frac{1}{m}\right] dP^{2m}(z^{2m}),$$

and one can apply Fubini's theorem for nonnegative measurable functions (see Rudin, 1987, Section 8.8) to the product measure P^{2m}, which gives:

$$\mathbb{P}_{D_{2m}}\left(\sup_{g\in\mathcal{G}}\left(R_{\tilde{D}_m}(g) - R_{\gamma,D_m}(g)\right) \geq \varepsilon - \frac{1}{m}\right) =$$

$$\int_{\mathcal{Z}^m} dP^m(z^m) \int_{\mathcal{Z}^m} \mathbb{1}\left[\sup_{g\in\mathcal{G}}\left(R_{\tilde{z}^m}(g) - R_{\gamma,z^m}(g)\right) \geq \varepsilon - \frac{1}{m}\right] dP^m(\tilde{z}^m).$$

In the inner integral, z^m is fixed. Let Q denote the following event:

$$Q = \left\{z^m = ((x_i,y_i))_{1\leq i\leq m} \in \mathcal{Z}^m : \sup_{g\in\mathcal{G}}\left(R(g) - R_{\gamma,z^m}(g)\right) > \varepsilon\right\}.$$

Restricting the integration domain to Q gives

$$\mathbb{P}_{D_{2m}}\left(\sup_{g\in\mathcal{G}}\left(R_{\tilde{D}_m}(g) - R_{\gamma,D_m}(g)\right) \geq \varepsilon - \frac{1}{m}\right) \geq$$

$$\int_{\mathcal{Q}} dP^m(z^m) \underbrace{\int_{\mathcal{Z}^m} \mathbb{1}\left[\sup_{g\in\mathcal{G}}\left(R_{\tilde{z}^m}(g) - R_{\gamma,z^m}(g)\right) \geq \varepsilon - \frac{1}{m}\right] dP^m(\tilde{z}^m)}_{I}. \tag{23}$$

I is an integral which is calculated for a fixed z^m satisfying

$$\sup_{g\in\mathcal{G}}\left(R(g) - R_{\gamma,z^m}(g)\right) > \varepsilon.$$

Consequently, there exists a function g^* in $\mathcal{G}$ such that

$$R(g^*) - R_{\gamma,z^m}(g^*) \geq \varepsilon.$$

By definition of g^*, the following inequality holds

$$I \geq \int_{\mathcal{Z}^m} \mathbb{1}\left[R_{\tilde{z}^m}(g^*) - R_{\gamma,z^m}(g^*) \geq \varepsilon - \frac{1}{m}\right] dP^m(\tilde{z}^m).$$

$$\left\{\begin{array}{l} R(g^*) - R_{\gamma,z^m}(g^*) \geq \varepsilon \\ R_{\tilde{z}^m}(g^*) - R(g^*) \geq -\frac{1}{m} \end{array}\right. \Longrightarrow R_{\tilde{z}^m}(g^*) - R_{\gamma,z^m}(g^*) \geq \varepsilon - \frac{1}{m}.$$

As a consequence

$$I \geq \int_{\mathcal{Z}^m} \mathbb{1}\left[R_{\tilde{z}^m}(g^*) - R(g^*) \geq -\frac{1}{m}\right] dP^m(\tilde{z}^m).$$

Furthermore

$$\int_{\mathcal{Z}^m} \mathbb{1}\left[R_{\tilde{z}^m}(g^*) - R(g^*) \geq -\frac{1}{m}\right] dP^m(\tilde{z}^m) = \mathbb{P}_{\tilde{D}_m}\left(mR_{\tilde{D}_m}(g^*) \geq mR(g^*) - 1\right). \tag{24}$$

By definition of $R(g^*)$ and $R_{\tilde{D}_m}(g^*)$, $mR_{\tilde{D}_m}(g^*)$ has a binomial distribution with parameters m and $R(g^*)$ $\left(mR_{\tilde{D}_m}(g^*) \hookrightarrow \mathcal{B}(m, R(g^*))\right)$. To bound from below the right-hand side of (24), we make use of a result on the median of random variables following a binomial distribution, Lemma 49. According to this lemma, $mR(g^*) - 1$ is inferior or equal to the median of $mR_{\tilde{D}_m}(g^*)$, and thus, by definition of the median, the right-hand side of (24) is superior or equal to $1/2$. By transitivity, I is also greater that $1/2$. Substituting this lower bound on I into (23) yields

$$\mathbb{P}_{D_{2m}}\left(\sup_{g\in\mathcal{G}}\left(R_{\tilde{D}_m}(g) - R_{\gamma,D_m}(g)\right) \geq \varepsilon - \frac{1}{m}\right) \geq \frac{1}{2}\int_{\mathcal{Q}} dP^m(z^m)$$

or equivalently, by definition of Q:

$$\mathbb{P}_{D_{2m}}\left(\sup_{g\in\mathcal{G}}\left(R_{\tilde{D}_m}(g) - R_{\gamma,D_m}(g)\right) \geq \varepsilon - \frac{1}{m}\right) \geq \frac{1}{2}\mathbb{P}_{D_m}\left(\sup_{g\in\mathcal{G}}\left(R(g) - R_{\gamma,D_m}(g)\right) > \varepsilon\right)$$

which is the result announced. ■

Lemma 51 *The distribution of the random variable* $\sup_{g\in\mathcal{G}}\left(R_{\tilde{D}_m}(g)-R_{\gamma,D_m}(g)\right)$ *is connected with the distribution of the random variable* $\max_{\overline{g}\in\overline{\mathcal{G}}(\gamma,D_{2m})}\left(R_{\gamma/2,\tilde{D}_m}(\overline{g})-R_{\gamma/2,D_m}(\overline{g})\right)$ *by the inequality*

$$\mathbb{P}_{D_{2m}}\left(\sup_{g\in\mathcal{G}}\left(R_{\tilde{D}_m}(g)-R_{\gamma,D_m}(g)\right)\geq\varepsilon-\frac{1}{m}\right)\leq$$

$$\mathbb{P}_{D_{2m}}\left(\max_{\overline{g}\in\overline{\mathcal{G}}(\gamma,D_{2m})}\left(R_{\gamma/2,\tilde{D}_m}(\overline{g})-R_{\gamma/2,D_m}(\overline{g})\right)\geq\varepsilon-\frac{1}{m}\right).$$

Proof $\forall g\in\mathcal{G},\forall(x_i,y_i)\in z^{2m}$,

$$\begin{cases}\Delta^{\#}g_{y_i}(x_i)\leq 0\\ d_{x^{2m}}(\Delta^{\#}_{\gamma}g,\Delta^{\#}_{\gamma}\overline{g})<\frac{\gamma}{2}\end{cases}\Longrightarrow\Delta^{\#}\overline{g}_{y_i}(x_i)<\frac{\gamma}{2}. \tag{25}$$

Similarly,

$$\begin{cases}\Delta^{\#}\overline{g}_{y_i}(x_i)<\frac{\gamma}{2}\\ d_{x^{2m}}(\Delta^{\#}_{\gamma}g,\Delta^{\#}_{\gamma}\overline{g})<\frac{\gamma}{2}\end{cases}\Longrightarrow\Delta^{\#}g_{y_i}(x_i)<\gamma. \tag{26}$$

From (25) it results that if $d_{x^{2m}}(\Delta^{\#}_{\gamma}g,\Delta^{\#}_{\gamma}\overline{g})<\frac{\gamma}{2}$, then

$$R_{z^m}(g)\leq R_{\gamma/2,z^m}(\overline{g}).$$

Similarly, it results from (26) that if $d_{x^{2m}}(\Delta^{\#}_{\gamma}g,\Delta^{\#}_{\gamma}\overline{g})<\frac{\gamma}{2}$, then

$$R_{\gamma/2,z^m}(\overline{g})\leq R_{\gamma,z^m}(g).$$

To sum up, for all g in $\mathcal{G}$, there exists $\overline{g}$ in $\overline{\mathcal{G}}(\gamma,x^{2m})$ such that

$$R_{z^m}(g)-R_{\gamma,z^m}(g)\leq R_{\gamma/2,z^m}(\overline{g})-R_{\gamma/2,z^m}(\overline{g})$$

and thus

$$\mathbb{P}_{D_{2m}}\left(\sup_{g\in\mathcal{G}}\left(R_{\tilde{D}_m}(g)-R_{\gamma,D_m}(g)\right)\geq\varepsilon-\frac{1}{m}\right)=$$

$$\int_{\mathcal{Z}^{2m}}\mathbb{1}\left[\sup_{g\in\mathcal{G}}\left(R_{z^m}(\overline{g})-R_{\gamma,z^m}(\overline{g})\right)\geq\varepsilon-\frac{1}{m}\right]dP^{2m}(z^{2m})\leq$$

$$\int_{\mathcal{Z}^{2m}}\mathbb{1}\left[\max_{\overline{g}\in\overline{\mathcal{G}}(\gamma,x^{2m})}\left(R_{\gamma/2,z^m}(\overline{g})-R_{\gamma/2,z^m}(\overline{g})\right)\geq\varepsilon-\frac{1}{m}\right]dP^{2m}(z^{2m})=$$

$$\mathbb{P}_{D_{2m}}\left(\max_{\overline{g}\in\overline{\mathcal{G}}(\gamma,D_{2m})}\left(R_{\gamma/2,\tilde{D}_m}(\overline{g})-R_{\gamma/2,D_m}(\overline{g})\right)\geq\varepsilon-\frac{1}{m}\right).$$

■

Lemma 52 *Let S_{2m} be a random variable described by the uniform distribution on $\mathfrak{T}_{2m}$. Then*

$$\mathbb{P}_{D_{2m}}\left(\max_{\overline{g}\in\overline{\mathcal{G}}(\gamma,D_{2m})}\left(R_{\gamma/2,\tilde{D}_m}(\overline{g})-R_{\gamma/2,D_m}(\overline{g})\right)\geq\varepsilon-\frac{1}{m}\right)\leq$$

$$\max_{z^{2m}\in\mathcal{Z}^{2m}}\sum_{\overline{g}\in\overline{\mathcal{G}}(\gamma,x^{2m})}\mathbb{P}_{S_{2m}}\left(R_{\gamma/2,S_{2m}(\tilde{z}^m)}(\overline{g})-R_{\gamma/2,S_{2m}(z^m)}(\overline{g})\geq\varepsilon-\frac{1}{m}\right).$$

Proof Since coordinate permutations preserve the product distribution P^{2m},

$$\mathbb{P}_{D_{2m}}\left(\max_{\overline{g}\in\overline{\mathcal{G}}(\gamma,D_{2m})}\left(R_{\gamma/2,\tilde{D}_m}(\overline{g})-R_{\gamma/2,D_m}(\overline{g})\right)\geq\varepsilon-\frac{1}{m}\right)$$

is not affected by a permutation σ. One thus obtains:

$$\forall\sigma\in\mathfrak{T}_{2m},\ \mathbb{P}_{D_{2m}}\left(\max_{\overline{g}\in\overline{\mathcal{G}}(\gamma,D_{2m})}\left(R_{\gamma/2,\tilde{D}_m}(\overline{g})-R_{\gamma/2,D_m}(\overline{g})\right)\geq\varepsilon-\frac{1}{m}\right)=$$

$$\int_{\mathcal{Z}^{2m}}\mathbb{1}\left[\max_{\overline{g}\in\overline{\mathcal{G}}(\gamma,x^{2m})}\left(R_{\gamma/2,\sigma(\tilde{z}^m)}(\overline{g})-R_{\gamma/2,\sigma(z^m)}(\overline{g})\right)\geq\varepsilon-\frac{1}{m}\right]dP^{2m}(z^{2m}).$$

Averaging the summand of the right-hand side over the whole set $\mathfrak{T}_{2m}$ gives:

$$\mathbb{P}_{D_{2m}}\left(\max_{\overline{g}\in\overline{\mathcal{G}}(\gamma,D_{2m})}\left(R_{\gamma/2,\tilde{D}_m}(\overline{g})-R_{\gamma/2,D_m}(\overline{g})\right)\geq\varepsilon-\frac{1}{m}\right)=$$

$$\frac{1}{|\mathfrak{T}_{2m}|}\sum_{\sigma\in\mathfrak{T}_{2m}}\int_{\mathcal{Z}^{2m}}\mathbb{1}\left[\max_{\overline{g}\in\overline{\mathcal{G}}(\gamma,x^{2m})}\left(R_{\gamma/2,\sigma(\tilde{z}^m)}(\overline{g})-R_{\gamma/2,\sigma(z^m)}(\overline{g})\right)\geq\varepsilon-\frac{1}{m}\right]dP^{2m}(z^{2m}).$$

Since the cardinality of $\mathfrak{T}_{2m}$ is finite, summation and integration can be interchanged as follows:

$$\mathbb{P}_{D_{2m}}\left(\max_{\overline{g}\in\overline{\mathcal{G}}(\gamma,D_{2m})}\left(R_{\gamma/2,\tilde{D}_m}(\overline{g})-R_{\gamma/2,D_m}(\overline{g})\right)\geq\varepsilon-\frac{1}{m}\right)=$$

$$\int_{\mathcal{Z}^{2m}}\frac{1}{|\mathfrak{T}_{2m}|}\sum_{\sigma\in\mathfrak{T}_{2m}}\mathbb{1}\left[\max_{\overline{g}\in\overline{\mathcal{G}}(\gamma,x^{2m})}\left(R_{\gamma/2,\sigma(\tilde{z}^m)}(\overline{g})-R_{\gamma/2,\sigma(z^m)}(\overline{g})\right)\geq\varepsilon-\frac{1}{m}\right]dP^{2m}(z^{2m})=$$

$$\int_{\mathcal{Z}^{2m}}\mathbb{P}_{S_{2m}}\left(\max_{\overline{g}\in\overline{\mathcal{G}}(\gamma,x^{2m})}\left(R_{\gamma/2,S_{2m}(\tilde{z}^m)}(\overline{g})-R_{\gamma/2,S_{2m}(z^m)}(\overline{g})\right)\geq\varepsilon-\frac{1}{m}\right)dP^{2m}(z^{2m})\leq$$

$$\max_{z^{2m}\in\mathcal{Z}^{2m}}\mathbb{P}_{S_{2m}}\left(\max_{\overline{g}\in\overline{\mathcal{G}}(\gamma,x^{2m})}\left(R_{\gamma/2,S_{2m}(\tilde{z}^m)}(\overline{g})-R_{\gamma/2,S_{2m}(z^m)}(\overline{g})\right)\geq\varepsilon-\frac{1}{m}\right).\tag{27}$$

By application of the union bound, the right-hand side of (27) can be bounded from above as follows:

$$\max_{z^{2m}\in\mathcal{Z}^{2m}}\mathbb{P}_{S_{2m}}\left(\max_{\overline{g}\in\overline{\mathcal{G}}(\gamma,x^{2m})}\left(R_{\gamma/2,S_{2m}(\tilde{z}^m)}(\overline{g})-R_{\gamma/2,S_{2m}(z^m)}(\overline{g})\right)\geq\varepsilon-\frac{1}{m}\right)\leq$$

$$\max_{z^{2m}\in\mathcal{Z}^{2m}} \sum_{\overline{g}\in\overline{\mathcal{G}}(\gamma,x^{2m})} \mathbb{P}_{S_{2m}}\left(R_{\gamma/2,S_{2m}(\tilde{z}^m)}(\overline{g}) - R_{\gamma/2,S_{2m}(z^m)}(\overline{g}) \geq \varepsilon - \frac{1}{m}\right).$$

■

Lemma 53 (Hoeffding's inequality, Hoeffding, 1963) *For $n \in \mathbb{N}^*$, let $(T_i)_{1\leq i\leq n}$ be a sequence of n independent random variables with zero means and bounded ranges: $a_i \leq T_i \leq b_i$. Then, for all $\eta \in \mathbb{R}_+^*$,*

$$\mathbb{P}\left(\sum_{i=1}^n T_i \geq \eta\right) \leq \exp\left(\frac{-2\eta^2}{\sum_{i=1}^n (b_i - a_i)^2}\right).$$

Lemma 54 *Let S_{2m} be a random variable described by the uniform distribution on $\mathfrak{T}_{2m}$. For all z^{2m} in $\mathcal{Z}^{2m}$ and for all $\overline{g}$ in $\overline{\mathcal{G}}\left(\gamma, x^{2m}\right)$,*

$$\mathbb{P}_{S_{2m}}\left(R_{\gamma/2,S_{2m}(\tilde{z}^m)}(\overline{g}) - R_{\gamma/2,S_{2m}(z^m)}(\overline{g}) \geq \varepsilon - \frac{1}{m}\right) \leq \exp\left(-\frac{m}{2}\left(\varepsilon - \frac{1}{m}\right)^2\right).$$

Proof To bound uniformly the probabilities $\mathbb{P}_{S_{2m}}\left(R_{\gamma/2,S_{2m}(\tilde{z}^m)}(\overline{g}) - R_{\gamma/2,S_{2m}(z^m)}(\overline{g}) \geq \varepsilon - \frac{1}{m}\right)$, we appeal to the classical law of large numbers. For any function $\overline{g}$ in $\overline{\mathcal{G}}\left(\gamma, x^{2m}\right)$, let $(\xi_i)_{1\leq i\leq m}$ be the sequence of losses $\left(\mathbb{1}_{\left\{\Delta^{\#}\overline{g}_{y_i}(x_i)<\gamma/2\right\}}\right)_{1\leq i\leq m}$ (sequence of losses on z^m) and $\left(\tilde{\xi}_i\right)_{1\leq i\leq m}$ the corresponding sequence of losses on $\tilde{z}^m$. Let $\alpha = (\alpha_i)_{1\leq i\leq m}$ be a Rademacher sequence. The terms of interest can then be rewritten as:

$$\mathbb{P}_{S_{2m}}\left(R_{\gamma/2,S_{2m}(\tilde{z}^m)}(\overline{g}) - R_{\gamma/2,S_{2m}(z^m)}(\overline{g}) \geq \varepsilon - \frac{1}{m}\right) = \mathbb{P}_\alpha\left(\frac{1}{m}\sum_{i=1}^m \alpha_i\left(\tilde{\xi}_i - \xi_i\right) \geq \varepsilon - \frac{1}{m}\right). \tag{28}$$

To bound from above the right-hand side of (28), Hoeffding's inequality (Lemma 53) can be used. Since the random variables $\alpha_i\left(\tilde{\xi}_i - \xi_i\right)$ take their values in $[-1,1]$ (more precisely in $[\![-1,1]\!]$), this gives:

$$\mathbb{P}_\alpha\left(\frac{1}{m}\sum_{i=1}^m \alpha_i\left(\tilde{\xi}_i - \xi_i\right) \geq \varepsilon - \frac{1}{m}\right) \leq \exp\left(-\frac{m}{2}\left(\varepsilon - \frac{1}{m}\right)^2\right).$$

■

Lemma 55 Kroon, 2003, Theorem 68 *Let $(\Omega, \mathcal{B}, \mathbb{P})$ be a probability space, let $K \in \mathbb{R}_+^*$ and let*

$$\{E(\alpha_1, \alpha_2, \delta) :\ 0 < \alpha_1, \alpha_2 \leq K,\ \delta \leq 1\}$$

be a set of events satisfying the following conditions:

1. *for all $0 < \alpha \leq K$ and $0 < \delta \leq 1$, $\mathbb{P}(E(\alpha,\alpha,\delta)) \leq \delta$;*
2. *for all $0 < a < 1$ and $0 < \delta \leq 1$, $\bigcup_{\alpha\in(0,K]} E(\alpha a, \alpha, \delta\alpha(1-a))$ is measurable;*

3. for all $0<\alpha_1\leq\alpha\leq\alpha_2\leq K$ *and* $0<\delta_1\leq\delta\leq 1$, $E(\alpha_1,\alpha_2,\delta_1)\subseteq E(\alpha,\alpha,\delta)$.

Then for $(a,\delta)\in(0,1)\times(0,1]$,

$$\mathbb{P}\left(\bigcup_{\alpha\in(0,K]} E\left(\alpha a,\alpha,\frac{\delta\alpha(1-a)}{K}\right)\right)\leq\delta.$$

Lemma 56 Kolmogorov and Tihomirov, 1961, Theorem IV *For every pseudo-metric space* (E,ρ), *every totally bounded subset* E' *of* E *and* $\varepsilon\in\mathbb{R}_+^*$,

$$\mathcal{M}(2\varepsilon,E',\rho)\leq\mathcal{N}^{(p)}(\varepsilon,E',\rho)\leq\mathcal{M}(\varepsilon,E',\rho).$$

Lemma 57 *For any class* $\mathcal{G}$ *of functions on* $\mathcal{X}$ *taking their values in* $[-M,M]^Q$ *and for any real number* η *in* $(0,M]$*:*

1. for every real number ε *satisfying* $0<\varepsilon\leq\eta/2$,

$$\textit{SN-dim}\left((\Delta\mathcal{G})^{(\eta)}\right)\leq\textit{N-dim}\left(\Delta\mathcal{G},\varepsilon\right);$$

2. for every real number ε *satisfying* $\varepsilon\geq 3\eta$ *and every* $x^n=(x_i)_{1\leq i\leq n}\in\mathcal{X}^n$,

$$\mathcal{M}\left(\varepsilon,\Delta^*\mathcal{G},d_{x^n}\right)\leq\mathcal{M}\left(2,(\Delta^*\mathcal{G})^{(\eta)},d_{x^n}\right).$$

Proof To prove the first proposition, it is enough to establish that any set strongly N-shattered by $(\Delta\mathcal{G})^{(\eta)}$ is also N-shattered with margin $\eta/2$ by $\Delta\mathcal{G}$. If $s_{\mathcal{X}^n}$, a subset of $\mathcal{X}$ of cardinality n, is strongly N-shattered by $(\Delta\mathcal{G})^{(\eta)}$, then according to Definition 34, there exists a set $I(s_{\mathcal{X}^n})$ of n couples of distinct indexes of categories and a vector v_b in $[\![-\lfloor M/\eta\rfloor+1,\lfloor M/\eta\rfloor-1]\!]^n$ such that for every vector $v_y=(y_i)\in\{-1,1\}^n$, there is a function g_y in $\mathcal{G}$ satisfying

$$\forall i\in[\![1,n]\!],\ \begin{cases}\text{if } y_i=\ \ 1, & \left(\Delta g_{y,i_1(x_i)}\right)^{(\eta)}(x_i)-b_i\geq 1\\ \text{if } y_i=-1, & \left(\Delta g_{y,i_2(x_i)}\right)^{(\eta)}(x_i)+b_i\geq 1\end{cases}.$$

Thus, we are looking for a vector $(b_i')_{1\leq i\leq n}$ such that $\left(\Delta g_{y,i_1(x_i)}\right)^{(\eta)}(x_i)-b_i\geq 1\Longrightarrow\Delta g_{y,i_1(x_i)}(x_i)-b_i'\geq\eta/2$ and $\left(\Delta g_{y,i_2(x_i)}\right)^{(\eta)}(x_i)+b_i\geq 1\Longrightarrow\Delta g_{y,i_2(x_i)}(x_i)+b_i'\geq\eta/2$. To that end, four cases must be considered.

1) $b_i\geq 0$ and $y_i=1$

$$\left(\Delta g_{y,i_1(x_i)}\right)^{(\eta)}(x_i)>0\Longrightarrow\eta\left(\Delta g_{y,i_1(x_i)}\right)^{(\eta)}(x_i)\leq\Delta g_{y,i_1(x_i)}(x_i)$$

thus

$$\left(\Delta g_{y,i_1(x_i)}\right)^{(\eta)}(x_i)-b_i\geq 1\Longrightarrow\Delta g_{y,i_1(x_i)}(x_i)-\eta(b_i+1/2)\geq\eta/2.$$

2) $b_i\geq 0$ and $y_i=-1$

$$\left(\Delta g_{y,i_2(x_i)}\right)^{(\eta)}(x_i)+b_i\geq 1\Longrightarrow\Delta g_{y,i_2(x_i)}(x_i)+\eta b_i\geq 0$$

or equivalently

$$\left(\Delta g_{y,i_2(x_i)}\right)^{(\eta)}(x_i)+b_i\geq 1\Longrightarrow \Delta g_{y,i_2(x_i)}(x_i)+\eta(b_i+1/2)\geq \eta/2.$$

3) $b_i<0$ and $y_i=1$

$$\left(\Delta g_{y,i_1(x_i)}\right)^{(\eta)}(x_i)-b_i\geq 1\Longrightarrow \Delta g_{y,i_1(x_i)}(x_i)-\eta b_i\geq 0$$

or equivalently

$$\left(\Delta g_{y,i_1(x_i)}\right)^{(\eta)}(x_i)-b_i\geq 1\Longrightarrow \Delta g_{y,i_1(x_i)}(x_i)-\eta(b_i-1/2)\geq \eta/2.$$

4) $b_i<0$ and $y_i=-1$

$$\left(\Delta g_{y,i_2(x_i)}\right)^{(\eta)}(x_i)>0\Longrightarrow \eta\left(\Delta g_{y,i_2(x_i)}\right)^{(\eta)}(x_i)\leq \Delta g_{y,i_2(x_i)}(x_i)$$

thus

$$\left(\Delta g_{y,i_2(x_i)}\right)^{(\eta)}(x_i)+b_i\geq 1\Longrightarrow \Delta g_{y,i_2(x_i)}(x_i)+\eta(b_i-1/2)\geq \eta/2.$$

To sum up, a satisfactory solution consists in setting $b'_i=\eta(b_i+1/2)$ if $b_i\geq 0$ and $b'_i=\eta(b_i-1/2)$ otherwise. By definition, the set of functions Δg_y, for v_y in $\{-1,1\}^n$, N-shatters $s_{\mathcal{X}^n}$ with margin $\eta/2$, for a set of couples of indexes and a vector of "biases" respectively equal to $I(s_{\mathcal{X}^n})$ and $v_{b'}=(b'_i)_{1\leq i\leq n}$. As a consequence, any set strongly N-shattered by $(\Delta\mathcal{G})^{(\eta)}$ is also N-shattered with margin $\eta/2$ by $\Delta\mathcal{G}$, which is precisely our claim.

To prove the second proposition, let us first notice that:

$$\forall (g,g')\in\mathcal{G}^2,\ \forall x\in\mathcal{X},\ \forall k\in[\![1,Q]\!],\ \forall\eta\in(0,M],$$

$$\left|\Delta^* g_k(x)-\Delta^* g'_k(x)\right|\geq 3\eta\Longrightarrow\left|\left(\Delta^* g_k\right)^{(\eta)}(x)-\left(\Delta^* g'_k\right)^{(\eta)}(x)\right|\geq 2.$$

Indeed, without loss of generality, we can make the hypothesis that $\Delta^* g_k(x)>\Delta^* g'_k(x)$. Then,

$$\left(\left(\Delta^* g'_k\right)^{(\eta)}(x)-1\right)\eta<\Delta^* g'_k(x)<\Delta^* g_k(x)<\left(\left(\Delta^* g_k\right)^{(\eta)}(x)+1\right)\eta.$$

Thus

$$\left(\left(\Delta^* g_k\right)^{(\eta)}(x)+1\right)\eta-\left(\left(\Delta^* g'_k\right)^{(\eta)}(x)-1\right)\eta>3\eta$$

and finally

$$\left(\Delta^* g_k\right)^{(\eta)}(x)-\left(\Delta^* g'_k\right)^{(\eta)}(x)>1,$$

from which the desired result springs directly, keeping in mind that the η-discretizations are integer numbers $\left(\left(\Delta^* g_k\right)^{(\eta)}(x)-\left(\Delta^* g'_k\right)^{(\eta)}(x)>1\Longrightarrow\left(\Delta^* g_k\right)^{(\eta)}(x)-\left(\Delta^* g'_k\right)^{(\eta)}(x)\geq 2\right)$.

Let $s_{\Delta^*\mathcal{G}}$ be a 3η-separated subset of $\Delta^*\mathcal{G}$ in the pseudo-metric d_{x^n}. It results from the definition of the pseudo-metric that:

$$\forall\ \left(\Delta^* g,\Delta^* g'\right)\in s^2_{\Delta^*\mathcal{G}},\ d_{x^n}\left(\Delta^* g,\Delta^* g'\right)\geq 3\eta\Longrightarrow$$

$$\max_{1\leq i\leq n}\left\|\Delta^* g(x_i)-\Delta^* g'(x_i)\right\|_\infty\geq 3\eta\Longrightarrow$$

$$\max_{1\leq i\leq n}\left\|\left(\Delta^* g\right)^{(\eta)}(x_i)-\left(\Delta^* g'\right)^{(\eta)}(x_i)\right\|_\infty \geq 2 \Longrightarrow$$
$$d_{x^n}\left(\left(\Delta^* g_k\right)^{(\eta)},\left(\Delta^* g_k'\right)^{(\eta)}\right)\geq 2.$$

We have thus proved the second proposition. ■

Note that a more interesting second proposition could have resulted from using a different definition of the η-discretization. Indeed, setting $\left(\Delta^{\#} g_k\right)^{(\eta)}(x)=\left\lfloor\frac{\Delta^{\#} g_k(x)}{\eta}\right\rfloor$ irrespective of the sign of $\Delta^{\#} g_k(x)$, one can easily establish that the following proposition, with a dependence between ε and η identical to the one of Alon et al. (1997), holds true: for every $\varepsilon\geq 2\eta$ and every $x^n\in\mathcal{X}^n$, $\mathcal{M}(\varepsilon,\Delta^*\mathcal{G},d_{x^n})\leq \mathcal{M}(2,(\Delta^*\mathcal{G})^{(\eta)},d_{x^n})$. The reason for our choice is to get an additional useful property, namely:

$$\forall\eta\in(0,M],\ \Delta^{\#} g_l(x)=-\Delta^{\#} g_k(x)\Longrightarrow\left(\Delta^{\#} g_l\right)^{(\eta)}(x)=-\left(\Delta^{\#} g_k\right)^{(\eta)}(x).$$

This property plays a central role in the derivation of our generalized Sauer-Shelah lemma (see for instance the proofs of Lemmas 35 and 37).

Lemma 58 *For all triplet (K_1,K_2,K_3) of positive integers such that $1\leq K_1\leq K_2$ and $K_3\geq 1$, let*

$$\Phi(K_1,K_2,K_3)=\sum_{i=0}^{K_1}\binom{K_2}{i}K_3^i.$$

The following bound is true:

$$\Phi(K_1,K_2,K_3)<\left(\frac{K_2K_3e}{K_1}\right)^{K_1},$$

where e is the base of the Neperian (or natural) logarithm.

Proof $\sum_{i=0}^{K_1}\binom{K_2}{i}K_3^i\leq K_3^{K_1}\sum_{i=0}^{K_1}\binom{K_2}{i}$. By application of Theorem 13.3. in Devroye et al. (1996), $\sum_{i=0}^{K_1}\binom{K_2}{i}$ can be bounded from above by $\left(\frac{K_2e}{K_1}\right)^{K_1}$, which concludes the proof. ■

Appendix B. Proof of Theorem 22

The proof is divided into several steps, following the structure proposed by Dudley (1978) and Pollard (1984, chap. II), structure also described, with variants, in Devroye et al. (1996, Chap. 12), Vapnik (1998, Chap. 4), Anthony and Bartlett (1999), and Schölkopf and Smola (2002, Chap. 5).

B.1 First Symmetrization

The first step is a symmetrization. The idea is to replace the true risk by an estimate computed on a m-sample $\tilde{D}_m$ independent of D_m. This symmetrization corresponds to Lemma 50, and thus gives:

$$\mathbb{P}_{D_m}\left(\sup_{g\in\mathcal{G}}\left(R(g)-R_{\gamma,D_m}(g)\right)>\varepsilon\right)\leq 2\mathbb{P}_{D_{2m}}\left(\sup_{g\in\mathcal{G}}\left(R_{\tilde{D}_m}(g)-R_{\gamma,D_m}(g)\right)\geq\varepsilon-\frac{1}{m}\right). \tag{29}$$

Note that at this point, the standard pathway consists in applying a second symmetrization to get rid of the "ghost sample" $\tilde{D}_m$ (see for example Pollard, 1984; Devroye et al., 1996). For the sake of simplicity, we do not develop this possibility here. Instead, we apply another symmetrization, to keep one single type of empirical measure of accuracy in the bound.

B.2 Second Symmetrization

The second symmetrization, resulting from Lemma 51, corresponds to the following upper bound :

$$\mathbb{P}_{D_{2m}}\left(\sup_{g\in\mathcal{G}}\left(R_{\tilde{D}_m}(g)-R_{\gamma,D_m}(g)\right)\geq\varepsilon-\frac{1}{m}\right)\leq$$

$$\mathbb{P}_{D_{2m}}\left(\max_{\overline{g}\in\overline{\mathcal{G}}(\gamma,D_{2m})}\left(R_{\gamma/2,\tilde{D}_m}(\overline{g})-R_{\gamma/2,D_m}(\overline{g})\right)\geq\varepsilon-\frac{1}{m}\right). \tag{30}$$

It is useful for two reasons. First, it completes, in some sense, the first one, by replacing the two different empirical measures of accuracy appearing in the right-hand side of (29) with two independent copies of the same random variable. Second, it makes it possible to substitute, in the forthcoming computations, the set $\mathcal{G}$ of possibly infinite cardinality with a subset of it of cardinality no more than $\mathcal{N}^{(p)}(\gamma/2,\Delta_\gamma^{\#}\mathcal{G},2m)$. This is exploited in the next step of the proof, to apply a standard union bound.

B.3 Maximal Inequality

To bound from above the right-hand side of (30) irrespective of P, and thus derive a distribution-free result, we introduce an auxiliary step of randomization. Let S_{2m} be a random variable described by the uniform distribution on $\mathfrak{T}_{2m}$. By application of Lemma 52,

$$\mathbb{P}_{D_{2m}}\left(\max_{\overline{g}\in\overline{\mathcal{G}}(\gamma,D_{2m})}\left(R_{\gamma/2,\tilde{D}_m}(\overline{g})-R_{\gamma/2,D_m}(\overline{g})\right)\geq\varepsilon-\frac{1}{m}\right)\leq$$

$$\max_{z^{2m}\in\mathcal{Z}^{2m}}\sum_{\overline{g}\in\overline{\mathcal{G}}(\gamma,x^{2m})}\mathbb{P}_{S_{2m}}\left(R_{\gamma/2,S_{2m}(z^m)}(\overline{g})-R_{\gamma/2,S_{2m}(z^m)}(\overline{g})\geq\varepsilon-\frac{1}{m}\right). \tag{31}$$

B.4 Exponential Bound

Using Lemma 54, the probabilities in the right-hand side of (31) are bounded uniformly by $\exp\left(-\frac{m}{2}\left(\varepsilon-\frac{1}{m}\right)^2\right)$. As a consequence,

$$\max_{z^{2m}\in\mathcal{Z}^{2m}}\sum_{\overline{g}\in\overline{\mathcal{G}}(\gamma,x^{2m})}\mathbb{P}_{S_{2m}}\left(R_{\gamma/2,S_{2m}(z^m)}(\overline{g})-R_{\gamma/2,S_{2m}(z^m)}(\overline{g})\geq\varepsilon-\frac{1}{m}\right)\leq$$

$$\max_{x^{2m}\in\mathcal{X}^{2m}}\left|\overline{\mathcal{G}}\left(\gamma,x^{2m}\right)\right|\exp\left(-\frac{m}{2}\left(\varepsilon-\frac{1}{m}\right)^2\right).$$

According to Definitions 15 and 19, $\max_{x^{2m}\in\mathcal{X}^{2m}}\left|\overline{\mathcal{G}}\left(\gamma,x^{2m}\right)\right|=\mathcal{N}^{(p)}\left(\gamma/2,\Delta_\gamma^{\#}\mathcal{G},2m\right)$, and thus

$$\max_{z^{2m}\in\mathcal{Z}^{2m}}\sum_{\overline{g}\in\overline{\mathcal{G}}(\gamma,x^{2m})}\mathbb{P}_{S_{2m}}\left(R_{\gamma/2,S_{2m}(z^m)}(\overline{g})-R_{\gamma/2,S_{2m}(z^m)}(\overline{g})\geq\varepsilon-\frac{1}{m}\right)\leq$$

$$\mathcal{N}^{(p)}\left(\gamma/2,\Delta_\gamma^{\#}\mathcal{G},2m\right)\exp\left(-\frac{m}{2}\left(\varepsilon-\frac{1}{m}\right)^2\right). \tag{32}$$

The combination of (29), (30), (31), and (32) provides us with the following bound:

$$\mathbb{P}_{D_m}\left(\sup_{g\in\mathcal{G}}\left(R(g)-R_{\gamma,D_m}(g)\right)>\varepsilon\right)\leq 2\mathcal{N}^{(p)}\left(\gamma/2,\Delta_\gamma^{\#}\mathcal{G},2m\right)\exp\left(-\frac{m}{2}\left(\varepsilon-\frac{1}{m}\right)^2\right). \tag{33}$$

Setting the right-hand side of (33) to δ and solving for ε finally gives:

$$R(g)\leq R_{\gamma,m}(g)+\sqrt{\frac{2}{m}\left(\ln\left(2\mathcal{N}^{(p)}\left(\gamma/2,\Delta_\gamma^{\#}\mathcal{G},2m\right)\right)-\ln(\delta)\right)}+\frac{1}{m}.$$

B.5 Uniform Bound Over the Margin Parameter γ

This last bound holds for a value of γ specified in advance. To make the bound useful, we would like to be able to select γ after observation of the trained machine on the training set. This can be done thanks to Lemma 55, extending Proposition 8 in Bartlett (1998), which allows us to produce a result that stands uniformly for all values of the margin parameter γ in the interval $(0,\Gamma]$. To apply Lemma 55 to the case of interest, let us define the function Θ as follows:

$$\Theta(t,u)=\sqrt{\frac{2}{m}\left(\ln\left(2\mathcal{N}^{(p)}\left(t,\Delta_\gamma^{\#}\mathcal{G},2m\right)\right)-\ln(u)\right)}.$$

One can readily verify that the measure $\mathbb{P}_{D_m}$ and the set of events $E(\alpha_1,\alpha_2,\delta)$ given by:

$$\sup_{g\in\mathcal{G}}\left(R(g)-R_{\alpha_2,D_m}(g)\right)\geq\Theta\left(\frac{\alpha_1}{2},\delta\right)+\frac{1}{m}$$

satisfy the hypotheses of Lemma 55. Its application gives, for all choice of the couple (a,δ) in $(0,1)\times(0,1]$,

$$\mathbb{P}_{D_m}\left[\bigcup_{\alpha\in(0,K]}\left(\sup_{g\in\mathcal{G}}\left(R(g)-R_{\alpha,D_m}(g)\right)\geq\Theta\left(\frac{\alpha a}{2},\frac{\delta\alpha(1-a)}{K}\right)+\frac{1}{m}\right)\right]\leq\delta.$$

Setting $\alpha=\gamma$, $K=\Gamma$ and choosing $a=1/2$ yields to:

$$\mathbb{P}_{D_m}\left[\bigcup_{\gamma\in(0,\Gamma]}\left(\sup_{g\in\mathcal{G}}\left(R(g)-R_{\gamma,D_m}(g)\right)\geq\Theta\left(\frac{\gamma}{4},\frac{\gamma\delta}{2\Gamma}\right)+\frac{1}{m}\right)\right]\leq\delta$$

and finally, by definition of Θ,

$$\mathbb{P}_{D_m}\left[\bigcup_{\gamma\in(0,\Gamma]}\left(\sup_{g\in\mathcal{G}}\left(R(g)-R_{\gamma,D_m}(g)\right)\geq\right.\right.$$

$$\left.\left.\sqrt{\frac{2}{m}\left(\ln\left(2\mathcal{N}^{(p)}\left(\gamma/4,\Delta_\gamma^{\#}\mathcal{G},2m\right)\right)-\ln\left(\frac{\gamma\delta}{2\Gamma}\right)\right)}+\frac{1}{m}\right)\right]\leq\delta,$$

which concludes the proof of Theorem 22.

References

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

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.

A. Ambroladze, E. Parrado-Hernandez, and J. Shawe-Taylor. Tighter PAC-Bayes bounds. In *Advances in Neural Information Processing Systems 19*, 2007. (to appear).

M. Anthony and P.L. Bartlett. *Neural Network Learning: Theoretical Foundations*. Cambridge University Press, Cambridge, 1999.

N. Aronszajn. Theory of reproducing kernels. *Transactions of the American Mathematical Society*, 68(3):337–404, 1950.

P.L. Bartlett. The sample complexity of pattern classification with neural networks: The size of the weights is more important than the size of the network. *IEEE Transactions on Information Theory*, 44(2):525–536, 1998.

P.L. Bartlett and J. Shawe-Taylor. Generalization performance of support vector machines and other pattern classifiers. In B. Schölkopf, C.J.C. Burges, and A. Smola, editors, *Advances in Kernel Methods - Support Vector Learning*, chapter 4, pages 43–54. The MIT Press, Cambridge, MA, 1999.

P.L. Bartlett, P.M. Long, and R.C. Williamson. Fat-shattering and the learnability of real-valued functions. *Journal of Computer and System Sciences*, 52(3):434–452, 1996.

S. Ben-David, N. Cesa-Bianchi, D. Haussler, and P.M. Long. Characterizations of learnability for classes of $\{0,\ldots,n\}$-valued functions. *Journal of Computer and System Sciences*, 50(1):74–86, 1995.

A. Berlinet and C. Thomas-Agnan. *Reproducing Kernel Hilbert Spaces in Probability and Statistics*. Kluwer Academic Publishers, Boston, 2004.

G. Blanchard, O. Bousquet, and P. Massart. Statistical performance of support vector machines. *The Annals of Statistics*, 2007. (to appear).

B. Boser, I. Guyon, and V.N. Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, pages 144–152, 1992.

S. Boucheron, O. Bousquet, and G. Lugosi. Theory of classification: A survey of some recent advances. *ESAIM: Probability and Statistics*, 9:323–375, 2005.

O. Bousquet. *Concentration Inequalities and Empirical Processes Theory Applied to the Analysis of Learning Algorithms*. PhD thesis, Ecole Polytechnique, 2002.

B. Carl and I. Stephani. *Entropy, Compactness and the Approximation of Operators*. Cambridge University Press, Cambridge, 1990.

O. Chapelle, V.N. Vapnik, O. Bousquet, and S. Mukherjee. Choosing multiple parameters for support vector machines. *Machine Learning*, 46(1):131–159, 2002.

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

K. Crammer and Y. Singer. On the algorithmic implementation of multiclass kernel-based vector machines. *Journal of Machine Learning Research*, 2:265–292, 2001.

K. Crammer and Y. Singer. On the learnability and design of output codes for multiclass problems. *Machine Learning*, 47(2):201–233, 2002.

Y. Darcy and Y. Guermeur. Radius-margin bound on the leave-one-out error of multi-class SVMs. Technical Report RR-5780, INRIA, 2005.

L. Devroye, L. Györfi, and G. Lugosi. *A Probabilistic Theory of Pattern Recognition*. Springer-Verlag, New York, 1996.

R.M. Dudley. Central limit theorems for empirical measures. *The Annals of Probability*, 6(6): 899–929, 1978.

R.M. Dudley. A course on empirical processes. In P.L. Hennequin, editor, *Ecole d'Eté de Probabilités de Saint-Flour XII - 1982*, volume 1097 of *Lecture Notes in Mathematics*, pages 1–142. Springer-Verlag, 1984.

R.M. Dudley. Universal Donsker classes and metric entropy. *The Annals of Probability*, 15(4): 1306–1326, 1987.

A. Elisseeff, Y. Guermeur, and H. Paugam-Moisy. Margin error and generalization capabilities of multi-class discriminant models. Technical Report NC-TR-99-051-R, NeuroCOLT2, 1999. (revised in 2001).

J. Fürnkranz. Round robin classification. *Journal of Machine Learning Research*, 2:721–747, 2002.

Y. Guermeur. Combining discriminant models with new multi-class SVMs. *Pattern Analysis and Applications*, 5(2):168–179, 2002.

Y. Guermeur, A. Elisseeff, and H. Paugam-Moisy. Estimating the sample complexity of a multi-class discriminant model. In *International Conference on Artificial Neural Networks*, pages 310–315. IEE, 1999.

Y. Guermeur, M. Maumy, and F. Sur. Model selection for multi-class SVMs. In *International Symposium on Applied Stochastic Models and Data Analysis*, pages 507–517, 2005.

L. Gurvits. A note on a scale-sensitive dimension of linear bounded functionals in Banach spaces. *Theoretical Computer Science*, 261(1):81–90, 2001.

T. Hastie, S. Rosset, R. Tibshirani, and J. Zhu. The entire regularization path for the support vector machine. *Journal of Machine Learning Research*, 5:1391–1415, 2004.

D. Haussler and P.M. Long. A generalization of Sauer's lemma. *Journal of Combinatorial Theory, Series A*, 71(2):219–240, 1995.

W. Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58:13–30, 1963.

K. Jogdeo and S.M. Samuels. Monotone convergence of binomial probabilities and a generalization of Ramanujan's equation. *The Annals of Mathematical Statistics*, 39(4):1191–1195, 1968.

M.J. Kearns and R.E. Schapire. Efficient distribution-free learning of probabilistic concepts. *Journal of Computer and System Sciences*, 48(3):464–497, 1994.

A.N. Kolmogorov and V.M. Tihomirov. ε-entropy and ε-capacity of sets in functional spaces. *American Mathematical Society Translations, series 2*, 17:277–364, 1961.

R.S. Kroon. Support vector machines, generalization bounds, and transduction. Master's thesis, University of Stellenbosch, South Africa, December 2003. `http://www.cs.sun.ac.za/~skroon/personal/pubs/kroon2003support.ps`.

M. Ledoux. On Talagrand's deviation inequalities for product measures. *ESAIM: Probability and Statistics*, 1:63–87, 1996.

Y. Lee and Z. Cui. Characterizing the solution path of multicategory support vector machines. *Statistica Sinica*, 16:391–409, 2006.

Y. Lee, Y. Lin, and G. Wahba. Multicategory support vector machines: Theory and application to the classification of microarray data and satellite radiance data. *Journal of the American Statistical Association*, 99(465):67–81, 2004.

G. Lugosi. Concentration-of-measure inequalities. Lecture notes, Summer School on Machine Learning at the Australian National University, Canberra, 2004.

P. Massart. Some applications of concentration inequalities to statistics. *Annales de la Faculté des Sciences de Toulouse*, 9(2):245–303, 2000.

E. Monfrini and Y. Guermeur. A quadratic loss multi-class SVM. Technical report, LORIA, 2007. (to appear).

B.K. Natarajan. On learning sets and functions. *Machine Learning*, 4(1):67–97, 1989.

J.C. Platt, N. Cristianini, and J. Shawe-Taylor. Large margin DAGs for multiclass classification. In *Advances in Neural Information Processing Systems 12*, pages 547–553, 2000.

D. Pollard. *Convergence of Stochastic Processes*. Springer-Verlag, New York, 1984.

R. Rifkin and A. Klautau. In defense of one-vs-all classification. *Journal of Machine Learning Research*, 5:101–141, 2004.

W. Rudin. *Real and Complex Analysis*. McGraw-Hill, New York, third edition, 1987.

N. Sauer. On the density of families of sets. *Journal of Combinatorial Theory (A)*, 13:145–147, 1972.

B. Schölkopf and A.J. Smola. *Learning with Kernels - Support Vector Machines, Regularization, Optimization, and Beyond*. The MIT Press, Cambridge, MA, 2002.

J. Shawe-Taylor, P.L. Bartlett, R.C. Williamson, and M. Anthony. Structural risk minimization over data-dependent hierarchies. *IEEE Transactions on Information Theory*, 44(5):1926–1940, 1998.

S. Shelah. A combinatorial problem: Stability and order for models and theories in infinitary languages. *Pacific Journal of Mathematics*, 41(1):247–261, 1972.

I. Steinwart and C. Scovel. Fast rates for support vector machines. In *Proceedings of the eighteenth annual Conference on Learning Theory*, pages 279–294, 2005.

M. Stone. Asymptotics for and against cross-validation. *Biometrika*, 64(1):29–35, 1977.

M. Talagrand. Concentration of measure and isoperimetric inequalities in product spaces. *Publications mathématiques de l'I.H.E.S.*, 81:73–205, 1995.

M. Talagrand. A new look at independence. *The Annals of Probability*, 24(1):1–34, 1996.

A. Tewari and P.L. Bartlett. On the consistency of multiclass classification methods. *Journal of Machine Learning Research*, 8:1007–1025, 2007.

L.G. Valiant. A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142, 1984.

A.W. van der Vaart and J.A. Wellner. *Weak Convergence and Empirical Processes - With Applications to Statistics*. Springer Series in Statistics. Springer-Verlag, New York, 1996.

V.N. Vapnik. *Estimation of Dependences Based on Empirical Data*. Springer-Verlag, New York, 1982.

V.N. Vapnik. *Statistical Learning Theory*. John Wiley & Sons, Inc., New York, 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.

G. Wahba. Support vector machines, reproducing kernel Hilbert spaces, and randomized GACV. In B. Schölkopf, C.J.C. Burges, and A.J. Smola, editors, *Advances in Kernel Methods - Support Vector Learning*, chapter 6, pages 69–88. The MIT Press, Cambridge, MA, 1999.

L. Wang, P. Xue, and K.L. Chan. Generalized radius-margin bounds for model selection in multiclass SVMs. Technical report, School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore, 639798, 2005.

J. Weston and C. Watkins. Multi-class support vector machines. Technical Report CSD-TR-98-04, Royal Holloway, University of London, Department of Computer Science, 1998.

R.C. Williamson, A.J. Smola, and B. Schölkopf. Entropy numbers of linear function classes. In *Proceedings of the Thirteenth Annual Workshop on Computational Learning Theory*, pages 309–319, 2000.

T. Zhang. Statistical analysis of some multi-category large margin classification methods. *Journal of Machine Learning Research*, 5:1225–1251, 2004.

Journal of Machine Learning Research 8 (2007) 2595-2628 Submitted 7/06; Revised 5/07; Published 11/07

Learning in Environments with Unknown Dynamics: Towards more Robust Concept Learners

Marlon Núñez MNUNEZ@LCC.UMA.ES
Raúl Fidalgo RFM@LCC.UMA.ES
Rafael Morales MORALES@LCC.UMA.ES
Departamento de Lenguajes y Ciencias de la Computación
ETSI Informática. Campus Teatinos. Universidad de Málaga
29071 Málaga, Spain

Editor: Claude Sammut

Abstract

In the process of concept learning, target concepts may have portions with short-term changes, other portions may support long-term changes, and yet others may not change at all. For this reason several local windows need to be handled. We suggest facing this problem, which naturally exists in the field of concept learning, by allocating windows which can adapt their size to portions of the target concept. We propose an incremental decision tree that is updated with incoming examples. Each leaf of the decision tree holds a time window and a local performance measure as the main parameter to be controlled. When the performance of a leaf decreases, the size of its local window is reduced. This learning algorithm, called OnlineTree2, automatically adjusts its internal parameters in order to face the current dynamics of the data stream. Results show that it is comparable to other batch algorithms when facing problems with no concept change, and it is better than evaluated methods in its ability to deal with concept drift when dealing with problems in which: concept change occurs at different speeds, noise may be present and, examples may arrive from different areas of the problem domain (virtual drift).

Keywords: incremental algorithms, online learning, concept drift, decision trees, robust learners

1. Introduction

Target concepts to be learnt, which change over time, are often handled by time windows of fixed or adaptive size on the training data (Widmer and Kubat, 1996; Klinkenberg and Renz, 1998; Klinkenberg, 2004) or by weighing data or parts of the hypothesis (Taylor et al., 1997; Krizakova and Kubat, 1992; Klinkenberg, 2004).

In case of fixed windows, their size is a compromise between fast adaptability in phases with short-term changes (small window) or good generalization in phases without concept change (large window). The basic idea of adaptive window management is to adjust the window size to the current rate of concept drift. In Section 3, we will illustrate how changes may affect only a portion of the whole concept. For example, in problems related to user interests, Widyantoro et al. (1999) found that for a user there will be some document categories of interest that change from day to day depending on particular work interests, while there are other categories of interest which may not change greatly because they fit in with a personal and/or professional profile.

On the other hand, when dealing with problems involving unknown dynamics (i.e., problems in which any circumstances may occur, such as concept drift, changes in noise level, distribution of

examples varies, etc), it is desirable that algorithms are able to adapt their parameters to improve learning (Potts and Sammut, 2005; Klinkenberg and Joachims, 2000). The adaptive capacity of the internal parameters makes learning algorithms simpler to use when dealing with real problems, where you can almost never guarantee that the dynamics of the problem will not change over time.

In order to tackle the aforementioned problems, the presented method, called OnlineTree2, is able to deal with problems in which the target concept, or portions of it, do not change or change at different speeds over time. This method incrementally learns a decision tree in which each leaf maintains a local window, used to forget examples when a concept change has been detected. Depending on the dynamics of the problem, OnlineTree2 adapts the decision tree structure and adjusts the local parameters with the aim of achieving more efficient learning in terms of: improving local performance, optimizing the number of stored examples and reducing processing time. The method may also face problems with a changing level of noise and/or virtual drift. Virtual drift occurs when the distribution of the observed examples changes over time but the concept remains the same.

This paper is organized as follows: Section 2 summarizes the related work; Section 3 describes what concept drift is and its different speeds; Section 4 explains the proposed algorithm; Section 5 presents experimentation results of OnlineTree2 and a set of algorithms on problems that may change over time and involves different conditions; Section 6 describes future work; and, Section 7 presents conclusions.

2. Related Work

Maloof and Michalski (2000) suggested three possibilities for managing the memory model when dealing with past training examples: *No instance memory*, in which the incremental learner retains no examples in memory (VFDT by Domingos and Hulten, 2000, VFDTc by Gama et al., 2003, CVFDT by Hulten et al., 2001, neural networks, Naïve Bayes and support vector machines are algorithms that follow this approach); *Full instance memory*, in which the method retains all past training examples (the incremental algorithm ITI by Utgoff et al., 1997, and most of batch algorithms, like C4.5 by Quinlan, 1993, and IBk by Aha et al., 1991, follow this approach); *Partial instance memory*, in which the incremental learner retains some of the past training examples that are within a window, mainly orientated to deal with concept drift. The window size may be fixed or adaptive. AQ11-PM (Maloof and Michalski, 2004) uses a global window with fixed size, FLORA (Widmer and Kubat, 1996), SVM-TC (Klinkenberg and Joachims, 2000) and DDM (Gama et al., 2004a) use a global window with adaptive size. OnlineTree2 uses several local windows with adaptive sizes.

Another interesting aspect is the management of the internal parameters for dealing with real applications. When facing problems in which concepts change over time, almost all learning systems (e.g., FRANN by Kubat and Widmer, 1995, CVFDT, FLORA, AQ11-PM) require a prior establishment of a series of parameters in order to treat a determined dynamic of the problem (e.g., to a noise level, speed of change—explained in next section—and temporal distribution of the examples). Recent research in the field of information retrieval (Klinkenberg and Joachims, 2000; Klinkenberg, 2004) (i.e., adaptive classification of documents) and machine learning (Núñez et al., 2005) have found that these parameterizations may be inadequate, particularly in those problems where the concepts obey certain subjectivity (user interests). Klinkenberg and Joachims (2000) maintain that the dynamics of problems themselves may change over time making previous parameterization of no use later for the same or for other new problems. They proposed a non-parameterized approach

for handling a global time window. OnlineTree2 automatically adjusts its internal parameters to face the changing dynamics of the problem.

A previous version of the proposed algorithm, called OnlineTree (Núñez et al., 2005), was able to detect concept drift from small data sets (less than 200 examples) and manage noise level in data, but it does not work when the data set has numerical features, a data stream is present, change in noise levels appears and/or the problem contains virtual drift. OnlineTree2 corrects these deficiencies, being able to deal with data streams containing unknown dynamics (that is, possible concept drifts, changes in noise level, virtual drift, continuous or symbolic features and different distribution of examples). Experimentation will show that OnlineTree2 achieves low error rates, improves the number of stored examples and has a reduced processing time.

Before explaining the algorithm in detail, let us first illustrate the importance of detecting the different speeds of change of the different concepts for an incremental learner.

3. Concept Drift Speeds

Concepts may change gradually or abruptly. Previous studies in the field of information retrieval (Widyantoro et al., 1999) and data mining (Fan, 2004) have found that target concepts may change with several speeds.

A problem with simultaneous target concepts that change at different speeds is illustrated in Figure 1. The concepts + and – change over time, from time t_1 to time t_5. The concepts of times t_1, t_2 and t_3 drift gradually. Note that they maintain a + subconcept (a portion of the whole concept) at the top of the two dimension domain and a – subconcept at the bottom, while the subconcept in the middle changes gradually (or slowly; that is, long-term changes). At t_4 there is an abrupt change (or fast; that is, short-term change), given that no subconcept is kept.

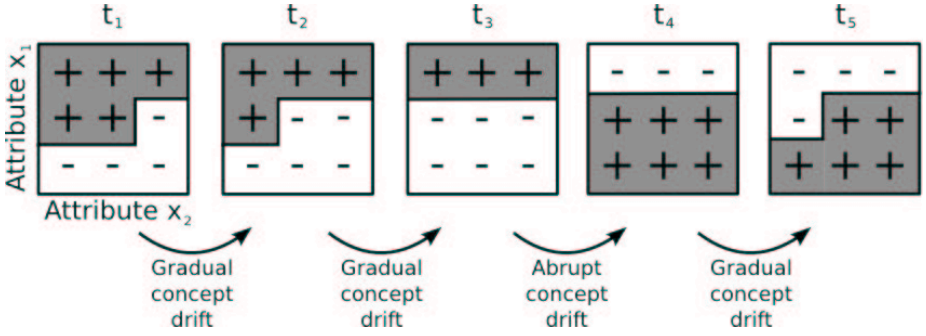


Figure 1: Illustration of gradual and abrupt concept drift for concepts described by two attributes (x_1 and x_2)

In order to learn concepts that change over time, current learners usually use a global window of examples. Looking at Figure 1, if a learner has a small global window, let us say 1, it will react to changes quickly, but it may forget those instances that have not been recently observed. If the global window is large, stationary subconcepts may be learnt well, but inconsistent instances will be accumulated for those subconcepts where changes have occurred. It is therefore of great importance to discover an adequate window size for each subconcept.

Some authors (Maloof and Michalski, 2004; Widmer and Kubat, 1996; Klinkenberg and Joachims, 2000; Gama et al., 2004a) propose a global adaptive window to face this problem, but these algorithms do not function properly when various speeds of change are present in the concept. Widyantoro et al. (1999) uses two windows simultaneously, one for short-term changing concepts and another for long-term changing concepts.

This problem, which naturally exists in the field of concept learning, could be overcome by using several adaptive windows, one for each portion of the target concept. These windows are able to adapt themselves to a rate of change. Since our proposal is an incremental decision tree updated with incoming examples, each of its leaves monitors its related subconcept by holding a local time window. As will be explained in Section 4, the size of these local windows shrinks or grows depending on changes detected in subconcepts. This adapts learning to the dynamics of the labelled data stream. The following section presents and describes the algorithm in detail.

4. Description of the OnlineTree2 Algorithm

OnlineTree2 is an algorithm for incremental induction of binary decision trees, which also supports adaptability to gradual and abrupt concept drift, virtual drift, robustness to noise in data, and the handling of symbolic and numeric attributes. This method develops a partial memory management; that is to say, it selectively forgets examples and stores the remaining examples in a subset according to local windows in the leaves of the tree. Depending on the dynamics of the problem OnlineTree2 expands or prunes subtrees and adjusts its internal parameters to improve the local performance measure. This method is intended to be used under unknown dynamics, and thus it is able to automatically adapt its parameters for each problem.

We consider data streams as sequences of examples labelled with a time stamp. An example is described by attribute-value pairs and a class label. This means that the method is capable of dealing with streams of examples in continuous (real) time. A consecutive index can also be used as time label.

The pseudo-algorithm is presented in Table 1. In order to describe it, let us consider that we have built a tree and that a new example from the data stream must be processed. Actions that are carried out may be summarized in three stages: downward revision of statistics, treatment of a leaf or a non coherent node, and upward updating of statistics.

Stage 1. Downward revision of statistics: OnlineTree2 moves the example down the tree. The nodes are checked for coherence against its related subconcept when they are visited. A node is coherent whenever its split contributes with the induction of the underlying concept. This stage finishes when the algorithm reaches a non coherent node or a leaf.

Stage 2. Treatment of a leaf or a non coherent node:

(a) Non coherent node treatment: Once OnlineTree2 stops at a non coherent node, the algorithm drops the example down to its corresponding leaf. Then, from the set of leaves below the non coherent node, OnlineTree2 orders those leaves whose performance decreases to forget, reducing its windows. After that, the node is converted into a leaf with the remaining examples in the set of leaves. Finally, and in order to adapt the pruned node to its subconcept, an attempt to draw a new split is performed.

Input: *tree*, *example*
Output: *tree*

OnlineTree2 Algorithm
IF *node* is not a leaf AND its split is useful THEN
Move *example* to the next node (recursive call) } Stage 1
ELSE
IF *node* is not a leaf THEN
Drop *example* down the *node* until reach a leaf and store in it
Adjust local windows in degraded leaves below *node*
Prune *node* (a)
Update statistics in the pruned *node*
Try to expand the pruned *node*
ELSE –*node* is a *leaf*–
Store *example* in *leaf*
Update *leaf* statistics
IF *leaf* improves THEN } Stage 2
Try to expand *leaf*
IF *leaf* was not expanded THEN (b)
Adjust local window in improved *leaf*
ENDIF
ELSE –*leaf* degrades–
Adjust local window in degraded *leaf*
ENDIF
ENDIF
RETURN *node* OR *leaf*
ENDIF
Update *node* statistics } Stage 3
RETURN *node*

Table 1: OnlineTree2 algorithm

(b) Leaf treatment: OnlineTree2 stores the example in the leaf and updates its statistics. If the leaf improves its performance measure, the algorithm tries to create a new decision node in order to better adapt to the subconcept. If it is not successful (i.e., no more improvements may be made in this leaf with its examples) the algorithm checks the leaf for stability, discarding old local examples in that case. If the leaf is not improving its performance measure, then an attempt to reduce its window is performed.

Stage 3. Upward updating of statistics: Once stage two has finished, the algorithm starts a bottom-up process, updating the statistics of each visited node.

In following sections, these three stages are explained in detail. For sake of reproducibility, Appendix A contains the OnlineTree2 pseudocode, as well as details about variables and functions used in it.

4.1 Stage 1: Downward Revision of Statistics

This stage describes how an example is moved down the tree, checking the coherence of nodes it finds in its path. The stage finish when a non coherent node is found or a leaf is reached.

We say a node is coherent when it brings something useful when inducing the subjacent subconcept of that node. To do this, we employ techniques that are widely used for pre-pruning in the induction of decision trees (Esposito et al., 1997). Specifically, in order to know if a node is coherent with a concept we use a χ^2 hypothesis test (with a significance level of 0.05) between the examples distribution of the node under analysis and the sum of the class distributions of its descendant nodes (Quinlan, 1986). Therefore, at each node the class distribution of the examples that are in the leaves below that node is updated. The objective is to confirm if each visited node brings something statistically significant to the concept.

If the node is coherent to the current concept, the example is directed along the appropriate branch depending on the value of the example for the splitting attribute in the node. When OnlineTree2 finds a non-coherent node or arrives at a leaf, this stage finishes and the next one begins.

4.2 Stage 2: Treatment of a Leaf or a Non Coherent Node

This section explains the actions that are carried out to modify, if necessary, a leaf or a non- coherent node. Following, we describe some preliminary concepts, as well as some local parameters to be adjusted; finally, the contents of the phases will be presented in details.

4.2.1 PRELIMINARY CONCEPTS

Before describing the content of the stages, which make up this section, we need to introduce some important concepts which will be used.

Performance

As said before, each leaf of the tree stores a quality measure. This is referred to as *performance*, and it is used by OnlineTree2 to make decisions about expansion and forgetting examples. *Performance* is calculated via the instantaneous accuracy of the leaf, that is, the ratio between examples well-classified and total examples in the leaf.

This instantaneous measurement varies greatly under certain conditions, mainly when dealing with noise and after the forgetting of several examples. Due to this high variability, instantaneous accuracy alone is inappropriate to make decisions.

For this reason, we had used a smoothing formula in order to make robust decisions based on instantaneous accuracy. Exponential smoothing is commonly used by control systems. For example, it is implemented in the nodes of a network to measure the congestion in the node lines, which is an important factor to take into account to make difficult decisions (e.g., to discard messages or to retransmit messages) in high variability of traffic conditions. Exponential smoothing needs a parameter to control the level of smoothness (α), that is, the importance of past history. The α parameter has been widely studied by Nagle (1987) and other authors (Postel, 1981; Paxson and Allman, 2000) in this environment, and a fix value of $\frac{7}{8}$ is recommended for any dynamics and problem condition.

By using exponential smoothing in a node, a weight is assigned to its instantaneous accuracy at present time, while past history becomes less important (in an exponential way). We have decided

to fix the same value for used by Nagle as default ($\alpha = \frac{7}{8}$), which has also been demonstrated in our experimentation to be valid for several problem conditions. This allows having a robust measure in order to make decisions and to quickly respond to changes in subconcepts.

So, the performance formula used by OnlineTree2 in each node is:

$$perf(t) = \frac{7}{8} perf(t-1) + \frac{1}{8} ia$$

where: *perf(t)* and *perf(t-1)* are the current and previous performance measure of the node, respectively; and, *ia* is the instantaneous accuracy of the node.

A change in the tendency of *performance* determines the *state* of the leaf.

States

Using the above mentioned performance measure, OnlineTree2 employs a state diagram in each leaf to find out if the leaf is in one of the following states:

- *Degradation State*: A leaf passes into this state when the performance worsens. This suggests that a change in subconcept has occurred and the leaf must react accordingly. A window of previous examples is generated, which is called local window in degradation state (explained in Section 4.2.2). Older examples outside this window are forgotten.

- *Improvement State*: a leaf passes into this state when its performance improves. This means that its subconcept is being learned adequately. In this state a local window is generated, referred to as local window in improvement state (explained in Section 4.2.2), which has been designed in such a way as to produce the forgetting of examples once the concept has been adequately learned, discarding old examples for new ones.

4.2.2 LOCAL PARAMETERS TO BE ADJUSTED

The OnlineTree2 algorithm adjusts three local parameters in each leaf with the aim of achieving more efficient learning in terms of: improving local performance, optimising the number of stored examples and reducing processing time. These local parameters are: *local window in degradation state*, *local window in improvement state* and *local majority/expansion factor*.

Local Window Size in Degradation State

When a leaf is in degradation state, it needs to adjust its window of examples to deal with a possible concept change and so improve its performance. This section explains in detail the mechanism to carry out the management of the local window size when the leaf is in this state.

Figure 2 shows how the performance of a leaf is updated with each example that arrives in it. The leaf remains in *Improvement State*, accumulating examples, while the performance grows. When the leaf performance starts to decline, the leaf enters in *Degradation State* noting the time of that example as *anomaly time* (the moment at which a concept drift is suspected to occur) and making a local window of examples to forget with the examples that are before that time (which we called *delayed window*). While the performance declines, the degradation persists (it is noted as *anomaly persistence*, that is, the number of examples arrived at the leaf after *anomaly time*) and a

drop in performance can be calculated as the difference between the performance at *anomaly time* and the performance at current time (*drop of performance*).

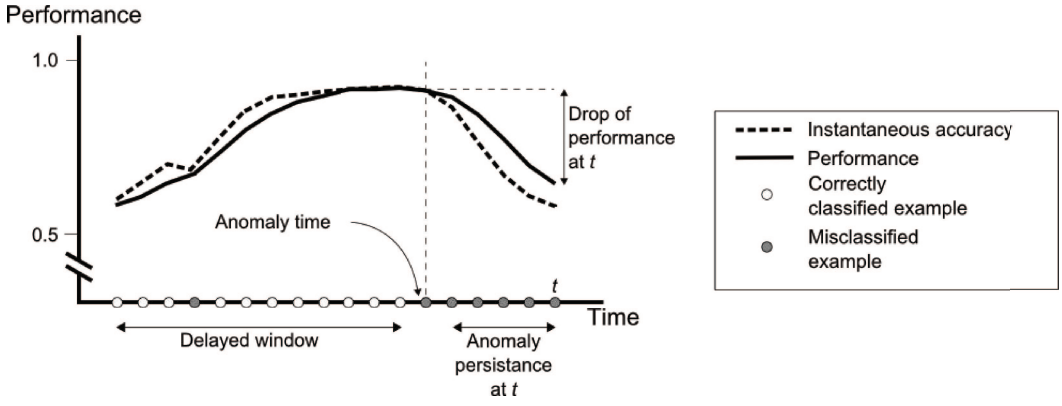


Figure 2: Illustration of the detection of a hypothesis anomaly and the delayed window at a leaf

Deterioration in the performance of a leaf reduces the size of the delayed window by a fraction. If the deterioration persists, concept drift is more probable, and a greater fraction of window needs to be discarded. In order to calculate the fraction of the delayed window to be forgotten we use the following equation:

$$wf = pers \cdot dp$$

where: wf is the window fraction of the delayed window to be forgotten; $pers$ is the anomaly persistence; and, dp is the drop of performance from anomaly time.

Thus, the new size of the delayed window will be:

$$df = \begin{cases} dw(1-wf) & \text{if } wf < 1 \\ 0 & \text{otherwise} \end{cases}$$

where: dw is the size of the delayed window; and, wf is the window fraction to be forgotten.

A reduction in size of the delayed window provokes the forgetting of older examples occurred before its anomaly time. Examples that are after the anomaly time are presumed to belong to the current concept and for that reason are maintained.

A special characteristic of the algorithm is that it treats the time in which the examples arrive in a continuous manner, that is to say, the examples may arrive in an asynchronous manner (i.e., at any time). This affects forgetting. Suppose that a burst of examples reached the leaf some time ago and few examples of a new concept have reached the leaf recently. If a window fraction of 10% must be discarded, our algorithm we would probably be left with only the most recent examples, removing the older examples that belonged to the previous concept. Other methods (Hulten et al., 2001; Maloof and Michalski, 2004; Widmer and Kubat, 1996; Gama et al., 2004a) eliminate older examples according to their number; in the previous example, if these methods need to discard a 10% of the examples within the window, they would maintain some examples from the burst. This means they could take longer to react to a concept drift.

Local Window Size in Improvement State

When a leaf is in improvement state it may lose examples. This is due to the fact that a well-constructed leaf with a high performance discards older examples on receiving new ones to avoid excessive accumulation of examples.

Therefore we make use of a heuristic, that can be describe as: the number of examples in the leaf should not be greater than the number of examples in its brother subtree, especially when the performance of the leaf is high. The formula which permits this to be carried out is:

$$|E_{leaf}| > \frac{|E_{brother}|}{perf_{leaf}(t)}$$

where: E_{leaf} is the set of examples stored in the leaf, $E_{brother}$ is the set of examples contained in the brother subtree of the leaf, and $perf_{leaf}(t)$ is the current performance measure in the leaf.

As can be seen, when performance is low the boundary permits the leaf to accumulate examples so that it can induct the concept with more data. If performance is high, the number of examples in the brother node is taken into consideration to control the number of examples in the leaf.

When the tree is functioning well (i.e., every node has high performance), the number of examples beneath the two branches of any node, whether leaf or subtree, should be balanced. Any imbalance results in an older example being forgotten by a leaf. Obviously the main objective here is to optimize the number of examples stored when the concept has been adequately learned.

Local Expansion/Labelling Parameter

Most supervised learning methods use some parameters to decide when to stop learning or when to make a decision. For example, C4.5 algorithm uses a majority threshold to label their leaves. This is not appropriate for our model as we do not want the user to have to study the problem to configure this kind of parameters of the algorithm. Something similar occurs with the expansion criteria. For example, algorithms able to deal with data streams like VFDT, CVFDT and others (Gama et al., 2004b; Gama and Medas, 2005), use Hoeffding bounds as expansion criteria (Hoeffding, 1963), but it also contains many parameters that may be arbitrarily set by users.

As far as we know there is no theoretically supported criterion to decide when to stop growing a tree, and which is capable of functioning without parameters. Therefore we make use of a heuristic which can be resumed as follow: a leaf is labelled when its number of majority class examples is greater than the number of examples of non-majority class in exponential factor; otherwise, OnlineTree2 may try to expand that leaf. The experimentation, conducted using many different problems of varying dynamics, led us to the conclusion that an exponential factor should be near to e. This heuristic can be formalized as:

$$\begin{cases} m < e^r & \text{, try to expand the leaf} \\ otherwise & \text{, label the leaf} \end{cases}$$

where: m is the number of examples with majority class in the leaf, and r is the rest of the examples in that leaf, that is: the total number of examples in the leaf minus m.

Once presented the adjustable local parameters of OnlineTree2, stages 2a and 2b will be better understood. For the sake of clarity, leaf treatment is explained first.

4.2.3 STAGE 2B: LEAF TREATMENT

When OnlineTree2 finish the previous stage (downwards revision of statistics) in a leaf, the example is stored in it and the algorithm updates local class distribution and local performance in that leaf.

At this point, OnlineTree2 will try to adjust the leaf to adapt better its related subconcept. The action to be performed depends on the state of the leaf: if it is degradation, OnlineTree2 adjust its local window to forget examples; otherwise, OnlineTree2 tries to expand the leaf. The latter is doing depending on the expansion/labelling parameter described in previous section. When an expansion is allowed, OnlineTree2 creates a subtree of only one level.

OnlineTree2 learns dichotomic trees from examples described by symbolic and numeric attributes. Utgoff et al. (1997) comment that the binarized forms of attributes (e.g., D([Colour= red])=true, false, D([Age40])=true, false) produce better results than the original multi-valued forms (e.g., D(Colour)=red, blue, green, D(Age)=[0...150]). In order to binarize the continuous attributes, we use the clustering algorithm k-means (MacQueen, 1967) using the attribute values of the examples of the leaf we want to expand (Dougherty et al., 1995). The reason that we decided to use this method is that a binary split is needed (k=2) and because 2-means has a linear complexity ($O(n)$), while the dichotomic split used by C4.5 and ITI is loglinear ($O(n \cdot log(n))$). To obtain the binary decision attribute of the node, OnlineTree2 uses normalized information gain (Quinlan, 1986).

Before substituting the leaf, OnlineTree2 checks the new subtree for coherence, similar to the way in which OnlineTree2 checks nodes in the previous stage. If the new subtree is coherent with the current concept, it replaces the leaf. If not, the leaf is labelled with the majority class, and OnlineTree2 adjusts the local window size in improvement state, as presented previously.

4.2.4 STAGE 2A: NON COHERENT NODE TREATMENT

If, after the first stage, the example has found an incoherent node with the current concept, that node may be adjusted by either; pruning and labelling, or grafting a new subtree that replaces the incoherent one.

First of all, OnlineTree2 stores the example in its corresponding leaf without revising nodes, updating the performance of that leaf. Then, the local window size of each leaf in degradation state below the non coherent node is adjusted as described in Section 4.2.2. After that, the remaining examples are collected and a new leaf, labelled with the majority class of these examples, is created. That leaf replaces the incoherent node. Finally an attempt to reconstruct the leaf is performed (as seen in the previous section).

4.3 Stage 3: Upward Updating of Statistics

Once stage 2 has been completed, OnlineTree2 travels back to the root, taking advantage of recursion used in Stage 1, updating statistics and performance of each visited node.

4.4 Complexity Issues

In order to calculate the complexity of our algorithm, we take into consideration different situations that may occur.

In the case of an established concept and once a tree has been constructed that does not change over time (no concept drifts arrive), the complexity in time of each example is calculated as the cost of the example moving down the tree (i.e., $O(log_2(t)$, where t is the number of nodes in the

tree), storing it on the leaf (i.e., $O(1)$) and using the returning mechanism of the recursion to update statistics (i.e., $O(log_2(t))$). In this case, we have $O(2 \cdot log_2(t) + 1) \equiv O(log_2(t))$.

If the new example provokes an adjustment of a subtree, the cost is higher. As before, the example moves down the tree until it meets the leaf or the badly adjusted node. In any case, the example is placed on its corresponding leaf (i.e., $O(log_2(t))$, where t represents the number of nodes below the current node). Subsequently it collects all the leaves hanging from this node (i.e., $O(t)$) and readjusts its windows (i.e., $O(l \cdot f)$, where l is the number of leaves and f is the number of examples that stay outside the window and must be forgotten). With the remaining examples, it tries to reconstruct the new node using information gain (i.e., $O(n \cdot a \cdot v)$, where n is the number of examples, a is the number of attributes of the problem and v is the number of values in each attribute). Finally, it updates the statistics of the ancestor nodes, taking advantage of the recursion (i.e., $O(log_2(t))$). In total we have $O(2 \cdot log_2(t) + t + (n \cdot a \cdot v)) \equiv O(n \cdot a \cdot v)$. As can be seen, the highest complexity is from the calculation of information gain when OnlineTree2 needs to expand a leaf.

As a whole, the complexity of our method is equivalent to that of other TDIDT incremental methods; however OnlineTree2 can also deal with concept drift.

5. Experimentation

This section shows results of the proposed algorithm when facing different kinds of problems, from classical (stationary) data sets to data streams with unknown dynamics.

To compare results, we have chosen several well-known algorithms based on different techniques on machine learning and data mining: C4.5 and CVFDT are tree-based algorithms, the former is commonly used to treat problems without concept changes, the latter creates trees capable of dealing with concept changes; IB-k is a nearest-neighbour classifier algorithm; SMO (Platt, 1998) is a support vector machine based on a polynomial kernel; MLP is a multilayer perceptron based algorithm; a Naïve Bayes algorithm (Duda and Hart, 1973); and DDM, a meta algorithm which is able to provide concept drift treatment to a base algorithm by controlling its online error rate (each example is tested using the model before learning from it). We used a suite for data mining called Weka (Witten and Frank, 2005) to obtain results with the algorithms describe above, except for CVFDT, in which we used the implementation provided in the VFML toolkit (Hulten and Domingos, 2003), and DDM in which we used our own implementation integrated in Weka. Otherwise stated, defaults parameters for all methods are used on all the experiments.

The following experiments have been done on a 3 GHz machine with 2 GBytes of memory, running GNU/Linux.

When using OnlineTree2, it is difficult to know when a false alarm regarding concept drift is produced (i.e., the detection of a concept change when there is none), because information about concept drift is distributed into the leaves of the tree. With this purpose, we define an external new measure, called *estimated rate of concept drift* (*ercd*), which is calculated with the arrival of each example:

$$ercd_{tree} = \frac{\sum_{l \in L_{degraded}} \frac{|E_l|}{|E_{tree}|} w_i}{|L_{degraded}|}$$

where, $L_{degraded}$ is the set of degraded leaves in tree, E_{node} is the set of examples stored below a node, and, wf_l is the window fraction forgotten in leaf l. When this value is greater than a specified

threshold (e.g., we use 10^{-4}), we suppose that a concept drift is detected. This metric is not part of the algorithm.

Throughout this section, low error levels and fast reaction to concept drift will show the relevance of using a strategy based on local windows when dealing with problems with unknown dynamics. Experiments suggest a high level of adaptability of our algorithm when facing unforeseen changes.

This section has been divided into two parts: Section 5.1 contains an analysis of how OnlineTree2 behaves to different conditions in various concept drifting data streams as well as comparisons with other algorithms; and Section 5.2 evaluates the performance of algorithms when facing with real problems whose concepts do not change over time.

5.1 Experiments in Problems with Concept Changes

This section presents problems whose dynamics come in an unknown manner to show how OnlineTree2 algorithm adapts to them automatically. To do so, we will present problems in which: a hyperplane changes its position in the attribute space gradually and/or abruptly, noise in examples can be increased/decreased, and the speed of the concept drift also changes. We are also interested in evaluating the incidence of virtual drift in which the concept remains the same but the distribution of the observed examples changes over time. Algorithms able to track concept drifts used to make comparisons are: CVFDT, IB1 with a global fixed window and DDM with Naïve Bayes as base algorithm (abbreviated as DDM+NB). To work with CVFDT, a previous discretization of numerical attributes is needed. This was carried out dividing these variables into 5 bins. The rest of the algorithms, including OnlineTree2, can deal with numerical attributes.

The rest of this section will be organized as follows. The incidence of noise and virtual drift within the concept change is evaluated in Section 5.1.1, while Section 5.1.2 presents the performance of the algorithms when facing a synthetic data stream involving unknown conditions, such as different degrees of concept change, virtual drift and noise. Finally, Section 5.1.3 evaluates the performance of the proposed algorithm dealing with a real problem where concept changes may occur.

5.1.1 THE INCIDENCE OF NOISE AND VIRTUAL DRIFT WITHIN THE CONCEPT CHANGE

In this section, we use a well-known data set in the concept drift community that is useful to evaluate and illustrate how an algorithm able to deal with concept drift should work. It is known as the SEA data set and was proposed by Street and Kim (2001). It is easily understood, simply by imagining a cross-section of a three-dimensional space. The examples are points in this space, and are labelled depending on their position in respect to the stated plane (when the example is below that plane, it is labelled as positive; otherwise it is labelled as negative). Following a number of examples the plane is moved, changing the labels of the examples, and thus producing a change in concept. It is hoped that algorithms capable of managing changes in concept can treat this data set.

The SEA data set can be described as follows: being x_i, $i \in \{1,2,3\}$, variable with real domain ($x_i \in [0,1]$), an example is composed of the values of these three variables and is labelled according to the actual concept. An example is labelled as positive when $x_1 + x_2 < b$, otherwise the example would have a negative label. A change in concept takes place changing the value of variable b ($b \in 8,9,7,9.5$). Examples are affected by noise at a level of 10% (that is, each example has a probability of 10% of being labelled randomly).

This data set can be considered as a stream of examples. Each training example occurs in a step of time. The data set contains 50000 time steps and four concepts (each concept contains 12500 time steps). An independent set of 10000 test examples is used to evaluate the classifiers every 500 time steps. Examples in the test set are noise free and are labelled according to the concept being evaluated. For each experiment based on this data set 30 runs are randomly generated which evaluate the algorithms and calculate classification errors and the number of examples maintained in our decision tree on each test point. On each test point a Wilcoxon hypothesis test (with significance level at 0.05) was used to compare algorithms.

In Figure 3 the misclassification error of the algorithms on this data set is shown. The horizontal axis represents time and the vertical axis represents the percentage of misclassification error achieved. The discontinued vertical lines show the moment of each concept change.

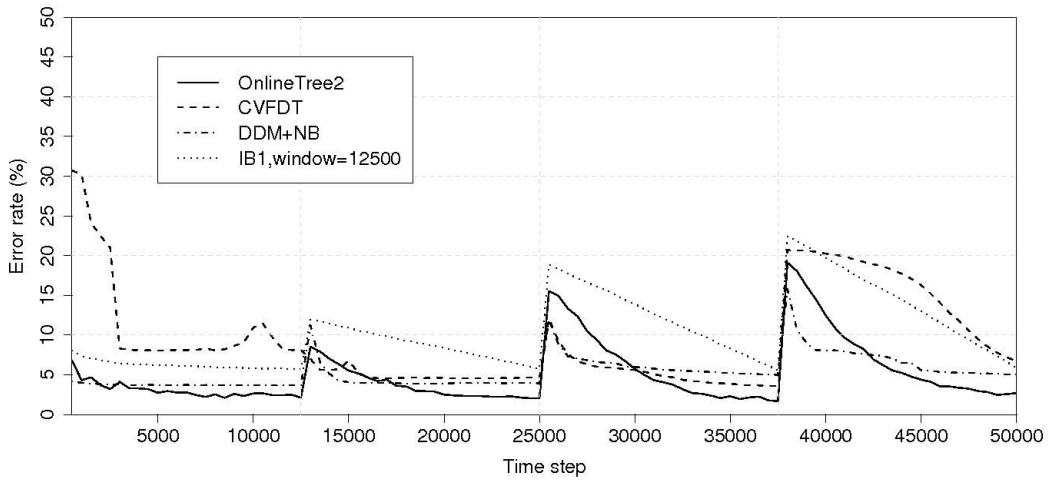


Figure 3: Misclassification errors on SEA data set with 10% noise

When facing concept drift problems, error rate curves show a common behaviour: just after a concept change, a sudden rise in the error rate occurs. This happens because models are evaluated with examples from the new concept and they have not adapted yet. From this point on, the slope of each algorithm is important because it shows their reaction capability when faced with changes in concepts. It is desirable that once a concept change occurs, the classifier detects it and quickly forgets a subset of examples and thus fit the new concept accordingly.

Results of IB1 using a global window, with fixed size of 12500 examples, show that reactions to concept change are slow, as the algorithm must wait until the window has forgotten all examples from a previous concept. For this experiment, once this occurs, error rates are at around 6.5% at the end of each concept. With respect to algorithms using a global adaptive window size (CVFDT and DDM+NB algorithms), reactions to concept drift are faster than the previous one, but insufficient in third and fourth concepts. Nevertheless, results of both DDM+NB and CVFDT are adequate but are also shown to be affected by noise.

In this data set, OnlineTree2 has a better overall performance than the rest of the algorithms because it achieves low error rates at the end of each concept (having a significantly better error

rate for the last three concepts), reacting quickly after each concept drift. This is due to the local window strategy: OnlineTree2 detects persistent changes in subconcepts, reacting by forgetting examples from them. The robustness to noise and adaptability to different levels of concept drifts of our algorithm can be seen. Regarding false alarms, OnlineTree2 detects concept change when there is none at the beginning of the experiment. This phenomenon is due to a lack of examples at that moment.

Time and memory statistics were also collected. CVFDT proved the fastest algorithm in processing the whole data set, followed by OnlineTree2 and DDM+NB. As expected, IB1 with a fixed window of 12500 examples was the slowest algorithm. With respect to memory requirements, from more to less, IB1 with a fixed window maintains 12500 examples in memory, OnlineTree2 needs to store 4000 examples on average in a tree with about 60 leaves, while CVFDT stores a model with about 80 leaves, and finally, DDM+NB needs to store about 500 examples after each concept drift and statistics about each attribute with respect to class labels.

Change in Concept Accompanied by Change in Level of Noise

In the next experiment, the SEA data set is modified to have different noise level in each concept, that is, level of noise changes with concept drift. Specifically, percentages of noise level at each concept are: 20%, no noise, 40% and 10%, respectively. This will allow us to know if our algorithm adapts to these conditions and the nature of its adaptation. Results are presented in Figure 4.

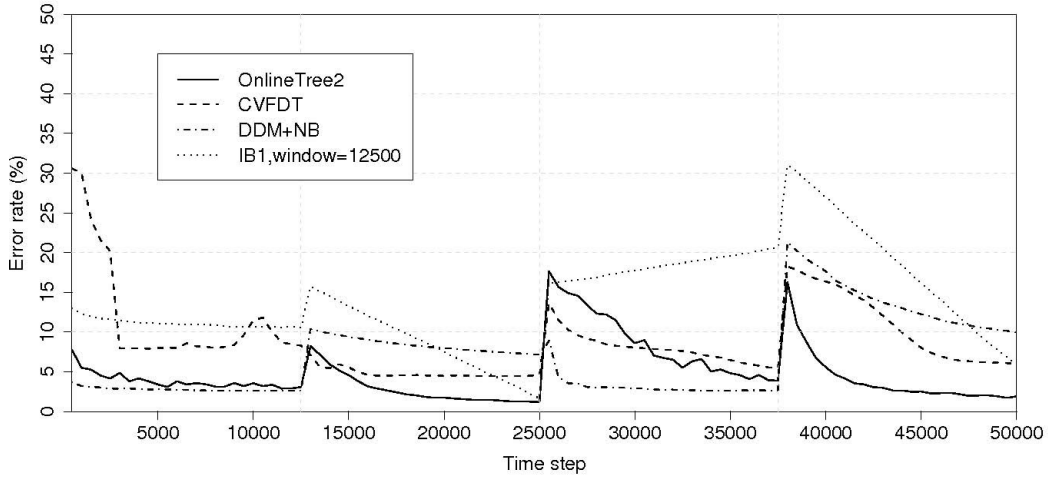


Figure 4: Misclassification error on SEA data set when the level of noise varies

The error curve of OnlineTree2 reveals factors related to noise level: if there is no noise in data (second concept) it is reasonable to have a low error rate (about 1.25%), when noise is at 10% (last concept) the error rate rises up to 2, if noise is 20% (first concept) error rate is about 2.5% and finally, when noise is 40% (third concept) error rate goes to 4.25%. This is a desirable characteristic because it demonstrates robustness. This is another advantage of making use of local adjustment of parameters and local windows.

This experiment shows that OnlineTree2 takes alarm about concept change at the beginning of the first concept, as in the previous experiment. During the third concept it also detects some concept drifts when there are none, because of the massive noise level within this concept (noise level at 40%). In the second and fourth concepts there are no false alarms.

DDM+NB and CVFDT algorithms do not work in this way. Moreover, DDM+NB has surprisingly good results for concepts with high rates of noise, but it is the worst when dealing with low rates of noise. This might be because DDM has difficulties in distinguishing concept changes when a previous concept has more noise level than the current concept. In the case of CVFDT, it is able to detect concept drifts but its convergence speed is low and it hardly managed to reduce error rate below 5%. As in the previous experiment, IB1 with global fixed window does not work well because it is sensitive to noise and it must wait until every example of the previous concept has been forgotten.

In these conditions, OnlineTree2 has shown better adaptability than other algorithms evaluated, reaching low error levels when dealing with concept changes and different levels of noise.

Noise, Virtual Drift and Concept Changes

When learning algorithms face real problems, examples do not arrive in a uniform manner. They usually show a part of the domain; moreover, they can change and come from another area but a concept change is not made. This is called virtual drift. In the following paragraphs we extend the original SEA data set definition to contain both concept drift and virtual drift.

The SEA data set is modified in such a way, that with each concept, two virtual drifts occur, and so the concepts are divided into three equal parts. In the first third, the values of the attributes for each example are uniformly distributed, in the second third they are distributed following a Normal distribution, N($b/2, b/4$), and in the last third they are once again distributed uniformly. This data set also has 10% level of noise.

Figure 5 shows results from this experiment. CVFDT and DDM+NB are affected by this phenomenon, detecting virtual drift as concept change, which implies degradation in its models. Contrary to these algorithms, IB1 with global fixed window and OnlineTree2 show a stable behaviour along this data set, producing models similar to that obtained in the experiment without virtual drift (see Figure 3). During this experiment, the number of false alarms detected by OnlineTree2 is the same as in the experiment without concept drift. This robustness to virtual drift allows OnlineTree2 to have lower error rates than the compared algorithms (significantly lower in second and third concepts).

5.1.2 THE INCIDENCE OF LEVEL OF CHANGE WITHIN THE CONCEPT CHANGE

In this section, we are interested in evaluating the performance of OnlineTree2 when dealing with data streams with unknown dynamics; that is to say, data streams with noise, concept changes, different speed of change, variable concept length, virtual drift, etc. In the following experiment a synthetic data stream is presented where these characteristics are present. The aim is to test the algorithm on a problem with situations as realistic as possible, and compare its results with ones obtained by actual methods able to deal with concept drift.

The domain of this problem, based on the experiment presented by Hulten et al. (2001) and Fan (2004), is defined by a ten dimensional space divided by a hyperplane. The examples are

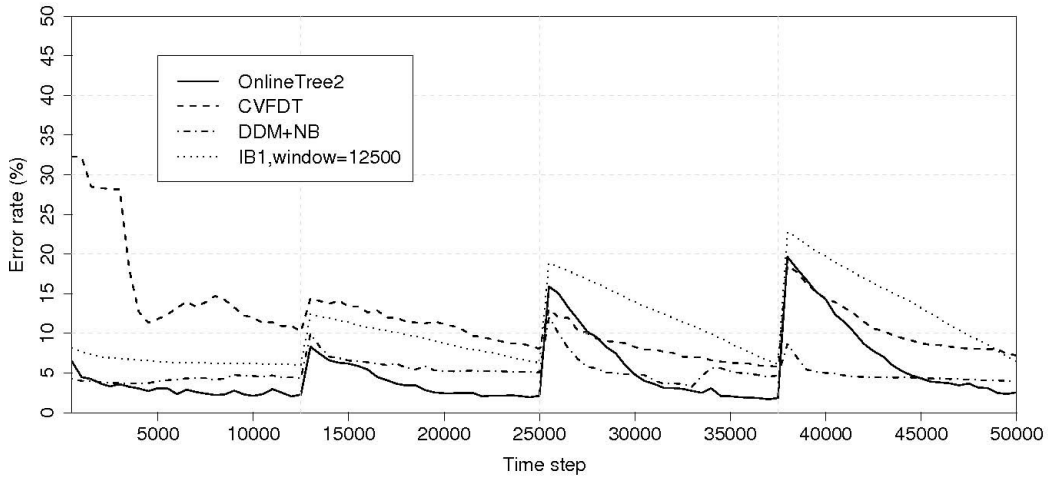


Figure 5: Misclassification error on SEA data set with virtual drift

points of this space labelled depending on the position within this hyperplane; if they are below the hyperplane they are labelled as positive, otherwise they are labelled as negative. To calculate the position of the hyperplane, the following equation is used: $\sum_{i=1}^{d} w_i x_i = w_0$, where: d is the number of dimensions ($d = 10$), w_i are weights associated to each dimension ($w_i \in [0 \dots 1]$), $w_0 = \frac{1}{2} \sum_{i=1}^{d} w_i$, and x_i are values for each dimension.

This data stream is divided into two phases, depending on the type of concept change:

- Gradual change phase. The initial concept is formed by a hyperplane were five dimensions are relevant in its calculation ($w_i = 1, i \in [1..5]$), two dimensions are sincronized ($x_5 = x_6$), and four attributes are irrelevant ($w_j = 0, j \in [5 \dots 10]$). This phase is composed of 16 different concepts (15 concept changes). Following each change the weight of one dimension is reduced ($\Delta w_i = -0.2$), producing a new concept easier to induce. Also, with the concept change the level of noise (nl) is increased by $frac5015$, thus the data stream starts without noise and finishes this phase with 50% noise. The length of each concept is calculated by: $cl = 2500 + w_0 \cdot nl \cdot 500$.

  Virtual drifts are added throughout this phase, the result of which is that the concentration of examples, in certain areas, changes over time whilst maintaining the current concept. This can create the impression that there is a concept change when in fact there is none. This time virtual drift is produced each 1000 examples by changing the distribution of examples using a Normal distribution (N(a, 0.5), where $a = vdl \cdot w_i$ and $vdl \in [0.25, 0.75]$ is the virtual drift level) on each dimension. This results in a slight imbalance of positive examples from 75% to 25%, and vice versa, every 50000 examples.

- Abrupt change phase. This phase starts using the last hyperplane made by the previous phase, but exchanging the labelled zones, that is, that which was previously positive is now negative and vice versa. This time there are five concept changes, each of them flip the labelled zones

from its previous concept and reduces the noise by 10%. Examples are uniformly distributed. The concept length is now calculated by: $cl = 100000 - 5000 \cdot nc$, where nc is the number of changes performed in this phase.

During this data stream the model induced by each learning algorithm is available. To evaluate the models, an independent set of examples is used. These test examples are noise free, have the same distribution of training examples at that time and are labelled depending on the concept present in each moment. Models are evaluated each 5000 training examples and at the end of each concept. This experiment consist on 10 runs of this data set. On each test point a Wilcoxon hypothesis test (with significance level at 0.05) has been used to compare algorithms.

In this study we are interested in measuring the performance of our algorithm and comparing it with others under conditions that include wide variability: speed of concept change, noise level and virtual drift. Results of error rate are presented in Figure 6.

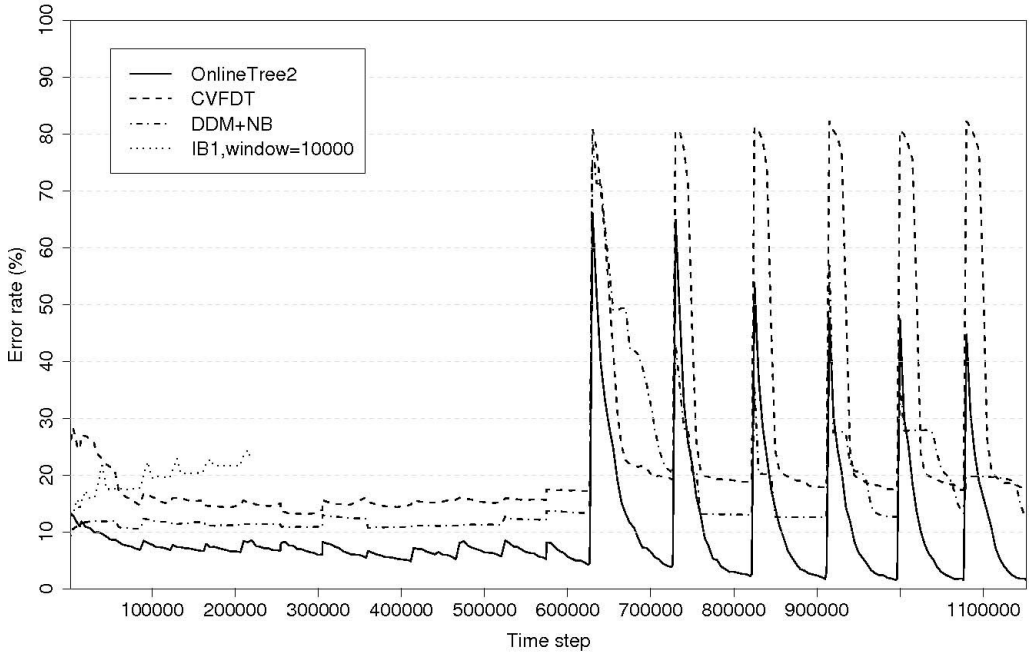


Figure 6: Misclassification error on the hyperplane data stream

Analysis of Gradual Change Phase

As can be seen in Figure 6, CVFDT starts to face this data stream with the highest error level of the algorithms tested. However after some examples, and in spite of its attribute discretization, its performance increases until it reaches the DDM+NB error rate. This Naïve Bayes based model maintains its curve around an error rate of 10% throughout this phase.

OnlineTree2 starts with a similar error rate of that obtained with DDM+NB, but our algorithm quickly improves the model, achieving (significantly) the lowest error rates of the experimenting

algorithms from time step 300000 until the end of this phase. The reason for this result is that OnlineTree2 adapts local windows and parameters of those leaves involved in gradual changes, which indicates robustness to noise, virtual drift and different levels of gradual changes. Regarding false alarms, OnlineTree2 detects concept drift when there is not due to massive noise in data a few times at the end of this phase.

IB1, using a global window with 10000 examples of fixed size, starts as well as OnlineTree2, but, as shown in previous experiments, because of this approach is not able to react to concept drift and suffers sensitivity to noise and its error rate grows quickly, being the worst algorithm of those evaluated. This algorithm was stopped at time step 220000, because its running time at this point was more than two days.

Analysis of Abrupt Change Phase

Faced with these abrupt concept changes (beyond time step 550000), the best policy is to quickly discard all examples from previous concepts and induce the new concept with fresh examples. As can be seen in Figure 6, in each concept of this phase, OnlineTree2 has a fast convergence speed. This is another advantage of the local drift detection, OnlineTree2 reacts better and faster to abrupt concept drifts than those global drift detection methods evaluated, and can control the selective forgetting of examples to reach high performance, independently of noise in data. With respect to the error rate at the end of each concept, OnlineTree2 obtains (significantly) the best results. Within this phase, the algorithm reacts to false alarms mainly when massive noise is present (first and second concept of this phase). DDM+NB has also a great convergence speed, except in the first concept of this phase. It seems to be sensitive to high rates of noise when dealing with abrupt concept changes. CVFDT seems to be more stable, but with less convergence speed than previous methods.

Overall Measurements

CVFDT process this data stream using a model with about 1000 leaves. DDM+NB stores 40 numeric measures to maintain the model and an average of 20000 examples because of its global adaptive window. IB1 with global window maintains 10000 examples as the model to update. Finally, OnlineTree2 model has approximately 600 leaves and stores an average of 50000 examples along this data set. Results from OnlineTree2 in this data stream (about 1.2 Million examples, 10 continuous attributes) shows that it is the best algorithm of those evaluated when dealing with different and unknown conditions such as different speed of change, virtual drift and different noise levels. Trees updated by the proposed algorithm are smaller than those created by CVFDT. In this problem, CVFDT processes about 2000 examples per second, DDM+NB processes about 6000 examples per second, while OnlineTree2 processes about 30 examples per second.

5.1.3 CASE STUDY: THE ELECTRICITY MARKET DATA SET

This section evaluates some learning algorithms in a real data stream which involves unknown dynamics. The data used in this experiment, collected from the Australian New South Wales Electricity Market, was first described by Harries et al. (1998). In this market, the prices are not fixed and are affected by demand and supply. The prices in this market are set every five minutes. Harries et al. (1998) show the dependence between prices on short-term events such as weather fluctuations, the

time evolution of the electricity market and adjacent areas to the one being analysed. This allowed for a more elaborated management of the supply. The excess production of one region could be sold on the adjacent region. A consequence of this expansion was a dampener of the extreme prices. This data stream is presumed to have concept drift, noise and virtual drift.

The Elec2 data set contains 45312 instances. Each example of the data set refers to a period of 30 minutes, and they have 5 fields: the day of week, the time stamp, the NSW electricity demand, the Vic electricity demand, the scheduled electricity transfer between states and the class label. The class label identifies the change of the price related to a moving average in the last 24 hours. The class level only reflects deviations of the price on a one day average and removes the impact of longer term price trends. This data set is interesting as it is a real-world data set which involves unknown dynamics, and it has been dealt with by other authors (Gama et al., 2004a).

Error rate is obtained by testing models from learning algorithms each week (i.e., each 336 examples). The test set is composed of examples from the following week; by having more examples can results in an evaluation with examples from different concepts. As error rate varies a lot (because of the small test sets), Figure 7 shows an exponential smoothed version (with $\alpha = \frac{7}{8}$) of this measure, which results in a clear graph. Of course, learning algorithms are not affected by doing this. Algorithms used to make comparisons are CVFDT, DDM+NB and MLP (with parameterization for on-line treatment[1]).

As can be seen in Figure 7, algorithms start with a similar error rate until time step 15000. From this point on, OnlineTree2 improves its model obtaining the lowest error rate until the end of the data stream. Also, from time step 20000, error rates from CVFDT and DDM+NB suggest that they do not detect concept drift because MLP (an incremental method not able to deal with concept drifting dynamics) error curve is below them. As this problem is suspected of having concept drift, OnlineTree2 is able to adapt to possibly different rates of changes (it detects 15 concept drifts) and/or noise in data.

5.2 Experimentation with Stationary Concepts

In this section, OnlineTree2 is evaluated with real problems that do not change over time. Table 2 shows a summary of twenty data sets used in this section that have been obtained from the UCI Machine Learning Repository (Asuncion and Newman, 2007). In the case of missing values, these are considered as another valid value (Quinlan, 1986). Performance of analysed algorithms is documented in terms of error rate, execution time, examples stored, and false alarms triggered. This time, algorithms used to compare results with are: CVFDT, C4.5, IB1 (no window used), SMO, MLP and Naïve Bayes.

The experiment was designed based on that proposed by Dietterich (1998). It can be described as follows: for each problem a) the examples are randomly distributed; b) a two fold cross validation is carried out with each learning algorithm; c) the resulting values in each fold are noted. These steps are repeated 5 times (producing a 5x2 CV), providing ten measurements for each algorithm and data set. These results were analysed using the recently proposed method by Demšar (2006), which focuses in comparing classifiers over multiple data sets.

The latter methodology assigns a rank to each algorithm on a data set, using its reported result based on error rates. The average rank of each algorithm is then calculated. These average values

1. Parameters modified from Weka implementation are: N (number of epochs), set to 1 and V (number of examples to validate), set to 0.

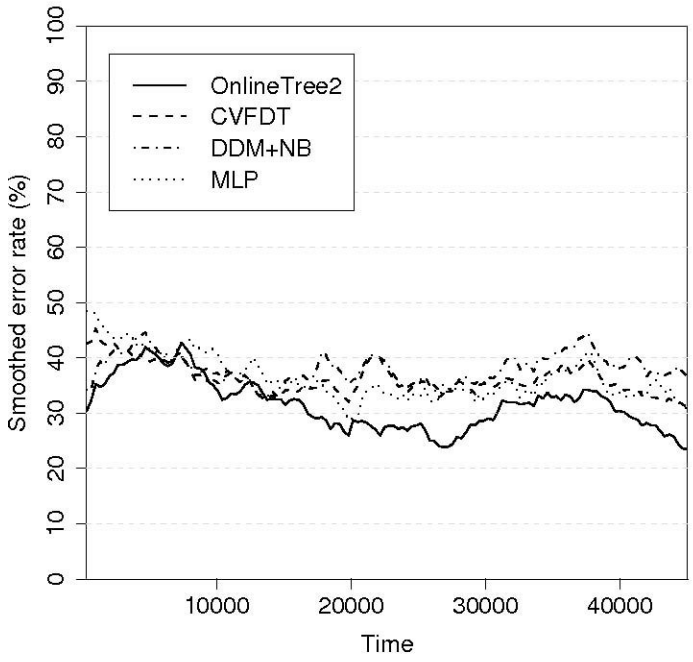


Figure 7: Results on Electricity Market Data Set

are used to decide, with a non parametrical F test, if there is a significant difference between the algorithms. If so, the critical difference measurement (CD) is calculated using the Nemenyi test. Any difference larger than this CD when analysing a pair of algorithms confirms that they are significantly different.

In Table 3, results on average error rate and standard deviation for each algorithm and data set are shown, as well as their ranks. Tails in average errors are solved by assigning an average of their ranks. *NA* error values for these measures were due to an algorithm taking more than one day to evaluate one fold of the cross validation on a data set, and the assigned rank is the largest. At the bottom of Table 3, the average of ranks for each algorithm is provided.

The Friedman test (with a significance level of 0.05) reveals that there are differences between the algorithms. Thus, a Nemenyi test (with a significance level of 0.05) is performed. The CD value for this experiment is 2.01. Figure 8 shows the critical difference diagram.

These results show that, for the data sets evaluated, there are two groups of algorithms that are not significantly different. OnlineTree2 is in the group with better ranking, and outperforms CVFDT. This demonstrates the effectiveness of Onlinetree2 in facing these kinds of problems.

On the other hand, in these experiments we observed that in the case of IB1, SMO and MLP, the average execution time was many times larger than the one obtained by OnlineTree2. For the non-incremental C4.5 algorithm, this time is nearly a half of OnlineTree2 execution time. Something similar occurs with Naïve Bayes. Neither C4.5 nor Naïve Bayes are able to deal with concept

Data Set	Examples	Classes	Num. att.	Symb. att.
Abalone	4177	21	7	1
Adult[2]	48842	2	8	7
Anneal[2]	898	6	9	29
Ann-thyroid	7200	3	6	15
Car	1728	4	0	6
Covertype	581012	7	10	44
Kr vs kv	28056	17	3	3
Letter	20000	26	16	0
Mushrooms[2]	8124	2	0	22
Nursery	12960	5	0	8
Optdigits	5620	10	64	0
Pendigit	10992	10	12	0
Segmentation	2310	7	19	0
Solar flares - 78	1066	2	10	0
Splice	3190	3	0	61
Tic-tac-toe	958	2	0	9
Vowel	990	11	10	0
Waveform	5000	3	21	0
Waveform-noise	5000	3	40	0
Yeast	1484	7	15	2

Table 2: Summary of characteristics of the evaluated data sets

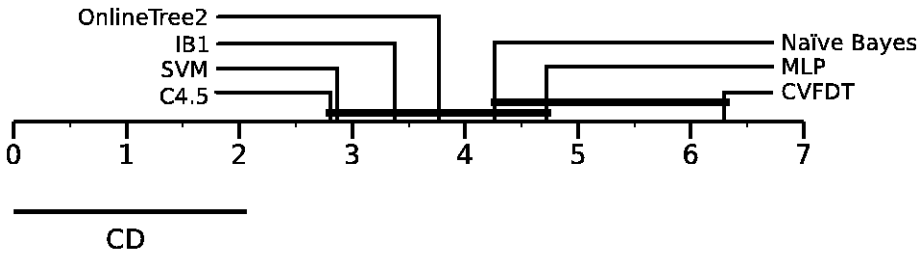


Figure 8: The critical difference diagram shows two groups (bold lines) of classifiers which are not significantly different

drifting environments. Execution time of CVFDT algorithm was shorter than OnlineTree2 execution time, but, as said before, it is significantly worse than our algorithm in these evaluations.

With respect to results of OnlineTree2 regarding number of drift detected, they shown that the algorithm detects false alarms when the algorithm starts to process each data set, presumably

2. Data set with missing values.

	OnlineTree2	Naïve Bayes	SVM	IB1	C4.5	MLP	CVFDT
Abalone	79.36 ± 0.60 (4)	76.78 ± 1.42 (2)	75.54 ± 0.55 (1)	80.2 ± 0.84 (5)	78.9 ± 0.75 (3)	83.67 ± 2.91 (6)	83.79 ± 0.72 (7)
Adult	17.2 ± 0.69 (4.5)	16.78 ± 0.27 (3)	15.11 ± 0.21 (2)	20.77 ± 0.23 (7)	13.97 ± 0.1 (1)	17.2 ± 0.95 (4.5)	17.22 ± 0.37 (6)
Anneal	6.95 ± 1.75 (2)	20.94 ± 3.47 (5)	14.05 ± 2.59 (4)	6.19 ± 0.96 (1)	10.91 ± 2.33 (3)	23.83 ± 1.9 (6.5)	23.83 ± 0.67 (6.5)
Ann-thyroid	2.08 ± 0.51 (2)	4.67 ± 0.38 (3)	6.37 ± 0.40 (4)	8.14 ± 0.41 (7)	0.4 ± 0.18 (1)	7.92 ± 0.72 (6)	7.3 ± 0.43 (5)
Car	7.93 ± 2.59 (1)	16.24 ± 1.45 (6)	8.16 ± 1.00(2)	12.28 ± 1.16 (3)	12.68 ± 1.72 (4)	14.49 ± 3.83 (5)	29.98 ± 1.04 (7)
Covertype	13.34 ± 0.48 (2)	36.82 ± 0.16 (5)	NA (6.5)	NA (6.5)	7.15 ± 0.07 (1)	27.14 ± 0.63 (3)	32.61 ± 0.14 (4)
Kr vs kv	45.35 ± 1.85 (2)	64.17 ± 0.32 (5)	56.64 ± 0.44 (4)	34.12 ± 0.41 (1)	51.30 ± 0.6 (3)	66.33 ± 1.54 (6)	83.77 ± 0.25 (7)
Letter	24.44 ± 1.48 (4)	36.02 ± 0.39 (5)	18.29 ± 3.33 (3)	5.82 ± 0.44 (1)	15.66 ± 0.55 (2)	95.08 ± 1.56 (6)	96.27 ± 0.16 (7)
Mushrooms	0.68 ± 0.42 (6)	4.81 ± 0.36 (7)	0.09 ± 0.22 (4)	0.02 ± 0.05 (1.5)	0.02 ± 0.06 (1.5)	0.22 ± 0.08 (4)	0.59 ± 0.12 (5)
Nursery	4.14 ± 0.76 (2)	9.77 ± 0.58 (6)	7.17 ± 0.24 (4)	3.84 ± 0.33 (1)	4.62 ± 0.32 (3)	8.78 ± 0.57 (5)	14.76 ± 2.16 (7)
Optdigits	23.04 ± 2.08 (6)	8.94 ± 0.49 (4)	1.95 ± 0.34 (2)	1.69 ± 0.18 (1)	11.31 ± 0.72 (5)	6.11 ± 0.54 (3)	89.14 ± 4.49 (7)
Pendigit	8.15 ± 0.78 (4)	14.26 ± 0.33 (6)	2.31 ± 0.16 (2)	0.8 ± 0.11 (1)	4.83 ± 0.46 (3)	14.12 ± 0.89 (5)	90.01 ± 0.18 (7)
Segmentation	7.86 ± 1.20 (3)	20.87 ± 2.31 (5)	9.00 ± 0.80 (4)	4.75 ± 0.37 (2)	4.64 ± 0.73 (1)	53.35 ± 4.64 (6)	86.72 ± 0.39 (7)
Solar Flares - 78	19.53 ± 1.61 (5)	20.69 ± 1.66 (6)	18.18 ± 1.12 (1)	22.31 ± 1.97 (7)	18.91 ± 0.92 (4)	18.86 ± 1.1 (2.5)	18.86 ± 0.96 (2.5)
Splice	10.23 ± 1.94 (5)	4.75 ± 0.43 (1)	7.84 ± 0.59 (4)	27.5 ± 1.22 (6)	7.65 ± 0.60 (3)	7.17 ± 0.92 (2)	48.12 ± 0.61 (7)
Tic-tac-toe	21.5 ± 7.15 (4)	29.29 ± 1.76 (5)	1.67 ± 0.62 (1)	4.45 ± 0.69 (2)	19.02 ± 2.69 (3)	30.4 ± 3.01 (6)	34.51 ± 1.65 (7)
Vowel	40.03 ± 7.86 (3)	40.61 ± 2.44 (4)	42.72 ± 3.37 (5)	7.84 ± 2.15 (1)	30.81 ± 3.22 (2)	91.4 ± 0.52 (6)	92.41 ± 0.47 (7)
Waveform	25.73 ± 1.53 (6)	19.09 ± 0.48 (3)	13.39 ± 0.62 (1)	23.3 ± 0.82 (4)	24.36 ± 1.00 (5)	14.73 ± 0.75 (2)	42.46 ± 1.92 (7)
Waveform-noise	26.46 ± 0.95 (5)	20.01 ± 0.52 (3)	14.09 ± 0.47 (1)	27.22 ± 0.96 (6)	25.02 ± 1.01 (4)	14.89 ± 0.47 (2)	47.74 ± 3.08 (7)
Yeast	49.97 ± 2.22 (5)	43.57 ± 1.02 (1)	45.30 ± 1.42 (2)	49.38 ± 1.05 (4)	46.33 ± 1.73 (3)	69.87 ± 1.71 (7)	68.80 ± 1.00 (6)
				Ranking			
Summary	3.78	4.25	2.83	3.4	2.78	4.68	6.3

Table 3: Misclassification rate results on stationary data sets

because of the low number of examples at that moment. Once several examples have been treated, this metric does not detect any false alarms.

In this experiment, the performance of our method is similar to that of algorithms used in analysis. However, it is important to note that, for each data set, the error rates achieved with OnlineTree2 have been obtained from trees which contain a reduced subset of the original data set (very small in some cases), and it is several times faster than other algorithms. Furthermore, although our method generates binary decision trees, we observed that it contains fewer leaves than the ones obtained by CVFDT and C4.5 algorithms.

6. Limitations and Future Lines

As previously mentioned OnlineTree2 is an algorithm with partial memory management, therefore its consumption of resources limits it to problems where a medium level processing capacity is available. For example, for a similar data stream of that used in Section 5.1.2, OnlineTree2 takes about 30 examples per second, while CVFDT processes about 2000 examples per second and DDM+NB about 6000 examples per second. If the problem requires processing with few resources (i.e., embedded systems, mobile phones, PDAs) and high-rate arrival of examples, this algorithm does not prove adequate.

Even though computers over time will be faster and have more and more memory, we can imagine a future line of investigation which would optimise the algorithm presented or reuse many of the ideas for a new algorithm with a no-memory management. This can be appropriate when used with equipment with limited resources. Following this line, we think that a good contribution should be to give a more intelligent method for split numerical attributes without damage the complexity showed in Section 4.4.

With the aim of improving the precision of this algorithm, although sacrificing representation of knowledge, a Naïve Bayes approach could be incorporated into the leaves as suggested by Gama et al. (2003).

7. Conclusions

An incremental decision tree learning method has been presented which is able to learn changing concepts with the presence of noise and virtual drift in examples for problems with unknown conditions.

Contrary to most of the current methods, OnlineTree2 uses local adaptive windows using a new strategy which forgets examples as a result of the leaf reducing its window size when the local performance decreases.

In general, OnlineTree2 is more robust to noise than current methods, as can be seen in the SEA data set evaluation and with different levels of noise in the data stream presented in Section 5.1.2. The proposed algorithm has less error rate at the end of each context than other methods studied in problems with gradual and abrupt concept drift and noise.

This algorithm has presented a low sensibility to change in the distribution of examples of data (virtual drift), as can be observed in Sections 5.1.1 and 5.1.2.

OnlineTree2s ability to adapt its internal parameters means it is able to face problems with unknown conditions. Current learning methods from data streams require the careful selection of the values of its user-defined parameters when faced with certain dynamics of the data stream;

making use of such methods is complicated for the user because they should be adjusted as dynamic of data stream changes.

Furthermore, if the studied dynamic changes over time, for example a change in distribution or noise level, the performance of the algorithms which cannot adapt their parameters may be very poor when dealing with new dynamics. For example IB1 with a large global fixed window could be adequate for gradual concept changes, but it is not adequate for faster changes, and in the same way IB1 with a small global fixed window is adequate for rapid changes and less adequate for very gradual changes or even when there is no concept change. Therefore OnlineTree2 is more suitable than current methods for problems with unknown dynamics with medium or high resource levels, which require a continual functioning.

We think that the OnlineTree2 processing speed of dozens or hundreds of examples per second, on an average personal computer, is sufficient for most applications. By using OnlineTree2, the user does not need to known the details of the algorithm and does not have to carry out any parameter configuration. On the other hand, we think that algorithms like CVFDT and DDM+NB are adequate for problems with a known dynamics and a data stream of thousands of examples per second.

With the goal that learning algorithms will be used by a wider spectrum of systems and users, we think that learning algorithms should be as simple to use as possible and that the models generated should be understandable. This has been a principal motivation in the design and evaluation of the OnlineTree2 algorithm. The way in which the knowledge is represented helps to understand both the patterns discovered in each moment and the problem itself. We consider the adaptive capacity of the internal parameters to be fundamental because it will allow them to deal with problems in the real world where you can never guarantee that the dynamic of the problem will not change over time.

Acknowledgments

The authors want to thank the editor and anonymous revisors for its useful comments and suggestions. This work has been partially supported by the MOISES-TA project (TIN2005-08832-C03) of the Ministry of Education and Science, Spain. The authors also thank Manuel Baena for provide us with the functional implementation of DDM algorithm used in Section 5.

Appendix A. Detailed Description of OnlineTree2 Algorithm

The aim of this appendix is to make OnlineTree2 reproducible. To do so, the algorithm appearing in Table 1 will be described in terms of functions and variables. Structural concepts for the decision tree and functions used will be defined. Notation of regular expressions is used throughout this appendix. In order to reference a variable of an object, we have used subscripts (e.g., $time_{node}$ refers to the time of the example).

The following section presents some definitions about information stored in nodes of the decision tree created and updated by OnlineTree2. Section A.2 describes the OnlineTree2 algorithm, and details the functions used in it.

A.1 Definitions

As seen in Section 4, OnlineTree2 needs to store some information in the decision tree to process examples incrementally and efficiently. This section defines this information and identifies the place where it is stored.

As stated in Section 4.2.1, every tree node has a state used to detect concept drifts.

Definition 1 (state)

$$state = (i/d, perf, \textit{prev-perf}, \textit{anom-time}, \textit{anom-perf}, pers)$$

where: *i/d* means either improvement state or degradation state, *perf*, *prev-perf* and *anom-perf* are current, previous and anomaly performance measures, respectively; *anom-time* is the anomaly time when the state enters in degradation state; and, *pers* counts how many examples arrived at the node when it is in degradation state (from last anomaly time).

The following defines the information contained in tree nodes.

Definition 2 (info)

$$info = (cd, ecc, state)$$

where: $cd = (class, frec)^+$ is the class distribution of the examples passed through the node and not forgotten (note that is the total number of this examples), *ecc* is the number of examples correctly classified below the node, and $state$ is the state of the node as defined above.

Once the information in a node has been defined, we treat tree nodes as decision nodes (*d-node*) and leaf nodes (*l-node*).

Definition 3 (d-node)

$$\textit{d-node} = (info, \textit{splitting-attribute}, children)$$

where: *info* contains node information as defined above, *splitting-attribute* is the decision attribute used to split with domain $D(\text{splitting-attribute}) = V_1, V_2$ and $children = (V_1, node_1), (V_2, node_2)$ with $node_1$ and $node_2$ as nodes associated with *d-node*.

Definition 4 (l-node)

$$\textit{l-node} = (info, E)$$

where: *info* contains leaf information as defined above, and E is a FIFO structure containing the examples stored in the leaf. Operations defined for E are: *head(E)*, returns the first example added to the queue; *tail(E)*, returns the last example added to the queue; *add(E, example)*, adds an example to the tail of E; *remove(E, i)*, removes i examples from the head of E; and, $join(E_1, E_2, \ldots)$ returns a queue with the examples from $E_1, E_2, \ldots$ ordered by their time stamp. Note that the label of the leaf can be calculated efficiently from cd_{l-node}.

Definition 5 (node) *A node can be a decision node (d-node) or a leaf node (l-node).*

```
OnlineTree2(node, example): node
    IF not IsRevisable(node) THEN                                          } Stage 1
        OnlineTree2(NextNode(node, example), example)
    ELSE                                                                   }
        IF node is a d-node                                                }
            Drop(node, example)                                            }
            AdjustLocalWindowInDegradedLeaf(leaves(node))        (a)       }
            Prune(node)                                                    }
            UpdatePerformance(node)                                        }
            TryToReconstruct(node)                                         }
        ELSE                                                               }
            UpdateLeaf(node, example)                                      }
            IF state_n is Improvement THEN                                 } Stage 2
                TryToReconstruct(node)                                     }
                IF node is a d-node THEN                                   }
                    AdjustLocalWindowInImprovedLeaf(node)        (b)       }
                ENDIF                                                      }
                ELSE                                                       }
                AdjustLocalWindowInDegradedLeaf(node)                      }
            ENDIF                                                          }
        ENDIF                                                              }
        RETURN node                                                        }
    ENDIF                                                                  }
    UpdateNode(node, example)                                              } Stage 3
    RETURN node                                                            }
```

Pseudocode 1: OnlineTree2 algorithm

Definition 6 (tree) *A binary decision tree is a node.*

Definition 7 (example)

$$example = ((at, v)^a, class, time)$$

where: a is the number of attributes in the problem, $(at, v)^a$ are a attribute-value pairs describing the example, *class* is the label and *time* is its time stamp.

A.2 OnlineTree2 Algorithm

Pseudocode 1 presents the *OnlineTree2* pseudo-algorithm. As in Section 4, the pseudo-algorithm has been described with three stages: downwards revision of statistics, treatment of a non coherent node or leaf in the second stage, and finally, updating the information stored in visited nodes. Following sections describe in details the functions of each stage.

IsRevisable(*node*): boolean
    RETURN not (*node* is a d-node $\wedge$ χ^2-*test*(cd_{node},$cd_{children}$))

Pseudocode 2: *IsRevisable* function

NextNode(*d-node*, *example*): *node*
    $value \leftarrow value/(splitting\text{-}attribute, value) \in (at, val)^{a}_{d-node}$
    RETURN *next-child* / $value \in A \wedge (A,next\text{-}child) \in children_{d-node}$

Pseudocode 3: *NextNode* procedure

A.2.1 DOWNWARDS REVISION OF STATISTICS

As discussed in Section 4.1, the first stage of the algorithm searches either a leaf or a node that does not fit with current concept (non coherent node) in the path of current example from root to a leaf. On each visited node, a call to *IsRevisable* function is performed (see Pseudocode 2).

It checks the node to be a leaf or for coherence with current concept by doing a χ^2 test with a significance level of 0.05 (Quinlan, 1986). If this test results in a coherent node, this procedure is recursively performed with the next node depending on the value in the example for the splitting attribute (by calling the *NextNode* function, see Pseudocode 3). If a leaf or a non coherent node is reached, the stage ends and the next one starts.

A.2.2 TREATMENT OF A LEAF OR A NON COHERENT NODE

As stated above, this stage tries to adjust tree structure to the dynamics of current concept in the data stream. It is divided into two parts, depending on the type of node returned by the first stage: non coherent node treatment or leaf treatment.

Non Coherent Node Treatment

Once the example reaches a non coherent node, OnlineTree2 drops the example to the correct leaf, which is updated accordingly, by calling the *Drop* procedure (see Pseudocode 4 and related procedures: *UpdateLeaf* in Pseudocode 6, *Store* in Pseudocode 5, *UpdateStatistics* in Pseudocode 7 and *UpdatePerformance* in Pseudocode 8).

Drop(*node*, *example*)
    IF *node* is a l-node THEN
        UpdateLeaf(*node*, *example*)
    ELSE
        Drop(*NextNode*(*node*, *example*), *example*)
    ENDIF

Pseudocode 4: *Drop* procedure

Store(*leaf*, *example*)
    add(E_{leaf},*example*)

Pseudocode 5: *Store* procedure

UpdateLeaf(*l-node*, *example*)
    Store(*node*, *example*)
    UpdateStatistics(*node*, *example*)
    UpdatePerformance(*node*, $time_{example}$)

Pseudocode 6: *UpdateLeaf* procedure

UpdateStatistics(*node*, *example*)

Let $cd_f = (class, freq)*$ be the class distribution of forgotten examples after stage two.

$$cd_{node} = \left\{ (class, freq') / (class, freq) \in cd_{node} \wedge freq' = \left\{ \begin{array}{ll} freq & \text{if } class \neq class_{example} \\ freq+1 & \text{if } class = class_{example} \end{array} \right\} \right.$$

$$cd_{node} = \left\{ (class, freq') / (class, freq) \in cd_{node} \wedge freq' = \left\{ \begin{array}{ll} freq & \text{if } (class, *) \notin cd_f \\ freq - freq_f & \text{if } (class, freq_f) \in cd_f \end{array} \right\} \right.$$

IF *node* is a l-node THEN
    ecc_{node} = max(*freq* / (*class*,*freq*) $\in cd_{node}$)
ELSE
    $ecc_{node} = \sum_{(V,child) \in children_{node}} ecc_{child}$
ENDIF

Pseudocode 7: *UpdateStatistics* procedure

UpdatePerformance(*node*, *time*)

Let $ia_{node} = ecc_{node}/ne_{node}$ the instantaneous accuracy of node, and $\alpha = 7/8$ the factor used for exponential smoothing

$prev\text{-}perf_{node} \leftarrow perf_{node}$
$perf_{node} \leftarrow \alpha \cdot prev\text{-}perf_{node} + (1-\alpha) \cdot ia_{node}$
IF $prev\text{-}perf_{node} \leq perf_{node}$ THEN
  $state_{node} \leftarrow (i, perf_{node}, prev\text{-}perf_{node}, n/a, n/a, n/a)$
ELSIF *node* is in Improvement State THEN
  $state_{node} \leftarrow (d, perf_{node}, prev\text{-}perf_{node}, time, prev\text{-}perf_{node}, 0)$
ELSE
  $state_{node} \leftarrow (d, perf_{node}, prev\text{-}perf_{node}, anom\text{-}time_{node}, anom\text{-}perf_{node}, pers_{node}+1)$
ENDIF

Pseudocode 8: *UpdatePerformance* procedure

AdjustLocalWindowInDegradedLeaf(*leaves*)
  $cd_{forgotten} \leftarrow \oslash$
  FOR each l in *leaves* DO
    IF l is in Degradation State THEN
      $dw_l \leftarrow [time_{head(E_l)}, anom\text{-}time_l)$
      $dp_l \leftarrow (anom\text{-}perf_l perf_l)$
      $ff_l \leftarrow \min(pers_l \cdot dp_l, 1)$
      $lw_l \leftarrow [time_{head(E_l)} + |dw_l| \cdot ff_l, tail(E_l)]$
      $fe_l \leftarrow \{ef \in E_l / time_{ef} \notin lw_l\}$
      $remove(E_l, |fe_l|)$
      $cd_{forgot} \leftarrow \{(class_i, freq_i)^* / freq_i = freq_i + |\{e \in fe_l / class_i = label(e)\}|\}$
    ENDIF
  ENDFOR

Pseudocode 9: *AdjustLocalWindowInDegradedLeaf* procedure

Then, each leaf below the non coherent node is ordered to forget examples (see *AdjustLocalWindowInDegradedLeaf* procedure in Pseudocode 9); examples forgotten are noted to update statistics of ascendant nodes in the next stage. After that, the node is converted into a leaf with the unforgotten examples (see Pseudocode 10) and its metrics for forgetting examples are updated (see Pseudocode 8). Then a reconstruction of the pruned node is attempted (see Pseudocode 11) in order to make a coherent node for the current concept.

Leaf Treatment

When the first stage ends in a leaf node, the example updates that leaf (see Pseudocode 6). If the leaf is improving its performance, the algorithm tries to create a new decision node (see Pseu-

Prune(*node*)
$E_{new} \leftarrow join(\{E_l / l \in leaves(node)\})$
$new\text{-}leaf \leftarrow create\text{-}l\text{-}node(E_{new})$
$state_{new-leaf} \leftarrow state_{node}$
$swap(node, new\text{-}leaf)$

Pseudocode 10: *Prune* procedure

TryToReconstruct(*leaf*)

Let *create-decision-node*: $E \rightarrow d\text{-}node$, a function that returns a decision node with two leaves as children using the gain ratio as splitting criteria (Quinlan, 1986) from a set of examples E.

$max\text{-}freq = max(\{freq/(class, freq) \in cd_{leaf}\})$
$total = ne_{leaf}$
IF $max\text{-}freq \leq e^{total - max\text{-}freq}$ THEN
    $node \leftarrow create\text{-}d\text{-}node(E_{leaf})$
    IF not *IsRevisable*(*node*) THEN
        $swap(leaf, node)$
    ENDIF
ENDIF

Pseudocode 11: *TryToReconstruct* procedure

AdjustLocalWindowInImprovedLeaf(*leaf*)

Let *brother* a node such as the parent of *leaf* and *brother* is the same.

IF $ne_{leaf} \cdot perf_{leaf} > ne_{brother}$ THEN
  $remove(E_{leaf}, 1)$
ENDIF

Pseudocode 12: *AdjustLocalWindowInImprovedLeaf* procedure

UpdateNode(*d-node*, *example*)
  $UpdateStatistics(d\text{-}node, example)$
  $UpdatePerformance(d\text{-}node)$

Pseudocode 13: *UpdateNode* procedure

docode 11). If there is no reconstruction, the algorithm tries to find out if the leaf has become stable, deciding to forget the oldest example of the leaf in that case, by calling the *AdjustLocalWindowInImprovedLeaf* procedure (see Pseudocode 12).

If the leaf is not improving its performance, then an attempt to reduce its local window is performed (see Pseudocode 9).

A.2.3 UPWARD UPDATING OF STATISTICS

Once stage two has finished, the algorithm starts a bottom-up process, updating the information of each visited node by calling to the *UpdateNode* procedure (see Pseudocode 13).

References

David Aha, Dennis Kibler, and Marc Albert. Instance-based learning algorithms. *Machine Learning*, 6:37–66, 1991.

Arthur Asuncion and David J. Newman. UCI machine learning repository. http://www.ics.uci.edu/mlearn/MLRepository.html, Irvine, CA: University of California, Department of Information and Computer Science, 2007.

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

Thomas G. Dietterich. Approximate statistical tests for comparing supervised classification learning algorithms. *Neural Computation*, 10(7):1895–1923, 1998.

Pedro Domingos and Geoff Hulten. Mining high-speed data streams. In *Proceedings of the 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 71–80, 2000.

James Dougherty, Ron Kohavi, and Mehran Sahami. Supervised and unsupervised discretization of continuous features. In *Proceedings of the 20th International Conference on Machine Learning*, pages 194–202, 1995.

Richard Duda and Peter Hart. *Pattern Classification and Scene Analysis*. Wiley-Interscience, 1973.

Floriana Esposito, Donato Malerba, and Giovanni Semeraro. A comparative analysis of methods for pruning decision trees. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19 (5):476–491, 1997.

Wei Fan. Systematic data selection to mine concept-drifting data streams. In *Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 128–137, 2004.

JoÃo Gama and Pedro Medas. Learning decision trees from dynamic data streams. *Journal of Universal Computer Science*, 11(8):1353–1366, 2005.

JoÃo Gama, Pedro Medas, Gladys Castillo, and Pedro Rodrigues. Learning with drift detection. In *Proceedings of the 17th Brazilian Symposium on Artificial Intelligence*, pages 286–295, 2004a.

JoÃo Gama, Pedro Medas, and Pedro Rodrigues. Learning in dynamic environments: Decision trees for data streams. *Pattern Recognition in Information Systems*, pages 149–158, 2004b.

JoÃo Gama, Ricardo Rocha, and Pedro Medas. Accurate decision trees for mining high-speed data streams. In *Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 523–528, 2003.

Michael Harries, Claude Sammut, and Kim Horn. Extracting hidden context. *Machine Learning*, 32(2):101–126, 1998.

Wassily Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58:13–30, 1963.

Geoff Hulten and Pedro Domingos. VFML – a toolkit for mining high-speed time-changing data streams. http://www.cs.washington.edu/dm/vfml/, 2003.

Geoff Hulten, Laurie Spencer, and Pedro Domingos. Mining time-changing data streams. In *Proceedings of the 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 97–106, 2001.

Ralf Klinkenberg. Learning drifting concepts: Example selection vs. example weighting. *Intelligent Data Analysis*, 8(3):281–300, 2004.

Ralf Klinkenberg and Thorsten Joachims. Detecting concept drift with support vector machines. In *Proceedings of the 17th International Conference on Machine Learning*, pages 487–494, 2000.

Ralf Klinkenberg and Ingrid Renz. Adaptive information filtering: Learning in the presence of concept drifts. In *Workshop Notes of the ICML-98 on Workshop on Learning for Text Categorization*, pages 33–40, 1998.

Ivana Krizakova and Miroslav Kubat. FAVORIT: Concept formation with ageing of knowledge. *Pattern Recognition Letters*, 13(1):19–25, 1992.

Miroslav Kubat and Gerhard Widmer. Adapting to drift in continuous domains (extended abstract). In *Proceedings of the 8th European Conference on Machine Learning*, pages 307–310, 1995.

James B. MacQueen. Some methods for classification and analysis of multivariate observations. In *Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, pages 281–297, 1967.

Marcus A. Maloof and Ryszard S. Michalski. Selecting examples for partial memory learning. *Machine Learning*, 41(1):27–52, 2000.

Marcus A. Maloof and Ryszard S. Michalski. Incremental learning with partial instance memory. *Artificial Intelligence*, 154:95–126, 2004.

John Nagle. On packets switches with infinite storage. *IEEE Transactions On Communications*, 35 (4):435–438, 1987.

Marlon Núñez, Raúl Fidalgo, and Rafael Morales. On-line learning of decision trees in problems with unknown dynamics. In *Proceedings of the 4th Mexican International Conference on Artificial Intelligence*, pages 443–453, 2005.

Vern Paxson and Mark Allman. RFC-2988: Computing TCPs transmission timer. Network Working Group Requests for Comment, 2000.

John Platt. Fast training of support vector machines using sequential minimal optimization. *Advances in Kernel Methods - Support Vector Learning*, pages 185–208, 1998.

Jon Postel. RFC-793: TCP specification. ARPANET Working Group Requests for Comment, DDN Network Information Center, SRI International, 1981.

Duncan Potts and Claude Sammut. Incremental learning of linear model trees. *Machine Learning*, 61:5–48, 2005.

J. Ross Quinlan. Induction of decision trees. *Machine Learning*, 1:81–106, 1986.

J. Ross Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1993.

W. Nick Street and YongSeog Kim. A streaming ensemble algorithm (SEA) for large-scale classification. In *Proceedings of the 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 377–382, 2001.

Charles Taylor, Gholamreza Nakhaeizadeh, and Carsten Lanquillon. Structural change and classification. In *Workshop Notes of the ICML-97 Workshop on Dynamically Changing Domains: Theory Revision and Context Dependence Issues*, pages 67–78, 1997.

Paul E. Utgoff, Neil C. Berkman, and Jeffery A. Clouse. Decision tree induction based on efficient tree restructuring. *Machine Learning*, 29(1):5–44, 1997.

Gerhard Widmer and Miroslav Kubat. Learning in the presence of concept drift and hidden contexts. *Machine Learning*, 23:69–101, 1996.

Dwi H. Widyantoro, Thomas R. Ioerger, and John Yen. An adaptive algorithm for learning changes in user interests. In *Proceedings of the 8th International Conference on Information and Knowledge Management*, pages 405–412, 1999.

Ian H. Witten and Eibe Frank. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann, 2005. (2nd Edition).

Journal of Machine Learning Research 8 (2007) 2629-2669 Submitted 6/03; Revised 8/07; Published 11/07

Hierarchical Average Reward Reinforcement Learning

Mohammad Ghavamzadeh MGH@CS.UALBERTA.CA
Department of Computing Science
University of Alberta
Edmonton, AB T6G 2E8, CANADA

Sridhar Mahadevan MAHADEVA@CS.UMASS.EDU
Department of Computer Science
University of Massachusetts
Amherst, MA 01003-4601, USA

Editor: Michael Littman

Abstract

Hierarchical reinforcement learning (HRL) is a general framework for scaling reinforcement learning (RL) to problems with large state and action spaces by using the task (or action) structure to restrict the space of policies. Prior work in HRL including HAMs, options, MAXQ, and PHAMs has been limited to the *discrete-time discounted reward semi-Markov decision process* (SMDP) model. The average reward optimality criterion has been recognized to be more appropriate for a wide class of continuing tasks than the discounted framework. Although average reward RL has been studied for decades, prior work has been largely limited to flat policy representations.

In this paper, we develop a framework for HRL based on the *average reward* optimality criterion. We investigate two formulations of HRL based on the average reward SMDP model, both for discrete-time and continuous-time. These formulations correspond to two notions of optimality that have been previously explored in HRL: *hierarchical optimality* and *recursive optimality*. We present algorithms that learn to find hierarchically and recursively optimal average reward policies under discrete-time and continuous-time average reward SMDP models.

We use two automated guided vehicle (AGV) scheduling tasks as experimental testbeds to study the empirical performance of the proposed algorithms. The first problem is a relatively simple AGV scheduling task, in which the hierarchically and recursively optimal policies are different. We compare the proposed algorithms with three other HRL methods, including a hierarchically optimal discounted reward algorithm and a recursively optimal discounted reward algorithm on this problem. The second problem is a larger AGV scheduling task. We model this problem using both discrete-time and continuous-time models. We use a hierarchical task decomposition in which the hierarchically and recursively optimal policies are the same for this problem. We compare the performance of the proposed algorithms with a hierarchically optimal discounted reward algorithm and a recursively optimal discounted reward algorithm, as well as a non-hierarchical average reward algorithm. The results show that the proposed hierarchical average reward algorithms converge to the same performance as their discounted reward counterparts.

Keywords: semi-Markov decision processes, hierarchical reinforcement learning, average reward reinforcement learning, hierarchical and recursive optimality

1. Introduction

Sequential decision making under uncertainty is a fundamental problem in artificial intelligence (AI). Many sequential decision making problems can be modeled using the Markov decision process (MDP) formalism. A MDP (Howard, 1960; Puterman, 1994) models a system that we are interested in controlling as being in some state at each time step. As a result of actions, the system moves through some sequence of states and receives a sequence of rewards. The goal is to select actions to maximize (minimize) some measure of long-term reward (cost), such as the expected discounted sum of rewards (costs), or the expected average reward (cost).

Reinforcement learning (RL) is a machine learning framework for solving sequential decision-making problems. Despite its successes in a number of different domains, including backgammon (Tesauro, 1994), job-shop scheduling (Zhang and Dietterich, 1995), dynamic channel allocation (Singh and Bertsekas, 1996), elevator scheduling (Crites and Barto, 1998), and helicopter flight control (Ng et al., 2004), current RL methods do not scale well to high dimensional domains—they can be slow to converge and require many training samples to be practical for many real-world problems. This issue is known as the *curse of dimensionality*: the exponential growth of the number of parameters to be learned with the size of any compact encoding of system state (Bellman, 1957). Recent attempts to combat the curse of dimensionality have turned to principled ways of exploiting abstraction in RL. This leads naturally to hierarchical control architectures and associated learning algorithms.

Hierarchical reinforcement learning (HRL) is a general framework for scaling RL to problems with large state spaces by using the task (or action) structure to restrict the space of policies. Hierarchical decision making represents policies over fully or partially specified temporally extended actions. Policies over temporally extended actions cannot be simply treated as single-step actions over a coarser time scale, and therefore cannot be represented in the MDP framework since actions take variable durations of time. Semi-Markov decision process (SMDP) (Howard, 1971; Puterman, 1994) is a well-known statistical framework for modeling temporally extended actions. Action duration in a SMDP can depend on the transition that is made. The state of the system may change continually between actions, unlike MDPs where state changes are only due to actions. Therefore, SMDPs have become the main mathematical model underlying HRL methods.

Prior work in HRL including hierarchies of abstract machines (HAMs) (Parr, 1998), options (Sutton et al., 1999; Precup, 2000), MAXQ (Dietterich, 2000), and programmable HAMs (PHAMs) (Andre and Russell, 2001; Andre, 2003) has been limited to the discrete-time discounted reward SMDP model. In these methods, policies are learned that maximize the long-term discounted sum of rewards. On the other hand, the average reward optimality criterion has been shown to be more appropriate for a wide class of *continuing* tasks than the well-studied discounted framework. A primary goal of continuing tasks, including manufacturing, scheduling, queuing, and inventory control, is to find a *gain-optimal policy* that maximizes (minimizes) the long-run average reward (cost) over time. Although average reward RL has been studied using both the discrete-time MDP model (Schwartz, 1993; Mahadevan, 1996; Tadepalli and Ok, 1996a,b, 1998; Marbach, 1998; Van-Roy, 1998) as well as the continuous-time SMDP model (Mahadevan et al., 1997b; Wang and Mahadevan, 1999), prior work has been limited to *flat* policy representations. In addition to being an appropriate optimality criterion for continuing tasks, average reward optimality allows for more efficient state abstraction in HRL than discounted reward optimality, as will be discussed in Section 5.

In this paper, we extend previous work on HRL to the average reward setting, and investigate two formulations of HRL based on average reward SMDPs. These two formulations correspond to two notions of optimality in HRL: *hierarchical optimality* and *recursive optimality* (Dietterich, 2000). We extend the MAXQ hierarchical RL method (Dietterich, 2000) and introduce a HRL framework for simultaneous learning of policies at multiple levels of a task hierarchy. We then use this HRL framework to derive discrete-time and continuous-time algorithms that learn to find hierarchically and recursively optimal average reward policies. In these algorithms, we assume that the overall task (the *root* of the hierarchy) is continuing. **Hierarchically optimal average reward RL** (HAR) algorithms find a hierarchical policy within the space of policies defined by the hierarchical decomposition that maximizes the *global gain*. **Recursively optimal average reward RL** (RAR) algorithms treat non-primitive subtasks as continuing average reward problems, where the goal at each subtask is to maximize its gain given the policies of its children. We investigate the conditions under which the policy learned by RAR algorithm at each subtask is independent of the context in which it is executed and therefore can be reused by other hierarchies. We use two automated guided vehicle (AGV) scheduling tasks as experimental testbeds to study the empirical performance of the proposed algorithms. The first problem is a relatively simple AGV scheduling task, in which the hierarchically and recursively optimal policies are different. We compare the proposed algorithms with three other HRL methods, including a hierarchically optimal discounted reward algorithm and a recursively optimal discounted reward algorithm on this problem. The second problem is a relatively larger AGV scheduling task. We model this problem using both discrete-time and continuous-time models. We use a hierarchical task decomposition where the hierarchically and recursively optimal policies are the same. We compare the performance of the proposed algorithms with a hierarchically optimal discounted reward algorithm and a recursively optimal discounted reward algorithm, as well as a non-hierarchical average reward algorithm. The results show that the proposed hierarchical average reward algorithms converge to the same performance as their discounted reward counterparts.

The rest of this paper is organized as follows. Section 2 provides a brief overview of HRL. In Section 3, we concisely describe discrete-time SMDPs, and discuss average reward optimality in this model. Section 4 describes the HRL framework, which is used to develop the algorithms of this paper. In Section 5, we extend the previous work on HRL to the average reward setting, and study two formulations of HRL based on the average reward SMDP model. In Section 5.1, we present discrete-time and continuous-time hierarchically optimal average reward RL (HAR) algorithms. In Section 5.2, we investigate different methods to formulate subtasks in a recursively optimal hierarchical average reward RL setting, and present discrete-time and continuous-time recursively optimal average reward RL (RAR) algorithms. We demonstrate the type of optimality achieved by HAR and RAR algorithms as well as their empirical performance and convergence speed compared to other algorithms using two AGV scheduling problems in Section 6. Section 7 summarizes the paper and discusses some directions for future work. For convenience, a table of the symbols used in this paper is given in Appendix A.

2. An Overview of Hierarchical Reinforcement Learning

Hierarchical reinforcement learning (HRL) is a class of learning algorithms that share a common approach to scaling up reinforcement learning (RL). The key principle underlying HRL is to develop learning algorithms that do not need to learn policies from scratch, but instead reuse existing

policies for simpler subtasks. Subtasks form the basis of hierarchical specifications of action sequences because they can include other subtasks in their definitions. It is similar to the familiar idea of subroutines from programming languages. A subroutine can call other subroutines as well as execute primitive commands. A subtask as an open-loop control policy is inappropriate for most interesting control purposes, especially the control of stochastic systems. HRL methods generalize the subtask idea to closed-loop policies or, more precisely, closed-loop partial policies because they are generally defined for a subset of the state space. The partial policies must also have well-defined termination conditions. The partial policies with well-defined termination conditions are sometimes called temporally extended actions. Work in HRL has followed three trends: focusing on subsets of the state space in a divide and conquer approach (state space decomposition), grouping sequences or sets of actions together (temporal abstraction), and ignoring differences between states based on the context (state abstraction). Much of the work in HRL falls into several of these categories. Barto and Mahadevan (2003) provide a more detailed introduction to HRL.

Mahadevan and Connell (1992) were among the first to systematically investigate the use of task structure to accelerate RL. In their work, the robot was given a pre-specified task decomposition, and learned a set of local policies instead of an entire global policy. Singh (1992) investigated reinforcement learning using abstract actions of different temporal granularity using a hierarchy of models with variable temporal resolution. Singh applied the mixture of experts framework as a special-purpose task selection architecture to switch between abstract actions. Kaelbling (1993a,b) investigated using subgoals to learn sub-policies. Dayan and Hinton (1993) describe Feudal RL, a hierarchical technique which uses both temporal abstraction and state abstraction to recursively partition the state space and the time scale from one level to the next.

One key limitation of all the above methods is that decisions in HRL are no longer made at synchronous time steps, as is traditionally assumed in RL. Instead, agents make decisions intermittently, where each epoch can be of variable length, such as when a distinguishing state is reached (e.g., an intersection in a robot navigation task), or a subtask is completed (e.g., the elevator arrives on the first floor). Fortunately, a well-known statistical model is available to treat variable length actions: the semi-Markov decision process (SMDP) model (Howard, 1971; Puterman, 1994). In a SMDP, state-transition dynamics is specified not only by the state where an action was taken, but also by parameters specifying the length of time since the action was taken. Early work on the SMDP model extended algorithms such as Q-learning to continuous-time (Bradtke and Duff, 1995; Mahadevan et al., 1997b). The early work on SMDP was then expanded to include hierarchical task models over fully or partially specified lower level subtasks, which led to developing powerful HRL models such as hierarchies of abstract machines (HAMs) (Parr, 1998), options (Sutton et al., 1999; Precup, 2000), MAXQ (Dietterich, 2000), and programmable HAMs (PHAMs) (Andre and Russell, 2001; Andre, 2003).

In the options framework policies are defined over not just primitive actions, but over fully specified lower-level policies. In the HAMs formulation, hierarchical learning could be achieved even when the policies for lower-level subtasks were only partially specified. The MAXQ model is one of the first methods to combine temporal abstraction with state abstraction. It provides a more comprehensive framework for hierarchical learning where instead of policies for subtasks, the learner is given pseudo-reward functions. Unlike options and HAMs, MAXQ does not rely directly on reducing the entire problem to a single SMDP. Instead, a hierarchy of SMDPs is created whose solutions can be learned simultaneously. The key feature of MAXQ is the decomposed representation of the value function. The MAXQ method views each subtask as a separate SMDP,

and thus represents the value of a state within that SMDP as composed of the reward for taking an action at that state (which might be composed of many rewards along a trajectory through a subtask) and the expected reward for completing the subtask. To isolate the subtask from the calling context, MAXQ uses the notion of a pseudo-reward. At the terminal states of a subtask, the agent is rewarded according to the pseudo-reward, which is set a priori by the designer, and does not depend on what happens after leaving the current subtask. Each subtask can then be treated in isolation from the rest of the problem with the caveat that the solutions learned are only recursively optimal. Each action in the recursively optimal policy is optimal with respect to the subtask containing the action, all descendant subtasks, and the pseudo-reward chosen by the designer of the system. Another important contribution of MAXQ is the idea that state abstraction can be done separately on the different components of the value function, which allows one to perform *dynamic abstraction*. We describe the MAXQ framework and related concepts such as recursive optimality and value function decomposition in Section 4. In the PHAM model, Andre and Russell extended HAMs and presented an agent-design language for RL. Andre and Russell (2002) also addressed the issue of safe state abstraction in their model. Their method yields state abstraction while maintaining hierarchical optimality.

3. Discrete-time Semi-Markov Decision Processes

Semi-Markov decision processes (SMDPs) (Howard, 1971; Puterman, 1994) extend the Markov decision process (MDP) (Howard, 1971; Puterman, 1994) model by allowing actions that can take multiple time steps to complete. Note that SMDPs do not theoretically provide additional expressive power but they do provide a convenient formalism for temporal abstraction. The duration of an action can depend on the transition that is made.[1] The state of the system may change continually between actions unlike MDPs where state changes are only due to actions. Thus, SMDPs have become the preferred language for modeling temporally extended actions (Mahadevan et al., 1997a) and, as a result, the main mathematical model underlying hierarchical reinforcement learning (HRL).

A SMDP is defined as a five tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, I \rangle$. All components are defined as in a MDP except the transition probability function $\mathcal{P}$ and the reward function $\mathcal{R}$. $\mathcal{S}$ is the set of states of the world, $\mathcal{A}$ is the set of possible actions from which the agent may choose on at each decision epoch, and $I : \mathcal{S} \rightarrow [0,1]$ is the initial state distribution. The transition probability function $\mathcal{P}$ now takes the duration of the actions into account. The transition probability function $\mathcal{P} : \mathcal{S} \times \mathbb{N} \times \mathcal{S} \times \mathcal{A} \rightarrow [0,1]$ is a multi-step transition probability function ($\mathbb{N}$ is the set of natural numbers), where $P(s', N|s, a)$ denotes the probability that action a will cause the system to transition from state s to state s' in N time steps. This transition is at decision epochs only. Basically, the SMDP model represents snapshots of the system at decision points, whereas the so-called **natural process** describes the evolution of the system over all times. If we marginalize $P(s', N|s, a)$ over N, we will obtain $F(s'|s, a)$ the transition probability for the embedded MDP. The term $F(s'|s, a)$ denotes the probability that the system occupies state s' at the next decision epoch, given that the decision maker chooses action a in state s at the current decision epoch. The key difference in the reward function for SMDPs is that the rewards can accumulate over the entire duration of an action. As a result, the reward in a SMDP for taking an action in a state depends on the evolution of the system during the execution of the action. Formally, the reward in a SMDP is modeled as a function $\mathcal{R} : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ ($\mathbb{R}$

1. Continuous-time SMDPs typically allow arbitrary continuous action durations.

is the set of real numbers), with $r(s,a)$ representing the expected total reward between two decision epochs, given that the system occupies state s at the first decision epoch and the agent chooses action a. This expected reward contains all necessary information about the reward to analyze the SMDP model. For each transition in a SMDP, the expected number of time steps until the next decision epoch is defined as

$$y(s,a) = \mathbf{E}[N|s,a] = \sum_{N\in\mathbb{N}} N \sum_{s'\in\mathcal{S}} P(s',N|s,a).$$

The notion of policy and the various forms of optimality are the same for SMDPs as for MDPs. In infinite-horizon SMDPs, the goal is to find a policy that maximizes either the expected discounted reward or the average expected reward. We discuss the average reward optimality criterion for the SMDP model in the next section.

3.1 Average Reward Semi-Markov Decision Processes

The theory of infinite-horizon SMDPs with the average reward optimality criterion is more complex than that for discounted models (Howard, 1971; Puterman, 1994). The aim of average reward SMDP algorithms is to compute policies that yield the highest average reward or gain. The average reward or gain of a policy μ at state s, $g^\mu(s)$, can be defined as the ratio of the expected total reward and the expected total number of time steps of following policy μ starting at state s

$$g^\mu(s) = \liminf_{n\to\infty} \frac{\mathbf{E}\left[\sum_{t=0}^{n-1} r(s_t,\mu(s_t))|s_0=s,\mu\right]}{\mathbf{E}\left[\sum_{t=0}^{n-1} N_t|s_0=s,\mu\right]}.$$

In this equation, N_t is the total number of time steps until the next decision epoch, when agent takes action $\mu(s_t)$ in state s_t.

A key observation that greatly simplifies the design of average reward algorithms is that for *unichain* SMDPs,[2] the gain of any policy is state independent, that is

$$g^\mu(s) = g^\mu = \liminf_{n\to\infty} \frac{\mathbf{E}\left[\sum_{t=0}^{n-1} r(s_t,\mu(s_t))|\mu\right]}{\mathbf{E}\left[\sum_{t=0}^{n-1} N_t|\mu\right]}, \qquad \forall s\in\mathcal{S}. \tag{1}$$

To simplify exposition, we consider only *unichain* SMDPs in this paper. When the state space of a SMDP, $\mathcal{S}$, is finite or countable, Equation 1 can be written as

$$g^\mu = \frac{\bar{F}^\mu r^\mu}{\bar{F}^\mu y^\mu}, \tag{2}$$

where F^μ and $\bar{F}^\mu = \lim_{n\to\infty} \frac{1}{n}\sum_{t=0}^{n-1}(F^\mu)^t$ are the transition probability matrix and the *limiting matrix* of the embedded Markov chain for policy μ, respectively,[3] and r^μ and y^μ are vectors with elements $r(s,\mu(s))$ and $y(s,\mu(s))$, for all $s\in\mathcal{S}$. Under the *unichain* assumption, $\bar{F}$ has equal rows, and therefore the right hand side of Equation 2 is a vector with elements all equal to g^μ.

2. In unichain SMDPs, the underlying Markov chain for every stationary policy has a single recurrent class, and a (possibly empty) set of transient states.
3. The *limiting matrix* $\bar{F}$ satisfies the equality $\bar{F}F = \bar{F}$.

In the average reward SMDP model, a policy μ is measured using a different value function, namely the average-adjusted sum of rewards earned following that policy[4]

$$H^\mu(s) = \lim_{n\to\infty} \mathbf{E}\left\{\sum_{t=0}^{n-1} [r(s_t,\mu(s_t)) - g^\mu y(s_t,\mu(s_t))] \,|s_0 = s,\mu\right\}.$$

The term H^μ is usually referred to as the **average-adjusted value function**. Furthermore, the average-adjusted value function satisfies the Bellman equation

$$H^\mu(s) = r(s,\mu(s)) - g^\mu y(s,\mu(s)) + \sum_{s'\in\mathcal{S},N\in\mathbb{N}} P(s',N|s,\mu(s))H^\mu(s').$$

Similarly, the **average-adjusted action-value function** for a policy μ, L^μ, is defined, and it satisfies the Bellman equation

$$L^\mu(s,a) = r(s,a) - g^\mu y(s,a) + \sum_{s'\in\mathcal{S},N\in\mathbb{N}} P(s',N|s,a)L^\mu(s',\mu(s')).$$

4. A Framework for Hierarchical Reinforcement Learning

In this section, we describe a general hierarchical reinforcement learning (HRL) framework for simultaneous learning of policies at multiple levels of a hierarchy. Our treatment builds upon existing methods, including HAMs (Parr, 1998), options (Sutton et al., 1999; Precup, 2000), MAXQ (Dietterich, 2000), and PHAMs (Andre and Russell, 2002; Andre, 2003), and, in particular, uses the MAXQ value function decomposition. We extend the MAXQ framework by including the three-part value function decomposition (Andre and Russell, 2002) to guarantee hierarchical optimality, as well as reward shaping (Ng et al., 1999) to reduce the burden of exploration. Rather than redundantly explain MAXQ and then our hierarchical framework, we will present our model and note throughout this section where the key pieces were inspired by or are directly related to MAXQ. In the next section, we will extend this framework to the average reward model and present our hierarchical average reward reinforcement learning algorithms.

4.1 Motivating Example

In the HRL framework, the designer of the system imposes a hierarchy on the problem to incorporate domain knowledge and thereby reduces the size of the space that must be searched to find a good policy. The designer recursively decomposes the overall task into a collection of subtasks that are important for solving the problem.

Let us illustrate the main ideas using a simple search task shown in Figure 1. Consider the domain of an office-type environment (with rooms and connecting corridors), where a robot is assigned the task of picking up trash from trash cans ($T1$ and $T2$) over an extended area and accumulating it into one centralized trash bin (*Dump*). For simplicity, we assume that the robot can observe its true location in the environment. The main subtasks in this problem are *root* (the whole trash-collection task), *collect trash at* $T1$ and $T2$, *navigate to* $T1$, $T2$, and *Dump*. Each of these subtasks is defined by a set of termination states. After defining subtasks, we must indicate, for each subtask, which

4. This limit assumes that all policies are aperiodic. For periodic policies, it changes to the Cesaro limit $H^\mu(s) = \lim_{n\to\infty} \frac{1}{n}\sum_{k=0}^{n-1} \mathbf{E}\left\{\sum_{t=0}^{k} [r(s_t,\mu(s_t)) - g^\mu y(s_t,\mu(s_t))] \,|s_0 = s,\mu\right\}$ (Puterman, 1994).

other subtasks or primitive actions it should employ to reach its goal. For example, *navigate to* $T1$, $T2$, and *Dump* use three primitive actions *find wall*, *align with wall*, and *follow wall*. *Collect trash at* $T1$ uses two subtasks, *navigate to* $T1$ and *Dump*, plus two primitive actions *Put* and *Pick*, and so on. Similar to MAXQ, all of this information can be summarized by a directed acyclic graph called **task graph**. The task graph for the trash-collection problem is shown in Figure 1. This hierarchical model is able to support **state abstraction** (while the agent is moving toward the *Dump*, the status of trash cans $T1$ and $T2$ is irrelevant and cannot affect this navigation process. Therefore, the variables defining the status of trash cans $T1$ and $T2$ can be removed from the state space of the *navigate to Dump* subtask), and **subtask sharing** (if the system could learn how to solve the *navigate to Dump* subtask once, then the solution could be shared by both *collect trash at* $T1$ and $T2$ subtasks.)

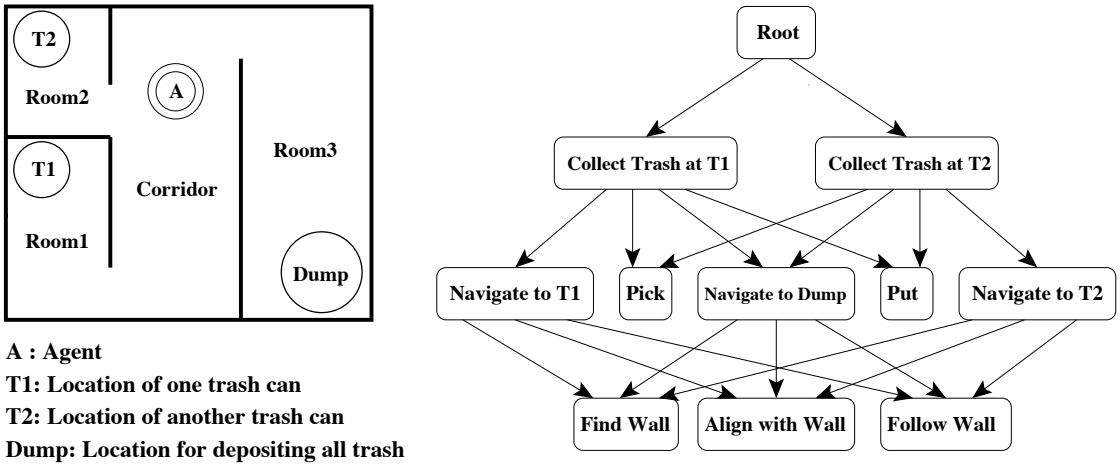


Figure 1: A robot trash-collection task and its associated task graph.

Like HAMs (Parr, 1998), options (Sutton et al., 1999; Precup, 2000), MAXQ (Dietterich, 2000), and PHAMs (Andre and Russell, 2001; Andre, 2003), this framework also relies on the theory of SMDPs. While SMDP theory provides the theoretical underpinnings of temporal abstraction by modeling actions that take varying amounts of time, the SMDP model provides little in the way of concrete representational guidance, which is critical from a computational point of view. In particular, the SMDP model does not specify how tasks can be broken up into subtasks, how to decompose value functions etc. We examine these issues next.

As in MAXQ, a task hierarchy such as the one illustrated above can be modeled by decomposing the overall task MDP $\mathcal{M}$, into a finite set of subtasks $\{M_0, M_1, \ldots, M_{m-1}\}$,[5] where M_0 is the *root* task. Solving M_0 solves the entire MDP $\mathcal{M}$.

Definition 1: Each **non-primitive** subtask M_i (M_i is not a primitive action) consists of five components $\langle S_i, I_i, T_i, A_i, R_i \rangle$:

5. m is the total number of subtasks in the hierarchy.

- S_i is the **state space** for subtask M_i. It is described by those state variables that are relevant to subtask M_i. The range of a state variable describing S_i might be a subset of its range in $\mathcal{S}$ (the state space of MDP $\mathcal{M}$).

- $I_i \subseteq S_i$ is the **initiation set** for subtask M_i. Subtask M_i can be initiated only in states belonging to I_i.

- $T_i \subseteq S_i$ is the **set of terminal states** for subtask M_i. Subtask M_i terminates when it reaches a state in T_i. A policy for subtask M_i can only be executed if the current state s belongs to $(S_i - T_i)$.

- A_i is the **set of actions** that can be performed to achieve subtask M_i. These actions can be either primitive actions from $\mathcal{A}$ (the set of primitive actions for MDP $\mathcal{M}$), or they can be other subtasks. Technically, A_i is a function of states, since it may differ from one state to another. However, we will suppress this dependence in our notation.

- R_i is the **reward structure** inside subtask M_i and could be different from the reward function of MDP $\mathcal{M}$. Here, we use the idea of reward shaping (Ng et al., 1999) and define a more general reward structure than MAXQ's. Reward shaping is a method for guiding an agent toward a solution without constraining the search space. Besides the reward of the overall task MDP $\mathcal{M}$, each subtask M_i can use additional rewards to guide its local learning. Additional rewards are only used inside each subtask and do not propagate to upper levels in the hierarchy. If the reward structure inside a subtask is different from the reward function of the overall task, we need to define two types of value functions for each subtask, internal value function and external value function. **Internal value function** is defined based on both the local reward structure of the subtask and the reward of the overall task, and only is used in learning the subtask. On the other hand, **external value function** is defined only based on the reward function of the overall task and is propagated to the higher levels in the hierarchy to be used in learning the global policy. This reward structure for each subtask in our framework is more general than the one in MAXQ, and of course, includes MAXQ's pseudo-reward.[6] □

Each primitive action a is a primitive subtask in this decomposition, such that a is always executable and it terminates immediately after execution. From now on in this paper, we use subtask to refer to non-primitive subtasks.

4.2 Policy Execution

If we have a policy for each subtask in a hierarchy, we can define a **hierarchical policy** for the model.

Definition 2: A hierarchical policy μ is a set of policies, one policy for each subtask in the hierarchy: $\mu = \{\mu_0, \ldots, \mu_{m-1}\}$. □

6. The MAXQ pseudo-reward function is defined only for transitions to terminal states, and is zero for non-terminal states.

The hierarchical policy is executed using a stack discipline, similar to ordinary programming languages. Each subtask policy takes a state and returns the name of a primitive action to execute or the name of a subtask to invoke. When a subtask is invoked, its name is pushed onto the **Task-Stack** and its policy is executed until it enters one of its terminal states. When a subtask terminates, its name is popped off the Task-Stack. If any subtask on the Task-Stack terminates, then all subtasks below it are immediately aborted, and control returns to the subtask that had invoked the terminated subtask. Hence, at any time, the *root* task is located at the bottom and the subtask which is currently being executed is located at the top of the Task-Stack.

Under a hierarchical policy μ, we define a multi-step transition probability $P_i^\mu : S_i \times \mathbb{N} \times S_i \rightarrow [0,1]$ for each subtask M_i in the hierarchy, where $P_i^\mu(s',N|s)$ denotes the probability that hierarchical policy μ will cause the system to transition from state s to state s' in N primitive steps at subtask M_i. We also define a multi-step abstract transition probability $F_i^\mu : S_i \times \mathbb{N} \times S_i \rightarrow [0,1]$ for each subtask M_i under the hierarchical policy μ. The term $F_i^\mu(s',N|s)$ denotes the N-step abstract transition probability from state s to state s' under hierarchical policy μ at subtask M_i, where N is the number of actions taken by subtask M_i, not the number of primitive actions taken in this transition. In this paper, we use the multi-step abstract transition probability F_i^μ to model state transition at the subtask level, and the multi-step transition probability P_i^μ to model state transition at the level of primitive actions. For $N = 1$, $F_i^\mu(s',1|s)$ is the transition probability for the embedded MDP at subtask M_i. We can write $F_i^\mu(s',1|s)$ as $F_i^\mu(s'|s)$, and it can also be obtained by marginalizing $P_i^\mu(s',N|s)$ over N as described in Section 3.

4.3 Local versus Global Optimality

Using a hierarchy reduces the size of the space that must be searched to find a good policy. However, a hierarchy constrains the space of possible policies so that it may not be possible to represent the optimal policy or its value function, and hence make it impossible to learn the optimal policy. If we cannot learn the optimal policy, the next best target would be to learn the best policy that is consistent with the given hierarchy. Two notions of optimality have been explored in the previous work on hierarchical reinforcement learning, **hierarchical optimality** and **recursive optimality** (Dietterich, 2000).

Definition 3: A hierarchically optimal policy for MDP $\mathcal{M}$ is a hierarchical policy which has the best performance among all policies consistent with the given hierarchy. In other words, hierarchical optimality is a global optimum consistent with the given hierarchy. In this form of optimality, the policy for each individual subtask is not necessarily locally optimal, but the policy for the entire hierarchy is optimal. The HAMQ HRL algorithm (Parr, 1998) and the SMDP Q-learning algorithm for a fixed set of options (Sutton et al., 1999; Precup, 2000) both converge to a hierarchically optimal policy. □

Definition 4: Recursive optimality is a weaker but more flexible form of optimality which only guarantees that the policy of each subtask is optimal given the policies of its children. It is an important and flexible form of optimality because it permits each subtask to learn a locally optimal policy while ignoring the behavior of its ancestors in the hierarchy. This increases the opportunity for subtask sharing and state abstraction. The MAXQ-Q HRL algorithm (Dietterich, 2000) con-

verges to a recursively optimal policy. □

4.4 Value Function Definitions

For recursive optimality, the goal is to find a hierarchical policy $\mu = \{\mu_0, \ldots, \mu_{m-1}\}$ such that for each subtask M_i in the hierarchy, the expected cumulative reward of executing policy μ_i and the policies of all descendants of M_i is maximized. In this case, the value function to be learned for subtask M_i under hierarchical policy μ must contain only the reward received during the execution of subtask M_i. We call this the **projected value function** after Dietterich (2000), and define it as follows:

Definition 5: The projected value function of a hierarchical policy μ on subtask M_i, denoted $\hat{V}^{\mu}(i,s)$, is the expected cumulative reward of executing policy μ_i and the policies of all descendants of M_i starting in state $s \in S_i$ until M_i terminates. □

The expected cumulative reward outside a subtask is not a part of its projected value function. It makes the projected value function of a subtask dependent only on the subtask and its descendants.

On the other hand, for hierarchical optimality, the goal is to find a hierarchical policy that maximizes the expected cumulative reward. In this case, the value function to be learned for subtask M_i under hierarchical policy μ must contain the reward received during the execution of subtask M_i, and the reward after subtask M_i terminates. We call this the **hierarchical value function**, following Dietterich (2000). The hierarchical value function of a subtask includes the expected reward outside the subtask and therefore depends on the subtask and all its ancestors up to the root of the hierarchy. In the case of hierarchical optimality, we need to consider the contents of the Task-Stack as an additional part of the state space of the problem, since a subtask might be shared by multiple parents.

Definition 6: Ω is the space of possible values of the Task-Stack for hierarchy $\mathcal{H}$. □

Let us define joint state space $\mathcal{X} = \Omega \times \mathcal{S}$ for the hierarchy $\mathcal{H}$ as the cross product of the set of the Task-Stack values Ω and the state space $\mathcal{S}$. We also define a transition probability function of the Markov chain that results from flattening the hierarchy using the hierarchical policy μ, $m^{\mu} : \mathcal{X} \times \mathcal{X} \rightarrow [0,1]$, where $m^{\mu}(x'|x)$ denotes the probability that the hierarchical policy μ will cause the system to transition from state $x = (\omega, s)$ to state $x' = (\omega', s')$ at the level of primitive actions. We will use this transition probability function in Section 5.1 to define *global gain* for a hierarchical policy. Finally, we define the hierarchical value function using the joint state space $\mathcal{X}$ as follows:

Definition 7: A hierarchical value function for subtask M_i in state $x = (\omega, s)$ under hierarchical policy μ, denoted $V^{\mu}(i,x)$, is the expected cumulative reward of following the hierarchical policy μ starting in state $s \in S_i$ and Task-Stack ω. □

The current subtask M_i is a part of the Task-Stack ω and as a result is a part of the state x. So we can exclude it from the hierarchical value function notation and write $V^{\mu}(i,x)$ as $V^{\mu}(x)$. However for clarity, we use $V^{\mu}(i,x)$ in the rest of this paper.

Theorem 1: Under a hierarchical policy μ, each subtask M_i can be modeled by a SMDP consisting of components $(S_i, A_i, P_i^\mu, \bar{R}_i)$, where $\bar{R}_i(s,a) = \hat{V}^\mu(a,s)$ for all $a \in \mathcal{A}_i$. □

This theorem is similar to Theorem 1 in Dietterich (2000). Using this theorem, we can define a recursive optimal policy for MDP $\mathcal{M}$ with hierarchical decomposition $\{M_0, M_1, \ldots, M_{m-1}\}$ as a hierarchical policy $\mu = \{\mu_0, \ldots, \mu_{m-1}\}$ such that for each subtask M_i, the corresponding policy μ_i is optimal for the SMDP defined by the tuple $(S_i, A_i, P_i^\mu, \bar{R}_i)$.

4.5 Value Function Decomposition

A value function decomposition splits the value of a state or a state-action pair into multiple additive components. Modularity in the hierarchical structure of a task allows us to carry out this decomposition along subtask boundaries. In this section, we first describe the two-part or MAXQ decomposition proposed by Dietterich (2000), and then the three-part decomposition proposed by Andre and Russell (2002). We use both decompositions in our hierarchical average reward framework depending on the type of optimality (hierarchical or recursive) that we are interested in.

The two-part value function decomposition is at the center of the MAXQ method. The purpose of this decomposition is to decompose the projected value function of the *root* task, $\hat{V}^\mu(0,s)$, in terms of the projected value functions of all the subtasks in the hierarchy. The projected value of subtask M_i at state s under hierarchical policy μ can be written as

$$\hat{V}^\mu(i,s) = \mathbf{E}\left[\sum_{k=0}^{\infty} \gamma^k r(s_k, a_k) | s_0 = s, \mu\right]. \tag{3}$$

Now, let us suppose that the first action chosen by μ_i is invoked and executed for a number of primitive steps N and terminates in state s' according to $P_i^\mu(s', N|s)$. We can rewrite Equation 3 as

$$\hat{V}^\mu(i,s) = \mathbf{E}\left[\sum_{k=0}^{N-1} \gamma^k r(s_k, a_k) + \sum_{k=N}^{\infty} \gamma^k r(s_k, a_k) | s_0 = s, \mu\right]. \tag{4}$$

The first summation on the right-hand side of Equation 4 is the discounted sum of rewards for executing subtask $\mu_i(s)$ starting in state s until it terminates. In other words, it is $\hat{V}^\mu(\mu_i(s), s)$, the projected value function of the child task $\mu_i(s)$. The second term on the right-hand side of the equation is the projected value of state s' for the current task M_i, $\hat{V}^\mu(i,s')$, discounted by γ^N, where s' is the current state when subroutine $\mu_i(s)$ terminates and N is the number of transition steps from state s to state s'. We can therefore write Equation 4 in the form of a Bellman equation:

$$\hat{V}^\mu(i,s) = \hat{V}^\mu(\mu_i(s), s) + \sum_{N, s' \in S_i} P_i^\mu(s', N|s) \gamma^N \hat{V}^\mu(i, s'). \tag{5}$$

Equation 5 can be restated for the projected action-value function as follows:

$$\hat{Q}^\mu(i,s,a) = \hat{V}^\mu(a,s) + \sum_{N, s' \in S_i} P_i^\mu(s', N|s,a) \gamma^N \hat{Q}^\mu(i, s', \mu_i(s')).$$

The right-most term in this equation is the expected discounted cumulative reward of completing subtask M_i after executing action a in state s. Dietterich called this term **completion function** and denoted it by

$$C^\mu(i,s,a) = \sum_{N, s' \in S_i} P_i^\mu(s', N|s,a) \gamma^N \hat{Q}^\mu(i, s', \mu_i(s')). \tag{6}$$

With this definition, we can express the projected action-value function recursively as

$$\hat{Q}^{\mu}(i,s,a) = \hat{V}^{\mu}(a,s) + C^{\mu}(i,s,a), \tag{7}$$

and we can rewrite the definition for projected value function as

$$\hat{V}^{\mu}(i,s) = \begin{cases} \hat{Q}^{\mu}(i,s,\mu_i(s)) & \text{if } M_i \text{ is a non-primitive subtask,} \\ r(s,i) & \text{if } M_i \text{ is a primitive action.} \end{cases} \tag{8}$$

Equations 6 to 8 are referred to as two-part value function decomposition equations for a hierarchy under a hierarchical policy μ. These equations recursively decompose the projected value function for the *root* into the projected value functions for the individual subtasks, $M_1, \ldots, M_{m-1}$, and the individual completion functions $C^{\mu}(j,s,a)$, $j = 1, \ldots, m-1$. The fundamental quantities that must be stored to represent this value function decomposition are the C values for all non-primitive subtasks and the V values for all primitive actions.[7] The two-part value function decomposition is summarized graphically in Figure 2. As mentioned in Section 4.4, since the expected reward after execution of subtask M_i is not a component of the projected action-value function, the two-part value function decomposition allows only for recursive optimality.

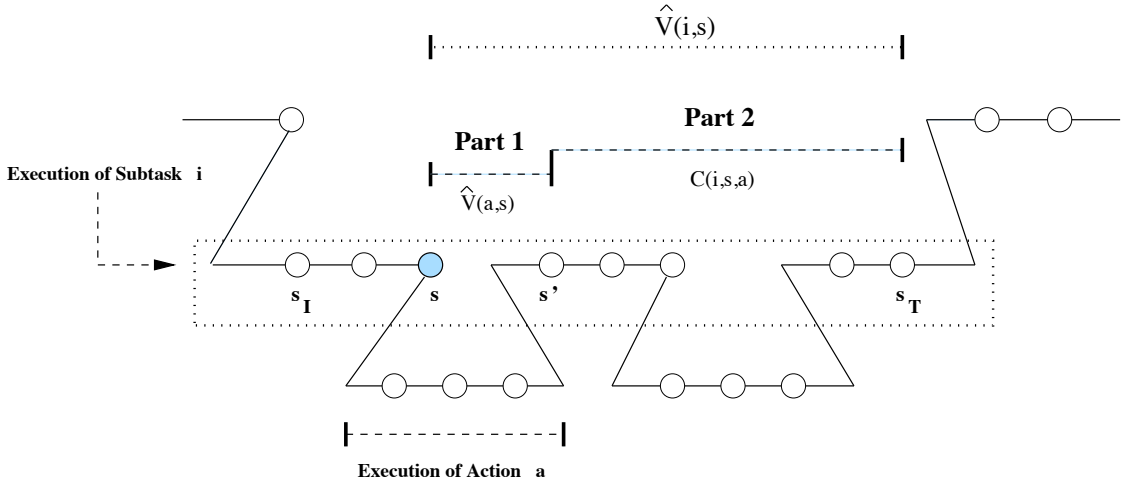


Figure 2: This figure shows the two-part decomposition for $\hat{V}(i,s)$, the projected value function of subtask M_i for the shaded state s. Each circle is a state of the SMDP visited by the agent. Subtask M_i is initiated at state s_I and terminates at state s_T. The projected value function $\hat{V}(i,s)$ is broken into two parts: **Part 1)** the projected value function of subtask M_a for state s, and **Part 2)** the completion function, the expected discounted cumulative reward of completing subtask M_i after executing action a in state s.

Andre and Russell (2002) proposed a three-part value function decomposition to achieve hierarchical optimality. They added a third component for the expected sum of rewards outside the current subtask to the two-part value function decomposition. This decomposition decomposes the hierarchical value function of each subtask into three parts. As shown in Figure 3, these three parts

7. The projected value function and value function are the same for a primitive action.

correspond to executing the current action (which might itself be a subtask), completing the rest of the current subtask (so far is similar to the MAXQ decomposition), and all actions outside the current subtask.

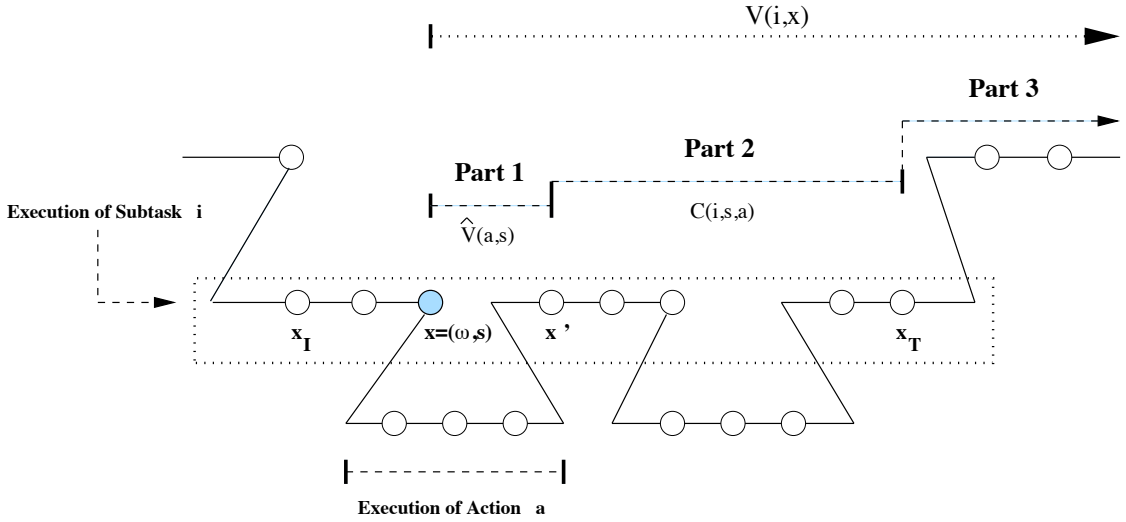


Figure 3: This figure shows the three-part decomposition for $V(i,x)$, the hierarchical value function of subtask M_i for the shaded state $x = (\omega, s)$. Each circle is a state of the SMDP visited by the agent. Subtask M_i is initiated at state x_I and terminates at state x_T. The hierarchical value function $V(i,x)$ is broken into three parts: **Part 1)** the projected value function of subtask M_a for state s, **Part 2)** the completion function, the expected discounted cumulative reward of completing subtask M_i after executing action a in state s, and **Part 3)** the sum of all rewards after termination of subtask M_i.

5. Hierarchical Average Reward Reinforcement Learning

As described in Section 1, the average reward formulation is more appropriate for a wide class of *continuing* tasks including manufacturing, scheduling, queuing, and inventory control than the more well-studied discounted framework. Moreover, average reward optimality allows for more efficient state abstraction in HRL than the discounted reward formulation. Consider the case that a set of state variables $\mathcal{Y}_a$ is irrelevant for the result distribution of action (subtask) M_a, when M_a is executed under subtask M_i. It means that for all policies executed by M_a and its descendants, and for all pairs of states s_1 and s_2 in S_i (the state space of subtask M_i) that differ only in their values for the state variables in $\mathcal{Y}_a$, we have

$$P_i^{\mu}(s', N|s_1, a) = P_i^{\mu}(s', N|s_2, a) \qquad , \qquad \forall s' \in S_i \quad , \quad \forall N \in \mathbb{N}.$$

Dietterich (2000) first defined this condition and called it *result distribution irrelevance*. If this condition is satisfied for subtask M_a, then the completion function values of its parent task M_i can be represented compactly, that is, all states $s \in S_i$ that differ only in their values for the state variables in $\mathcal{Y}_a$ have the same completion function, and therefore their completion function values can be represented only by one quantity $C^{\mu}(i,s,a)$, defined by Equation 6.

The definition of result distribution irrelevance can be weakened to eliminate N, the number of steps. All that is needed is that for all pairs of states s_1 and s_2 in S_i that differ only in the irrelevant state variables, $F_i^\mu(s'|s_1,a) = F_i^\mu(s'|s_2,a)$ for all $s' \in S_i$. Although the result distribution irrelevance condition would rarely be satisfied, we often find cases where the weakened result distribution irrelevance condition is true.

Under this revised definition, the compact representation of a completion function still holds in the undiscounted case, but not in the discounted formulation. Consider, for example, the *collect trash at T1* subtask in the robot trash-collection problem described in Section 4.1. No matter what location the robot has in state s, it will be at the *Dump* location when the *collect trash at T1* subtask terminates. Hence, the starting location is irrelevant to the resulting location of the robot, and $F^\mu_{Root}(s'|s_1, collect\ trash\ at\ T1) = F^\mu_{Root}(s'|s_2, collect\ trash\ at\ T1)$ for all states s_1 and s_2 in S_{Root} that differ only in the robot's location. However, if we were using discounted reward optimality, the robot's location would not be irrelevant, because the probability that the *collect trash at T1* subtask will terminate in N steps would depend on the location of the robot, which could differ in states s_1 and s_2. Different values of N will produce different amounts of discounting in Equation 6, and hence we cannot ignore the robot location when representing the completion function for the *collect trash at T1* subtask. When we use undiscounted optimality, such as average reward, we can use the weakened result distribution irrelevance and still represent the completion function for the *collect trash at T1* subtask with only one quantity.

In this section, we extend previous work on hierarchical reinforcement learning (HRL) to the average reward framework, and investigate two formulations of HRL based on the average reward SMDP model. These two formulations correspond to two notions of optimality in HRL: *hierarchical optimality* and *recursive optimality* described in Section 4.3. We present discrete-time and continuous-time algorithms to find hierarchically and recursively optimal average reward policies. In these algorithms, we assume that the overall task (the *root* of the hierarchy) is continuing. In the **hierarchically optimal average reward RL** (HAR) algorithms, the aim is to find a hierarchical policy within the space of policies defined by the hierarchical decomposition that maximizes the *global gain* (Ghavamzadeh and Mahadevan, 2002). In the **recursively optimal average reward RL** (RAR) algorithms, we treat subtasks as continuing average reward problems, where the goal at each subtask is to maximize its gain given the policies of its children (Ghavamzadeh and Mahadevan, 2001). We investigate the conditions under which the policy learned by the RAR algorithm at each subtask is independent of the context in which it is executed and therefore can be reused by other hierarchies. In Section 6, we use two automated guided vehicle (AGV) scheduling tasks as experimental testbeds to study the empirical performance of the proposed algorithms. We model the second AGV task using both discrete-time and continuous-time models. We compare the performance of our proposed algorithms with other HRL methods and a non-hierarchical average reward RL algorithm in this problem.

5.1 Hierarchically Optimal Average Reward RL Algorithm

Given the basic concepts of the average reward SMDP model described in Section 3.1, the fundamental principles of HRL, and the HRL framework in Section 4, we now describe a hierarchically optimal average reward RL formulation. Since we are interested in hierarchical optimality, we include the contents of the Task-Stack as a part of the state space of the problem. In this section, we

consider HRL problems for which the following assumptions hold.

Assumption 1 (Continuing Root Task): The *root* of the hierarchy is a *continuing* task, that is, the root task continues without termination. □

Assumption 2: For every hierarchical policy μ, the Markov chain that results from flattening the hierarchy using the hierarchical policy μ, represented by the transition probability matrix m^μ (defined in Section 4.4), has a single recurrent class and a (possibly empty) set of transient states. □

If Assumptions 1 and 2 hold, the gain[8]

$$g^\mu = \left(\lim_{n\to\infty} \frac{1}{n} \sum_{t=0}^{n-1} (m^\mu)^t \right) r^\mu = \overline{m}^\mu r^\mu \tag{9}$$

is well defined for every hierarchical policy μ and does not depend on the initial state. In Equation 9, $\overline{m}^\mu$ is the *limiting matrix* of the Markov chain that results from flattening the hierarchy using the hierarchical policy μ, and satisfies the equality $\overline{m}^\mu m^\mu = \overline{m}^\mu$, and r^μ is a vector with elements $r(x,\mu(x))$, for all $x \in \mathcal{X}$. We call g^μ the **global gain** under the hierarchical policy μ. The *global gain*, g^μ, is the gain of the Markov chain that results from flattening the hierarchy using the hierarchical policy μ.

Here, we are interested in finding a hierarchical policy μ^* that maximizes the *global gain*

$$g^{\mu^*} \geq g^\mu, \qquad \text{for all } \mu. \tag{10}$$

We refer to a hierarchical policy μ^* which satisfies Equation 10 as a *hierarchically optimal average reward* policy, and to g^{μ^*} as the *hierarchically optimal average reward* or the *hierarchically optimal gain*.

We replace the value and the action-value functions in the HRL framework of Section 4 with the average-adjusted value and the average-adjusted action-value functions described in Section 3.1. The hierarchical average-adjusted value function for hierarchical policy μ and subtask M_i, denoted $H^\mu(i,x)$, is the average-adjusted sum of rewards earned by following hierarchical policy μ starting in state $x = (\omega, s)$ until M_i terminates, plus the expected average-adjusted reward outside subtask M_i

$$H^\mu(i,x) = \lim_{N\to\infty} \mathbf{E}\left\{ \sum_{k=0}^{N-1} \left[r^\mu(x_k,a_k) - g^\mu y^\mu(x_k,a_k) \right] | x_0 = x, \mu \right\}. \tag{11}$$

Here, the rewards are adjusted with g^μ, the *global gain* under the hierarchical policy μ.

Now, let us suppose that the first action chosen by μ_i is executed for a number of primitive steps N_1 and terminates in state $x_1 = (\omega, s_1)$ according to multi-step transition probability $P_i^\mu(x_1, N_1 | x, \mu_i(x))$, and then subtask M_i itself executes for N_2 steps at the level of subtask M_i (N_2 is the number of actions taken by subtask M_i, not the number of primitive actions) and terminates in state $x_2 = (\omega, s_2)$ according to multi-step abstract transition probability $F_i^\mu(x_2, N_2 | x_1)$. We can rewrite Equation 11 in the form of a Bellman equation as

8. Under the *unichain* assumption, $\overline{m}^\mu$ has equal rows. Therefore, the right hand side of Equation 9 is a vector with elements all equal to g^μ.

$$H^{\mu}(i,x) = r_i^{\mu}(x,\mu_i(x)) - g^{\mu} y_i^{\mu}(x,\mu_i(x)) + \sum_{N_1,s_1 \in S_i} P_i^{\mu}(x_1,N_1|x,\mu_i(x)) \left[\hat{H}^{\mu}(i,x_1) + \sum_{N_2,s_2 \in S_i} F_i^{\mu}(x_2,N_2|x_1) H^{\mu}(Parent(i),(\omega \nearrow i,s_2)) \right], \tag{12}$$

where $\hat{H}^{\mu}(i,.)$ is the projected average-adjusted value function of the hierarchical policy μ and subtask M_i, $y_i^{\mu}(x,\mu_i(x))$ is the expected number of time steps until the next decision epoch of subtask M_i after taking action $\mu_i(x)$ in state x and following the hierarchical policy μ afterward, and $\omega \nearrow i$ is the content of the Task-Stack after popping subtask M_i off. Notice that $\hat{H}$ does not contain the average-adjusted rewards outside the current subtask and should be distinguished from the hierarchical average-adjusted value function H, which includes the sum of average-adjusted rewards outside the current subtask.

Since $r_i^{\mu}(x,\mu_i(x))$ is the expected reward between two decision epochs of subtask M_i, given that the system occupies state x at the first decision epoch, and the agent chooses action $\mu_i(x)$, we have

$$r_i^{\mu}(x,\mu_i(x)) = \hat{V}^{\mu}(\mu_i(x),(\mu_i(x) \searrow \omega,s)) = \hat{H}^{\mu}(\mu_i(x),(\mu_i(x) \searrow \omega,s)) + g^{\mu} y_i^{\mu}(x,\mu_i(x)),$$

where $\mu_i(x) \searrow \omega$ is the content of the Task-Stack after pushing subtask $\mu_i(x)$ onto it. By replacing $r_i^{\mu}(x,\mu_i(x))$ from the above expression, Equation 12 can be written as

$$H^{\mu}(i,x) = \hat{H}^{\mu}(\mu_i(x),(\mu_i(x) \searrow \omega,s)) + \sum_{N_1,s_1 \in S_i} P_i^{\mu}(x_1,N_1|x,\mu_i(x)) \left[\hat{H}^{\mu}(i,x_1) + \sum_{N_2,s_2 \in S_i} F_i^{\mu}(x_2,N_2|x_1) H^{\mu}(Parent(i),(\omega \nearrow i,s_2)) \right]. \tag{13}$$

We can restate Equation 13 for hierarchical average-adjusted action-value function as

$$L^{\mu}(i,x,a) = \hat{H}^{\mu}(a,(a \searrow \omega,s)) + \sum_{N_1,s_1 \in S_i} P_i^{\mu}(x_1,N_1|x,a) \left[\hat{H}^{\mu}(i,x_1) + \sum_{N_2,s_2 \in S_i} F_i^{\mu}(x_2,N_2|x_1) L^{\mu}(Parent(i),(\omega \nearrow i,s_2),\mu_{parent(i)}(\omega \nearrow i,s_2)) \right]. \tag{14}$$

From Equation 14, we can rewrite the hierarchical average-adjusted action-value function L recursively as

$$L^{\mu}(i,x,a) = \hat{H}^{\mu}(a,(a \searrow \omega,s)) + C^{\mu}(i,x,a) + CE^{\mu}(i,x,a), \tag{15}$$

where

$$C^{\mu}(i,x,a) = \sum_{N_1,s_1 \in S_i} P_i^{\mu}(x_1,N_1|x,a) \hat{H}^{\mu}(i,x_1), \tag{16}$$

and

$$CE^{\mu}(i,x,a) = \sum_{N_1,s_1 \in S_i} P_i^{\mu}(x_1,N_1|x,a) \left[\sum_{N_2,s_2 \in S_i} F_i^{\mu}(x_2,N_2|x_1) L^{\mu}(Parent(i),(\omega \nearrow i,s_2),\mu_{parent(i)}(\omega \nearrow i,s_2)) \right]. \tag{17}$$

The term $C^{\mu}(i,x,a)$ is the expected average-adjusted reward of completing subtask M_i after executing action a in state $x = (\omega,s)$. We call this term **completion function** after Dietterich (2000). The term $CE^{\mu}(i,x,a)$ is the expected average-adjusted reward received after subtask M_i terminates. We call this term **external completion function** after Andre and Russell (2002).

We can rewrite the definition of $\hat{H}$ as

$$\hat{H}^{\mu}(i,x) = \begin{cases} \hat{L}^{\mu}(i,x,\mu_i(x)) & \text{if } M_i \text{ is a non-primitive subtask,} \\ r(s,i) - g^{\mu} & \text{if } M_i \text{ is a primitive action,} \end{cases} \tag{18}$$

where $\hat{L}^{\mu}$ is the projected average-adjusted action-value function and can be written as

$$\hat{L}^{\mu}(i,x,a) = \hat{H}^{\mu}(a,(a \searrow \omega,s)) + C^{\mu}(i,x,a). \tag{19}$$

Equations 15 to 19 are the decomposition equations under a hierarchical policy μ. These equations recursively decompose the hierarchical average-adjusted value function for *root*, $H^{\mu}(0,x)$, into the projected average-adjusted value functions $\hat{H}^{\mu}$ for the individual subtasks, $M_1,\ldots,M_{m-1}$, in the hierarchy, the individual completion functions $C^{\mu}(i,x,a)$, $i = 1,\ldots,m-1$, and the individual external completion functions $CE^{\mu}(i,x,a)$, $i = 1,\ldots,m-1$. The fundamental quantities that must be stored to represent the hierarchical average-adjusted value function decomposition are the C and the CE values for all non-primitive subtasks, the $\hat{H}$ values for all primitive actions, and the *global gain* g. The decomposition equations can be used to obtain update equations for $\hat{H}$, C, and CE in this hierarchically optimal average reward model. Pseudo-code for the discrete-time *hierarchically optimal average reward RL* (HAR) algorithm is shown in Algorithm 1. In this algorithm, primitive subtasks update their projected average-adjusted value functions $\hat{H}$ (Line 5), while non-primitive subtasks update both their completion functions C (Line 17), and external completion functions CE (Lines 20 and 22). We store only one *global gain* g and update it after each non-random primitive action (Line 7). In the update formula on Line 17, the projected average-adjusted value function $\hat{H}(a^*,(a^* \searrow \omega,s'))$ is the average-adjusted reward of executing action a^* in state $(a^* \searrow \omega,s')$ and is recursively calculated by subtask M_{a^*} and its descendants using Equations 18 and 19. Notice that the hierarchical average-adjusted action-value function L on Lines 15, 19, and 20 is recursively evaluated using Equation 15.

This algorithm can be easily extended to continuous-time by changing the update formulas for $\hat{H}$ and g on Lines 5 and 7 as

$$\hat{H}_{t+1}(i,x) \leftarrow [1-\alpha_t(i)]\hat{H}_t(i,x) + \alpha_t(i)\left[k(s,i) + r(s,i)\tau(s,i) - g_t\tau(s,i)\right],$$

$$g_{t+1} = \frac{r_{t+1}}{t_{t+1}} = \frac{r_t + k(s,i) + r(s,i)\tau(s,i)}{t_t + \tau(s,i)},$$

where $\tau(s,i)$ is the time elapsing between state s and the next state, $k(s,i)$ is the fixed reward of taking action M_i in state s, and $r(s,i)$ is the reward rate for the time between state s and the next state.

Algorithm 1 : Discrete-time hierarchically optimal average reward RL (HAR) algorithm.

Function HAR(Task M_i, State $x = (\omega, s)$)
let $Seq = \{\}$ be the sequence of *states visited* while executing subtask M_i
if M_i is a primitive action **then**
  execute action M_i in state $x = (\omega, s)$, observe state $x' = (\omega, s')$ and reward $r(s,i)$
  $\hat{H}_{t+1}(i,x) \leftarrow [1-\alpha_t(i)]\hat{H}_t(i,x) + \alpha_t(i)[r(s,i) - g_t]$
  if M_i and all its ancestors are non-random actions **then**
    update the global gain $g_{t+1} = \frac{r_{t+1}}{n_{t+1}} = \frac{r_t + r(s,i)}{n_t + 1}$
  end if
  push *state* $x_1 = (\omega \nearrow i, s)$ into the beginning of *Seq*
else
  while M_i has not terminated **do**
    choose action (subtask) M_a according to the current exploration policy $\mu_i(x)$
    let *ChildSeq* = HAR(M_a, $(a \searrow \omega, s)$), where *ChildSeq* is the sequence of states visited while executing subtask M_a
    observe result state $x' = (\omega, s')$
    let $a^* = \arg\max_{a' \in A_i(s')} L_t(i, x', a')$
    for each $x = (\omega, s)$ in *ChildSeq* from the beginning **do**
      $C_{t+1}(i,x,a) \leftarrow [1-\alpha_t(i)]C_t(i,x,a) + \alpha_t(i)\left[\hat{H}_t(a^*, (a^* \searrow \omega, s')) + C_t(i,x',a^*)\right]$
      if $s' \in T_i$ (s' belongs to the set of terminal states of subtask M_i) **then**
        $a'' = \arg\max_{a' \in A_{Parent(i)}} L_t(Parent(i), (\omega \nearrow i, s'), a')$
        $CE_{t+1}(i,x,a) \leftarrow [1-\alpha_t(i)]CE_t(i,x,a) + \alpha_t(i)L_t(Parent(i), (\omega \nearrow i, s'), a'')$
      else
        $CE_{t+1}(i,x,a) \leftarrow [1-\alpha_t(i)]CE_t(i,x,a) + \alpha_t(i)CE_t(i,x',a^*)$
      end if
      replace state $x = (\omega, s)$ with $(\omega \nearrow i, s)$ in the *ChildSeq*
    end for
    append *ChildSeq* onto the front of *Seq*
    $x = x'$
  end while
end if
return *Seq*
end HAR

5.2 Recursively Optimal Average Reward RL

In the previous section, we introduced discrete-time and continuous-time hierarchically optimal average reward RL (HAR) algorithms. In HAR algorithms, we define only a *global gain* for the entire hierarchy to guarantee hierarchical optimality for the overall task. HAR algorithms find a hierarchical policy that has the highest *global gain* among all policies consistent with the given hierarchy. However, there may exist subtasks where their policies must be locally suboptimal so that the overall policy becomes optimal. Recursive optimality is a kind of local optimality in which the policy at each node is optimal given the policies of its children (see Section 4.3). Thus, the goal at *root* is to maximize its gain given the policies for its descendants. The reason seeking recursive optimality rather than hierarchical optimality is that recursive optimality makes it possible to solve

each subtask without reference to the context in which it is executed, and therefore the learned subtask can be reused by other hierarchies. This leaves open the question of what local optimality criterion should be used for each subtask in a recursively optimal average reward RL setting.

One approach pursued by Seri and Tadepalli (2002) is to optimize subtasks using their expected total average-adjusted reward with respect to the *global gain*. Seri and Tadepalli introduced a model-based algorithm called *hierarchical H-Learning* (HH-Learning). For every subtask, this algorithm learns the action model and maximizes the expected total average-adjusted reward with respect to the *global gain* at each state. In this method, the projected average-adjusted value functions with respect to the *global gain* satisfy the following equations:

$$\hat{H}^{\mu}(i,s) = \begin{cases} r(s,i) - g^{\mu} & \text{if } M_i \text{ is a primitive action,} \\ 0 & \text{if } s \in T_i \text{ (s is a terminal state of subtask } M_i\text{),} \\ \max_{a \in A_i(s)} [\hat{H}^{\mu}(a,s) + \sum_{N,s' \in S_i} P_i^{\mu}(s',N|s,a)\hat{H}^{\mu}(i,s')] & \text{otherwise.} \end{cases} \tag{20}$$

The first term of the last part of Equation 20, $\hat{H}^{\mu}(a,s)$, denotes the expected total average-adjusted reward during the execution of subtask M_a (the projected average adjusted value function of subtask M_a), and the second term denotes the expected total average-adjusted reward from then on until the completion of subtask M_i (the completion function of subtask M_i after execution of subtask M_a). Since the expected average-adjusted reward after execution of subtask M_i is not a component of the average-adjusted value function of subtask M_i, this approach does not necessarily allow for hierarchical optimality, as we will show in the experiments of Section 6. Moreover, the policy learned for each subtask using this approach is not context free, since each subtask maximizes its average-adjusted reward with respect to the *global gain*. However, Seri and Tadepalli (2002) showed that this method finds the hierarchically optimal average reward policy when the *result distribution invariance* condition holds.

Definition 8 (Result Distribution Invariance Condition): For all subtasks M_i and states s in the hierarchy, the distribution of states reached after the execution of any subtask M_a (M_a is one of M_i's children) is independent of the policy μ_a of subtask M_a and the policies of M_a's descendants, that is, $P_i^{\mu}(s'|s,a) = P_i(s'|s,a)$. □

In other words, states reached after the execution of a subtask cannot be changed by altering the policies of the subtask and its descendants. Note that the *result distribution invariance* condition does not hold for every problem, and therefore HH-Learning is neither hierarchically nor recursively optimal in general.

Another approach is to formulate subtasks as continuing average reward problems, where the goal at each subtask is to maximize its gain given the policies of its children (Ghavamzadeh and Mahadevan, 2001). We describe this approach in detail in Sections 5.2.1 and 5.2.2. In Section 5.2.3, we use this method to find recursively optimal average reward policies, and present discrete-time and continuous-time *recursively optimal average reward RL* (RAR) algorithms. Finally, in Section 5.2.4, we investigate the conditions under which the policy learned by RAR algorithm at each subtask is independent of the context in which it is executed and therefore can be reused by other hierarchies.

5.2.1 ROOT TASK FORMULATION

In our recursively optimal average reward RL approach, we consider those problems for which Assumption 1 (*Continuing Root Task*) and the following assumption hold.

Assumption 3 (Root Task Recurrence): There exists a state $s_0^* \in S_0$ such that, for every hierarchical policy μ and for every state $s \in S_0$, we have[9]

$$\sum_{N=1}^{|S_0|} F_0^{\mu}(s_0^*, N|s) > 0,$$

where F_0^{μ} is the multi-step abstract transition probability function of *root* under the hierarchical policy μ described in Section 4.2, and $|S_0|$ is the number of states in the state space of *root*. □

Assumption 3 is equivalent to assuming that the underlying Markov chain at *root* for every hierarchical policy μ has a single recurrent class, and state s_0^* is a recurrent state. If Assumptions 1 and 3 hold, the gain at the *root* task under the hierarchical policy μ, g_0^{μ}, is well defined for every hierarchical policy μ and does not depend on the initial state. When the state space at *root* is finite or countable, the gain at *root* can be written as[10]

$$g_0^{\mu} = \frac{\bar{F}_0^{\mu} r_0^{\mu}}{\bar{F}_0^{\mu} y_0^{\mu}},$$

where r_0^{μ} and y_0^{μ} are vectors with elements $r_0^{\mu}(s,\mu_0(s))$ and $y_0^{\mu}(s,\mu_0(s))$, for all $s \in S_0$. $r_0^{\mu}(s,\mu_0(s))$ and $y_0^{\mu}(s,\mu_0(s))$ are the expected total reward and the expected number of time steps between two decision epochs at *root*, given that the system occupies state s at the first decision epoch and the agent chooses its actions according to the hierarchical policy μ. The terms F_0^{μ} and $\bar{F}_0^{\mu} = \lim_{n\to\infty} \frac{1}{n} \sum_{t=0}^{n-1} (F_0^{\mu})^t$ are the transition probability matrix and the *limiting matrix* of the embedded Markov chain at *root* for hierarchical policy μ, respectively. The transition probability F_0^{μ} is obtained by marginalizing the multi-step transition probability P_0^{μ}. The term $F_0^{\mu}(s'|s,\mu_0(s))$ denotes the probability that the SMDP at *root* occupies state s' at the next decision epoch, given that the agent chooses action $\mu_0(s)$ in state s at the current decision epoch and follows the hierarchical policy μ.

5.2.2 SUBTASK FORMULATION

In Section 5.2.1, we described the average reward formulation for the *root* task of a hierarchical decomposition. In this section, we illustrate how we formulate all other subtasks in a hierarchy as average reward problems. From now on in this section, we use subtask to refer to non-primitive subtasks in a hierarchy except *root*.

In HRL methods, we typically assume that every time a subtask M_i is executed, it starts at one of its initial states ($\in I_i$) and terminates at one of its terminal states ($\in T_i$) after a finite number of time steps. Therefore, we can make the following assumption for every subtask M_i in the hierarchy.

9. Notice that the *root* task is represented as subtask M_0 in the HRL framework described in Section 4. Thus, we use index 0 to represent components of the *root* task.
10. When the underlying Markov chain at *root* for every hierarchical policy μ has a single recurrent class, $\bar{F}_0^{\mu}$ has equal rows, and the right hand side of the equation is a vector with elements all equal to g_0^{μ}.

Under this assumption, each subtask can be considered an episodic problem and each instantiation of a subtask can be considered an episode.

Assumption 4 (Subtask Termination): There exists a dummy state s_i^* such that, for every action $a \in A_i$ and every terminal state $s \in T_i$, we have

$$r_i(s,a) = 0 \qquad \text{and} \qquad P_i(s_i^*, 1|s,a) = 1$$

and for all hierarchical stationary policies μ and non-terminal states $s \in (S_i - T_i)$, we have

$$F_i^{\mu}(s_i^*, 1|s) = 0$$

and finally for all states $s \in S_i$, we have

$$F_i^{\mu}(s_i^*, |S_i||s) > 0$$

where F_i^{μ} is the multi-step abstract transition probability function of subtask M_i under the hierarchical policy μ described in Section 4.2, and $|S_i|$ is the number of states in the state space of subtask M_i. □

Although subtasks are episodic problems, when the overall task (*root* of the hierarchy) is continuing as we assumed in this section (Assumption 1), they are executed an infinite number of times, and therefore can be modeled as continuing problems using the model described in Figure 4. In this model, each subtask M_i terminates at one of its terminal states $s \in T_i$. All terminal states transit with probability 1 and reward 0 to a dummy state s_i^*. Finally, the dummy state s_i^* transits with reward zero to one of the initial states ($\in I_i$) of subtask M_i upon the next instantiation of M_i. These are dummy transitions and do not add any time-step to the cycle of subtask M_i and therefore are not taken into consideration when the average reward of subtask M_i is calculated. It is important for the validity of the model to fix the value of dummy states to zero.

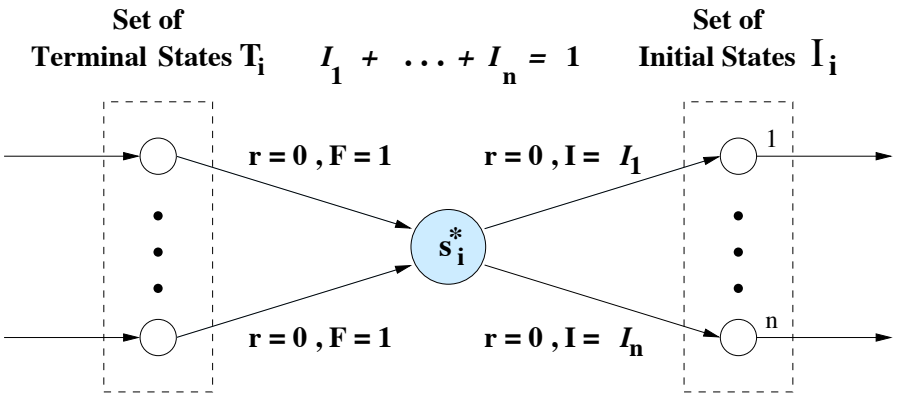


Figure 4: This figure shows how each subtask in a hierarchical decomposition of a continuing problem can be modeled as a continuing task.

Under this model, for every hierarchical policy μ, we define a new SMDP for each subtask M_i in the hierarchy with the following multi-step transition probabilities and rewards:

$$P^{\mu}_{I_i}(s',N|s,\mu_i(s)) = \begin{cases} P^{\mu}_i(s',N|s,\mu_i(s)) & s,s' \neq s^*_i \quad , \quad \forall N \in \mathbb{N}, \\ I_i(s') & s = s^*_i \quad , \quad N = 1, \\ 1 & s' = s^*_i \quad , \quad s \in T_i \quad , \quad N = 1, \\ 0 & \text{otherwise.} \end{cases}$$

$$r^{\mu}_{I_i}(s,\mu_i(s)) = \begin{cases} r^{\mu}_i(s,\mu_i(s)) & s \in (S_i - T_i), \\ 0 & s = s^*_i \quad \text{or} \quad s \in T_i. \end{cases} \tag{21}$$

where $I_i(s)$ is the probability that subtask M_i starts at state $s \in I_i$. The SMDP defined by Equation 21 has an embedded MDP with the following transition probability function:

$$F^{\mu}_{I_i}(s'|s,\mu_i(s)) = \begin{cases} F^{\mu}_i(s'|s,\mu_i(s)) & s,s' \neq s^*_i, \\ I_i(s') & s = s^*_i, \\ 1 & s' = s^*_i \quad , \quad s \in T_i, \\ 0 & s' = s^*_i \quad , \quad s \in (S_i - T_i). \end{cases} \tag{22}$$

Lemma 1: Let Assumption 4 (*Subtask Termination*) hold. Then, for every $F^{\mu}_{I_i}$ and every state $s \in S_i$, we have $\sum_{N=1}^{|S_i|} F^{\mu}_{I_i}(s^*_i,N|s) > 0$.[11] □

Lemma 1 is equivalent to assuming that for every subtask M_i in the hierarchy, the underlying Markov chain for every hierarchical policy μ has a single recurrent class and state s^*_i is its recurrent state. Under this model, the gain of subtask M_i under the hierarchical policy μ, g^{μ}_i, is well defined for every hierarchical policy μ and does not depend on the initial state. When the state space of subtask M_i is finite or countable, the gain of subtask M_i can be written as[12]

$$g^{\mu}_i = \frac{\bar{F}^{\mu}_{I_i} r^{\mu}_{I_i}}{\bar{F}^{\mu}_{I_i} y^{\mu}_{I_i}},$$

where $r^{\mu}_{I_i}$ and $y^{\mu}_{I_i}$ are vectors with elements $r^{\mu}_{I_i}(s,\mu_i(s))$ and $y^{\mu}_{I_i}(s,\mu_i(s))$, for all $s \in S_i$. $r^{\mu}_{I_i}(s,\mu_i(s))$ and $y^{\mu}_{I_i}(s,\mu_i(s))$ are the expected total reward and the expected number of time steps between two decision epochs of the SMDP defined by Equation 21 at subtask M_i, given that the system occupies state s at the first decision epoch and the agent chooses its actions according to hierarchical policy μ. The term $\bar{F}^{\mu}_{I_i} = \lim_{n\to\infty} \frac{1}{n} \sum_{t=0}^{n-1} (F^{\mu}_{I_i})^t$ is the *limiting matrix* of the Markov chain defined by Equation 22 at subtask M_i.

5.2.3 A RECURSIVELY OPTIMAL AVERAGE REWARD RL ALGORITHM

In this section, we present discrete-time and continuous-time recursively optimal average reward RL (RAR) algorithms using the formulation described in Sections 5.2.1 and 5.2.2. We consider

11. This lemma is a restatement of Lemma 5 on page 34 of Peter Marbach's thesis (Marbach, 1998).
12. When the underlying Markov chain for every hierarchical policy μ at subtask M_i has a single recurrent class, $\bar{F}^{\mu}_{I_i}$ has equal rows, and thus the right hand side of the equation is a vector with elements all equal to g^{μ}_i.

problems for which Assumptions 1, 3, and 4 (*Continuing Root-Task*, *Root-Task Recurrence*, and *Subtask Termination*) hold, *root* is modeled as an average reward problem as described in Section 5.2.1, and every other non-primitive subtask in the hierarchy is modeled as an average reward problem using the model described in Section 5.2.2. Under these assumptions, the average reward for every non-primitive subtask in the hierarchy including *root* is well defined for every hierarchical policy and does not vary with initial state. Since we are interested in finding a recursively optimal average reward policy, we do not need to include the contents of the Task-Stack as a part of the state space of the problem. We also replace the projected value and action-value functions in the hierarchical model of Section 4 with the projected average-adjusted value and projected average-adjusted action-value functions described in Section 3.1.

We show how the overall projected average-adjusted value function $\hat{H}^{\mu}(0,s)$ is decomposed into a collection of projected average-adjusted value functions of individual subtasks $\hat{H}^{\mu}(i,s), i = 1,\ldots,m-1$, in RAR algorithm. The projected average-adjusted value function of hierarchical policy μ at subtask M_i is the average-adjusted (with respect to the local gain g_i^{μ}) sum of rewards earned by following policy μ_i and the policies of all descendants of subtask M_i starting in state s until M_i terminates. Now, let us suppose that the first action chosen by μ_i is executed for a number of primitive steps N and terminates in state s' according to multi-step transition probability $P_i^{\mu}(s',N|s,\mu_i(s))$. We can write the projected average-adjusted value function in the form of a Bellman equation as

$$\hat{H}^{\mu}(i,s) = r_i^{\mu}(s,\mu_i(s)) - g_i^{\mu} y_i^{\mu}(s,\mu_i(s)) + \sum_{N,s' \in S_i} P_i^{\mu}(s',N|s,\mu_i(s))\hat{H}^{\mu}(i,s'). \tag{23}$$

Since $r_i^{\mu}(s,\mu_i(s))$ is the expected total reward between two decision epochs of subtask M_i, given that the system occupies state s at the first decision epoch, the agent chooses action $\mu_i(s)$, and the number of time steps until the next decision epoch is defined by $y_i^{\mu}(s,\mu_i(s))$, we have

$$r_i^{\mu}(s,\mu_i(s)) = \begin{cases} \hat{V}^{\mu}(\mu_i(s),s) = \hat{H}^{\mu}(\mu_i(s),s) + g_{\mu_i(s)}^{\mu} y_i^{\mu}(s,\mu_i(s)) & \\ & \text{if } M_{\mu_i(s)} \text{ is a non-primitive subtask,} \\ \hat{V}^{\mu}(\mu_i(s),s) & \\ & \text{if } M_{\mu_i(s)} \text{ is a primitive action.} \end{cases}$$

By replacing $r_i^{\mu}(s,\mu_i(s))$ from the above expression, and the fact that $y_i^{\mu}(s,\mu_i(s))$ equals 1 when $M_{\mu_i(s)}$ is a primitive action, Equation 23 can be written as

$$\hat{H}^{\mu}(i,s) = \begin{cases} \hat{H}^{\mu}(\mu_i(s),s) - (g_i^{\mu} - g_{\mu_i(s)}^{\mu}) y_i^{\mu}(s,\mu_i(s)) + \sum_{N,s' \in S_i} P_i^{\mu}(s',N|s,\mu_i(s))\hat{H}^{\mu}(i,s') & \\ & \text{if } M_{\mu_i(s)} \text{ is a non-primitive subtask,} \\ \hat{V}^{\mu}(\mu_i(s),s) - g_i^{\mu} + \sum_{s' \in S_i} P_i^{\mu}(s'|s,\mu_i(s))\hat{H}^{\mu}(i,s') & \\ & \text{if } M_{\mu_i(s)} \text{ is a primitive action.} \end{cases} \tag{24}$$

We can restate Equations 24 for the projected action-value function as follows:

$$\hat{L}^{\mu}(i,s,a) = \begin{cases} \hat{H}^{\mu}(a,s) - (g_i^{\mu} - g_a^{\mu}) y_i^{\mu}(s,a) + \sum_{N,s' \in S_i} P_i^{\mu}(s',N|s,a) \hat{L}^{\mu}(i,s',\mu_i(s')) \\ \qquad \text{if } M_a \text{ is a non-primitive subtask,} \\ \hat{V}^{\mu}(a,s) - g_i^{\mu} + \sum_{s' \in S_i} P_i^{\mu}(s'|s,a) \hat{L}^{\mu}(i,s',\mu_i(s')) \\ \qquad \text{if } M_a \text{ is a primitive action.} \end{cases} \tag{25}$$

By defining

$$C^{\mu}(i,s,a) = \begin{cases} -(g_i^{\mu} - g_a^{\mu}) y_i^{\mu}(s,a) + \sum_{N,s' \in S_i} P_i^{\mu}(s',N|s,a) \hat{L}^{\mu}(i,s',\mu_i(s')) \\ \qquad \text{if } M_a \text{ is a non-primitive subtask,} \\ -g_i^{\mu} + \sum_{s' \in S_i} P_i^{\mu}(s'|s,a) \hat{L}^{\mu}(i,s',\mu_i(s')) \\ \qquad \text{if } M_a \text{ is a primitive action,} \end{cases} \tag{26}$$

we can express the average-adjusted action-value function $\hat{L}^{\mu}$ recursively as

$$\hat{L}^{\mu}(i,s,a) = \begin{cases} \hat{H}^{\mu}(a,s) + C^{\mu}(i,s,a) & \text{if } M_a \text{ is a non-primitive subtask,} \\ \hat{V}^{\mu}(a,s) + C^{\mu}(i,s,a) & \text{if } M_a \text{ is a primitive action,} \end{cases} \tag{27}$$

where

$$\hat{H}^{\mu}(i,s) = \hat{L}^{\mu}(i,s,\mu_i(s)). \tag{28}$$

We call $C^{\mu}(i,s,a)$ defined by Equation 26 **completion function**.

Equations 24 to 28 are the decomposition equations for the projected average-adjusted value and projected average-adjusted action-value functions. They can be used to obtain update formulas for $\hat{H}$ and C in this recursively optimal average reward model. Pseudo-code for the discrete-time *recursively optimal average reward RL* (RAR) algorithm is shown in Algorithm 2. In this algorithm, a gain is defined for every non-primitive subtask in the hierarchy and this gain is updated every time a subtask is non-randomly chosen. Primitive subtasks store their projected value functions, and update them using the equation on Line 5. Non-primitive subtasks store their completion functions and gains, and update them using equations on Lines 17, 19, and 23. The projected average-adjusted action-value function $\hat{L}$ on Lines 12, 17, and 19 is recursively calculated using Equations 26 to 28.

This algorithm can be easily extended to continuous-time (Ghavamzadeh and Mahadevan, 2001). In continuous-time RAR algorithm, in addition to visited state and reward, we need to insert the execution time of primitive actions τ into the sequence *Seq*. Therefore, $N = N + 1$ on Line 15 of the algorithm is changed to $T = T + \tau$. We also need to change the update formulas for $\hat{V}, C$, and g_i on Lines 5, 17, 19, and 23 as

$$\hat{V}_{t+1}(i,s) \leftarrow [1 - \alpha_t(i)] \hat{H}_t(i,s) + \alpha_t(i) \left[k(s,i) + r(s,i) \tau(s,i) \right],$$

$$C_{t+1}(i,s,a) \leftarrow [1 - \alpha_t(i)] C_t(i,s,a) + \alpha_t(i) [\hat{L}_t(i,s',a^*) - g_t(i) T],$$

$$C_{t+1}(i,s,a) \leftarrow [1 - \alpha_t(i)] C_t(i,s,a) + \alpha_t(i) [\hat{L}_t(i,s',a^*) - (g_t(i) - g_t(a)) T],$$

Algorithm 2 : Discrete-time recursively optimal average reward RL (RAR) algorithm.

Function RAR(Task M_i, State s)
let $Seq = \{\}$ be the sequence of (*state visited*, *reward*) while executing subtask M_i
if M_i is a primitive action **then**
  execute action M_i in state s, observe state s' and reward $r(s,i)$
  $\hat{V}_{t+1}(i,s) \leftarrow [1-\alpha_t(i)]\hat{V}_t(i,s) + \alpha_t(i) r(s,i)$
  push (*state* s, *reward* $r(s,i)$) into the beginning of *Seq*
else
  while M_i has not terminated **do**
    choose action (subtask) M_a according to the current exploration policy $\mu_i(s)$
    let $ChildSeq = \text{RAR}(M_a, s)$, where *ChildSeq* is the sequence of (*state visited*, *reward*) while executing subtask M_a
    observe result state s'
    let $a^* = \arg\max_{a' \in A_i(s')} \hat{L}_t(i,s',a')$
    let $N = 0$; $\rho = 0$;
    for each (s,r) in *ChildSeq* from the beginning **do**
      $N = N + 1$; $\rho = \rho + r$;
      if a is a primitive action **then**
        $C_{t+1}(i,s,a) \leftarrow [1-\alpha_t(i)]C_t(i,s,a) + \alpha_t(i)[\hat{L}_t(i,s',a^*) - g_t(i)N]$
      else
        $C_{t+1}(i,s,a) \leftarrow [1-\alpha_t(i)]C_t(i,s,a) + \alpha_t(i)[\hat{L}_t(i,s',a^*) - (g_t(i) - g_t(a))N]$
      end if
    end for
    if a and all its ancestors are non-random actions **then**
      update the gain of subtask M_i $g_{t+1}(i) = \frac{r_{t+1}(i)}{n_{t+1}(i)} = \frac{r_t(i)+\rho}{n_t(i)+N}$
    end if
    append *ChildSeq* onto the front of *Seq*
    $s = s'$
  end while
end if
return *Seq*
end RAR

$$g_{t+1}(i) = \frac{r_{t+1}(i)}{t_{t+1}(i)} = \frac{r_t(i)+\rho}{t_t(i)+T},$$

where $\tau(s,i)$ is the time elapsing between state s and the next state, $k(s,i)$ is the fixed reward of taking action M_i in state s, and $r(s,i)$ is the reward rate for the time between state s and the next state.

5.2.4 ANALYSIS OF THE RAR ALGORITHM

In this section, we study the optimality achieved by RAR algorithm. As described earlier, the expected average-adjusted sum of rewards after execution of subtask M_i is not a component of the average-adjusted value function of subtask M_i in RAR algorithm. Therefore, the algorithm fails to

find a hierarchically optimal average reward policy in general, as was discussed in Seri and Tadepalli (2002) and will be demonstrated in the experiments of Section 6.

To achieve recursive optimality, the policy learned for each subtask must be context free, that is, each subtask should maximize its local gain given the policies of its descendants. In RAR algorithm, although each subtask maximizes its local gain given the policies of its descendants, the policy learned for each subtask is not necessarily context free, and as a result, the algorithm does not find a recursively optimal average reward policy in general. The reason is, the local gain g_i for each subtask M_i does not depend only on the policies of its descendants. The local gain g_i is the gain of the SMDP defined by Equation 21 and therefore depends on the initial state distribution $I_i(s)$. The initial state distribution $I_i(s)$, the probability of being in state s at the next instantiation of subtask M_i, depends not only on the policies of subtask M_i and all its descendants, but also on the policies of its ancestors. It makes the local gain g_i learned by RAR algorithm context dependent. However, the algorithm finds a recursively optimal average reward policy when the *initial distribution invariance* (IDI) condition holds. Under the IDI condition, the policy learned by RAR algorithm at each subtask is independent of the context in which it is executed and therefore can be reused by other hierarchies.

Definition 9 (Initial Distribution Invariance Condition): The initial state distribution for each non-primitive subtask in the hierarchy is independent of the policies of its ancestors. □

In other words, the initial state distribution for each non-primitive subtask cannot be changed by altering the policies of its ancestors. One special case that satisfies the IDI condition is when each non-primitive subtask in the hierarchy has only one initiation state, $|I_i| = 1, i = 1, \ldots, m-1$, and M_i is a non-primitive subtask.

6. Experimental Results

The goal of this section is to show the type of optimality achieved by the *hierarchically optimal average reward RL* (HAR) and the *recursively optimal average reward RL* (RAR) algorithms proposed in Sections 5.1 and 5.2, as well as their performance and speed compared to other algorithms. We describe two sets of experiments. In Section 6.1, we apply five HRL algorithms to a simple discrete-time automated guided vehicle (AGV) scheduling problem. Since we use a hierarchical task decomposition in which the hierarchically and recursively optimal policies are different for this problem, our experimental results clearly demonstrate the difference between the optimality achieved by these algorithms. Then, we turn to a relatively large AGV scheduling task in Section 6.2. We model this AGV scheduling task as discrete time and continuous-time problems. In the discrete-time model, we compare the performance of HAR and RAR algorithms with a hierarchically optimal discounted reward algorithm and a recursively optimal discounted reward algorithm, as well as a non-hierarchical (flat) average reward algorithm. In the continuous-time model, we compare the performance of HAR and RAR algorithms with a recursively optimal discounted reward algorithm. We do not use pseudo-reward or reward shaping in the experiments of this section. The first problem is simple and can be solved easily without reward shaping. There are rewards associated with the terminal states of the subtasks in the original MDP of the second problem. Therefore, the agent can find out about the desirability of the terminal states upon completing the subtasks, without using pseudo-reward or reward shaping.

6.1 A Simple AGV Scheduling Problem

In this section, we apply the *discrete-time hierarchically optimal average reward RL* (HAR) algorithm described in Section 5.1, the *discrete-time recursively optimal average reward RL* (RAR) algorithm described in Section 5.2, and *HH-Learning*, the algorithm proposed by Seri and Tadepalli (2002), to a small AGV scheduling task. We also test MAXQ-Q, the recursively optimal discounted reward HRL algorithm proposed by Dietterich (2000), and a *hierarchically optimal discounted reward RL* algorithm (HDR) on this task. The HDR algorithm is an extension of MAXQ-Q using the three-part value function decomposition (Andre and Russell, 2002) described in Section 4.5.

A simple AGV domain is depicted in Figure 5. In this domain there are two machines $M1$ and $M2$ that produce parts to be delivered to the corresponding destination stations $G1$ and $G2$. Since machines and destination stations are in two different rooms, the AGV has to pass one of the two doors $D1$ and $D2$ every time it goes from one room to another. Part 1 is more important than part 2, therefore the AGV gets a reward of 20 when part 1 is delivered to destination $G1$ and a reward of 1 when part 2 is delivered to destination $G2$. The AGV receives a reward of -1 for all other actions. Note that within subtasks "Go to Machine" and "Go to Door", the agent must choose which machine to go to, and which door to pass through, respectively. This task is deterministic and the state variables are AGV's *location* and *status* (empty, carry part 1, carry part 2), which is a total of $26 \times 3 = 78$ states. In all experiments, we use the task graph shown in Figure 5 and set the discount factor to 0.9 for the discounted reward algorithms. We tried several discounting factors and $\gamma = 0.9$ yielded the best performance. Using this task graph, hierarchically and recursively optimal policies are different. Since delivering part 1 has more reward than part 2, the hierarchically optimal policy is one in which the AGV always serves machine $M1$. In the recursively optimal policy, the AGV switches from serving machine $M1$ to serving machine $M2$ and vice versa. In this policy, the AGV goes to machine $M1$, picks up a part of type 1, goes to goal $G1$ via door $D1$, drops the part there, then passes through door $D2$, goes to machine $M2$, picks up a part of type 2, goes to goal $G2$ via door $D2$ and then switches again to machine $M1$ and so on so forth.

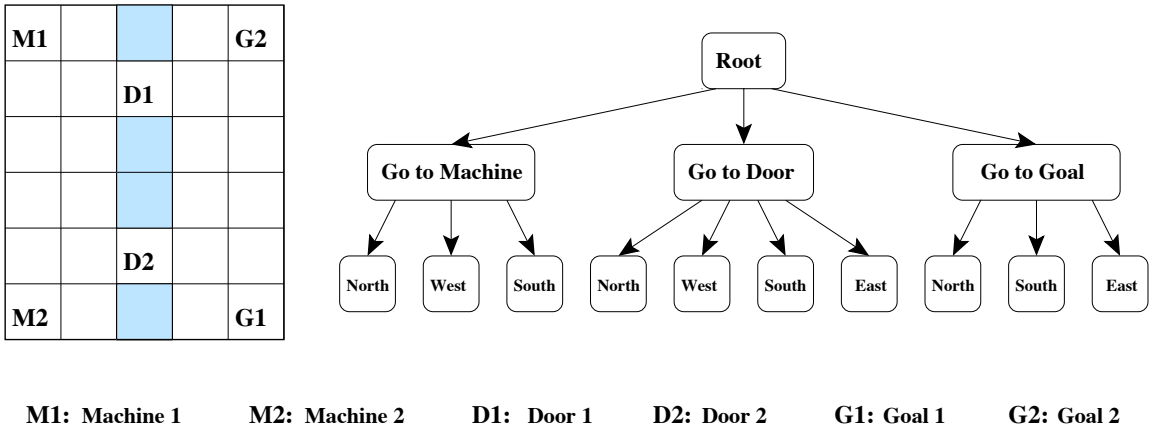


Figure 5: A simple AGV scheduling task and its associated task graph. Note that within subtasks "Go to Machine" and "Go to Door", the AGV must choose which machine to go to, and which door to pass through, respectively.

Among the algorithms we applied to this task, the hierarchically optimal average reward RL (HAR) and the hierarchically optimal discounted reward RL (HDR) algorithms find the hierarchically optimal policy, where the other algorithms only learn the recursively optimal policy. Figure 6 demonstrates the throughput of the system for the above algorithms. The hierarchically optimal algorithms learn more slowly than the recursively optimal algorithms due to more parameters to be learned. Since this problem is deterministic, the HH-Learning algorithm, which is the only model-based RL algorithm used in this experiment, learns the model of the environment quickly, and therefore converges much faster than the other algorithms. In this figure, the throughput of the system is the number of parts deposited at destination stations weighted by their rewards ($part1 \times 20 + part2 \times 1$) in 250 time steps. Each experiment was conducted twenty times and the results were averaged.

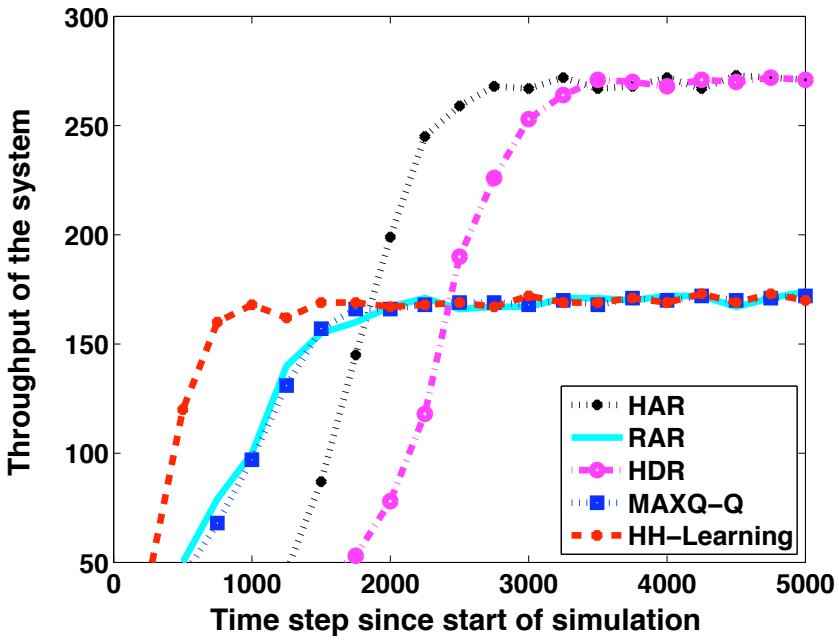


Figure 6: This figure shows that HDR and HAR algorithms (the two top curves) learn the hierarchically optimal policy while RAR, MAXQ-Q, and HH-Learning (the three bottom curves) only find the recursively optimal policy for the small AGV scheduling task.

6.2 AGV Scheduling Problem (Discrete and Continuous Time Models)

In this section, we describe two sets of experiments on the AGV scheduling problem shown in Figure 7. $M1$ to $M3$ are workstations in this environment. Parts of type i have to be carried to the drop-off station at workstation i (D_i), and the assembled parts brought back from pick-up stations of workstations (P_i's) to the warehouse. The AGV travel is unidirectional as the arrows show. The AGV receives a reward of 20 when it picks up a part at the warehouse, delivers a part to a drop-off station, picks up an assembled part from a pick-up station, or delivers an assembled part to the warehouse. It

also gets a reward of -5 when it attempts to execute Put1–Put3, Pick1–Pick3, Load1–Load3, Unload, and Idle actions illegally. There is a reward of -1 for all other actions. We model this AGV scheduling task using both discrete-time and continuous-time models. In the discrete-time model, we show the performance of four HRL algorithms: *hierarchically optimal average reward RL* (HAR), *recursively optimal average reward RL* (RAR), *hierarchically optimal discounted reward RL* (HDR), and *recursively optimal discounted reward RL* (MAXQ-Q), as well as a *non-hierarchical average reward* algorithm. In the continuous-time model, we compare the performance of HAR and RAR algorithms with the continuous-time MAXQ-Q algorithm (Ghavamzadeh and Mahadevan, 2001). We use the task graph shown in Figure 8 in both experiments. Using this task graph, hierarchical and recursive optimal policies are the same, and therefore hierarchical and recursive optimal algorithms should converge to the same performance.

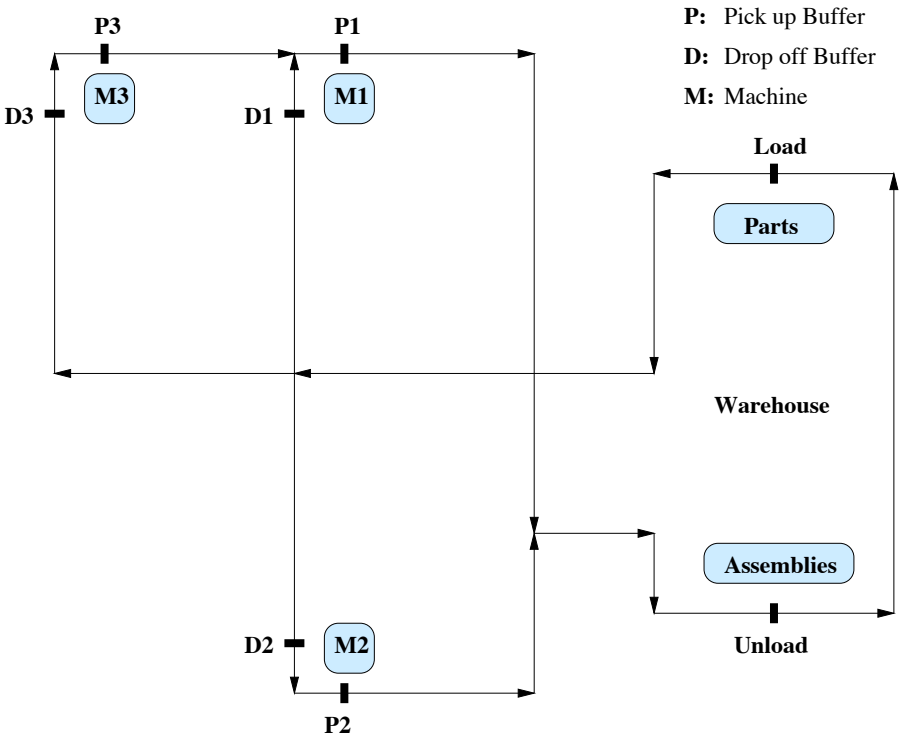


Figure 7: An AGV scheduling task. An AGV agent (not shown) carries raw materials and finished parts between machines (M1–M3) and warehouse.

The state of the environment consists of the number of parts in the pick-up and drop-off stations of each machine and whether the warehouse contains parts of each of the three types. In addition, the agent keeps track of its own location and status as a part of its state space. Thus, in the flat case, the state space consists of 33 locations, 6 buffers of size 2, 7 possible states of the AGV (carrying part1–part3, carrying assembly1–assembly3, empty), and 2 values for each part in the warehouse, that is, $33 \times 3^6 \times 7 \times 2^3 = 1,347,192$ states. Since there are 14 primitive actions (Left, Forward, Right, Put1–Put3, Pick1–Pick3, Load1–Load3, Unload, Idle) in this problem, the total number of parameters that must be learned (the size of the action-value function table) in the

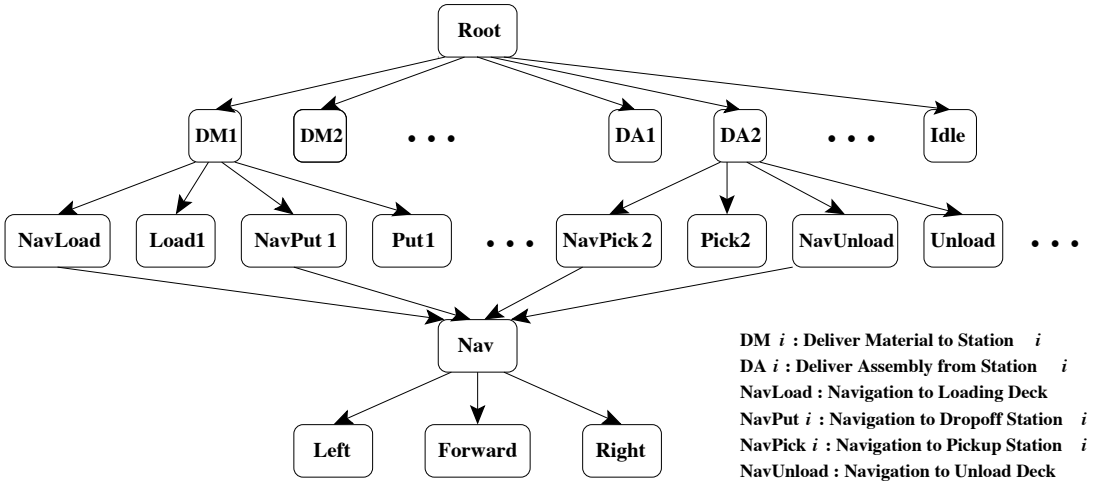


Figure 8: Task graph for the AGV scheduling task.

flat case is $1,347,192 \times 14 = 18,860,688$. *State abstraction* helps in reducing the state space considerably. Only the relevant state variables are used while storing the value functions in each node of the task graph. For example, for the 8 *Navigation* subtasks, only the location state variable is relevant and each of these subtasks can be learned with only 33 values. Tables 1 and 2 show the relevant state variables and the number of relevant states for non-primitive and primitive subtasks in the AGV scheduling problem, respectively. These tables also contain the number of parameters that must be stored by these subtasks, that is, completion function values, C, and external completion function values, CE, for non-primitive subtasks, and V values for primitive actions. The number of parameters that must be stored by a subtask is its number of relevant states times its number of children. Using Tables 1 and 2, the total number of parameters that must be learned in hierarchically and recursively optimal algorithms for this problem are equal to $10,809,150$ and $10,834,890$, respectively.[13] Both these numbers are smaller than the number of parameters that must be learned in the flat case. This state abstraction gives us a compact way of representing the value functions and speeds up the hierarchical algorithms.

The discrete-time experimental results were generated with the following model parameters. The inter-arrival time for parts at the warehouse is distributed according to a Poisson distribution.[14] The percentage of *Part1*, *Part2*, and *Part3* in the part-arrival process are 40, 35, and 25 respectively. The time required for assembling the various parts are Gamma random variables.[15] Since this is a discrete-time model for the AGV problem, we round the time x generated by these Gamma distributions to the nearest integer less than or equal to x. Table 3 shows the parameters of the

13. Note that in both recursively and hierarchically optimal algorithms, only one completion function needs to be defined at the *Root* of the hierarchy.

14. A random variable $x = 0, 1, 2, \ldots$ is said to be a Poisson random variable with parameter $\lambda > 0$, if $\Pr(x = n) = e^{-\lambda}\frac{\lambda^n}{n!}$. The mean and variance of the Poisson random variable x are both equal to λ.

15. A random variable $x \geq 0$ is said to have a Gamma distribution with parameters (κ, λ), $\kappa, \lambda > 0$, if its density function is given by $f(x) = \frac{\lambda e^{-\lambda x}(\lambda x)^{\kappa-1}}{\Gamma(\kappa)}$. The mean and variance of the Gamma random variable x are $\frac{\kappa}{\lambda}$ and $\frac{\kappa}{\lambda^2}$ respectively.

Subtask	Relevant States	Num. of Relevant States	Num. of C (CE) Values
Root	entire state space	$33 \times 3^6 \times 7 \times 2^3 = 1,347,192$	$1,347,192 \times 7 = 9,430,344$
DMi	AGV location, AGV status, status of input buffer i, whether part i exists in the warehouse	$33 \times 7 \times 3 \times 2 = 1,386$	$1,386 \times 4 = 5,544$
DAi	AGV location, AGV status, status of output buffer i	$33 \times 7 \times 3 = 693$	$693 \times 4 = 2,772$
Nav	AGV location	33	$33 \times 3 = 99$

Table 1: This table shows the relevant state variables, the number of relevant states, and the number of completion (external completion) function values C (CE) for non-primitive subtasks in the AGV scheduling problem.

Subtask	Relevant States	Num. of Relevant States = Num. of V Values
Left , Forward , Right	AGV location	33
Put$i$, Pick$i$	AGV location, AGV status, status of input/output buffer i	$33 \times 7 \times 3 = 693$
Load$i$	AGV location, AGV status, whether part i exists in the warehouse	$33 \times 7 \times 2 = 462$
Unload	AGV location, AGV status	$33 \times 7 = 231$
Idle	entire state space	$33 \times 3^6 \times 7 \times 2^3 = 1,347,192$

Table 2: This table shows the relevant state variables and the number of relevant states (which is equal to the number of V values) for primitive actions in the AGV scheduling problem.

discrete-time model. In these experiments, we used discount factors 0.9 and 0.95 for the discounted reward algorithms. Using the discount factor of 0.95 yielded a better performance.

Parameter	Distribution	Mean (steps)	Var (steps)
Assembly Time for Part1	Gamma ($\kappa = 180, \lambda = 3$)	60	20
Assembly Time for Part2	Gamma ($\kappa = 250, \lambda = 2.5$)	100	40
Assembly Time for Part3	Gamma ($\kappa = 288, \lambda = 2.4$)	120	50
Inter-Arrival Time for Parts	Poisson ($\lambda = 80$)	80	80

Table 3: Parameters of the Discrete-Time Model

The continuous-time experimental results were generated with the following model parameters. The time required for execution of each primitive action is uniformly distributed. The inter-arrival time for parts at the warehouse is distributed according to a Poisson distribution. The percentage of *Part1*, *Part2*, and *Part3* in the part-arrival process are 40, 35, and 25, respectively. The time required

for assembling the various parts are Gamma random variables. Table 4 contains the parameters of the continuous-time model.

Parameter	**Distribution**	**Mean (sec)**	**Var (sec)**
Assembly Time for Part1	Gamma ($\kappa = 180, \lambda = 3$)	60	20
Assembly Time for Part2	Gamma ($\kappa = 250, \lambda = 2.5$)	100	40
Assembly Time for Part3	Gamma ($\kappa = 288, \lambda = 2.4$)	120	50
Inter-Arrival Time for Parts	Poisson ($\lambda = 80$)	80	80
Execution Time for Primitive Actions	Uniform ($6 < t < 14$)	10	5.33

Table 4: Parameters of the Continuous-Time Model

Figure 9 compares the performance of the discrete-time hierarchically (HAR) and recursively (RAR) optimal average reward algorithms with the performance of the discrete-time discounted reward hierarchically (HDR) and recursively (MAXQ-Q) optimal algorithms on the AGV scheduling problem. All these algorithms eventually converge to the same system performance. The hierarchically optimal algorithms learn slower than the recursively optimal algorithms due to more parameters to be learned. This figure also shows the performance of relative value iteration (RVI) Q-learning (Abounadi et al., 2001), a non-hierarchical average reward RL algorithm. As shown in this figure, RVI Q-learning does not converge to the optimal throughput after 10^5 time steps. Figure 10 shows the performance of RVI Q-learning for 3×10^6 time steps. The RVI Q-learning algorithm converges to the optimal performance after over 2×10^6 time steps, where the hierarchical algorithms converge to this performance in less than 10^5 time steps as shown in Figure 9. The difference in convergence speed between flat and hierarchical algorithms becomes more significant as we increase the number of states. All the graphs in these figures are averaged over twenty runs, except the RVI Q-learning graph, which is averaged over thirty runs.

With the inter-arrival time and assembly-time parameters used in this experiment, there are time steps in which there is no part left in the warehouse. This is when the AGV must learn to take the *idle* action and wait until new parts appear in the warehouse. At first, the AGV does not serve the machines properly, and therefore parts are accumulated in the warehouse. As the AGV learns to serve the machines, the system performance goes up until the parts accumulated in the warehouse at the first learning steps are all processed. Then, the system performance goes down and eventually converges to its optimal value. This is why in Figures 9 and 10, the performance of the algorithms reaches a peak before it converges to its optimal value.

Figure 11 compares the performance of the continuous-time hierarchically (HAR) and recursively (RAR) optimal average reward algorithms with the performance of continuous-time MAXQ-Q, a continuous-time recursively optimal discounted reward RL algorithm, first presented by Ghavamzadeh and Mahadevan (2001), on the AGV scheduling problem. All the algorithms converge to the same system performance. The discounted reward algorithm, continuous-time MAXQ-Q, learns faster than both the average reward algorithms, HAR and RAR. Moreover, the hierarchically optimal average reward algorithm (HAR) learns more slowly than the recursively optimal average reward algorithm (RAR) due to more parameters to be learned. All the graphs in this figure are averaged over fifty runs.

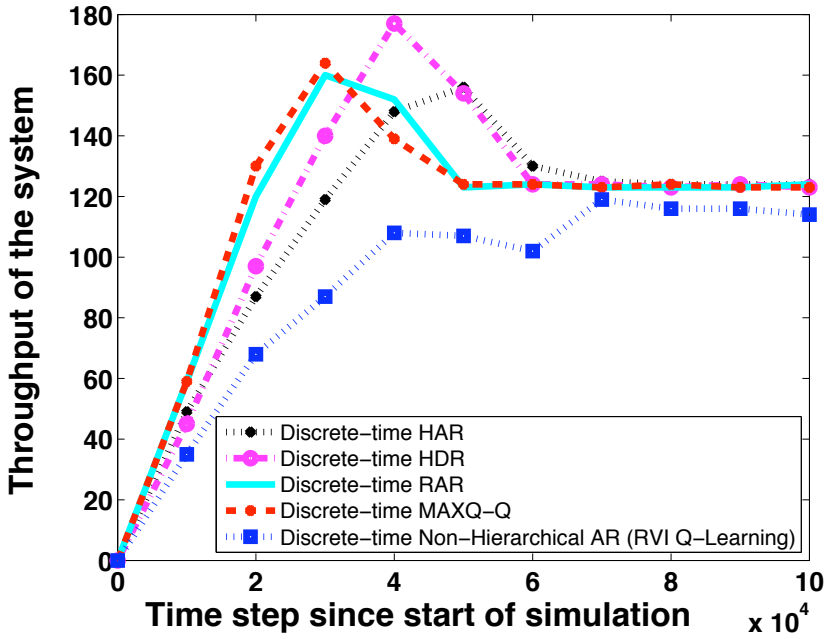


Figure 9: This figure compares the performance of the discrete-time hierarchically (HAR) and recursively (RAR) optimal average reward algorithms with the performance of the hierarchically (HDR) and recursively (MAXQ-Q) optimal discounted reward algorithms on the AGV scheduling problem. It also demonstrates the faster convergence of the hierarchical algorithms comparing to RVI Q-learning, a non-hierarchical average reward RL algorithm.

7. Conclusions and Future Work

Hierarchical reinforcement learning (HRL) is a general framework for scaling reinforcement learning (RL) to problems with large state spaces by using task (or action) structure to restrict the space of policies. Prior work in HRL, including hierarchies of abstract machines (HAMs) (Parr, 1998), options (Sutton et al., 1999; Precup, 2000), MAXQ (Dietterich, 2000), and programmable HAMs (PHAMs) (Andre and Russell, 2001; Andre, 2003), has been limited to the discrete-time discounted reward semi-Markov decision process (SMDP) model. These methods aim to find policies that maximize the long-term discounted sum of rewards. On the other hand, the average reward optimality criterion has been shown to be more appropriate for a wide class of continuing tasks than the more well-studied discounted formulation. A primary goal of continuing tasks, including manufacturing, scheduling, queuing, and inventory control, is to find policies that yield the highest expected payoff per step. Moreover, average reward optimality allows for more efficient state abstraction in HRL than the discounted reward optimality, as discussed in Section 5. Although average reward RL has been studied using both the discrete-time MDP model (Schwartz, 1993; Mahadevan, 1996; Tadepalli and Ok, 1996a,b, 1998; Marbach, 1998; Van-Roy, 1998) as well as the continuous-time SMDP

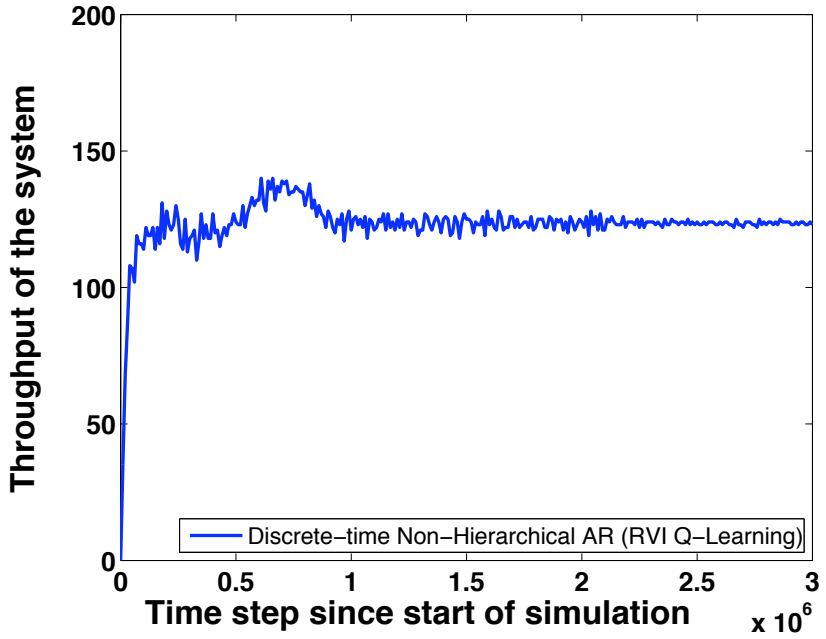


Figure 10: This figure shows the performance of RVI Q-learning, a non-hierarchical average reward algorithm, on the AGV scheduling problem. The RVI Q-learning algorithm converges to the optimal performance after over 2×10^6 time steps, where the hierarchical algorithms converge to this performance in less than 10^5 time steps as shown in Figure 9.

model (Mahadevan et al., 1997b; Wang and Mahadevan, 1999), prior work has been limited to *flat* policy representations.

In this paper, we extended previous work on HRL to the average reward setting, and presented new discrete-time and continuous-time *hierarchically optimal average reward RL* (HAR) and *recursively optimal average reward RL* (RAR) algorithms. These algorithms are based on the average reward SMDP model, and correspond to two notions of optimality in HRL: *hierarchical optimality* and *recursive optimality* (Dietterich, 2000). The HAR algorithm searches the space of policies defined by the hierarchical decomposition to find a hierarchical policy with maximum *global gain* (the gain of the Markov chain that results from flattening the hierarchy using a hierarchical policy). In the *recursively optimal average reward RL* setting, the formulation of learning algorithms directly depends on the local optimality criterion used for each subtask in the hierarchy. The RAR algorithm treats non-primitive subtasks as continuing average reward problems and solve them by maximizing their local gain given the policies of their children. We demonstrated that the policy learned for each subtask by the RAR algorithm is not necessarily context free, and as a result the algorithms do not find a recursively optimal average reward policy in general. However, we showed that the RAR algorithm finds a recursively optimal average reward policy when the *initial distribution invariance* condition holds. We used two automated guided vehicle (AGV) scheduling tasks as experimental testbeds to study the empirical performance of the proposed algorithms. The first problem is a relatively simple AGV scheduling task, in which the hierarchically and recursively optimal policies

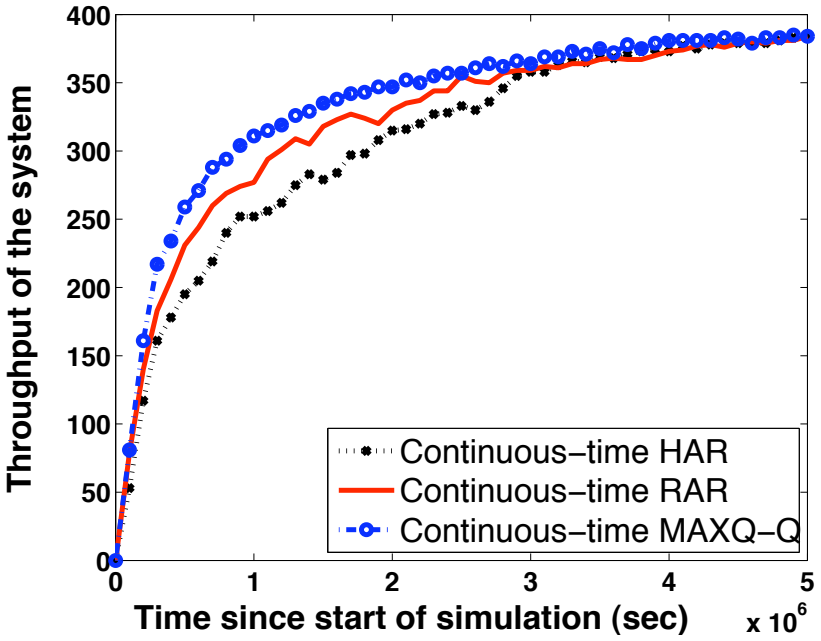


Figure 11: This figure compares the performance of the continuous-time hierarchically (HAR) and recursively (RAR) optimal average reward algorithms with the performance of continuous-time MAXQ-Q, a continuous-time recursively optimal discounted reward RL algorithm, on the AGV scheduling problem.

are different. We compared the proposed algorithms with three other HRL methods, including a hierarchically optimal discounted reward algorithm and a recursively optimal discounted reward algorithm on this problem. The results demonstrate the difference between the optimalities achieved by these algorithms. The second problem is a relatively larger AGV scheduling task. We modeled this problem using both discrete-time and continuous-time models. We used a hierarchical task decomposition with which the hierarchically and recursively optimal policies are the same for this problem. We compared the performance of the proposed algorithms with a hierarchically optimal discounted reward algorithm and a recursively optimal discounted reward algorithm, as well as a flat average reward algorithm in this problem. The results showed that the proposed hierarchical average reward algorithms converge to the same performance as their discounted reward counterparts.

There are a number of directions for future work. An immediate question that arises is proving the asymptotic convergence of the algorithms to hierarchically and recursively optimal policies. These results should provide some theoretical validity to the proposed algorithms, in addition to their empirical efficiency demonstrated in this paper. Studying other local optimality criteria for subtasks in a hierarchy is an interesting problem that needs to be addressed. It helps to develop more efficient *recursively optimal average reward RL* algorithms. It is also clear that our hierarchical average reward framework can be applied to many other manufacturing and robotics problems besides the AGV task.

Acknowledgments

We would like to thank Prasad Tadepalli, David Andre, Bernhard Hengst, and Balaraman Ravindran for their comments. The computational experiments were carried out in Autonomous Agents Laboratory in the Department of Computer Science and Engineering at Michigan State University under the Defense Advanced Research Projects Agency, DARPA contract No. DAANO2-98-C-4025, and in Autonomous Learning Laboratory in the Department of Computer Science at the University of Massachusetts Amherst under NASA contract No. NAg-1445 #1. Support for this research was also provided in part by the National Science Foundation under grants NSF IIS-0534999 and ECS 0218125. Part of this article was written while the first author was supported by iCORE Canada at the Department of Computing Science at the University of Alberta.

Appendix A. Index of Symbols

Here we present a list of the symbols used in this paper to provide a handy reference.

Notation	Definition
$\mathbb{R}$	set of real numbers
$\mathbb{N}$	set of natural numbers
$\mathbf{E}$	expected value
$\mathcal{M}$	a MDP model
$\mathcal{S}$	set of states of a SMDP
$\mathcal{A}$	set of actions of a SMDP
$\mathcal{P}$	multi-step transition probability function of a SMDP
$\mathcal{R}$	reward function of a SMDP
$r(s,a)$	reward of taking action a in state s
I	initial state distribution of a SMDP
μ	a policy
$\mu(a\|s)$	probability that policy μ selects action a in state s
μ^*	optimal policy
γ	discount factor
α	learning rate parameter
V^μ	hierarchical value function of hierarchical policy μ
$\hat{V}^\mu$	projected value function of hierarchical policy μ
V^*	optimal value function
Q^μ	hierarchical action-value function of hierarchical policy μ
$\hat{Q}^\mu$	projected action-value function of hierarchical policy μ
Q^*	optimal action-value function
g^μ	average reward or gain of policy μ
g^μ	global gain under hierarchical policy μ
g_i^μ	local gain of subtask M_i under hierarchical policy μ
g^*	optimal gain or gain of optimal policy
H^μ	average-adjusted value function of policy μ
H^μ	hierarchical average-adjusted value function of hierarchical policy μ
$\hat{H}^\mu$	projected average-adjusted value function of hierarchical policy μ
H^*	optimal average-adjusted value function

Notation	Definition
L^μ	average-adjusted action-value function of policy μ
L^μ	hierarchical average-adjusted action-value function of hierarchical policy μ
$\hat{L}^\mu$	projected average-adjusted action-value function of hierarchical policy μ
L^*	optimal average-adjusted action-value function
$P(s',N\|s,a)$	probability that action a will cause the system to transition from state s to state s' in N time steps
$F(s'\|s,a)$	probability that a SMDP occupies state s' at the next decision epoch given that the agent takes action a in state s at the current decision epoch
F^μ	transition probability matrix of the embedded Markov chain of a SMDP for policy μ
$\bar{F}^\mu$	limiting matrix of the embedded Markov chain of a SMDP for policy μ
$y(s,a)$	expected number of transition steps until the next decision epoch in a SMDP
$\mathcal{H}$	a hierarchy
M_i	subtask M_i in a hierarchy
S_i	set of states for subtask M_i in a hierarchy
$\|S_i\|$	cardinality of set of states S_i
A_i	set of actions for subtask M_i in a hierarchy
R_i	reward function for subtask M_i in a hierarchy
I_i	initiation set for subtask M_i in a hierarchy
T_i	termination set for subtask M_i in a hierarchy
μ_i	a policy for subtask M_i in a hierarchy
μ	a hierarchical policy
P_i^μ	multi-step transition probability function of subtask M_i
$P_i^\mu(s',N\|s)$	probability that action $\mu_i(s)$ causes transition from state s to state s' in N primitive steps under hierarchical policy μ
F_i^μ	multi-step abstract transition probability function of subtask M_i
$F_i^\mu(s',N\|s)$	probability of transition from state s to state s' in N abstract actions taken by subtask M_i under hierarchical policy μ
$F_i^\mu(s',1\|s)$	transition probability of the embedded Markov chain at subtask M_i under hierarchical policy μ (same as $F_i^\mu(s'\|s)$)
m^μ	transition probability function of the Markov chain that results from flattening the hierarchy using the hierarchical policy μ
$m^\mu(s'\|s)$	probability that hierarchical policy μ will cause the system to transition from state s to state s' at the level of primitive actions
m^μ	transition probability matrix of the Markov chain that results from flattening the hierarchy using the hierarchical policy μ
$\overline{m}^\mu$	limiting matrix of the Markov chain that results from flattening the hierarchy using the hierarchical policy μ
Ω	set of possible values for Task-Stack in a hierarchy
$X = \Omega \times S$	joint state space of Task-Stack values and states in a hierarchy
$x = (\omega, s)$	joint state value x formed by Task-Stack value ω and state value s in a hierarchy
$\omega \nearrow i$	popping subtask M_i off Task-Stack with content ω in a hierarchy
$i \searrow \omega$	pushing subtask M_i onto Task-Stack with content ω in a hierarchy
C^μ	completion function of hierarchical policy μ
CE^μ	external completion function of hierarchical policy μ

References

J. Abounadi, D. P. Bertsekas, and V. S. Borkar. Learning algorithms for Markov decision processes with average cost. *SIAM Journal on Control and Optimization*, 40:681–698, 2001.

D. Andre. *Programmable Reinforcement Learning Agents*. PhD thesis, University of California at Berkeley, 2003.

D. Andre and S. J. Russell. Programmable reinforcement learning agents. In *Proceedings of Advances in Neural Information Processing Systems 13*, pages 1019–1025. MIT Press, 2001.

D. Andre and S. J. Russell. State abstraction for programmable reinforcement learning agents. In *Proceedings of the Eighteenth National Conference on Artificial Intelligence*, pages 119–125, 2002.

A. Barto and S. Mahadevan. Recent advances in hierarchical reinforcement learning. *Discrete Event Systems (Special Issue on Reinforcement Learning)*, 13:41–77, 2003.

R. Bellman. *Dynamic Programming*. Princeton University Press, 1957.

S. Bradtke and M. Duff. Reinforcement learning methods for continuous-time Markov decision problems. In *Proceedings of Advances in Neural Information Processing Systems 7*, pages 393–400. MIT Press, 1995.

R. Crites and A. Barto. Elevator group control using multiple reinforcement learning agents. *Machine Learning*, 33:235–262, 1998.

P. Dayan and G. Hinton. Feudal reinforcement learning. In *Proceedings of Advances in Neural Information Processing Systems 5*, pages 271–278, 1993.

T. Dietterich. Hierarchical reinforcement learning with the MAXQ value function decomposition. *Journal of Artificial Intelligence Research*, 13:227–303, 2000.

M. Ghavamzadeh and S. Mahadevan. Continuous-time hierarchical reinforcement learning. In *Proceedings of the Eighteenth International Conference on Machine Learning*, pages 186–193, 2001.

M. Ghavamzadeh and S. Mahadevan. Hierarchically optimal average reward reinforcement learning. In *Proceedings of the Nineteenth International Conference on Machine Learning*, pages 195–202, 2002.

R. Howard. *Dynamic Programming and Markov Processes*. MIT Press, 1960.

R. Howard. *Dynamic Probabilistic Systems: Semi-Markov and Decision Processes*. John Wiley and Sons., 1971.

L. Kaelbling. Hierarchical reinforcement learning: Preliminary results. In *Proceedings of the Tenth International Conference on Machine Learning*, pages 167–173, 1993a.

L. Kaelbling. Learning to achieve goals. In *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence*, pages 1094–1098, 1993b.

S. Mahadevan. Average reward reinforcement learning: foundations, algorithms, and empirical results. *Machine Learning*, 22:159–196, 1996.

S. Mahadevan and J. Connell. Automatic programming of behavior-based robots using reinforcement learning. *Artificial Intelligence*, 55(2-3):311–365, 1992.

S. Mahadevan, N. Khaleeli, and N. Marchalleck. Designing agent controllers using discrete-event Markov models. In *Proceedings of the AAAI Fall Symposium on Model-Directed Autonomous Systems*, 1997a.

S. Mahadevan, N. Marchalleck, T. Das, and A. Gosavi. Self-improving factory simulation using continuous-time average reward reinforcement learning. In *Proceedings of the Fourteenth International Conference on Machine Learning*, pages 182–190, 1997b.

P. Marbach. *Simulated-Based Methods for Markov Decision Processes*. PhD thesis, Massachusetts Institute of Technology, 1998.

A. Ng, D. Harada, and S. Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In *Proceedings of the Sixteenth International Conference on Machine Learning*, pages 278–287, 1999.

A. Ng, H. Kim, M. Jordan, and S. Sastry. Autonomous helicopter flight via reinforcement learning. In *Proceedings of Advances in Neural Information Processing Systems 16*. MIT Press, 2004.

R. Parr. *Hierarchical Control and Learning for Markov Decision Processes*. PhD thesis, University of California at Berkeley, 1998.

D. Precup. *Temporal Abstraction in Reinforcement Learning*. PhD thesis, University of Massachusetts Amherst, 2000.

M. Puterman. *Markov Decision Processes*. Wiley Interscience, 1994.

A. Schwartz. A reinforcement learning method for maximizing undiscounted rewards. In *Proceedings of the Tenth International Conference on Machine Learning*, pages 298–305, 1993.

S. Seri and P. Tadepalli. Model-based hierarchical average-reward reinforcement learning. In *Proceedings of the Nineteenth International Conference on Machine Learning*, pages 562–569, 2002.

S. Singh. Transfer of learning by composing solutions of elemental sequential tasks. *Machine Learning*, 8:323–339, 1992.

S. Singh and D. Bertsekas. Reinforcement learning for dynamic channel allocation in cellular telephone systems. In *Proceedings of Advances in Neural Information Processing Systems 9*, pages 974–980, 1996.

R. Sutton, D. Precup, and S. Singh. Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, 112:181–211, 1999.

P. Tadepalli and D. Ok. Scaling up average reward reinforcement learning by approximating the domain models and the value function. In *Proceedings of the Thirteenth International Conference on Machine Learning*, pages 471–479, 1996a.

P. Tadepalli and D. Ok. Auto-exploratory average reward reinforcement learning. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, pages 881–887, 1996b.

P. Tadepalli and D. Ok. Model-based average reward reinforcement learning. *Artificial Intelligence*, 100:177–224, 1998.

G. Tesauro. TD-Gammon, a self-teaching backgammon program, achieves master-level play. *Neural Computation*, 6:215–219, 1994.

B. Van-Roy. *Learning and Value Function Approximation in Complex Decision Processes*. PhD thesis, Massachusetts Institute of Technology, 1998.

G. Wang and S. Mahadevan. Hierarchical optimization of policy-coupled semi-Markov decision processes. In *Proceedings of the Sixteenth International Conference on Machine Learning*, pages 464–473, 1999.

W. Zhang and T. Dietterich. A reinforcement learning approach to job-shop scheduling. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, pages 1114–1120, 1995.

Journal of Machine Learning Research 8 (2007) 2671-2699 Submitted 11/06; Revised 2/07; Published 12/07

Ranking the Best Instances

Stéphan Clémençon CLEMENCO@ENST.FR
Département TSI Signal et Images - LTCI UMR GET/CNRS 5141
Ecole Nationale Supérieure des Télécommunications
37-39, rue Dareau
75014 Paris, France

Nicolas Vayatis VAYATIS@CMLA.ENS-CACHAN.FR
Centre de Mathématiques et de Leurs Applications - UMR CNRS 8536
Ecole Normale Supérieure de Cachan - UniverSud
61, avenue du Président Wilson
94 235 Cachan cedex, France

Editor: Yoram Singer

Abstract

We formulate a local form of the bipartite ranking problem where the goal is to focus on the best instances. We propose a methodology based on the construction of real-valued scoring functions. We study empirical risk minimization of dedicated statistics which involve empirical quantiles of the scores. We first state the problem of *finding* the best instances which can be cast as a classification problem with mass constraint. Next, we develop special performance measures for the local ranking problem which extend the Area Under an ROC Curve (AUC) criterion and describe the optimal elements of these new criteria. We also highlight the fact that the goal of ranking the best instances cannot be achieved in a stage-wise manner where first, the best instances would be tentatively identified and then a standard AUC criterion could be applied. Eventually, we state preliminary statistical results for the local ranking problem.

Keywords: ranking, ROC curve and AUC, empirical risk minimization, fast rates

1. Introduction

The first takes all the glory, the second takes nothing. In applications where ranking is at stake, people often focus on the best instances. When scanning the results from a query on a search engine, we rarely go beyond the one or two first pages on the screen. In the different context of credit risk screening, credit establishments elaborate scoring rules as reliability indicators and their main concern is to identify risky prospects especially among the top scores. In medical diagnosis, test scores indicate the odds for a patient to be healthy given a series of measurements (age, blood pressure, ...). There again a particular attention is given to the "best" instances not to miss a possible diseased patient among the highest scores. These various situations can be formulated in the setup of bipartite ranking where one observes i.i.d. copies of a random pair (X,Y) with X being an observation vector describing the instance (web page, debtor, patient) and Y a binary label assigning to one population or the other (relevant vs. non relevant, good vs. bad, healthy vs. diseased). In this problem, the goal is to rank the instances instead of simply classifying them. There is a growing literature on the ranking problem in the field of Machine Learning but most of it considers the Area under the ROC Curve (also known as the AUC) criterion as a measure of performance of the

ranking rule (Cortes and Mohri, 2004; Freund et al., 2003; Rudin et al., 2005; Agarwal et al., 2005). In a previous work, we have mentioned that the bipartite ranking problem under the AUC criterion could be interpreted as a classification problem with pairs of observations (Clémençon et al., 2005). But the limit of this approach is that it weights uniformly the pairs of items which are badly ranked. Therefore it does not permit to distinguish between ranking rules making the same number of mistakes but in very different parts of the ROC curve. The AUC is indeed a global criterion which does not allow to concentrate on the "best" instances. Special performance measures, such as the Discounted Cumulative Gain (DCG) criterion, have been introduced by practitioners in order to weight instances according to their rank (Järvelin and Kekäläinen, 2000) but providing theory for such criteria and developing empirical risk minimization strategies still is a very open issue. Recent works by Rudin (2006), Cossock and Zhang (2006), and Li et al. (2007) reveal that there are several possibilities when designing ranking algorithms with focus on the top-rated instances. In the present paper, we focus on statistical aspects rather than algorithmic. We extend the results of our previous work in Clémençon et al. (2005) and set theoretical grounds for the problem of local ranking. The methodology we propose is based on the selection of a real-valued scoring function for which we formulate appropriate performance measures generalizing the AUC criterion. We point out that ranking the best instances is an involved task as it is a two-fold problem: (i) find the best instances and (ii) provide a good ranking on these instances. The fact that these two goals cannot be considered independently will be highlighted in the paper. Despite this observation, we will first formulate the issue of finding the best instances which is to be understood as a toy problem for our purpose. This problem corresponds to a *binary classification problem with a mass constraint* (where the proportion u_0 of +1 labels predicted by the classifiers is fixed) and it might present an interest *per se*. The main complication here has to do with the necessity of performing quantile estimation which affects the performance of statistical procedures. Our proof technique was inspired by the former work of Koul (2002) in the context of R-estimation where similar statistics, known as linear signed rank statistics, arise. By exploiting the structure of such statistics, we are able to establish noise conditions in a similar way as in Clémençon et al. (To appear) where we had to deal with performance criteria based on U-statistics. Under such conditions, we prove that rates of convergence up to $n^{-2/3}$ can be guaranteed for the empirical risk minimizer in the classification problem with mass constraint. Another contribution of the paper lies in our study of the optimality issue for the local ranking problem. We discuss how focusing on best instances affects the ROC curve and the AUC criterion. We propose a family of possible performance measures for the problem of ranking the best instances. In particular, we show that widespread ideas in the biostatistics literature about the partial AUC (see Dodd and Pepe, 2003) turn out to be questionable with respect to optimality considerations. We also point out that the empirical risks for local ranking are closely related to generalized Wilcoxon statistics.

The rest of the paper is organized as follows. We first state the problem of finding the best instances and study the performance of empirical risk minimization in this setup (Section 2). We also explore the conditions on the distribution in order to recover fast rates of convergence. In Section 3 we formulate performance measures for local ranking and provide extensions of the AUC criterion. Eventually (Section 4), we state some preliminary statistical results on empirical risk minimization of these new criteria.

2. Finding the Best Instances

In the present section, we have a limited goal which is only to determine the best instances without bothering with their order in the list. By considering this subproblem, we will identify the main technical issues involved in the sequel. It also permits to introduce the main notations of the paper.

Just as in standard binary classification, we consider the pair of random variables (X,Y) where X is an observation vector in a measurable space $\mathcal{X}$ and Y is a binary label in $\{-1,+1\}$. The distribution of (X,Y) can be described by the pair (μ,η) where μ is the marginal distribution of X and η is the a posteriori distribution defined by $\eta(x) = \mathbb{P}\{Y = 1 \mid X = x\}$, $\forall x \in \mathcal{X}$. We define the *rate of best instances* as the proportion of best instances to be considered and denote it by $u_0 \in (0,1)$. We denote by $Q(\eta, 1-u_0)$ the $(1-u_0)$-quantile of the random variable $\eta(X)$. Then the *set of best instances at rate* u_0 is given by:

$$C^*_{u_0} = \{x \in \mathcal{X} \mid \eta(x) \geq Q(\eta, 1-u_0)\}\,.$$

We mention two trivial properties of the set $C^*_{u_0}$ which will be important in the sequel:

- MASS CONSTRAINT: we have $\mu\big(C^*_{u_0}\big) = \mathbb{P}\left\{X \in C^*_{u_0}\right\} = u_0$,
- INVARIANCE PROPERTY: as a functional of η, the set $C^*_{u_0}$ is invariant to strictly increasing transforms of η.

The problem of finding a proportion u_0 of the best instances boils down to the estimation of the unknown set $C^*_{u_0}$ on the basis of empirical data. Before turning to the statistical analysis of the problem, we first relate it to binary classification.

2.1 A Classification Problem with a Mass Constraint

A classifier is a measurable function $g : \mathcal{X} \to \{-1,+1\}$ and its performance is measured by the classification error $L(g) = \mathbb{P}\{Y \neq g(X)\}$. Let $u_0 \in (0,1)$ be fixed. Denote by $g^*_{u_0} = 2\mathbb{I}_{C^*_{u_0}} - 1$ the classifier predicting +1 on the set of best instances $C^*_{u_0}$ and -1 on its complement. The next proposition shows that $g^*_{u_0}$ is an optimal element for the problem of minimization of $L(g)$ over the family of classifiers g satisfying the *mass constraint* $\mathbb{P}\{g(X) = 1\} = u_0$.

Proposition 1 *For any classifier $g : \mathcal{X} \to \{-1,+1\}$ such that $g(x) = 2\mathbb{I}_C(x) - 1$ for some subset C of $\mathcal{X}$ and $\mu(C) = \mathbb{P}\{g(X) = 1\} = u_0$, we have*

$$L^*_{u_0} \stackrel{\circ}{=} L\big(g^*_{u_0}\big) \leq L(g)\,.$$

Furthermore, we have

$$L^*_{u_0} = 1 - Q(\eta, 1-u_0) + (1-u_0)(2Q(\eta, 1-u_0) - 1) - \mathbb{E}\left(|\eta(X) - Q(\eta, 1-u_0)|\right),$$

and

$$L(g) - L\big(g^*_{u_0}\big) = 2\mathbb{E}\left(|\eta(X) - Q(\eta, 1-u_0)|\, \mathbb{I}_{C^*_{u_0} \Delta C}(X)\right),$$

where Δ denotes the symmetric difference operation between two subsets of $\mathcal{X}$.

PROOF. For simplicity, we temporarily change the notation and set $q = Q(\eta, 1-u_0)$. Then, for any classifier g satisfying the constraint $\mathbb{P}\{g(X)=1\} = u_0$, we have

$$L(g) = \mathbb{E}\left((\eta(X)-q)\mathbb{I}_{[g(X)=-1]} + (q-\eta(X))\mathbb{I}_{[g(X)=+1]}\right) + (1-u_0)q + (1-q)u_0 \ .$$

The statements of the proposition immediately follow. ■

There are several progresses in the field of classification theory where the aim is to introduce constraints in the classification procedure or to adapt it to other problems. We relate our formulation to other approaches in the following remarks.

Remark 2 (CONNECTION TO HYPOTHESIS TESTING). *The implicit asymmetry in the problem due to the emphasis on the best instances is reminiscent of the statistical theory of hypothesis testing. We can formulate a test of simple hypothesis by taking the* null assumption *to be* $H_0 : Y = -1$ *and the* alternative assumption *being* $H_1 : Y = +1$. *We want to decide which hypothesis is true given the observation* X. *Each classifier* g *provides a test statistic* $g(X)$. *The performance of the test is then described by its type I error* $\alpha(g) = \mathbb{P}\{g(X)=1 \mid Y=-1\}$ *and its power* $\beta(g) = \mathbb{P}\{g(X)=1 \mid Y=+1\}$. *We point out that if the classifier* g *satisfies a mass constraint, then we can relate the classification error with the type I error of the test defined by* g *through the relation:*

$$L(g) = 2(1-p)\alpha(g) + p - u_0$$

where $p = \mathbb{P}\{Y=1\}$, *and similarly, we have:* $L(g) = 2p(1-\beta(g)) - p - u_0$. *Therefore, the optimal classifier minimizes the type I error (maximizes the power) among all classifiers with the same mass constraint. In some applications, it is more relevant to fix a constraint on the probability of a false alarm (type I error) and maximize the power. This question is explored in a recent paper by Scott (2005) (see also Scott and Nowak, 2005).*

Remark 3 (CONNECTION WITH REGRESSION LEVEL SET ESTIMATION) *We mention that the estimation of the level sets of the regression function has been studied in the statistics literature (Cavalier, 1997) (see also Tsybakov, 1997 and Willett and Nowak, 2006) as well as in the learning literature, for instance in the context of anomaly detection (Steinwart et al., 2005; Scott and Davenport, 2006, to appear; Vert and Vert, 2006). In our framework of classification with mass constraint, the threshold defining the level set involves the quantile of the random variable* $\eta(X)$.

Remark 4 (CONNECTION WITH THE MINIMUM VOLUME SET APPROACH) *Although the point of view adopted in this paper is very different, the problem described above may be formulated in the framework of* minimum volume sets *learning as considered in Scott and Nowak (2006). As a matter of fact, the set* $C^*_{u_0}$ *may be viewed as the solution of the constrained optimization problem:*

$$\min_C \mathbb{P}\{X \in C \mid Y=-1\}$$

over the class of measurable sets C, *subject to*

$$\mathbb{P}\{X \in C\} \geq u_0 \ .$$

The main difference in our case comes from the fact that the constraint on the volume set has to be estimated using the data while in Scott and Nowak (2006) it is computed from a known reference

measure. We believe that learning methods for minimum volume set estimation may hopefully be extended to our setting. A natural way to do it would consist in replacing conditional distribution of X given $Y=-1$ by its empirical counterpart. This is beyond the scope of the present paper but will be the subject of future investigation.

2.2 Empirical Risk Minimization

We now investigate the estimation of the set $C^*_{u_0}$ of best instances at rate u_0 based on training data. Suppose that we are given n i.i.d. copies $(X_1,Y_1),\cdots,(X_n,Y_n)$ of the pair (X,Y). Since we have the ranking problem in mind, our methodology will consist in building the candidate sets from a class $\mathcal{S}$ of real-valued scoring functions $s:\mathcal{X}\to\mathbb{R}$. Indeed, we consider sets of the form

$$C_s \stackrel{\circ}{=} C_{s,u_0} = \{x\in\mathcal{X} \mid s(x)\geq Q(s,1-u_0)\}\,,$$

where s is an element of $\mathcal{S}$ and $Q(s,1-u_0)$ is the $(1-u_0)$-quantile of the random variable $s(X)$. Note that such sets satisfy the same properties of $C^*_{u_0}$ with respect to mass constraint and invariance to strictly increasing transforms of s.

From now on, we will take the simplified notation:

$$L(s) \stackrel{\circ}{=} L(s,u_0) \stackrel{\circ}{=} L(C_s) = \mathbb{P}\{Y\cdot(s(X)-Q(s,1-u_0))<0\}\,.$$

A scoring function minimizing the quantity

$$L_n(s) = \frac{1}{n}\sum_{i=1}^{n}\mathbb{I}\{Y_i\cdot(s(X_i)-Q(s,1-u_0))<0\}\,.$$

is expected to approximately minimize the true error $L(s)$, but the quantile depends on the unknown distribution of X. In practice, one has to replace $Q(s,1-u_0)$ by its empirical counterpart $\widehat{Q}(s,1-u_0)$ which corresponds to the empirical quantile. We will thus consider, instead of $L_n(s)$, the *empirical error*:

$$\widehat{L}_n(s) = \frac{1}{n}\sum_{i=1}^{n}\mathbb{I}\{Y_i\cdot(s(X_i)-\widehat{Q}(s,1-u_0))<0\}\,.$$

Note that $\widehat{L}_n(s)$ is a complicated statistic since the empirical quantile involves all the instances $X_1,\ldots,X_n$. We also mention that $\widehat{L}_n(s)$ is a biased estimate of the classification error $L(s)$ of the classifier $g_s(x)=2\mathbb{I}\{s(x)\geq Q(s,1-u_0)\}-1$.

We introduce some more notations. Set, for all $t\in\mathbb{R}$:

- $F_s(t)=\mathbb{P}\{s(X)\leq t\}$
- $G_s(t)=\mathbb{P}\{s(X)\leq t \mid Y=+1\}$
- $H_s(t)=\mathbb{P}\{s(X)\leq t \mid Y=-1\}\,.$

The functions F_s (respectively G_s, H_s) denote the cumulative distribution function (cdf) of $s(X)$ (respectively, given $Y=1$, given $Y=-1$). We recall that the definition of the quantiles of (the distribution of) a random variable involves the notion of generalized inverse F^{-1} of a function F:

$$F^{-1}(z)=\inf\{t\in\mathbb{R} \mid F(t)\geq z\}\,.$$

Thus, we have, for all $v \in (0,1)$:

$$Q(s,v) = F_s^{-1}(v) \quad \text{and} \quad \hat{Q}(s,v) = \hat{F}_s^{-1}(v)$$

where $\hat{F}_s$ is the empirical cdf of $s(X)$: $\hat{F}_s(t) = \frac{1}{n}\sum_{i=1}^{n} \mathbb{I}\{s(X_i) \leq t\}, \forall t \in \mathbb{R}$.

Without loss of generality, we will assume that all scoring functions in $\mathcal{S}$ take their values in $(0,\lambda)$ for some $\lambda > 0$. We now turn to study the performance of minimizers of $\hat{L}_n(s)$ over a class $\mathcal{S}$ of scoring functions defined by

$$\hat{s}_n = \arg\min_{s \in \mathcal{S}} \hat{L}_n(s).$$

Our first main result is an excess risk bound for the empirical risk minimizer $\hat{s}_n$ over a class $\mathcal{S}$ of uniformly bounded scoring functions. In the following theorem, we consider that the level sets of scoring functions from the class $\mathcal{S}$ form a Vapnik-Chervonenkis (VC) class of sets.

Theorem 5 *We assume that*

(i) the class $\mathcal{S}$ is symmetric (that is, if $s \in \mathcal{S}$ then $\lambda - s \in \mathcal{S}$) and is a VC major class of functions with VC dimension V.

(ii) the family $\mathcal{K} = \{ G_s, H_s \, : \, s \in \mathcal{S} \}$ of cdfs satisfies the following property: any $K \in \mathcal{K}$ has left and right derivatives, denoted by K'_+ and K'_-, and there exist strictly positive constants b, B such that $\forall (K,t) \in \mathcal{K} \times (0,\lambda)$,

$$b \leq \left|K'_+(t)\right| \leq B \quad \text{and} \quad b \leq \left|K'_-(t)\right| \leq B \, .$$

For any $\delta > 0$, we have, with probability larger than $1-\delta$,

$$L(\hat{s}_n) - \inf_{s \in \mathcal{S}} L(s) \leq c_1 \sqrt{\frac{V}{n}} + c_2 \sqrt{\frac{ln(1/\delta)}{n}},$$

for some positive constants c_1, c_2.

The following remarks provide some insights on conditions (i) and (ii) of the theorem.

Remark 6 (ON THE COMPLEXITY ASSUMPTION) *On the terminology of major sets and major classes, we refer to Dudley (1999). In the proof, we need to control empirical processes indexed by sets of the form $\{x \, : \, s(x) \geq t\}$ or $\{x \, : \, s(x) \leq t\}$. Condition (i) guarantees that these sets form a VC class of sets.*

Remark 7 (ON THE CHOICE OF THE CLASS $\mathcal{S}$ OF SCORING FUNCTIONS) *In order to grasp the meaning of condition (ii) of the theorem, we consider the one-dimensional case with real-valued scoring functions. Assume that the distribution of the random variable X_i has a bounded density f with respect to Lebesgue measure. Assume also that scoring functions s are differentiable except, possibly, at a finite number of points, and derivatives are denoted by s'. Denote by f_s the density of $s(X)$. Let $t \in (0,\lambda)$ and denote by x_1, ..., x_p the real roots of the equation $s(x) = t$. We can express the density of $s(X)$ thanks to the change-of-variable formula (see, for instance, Papoulis, 1965):*

$$f_s(t) = \frac{f(x_1)}{s'(x_1)} + \ldots + \frac{f(x_p)}{s'(x_p)} \, .$$

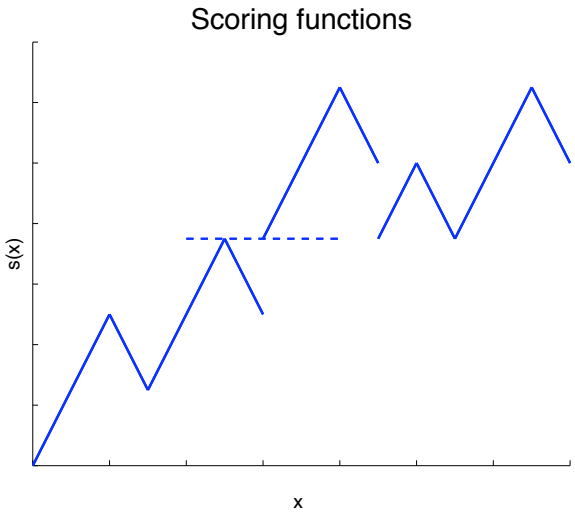


Figure 1: Typical example of a scoring function.

This shows that the scoring functions should not present neither flat nor steep parts. We can take for instance, the class $\mathcal{S}$ to be the class of linear-by-parts functions with a finite number of local extrema and with uniformly bounded left and right derivatives: $\forall s \in \mathcal{S}, \forall x, m \leq s'_+(x) \leq M$ and $m \leq s'_-(x) \leq M$ for some strictly positive constants m, and M (see Figure 1). Note that any subinterval of $[0,\lambda]$ has to be in the range of scoring functions s (if not, some elements of $\mathcal{K}$ will present a plateau*). In fact, the proof requires such a behavior only in the vicinity of the points corresponding to the quantiles $Q(s,1-u_0)$ for all $s \in \mathcal{S}$.*

PROOF. Set $v_0 = 1-u_0$. By a standard argument (see, for instance, Devroye et al., 1996), we have:

$$\begin{aligned} L(\hat{s}_n) - \inf_{s\in\mathcal{S}} L(s) &\leq 2\sup_{s\in\mathcal{S}} \left|\hat{L}_n(s) - L(s)\right| \\ &\leq 2\sup_{s\in\mathcal{S}} \left|\hat{L}_n(s) - L_n(s)\right| + 2\sup_{s\in\mathcal{S}} |L_n(s) - L(s)| \ . \end{aligned}$$

Note that the second term in the bound is an empirical process whose behavior is well-known. In our case, assumption (i) implies that the class of sets $\{x : s(x) \geq Q(s,v_0)\}$ indexed by scoring functions s has a VC dimension smaller than V. Hence, we have by a concentration argument combined with a VC bound for the expectation of the supremum (see, for instance, Lugosi), for any $\delta > 0$, with probability larger than $1-\delta$,

$$\sup_{s\in\mathcal{S}} |L_n(s) - L(s)| \leq c\sqrt{\frac{V}{n}} + c'\sqrt{\frac{ln(1/\delta)}{n}}$$

for universal constants c, c'.

The novel part of the analysis lies in the control of the first term and we now show how to handle it. Following the work of Koul (2002), we set the following notations:

$$M(s,v) = \mathbb{P}\left\{Y \cdot \left(s(X) - Q(s,v)\right) < 0\right\},$$

$$U_n(s,v) = \frac{1}{n}\sum_{i=1}^{n} \mathbb{I}\{Y_i \cdot \big(s(X_i) - Q(s,v)\big) < 0\} - M(s,v)\ .$$

and note that $U_n(s,v)$ is centered. In particular, we have:

$$L_n(s) = U_n(s,v_0) + M(s,v_0)\ .$$

As $Q(s,v) = F_s^{-1}(v)$, we have $Q(s, F_s \circ \hat{F}_s^{-1}(v)) = \hat{F}_s^{-1}(v) = \hat{Q}(s,v)$ and then

$$\hat{L}_n(s) = U_n(s, F_s \circ \hat{F}_s^{-1}(v_0)) + M(s, F_s \circ \hat{F}_s^{-1}(v_0))\ .$$

Note that $M(s, F_s \circ \hat{F}_s^{-1}(v_0)) = \mathbb{P}\left\{Y \cdot \big(s(X) - \hat{Q}(s,v)\big) < 0 \mid D_n\right\}$ where D_n denotes the sample $(X_1,Y_1), \cdots, (X_n,Y_n)$.

We then have the following decomposition, for any $s \in \mathcal{S}$ and $v_0 \in (0,1)$:

$$\left|\hat{L}_n(s) - L_n(s)\right| \leq \left|U_n(s, F_s \circ \hat{F}_s^{-1}(v_0)) - U_n(s,v_0)\right| + \left|M(s, F_s \circ \hat{F}_s^{-1}(v_0)) - M(s,v_0)\right|\ .$$

Recall the notation $p = \mathbb{P}\{Y = 1\}$. Since $M(s,v) = (1-p)(1 - H_s \circ F_s^{-1}(v)) + pG_s \circ F_s^{-1}(v)$ and $F_s = pG_s + (1-p)H_s$, the mapping $v \mapsto M(s,v)$ is Lipschitz by assumption (ii). Thus, there exists a constant $\kappa < \infty$, depending only on p, b and B, such that:

$$\left|M(s, F_s \circ \hat{F}_s^{-1}(v_0)) - M(s,v_0)\right| \leq \kappa \left|F_s \circ \hat{F}_s^{-1}(v_0) - v_0\right|.$$

Moreover, we have, for any $s \in \mathcal{S}$:

$$\begin{aligned}\left|F_s \circ \hat{F}_s^{-1}(v_0) - v_0\right| &\leq \left|F_s \circ \hat{F}_s^{-1}(v_0) - \hat{F}_s \circ \hat{F}_s^{-1}(v_0)\right| + \left|\hat{F}_s \circ \hat{F}_s^{-1}(v_0) - v_0\right| \\ &\leq \sup_{t \in (0,\lambda)} \left|F_s(t) - \hat{F}_s(t)\right| + \frac{1}{n}\ .\end{aligned}$$

Here again, we can use assumption (i) and a classical VC bound from Lugosi in order to control the empirical process, with probability larger than $1-\delta$:

$$\sup_{(s,t) \in \mathcal{S} \times (0,\lambda)} \left|F_s(t) - \hat{F}_s(t)\right| \leq c\sqrt{\frac{V}{n}} + c'\sqrt{\frac{ln(1/\delta)}{n}}$$

for some constants c, c'.

It remains to control the term involving the process U_n:

$$\left|U_n(s, F_s \circ \hat{F}_s^{-1}(v_0)) - U_n(s,v_0)\right| \leq \sup_{v \in (0,1)} |U_n(s,v) - U_n(s,v_0)| \leq 2 \sup_{v \in (0,1)} |U_n(s,v)|.$$

Using that the class of sets of the form $\{x : s(x) \geq Q(s,v)\}$ for $v \in (0,1)$ is included in the class of sets of the form $\{x : s(x) \geq t\}$ where $t \in (0,\lambda)$, we then have

$$\sup_{v \in (0,1)} |U_n(s,v)| \leq \sup_{t \in (0,\lambda)} \left| \frac{1}{n}\sum_{i=1}^{n} \mathbb{I}\{Y_i \cdot \big(s(X_i) - t\big) < 0\} - \mathbb{P}\left\{Y \cdot \big(s(X) - t\big) < 0\right\} \right|\ ,$$

which leads again to an empirical process indexed by a VC class of sets and can be bounded as before. ■

2.3 Fast Rates of Convergence

We now propose to examine conditions leading to fast rates of convergence (faster than $n^{-1/2}$). It has been noticed (see Mammen and Tsybakov, 1999; Tsybakov, 2004; Massart and Nédélec, 2006) that it is possible to derive such rates of convergence in the classification setup under additional assumptions on the distribution. We propose here to adapt these assumptions for the problem of classification with mass constraint.

Our concern here is to formulate the type of conditions which render the problem easier from a statistical perspective. For this reason and to avoid technical issues, we will consider a quite restrictive setup where it is assumed that:

- the class $\mathcal{S}$ of scoring functions is a finite class with N elements,
- an optimal scoring rule s^* is contained in $\mathcal{S}$.

We have found that the following additional conditions on the distribution and the class $\mathcal{S}$ allow to derive fast rates of convergence for the excess risk in our problem.

(iii) There exist constants $\alpha \in (0,1)$ and $D > 0$ such that, for all $t \geq 0$,

$$\mathbb{P}\{|\eta(X) - Q(\eta, 1-u_0)| \leq t\} \leq D t^{\frac{\alpha}{1-\alpha}} .$$

(iv) the family $\mathcal{K} = \{ G_s, H_s \ : \ s \in \mathcal{S} \}$ of cdfs satisfies the following property: for any $s \in \mathcal{S}$, G_s and H_s are twice differentiable at $Q(s, 1-u_0) = F_s^{-1}(1-u_0)$.

Note that condition (iii) simply extends the standard low noise assumption introduced by Tsybakov (2004) (see also Boucheron et al., 2005, for an account on this) where the level 1/2 is replaced by the $(1-u_0)$-quantile of $\eta(X)$. Condition (iv) is a technical requirement needed in order to derive an approximation of the statistics involved in empirical risk minimization.

Remark 8 (CONSEQUENCE OF CONDITION (III)) *We recall here the various equivalent formulations of condition (iii) as they are described in Section 5.2 from the survey paper by Boucheron et al. (2005). A slight variation in our setup is due to the presence of the quantile $Q(\eta, 1-u_0)$ but we can easily adapt the corresponding conditions. Hence, we have, under condition (iii), the variance control, for any $s \in \mathcal{S}$:*

$$\mathrm{Var}(\mathbb{I}\{Y \neq 2\mathbb{I}_{C_s}(X) - 1\} - \mathbb{I}\{Y \neq 2\mathbb{I}_{C^*_{u_0}}(X) - 1\}) \leq c \, (L(s) - L^*_{u_0})^{\alpha} ,$$

or, equivalently,

$$\mathbb{E}\big(\mathbb{I}_{C_s \Delta C^*_{u_0}}(X)\big) \leq c \, (L(s) - L^*_{u_0})^{\alpha} .$$

Recall that $L_n(s) = \frac{1}{n}\sum_{i=1}^n \mathbb{I}\{Y_i \cdot (s(X_i) - Q(s, 1-u_0)) < 0\}$. We point out that $L_n(s)$ is not an empirical criterion since the quantile $Q(s, 1-u_0)$ depends on the distribution. However, we can introduce the minimizer of this functional:

$$s_n = \arg\min_{s \in \mathcal{S}} L_n(s) ,$$

for which we can use the same argument as in the classification setup. We then have, by a standard argument based on Bernstein's inequality (which will be provided for completeness in the proof of Theorem 10 below), with probability $1-\delta$,

$$L(s_n) - L^*_{u_0} \leq c \left(\frac{log(N/\delta)}{n} \right)^{\frac{1}{2-\alpha}} .$$

for some positive constant c. We will show below how to obtain a similar rate when the true quantile $Q(s,1-u_0)$ is replaced by the empirical quantile $\widehat{Q}(s,1-u_0)$ in the criterion to be minimized.

We point out that conditions (ii) and (iii) are not completely independent. We offer the following proposition which will be useful in the sequel.

Proposition 9 *If (G_η, H_η) belongs to the class $\mathcal{K}$ fulfilling condition (ii), then F_η is Lipschitz and condition (iii) is satisfied with $\alpha = 1/2$.*

PROOF. We recall that $F_\eta = pG_\eta + (1-p)H_\eta$ and assume for simplicity that G_η and H_η are differentiable. By condition (ii), we then have $|F'_\eta| = p|G'_\eta| + (1-p)|H'_\eta| \leq pB + (1-p)B = B$. Set $q = Q(\eta, 1-u_0)$. Then, by the mean value theorem, there exists a constant c such that, for all $t \geq 0$:

$$\mathbb{P}\{|\eta(X) - q| \leq t\} = F_\eta(t+q) - F_\eta(-t+q) \leq B(t+q-(-t+q)) = 2Bt \ .$$

We have proved that condition (iii) is fulfilled with $D = 2B$ and $\alpha = 1/2$. ■

The novel part of the analysis below lies in the control of the bias induced by plugging the empirical quantile $\widehat{Q}(s,1-u_0)$ in the risk functional. The next theorem shows that faster rates of convergence up to the order of $n^{-2/3}$ can be obtained under the previous assumptions.

Theorem 10 *We assume that the class $\mathcal{S}$ of scoring functions is a finite class with N elements, and that it contains an optimal scoring rule s^*. Moreover, we assume that conditions (i)-(iv) are satisfied. We recall that $\widehat{s}_n = \arg\min_{s\in\mathcal{S}} \widehat{L}_n(s)$. Then, for any $\delta > 0$, we have, with probability $1-\delta$:*

$$L(\widehat{s}_n) - L^*_{u_0} \leq c \left(\frac{log(N/\delta)}{n} \right)^{\frac{2}{3}},$$

for some constant c.

Remark 11 (ON THE RATE $n^{-2/3}$) *This result highlights the fact that rates faster than the one obtained in Theorem 5 can be obtained in this setup with additional regularity assumptions. However, it is noteworthy that the standard low noise assumption (iii) is already contained, by Proposition 9, in assumption (ii) which is required in proving the typical $n^{-1/2}$ rate. The consequence of this observation is that there is no hope of getting rates up to n^{-1} unless assumption (ii) is weakened.*

Remark 12 (ON THE ASSUMPTION $s^* \in \mathcal{S}$) *This assumption is not important and can be removed. For a neat argument, check the proof of Theorem 5 from Clémençon et al. (To appear) which uses a result by Massart (2006).*

The proof of the previous theorem is based on two arguments: the structure of *linear signed rank statistics* and the variance control assumption. The situation is similar to the one we encountered in Clémençon et al. (To appear) where we were dealing with U-statistics and we had to invoke Hoeffding's decomposition in order to grasp the behavior of the underlying U-processes. Here we require a similar argument to describe the structure of the empirical risk functional $\widehat{L}_n(s)$ under study. This statistic can be interpreted as a linear signed rank statistic and the key decomposition has been used in the context of nonparametric hypotheses testing and R-estimation. We mainly

refer to Hájek and Sidák (1967), Dupac and Hájek (1969), Koul (1970), and Koul and R.G. Staudte (1972) for an account on rank statistics.

We now prepare for the proof by stating the main ideas in the next propositions, but first we need to introduce some notations. Set:

$$\forall v \in [0,1]\ , \quad K(s,v) = \mathbb{E}\left(Y \mathbb{I}\{s(X) \leq Q(s,v)\}\right) = pG_s(Q(s,v)) - (1-p)H_s(Q(s,v)),$$
$$\hat{K}_n(s,v) = \frac{1}{n}\sum_{i=1}^{n} Y_i \mathbb{I}\{s(X_i) \leq \widehat{Q}(s,v)\}\ .$$

Then we can write:

$$L(s) = 1 - p + K(s, 1-u_0),$$
$$\hat{L}_n(s) = \frac{n_-}{n} + \hat{K}_n(s, 1-u_0)\ ,$$

where $n_- = \sum_{i=1}^{n} \mathbb{I}\{Y_i = -1\}$.

We note that the statistic $\hat{L}_n(s)$ is related to linear signed rank statistics.

Definition 13 (Linear signed rank statistic) *Consider $Z_1, \ldots, Z_n$ an i.i.d. sample with distribution F and a real-valued score generating function Φ. Denote by $R_i^+ = rank(|Z_i|)$ the rank of $|Z_i|$ in the sample $|Z_1|, \ldots, |Z_n|$. Then the statistic*

$$\sum_{i=1}^{n} \Phi\left(\frac{R_i^+}{n+1}\right)\ \mathrm{sgn}(Z_i)$$

is a linear signed rank statistic.

Proposition 14 *For fixed s and v, the statistic $\hat{K}_n(s,v)$ is a linear signed rank statistic.*

PROOF. Take $Z_i = Y_i s(X_i)$. The random variables Z_i have their absolute value distributed according to F_s and have the same sign as Y_i. It is easy to see that the statistic $\hat{K}_n(s,v)$ is a linear signed rank statistic with score generating function $\Phi(x) = \mathbb{I}_{[x \leq v]}$. ■

A decomposition of Hoeffding's type for such statistics can be formulated. Set first:

$$Z_n(s,v) = \frac{1}{n}\sum_{i=1}^{n} \left(Y_i - K'(s,v)\right) \mathbb{I}\{s(X_i) \leq Q(s,v)\} - K(s,v) + vK'(s,v)\ ,$$

where $K'(s,v)$ denotes the derivative of the function $v \mapsto K(s,v)$. Note that $Z_n(s,v)$ is a centered random variable with variance:

$$\sigma^2(s,v) = v - K(s,v)^2 + v(1-v)K'^2(s,v) - 2(1-v)K'(s,v)K(s,v)\ .$$

The next result is due to Koul (1970) and we provide an alternate proof in the Appendix.

Proposition 15 *Assume that condition (iv) holds. We have, for all $s \in \mathcal{S}$ and $v \in [0,1]$:*

$$\hat{K}_n(s,v) = K(s,v) + Z_n(s,v) + \Lambda_n(s) \ .$$

with

$$\Lambda_n(s) = O_{\mathbb{P}}(n^{-1}) \ \ as \ \ n \to \infty \ .$$

This asymptotic expansion highlights the structure of the statistic $\hat{L}_n(s)$ for fixed s:

$$\hat{L}_n(s) = \frac{n_-}{n} + K(s, 1-u_0) + Z_n(s, 1-u_0) + \Lambda_n(s) \ .$$

Once centered, the leading term $Z_n(s, 1-u_0)$ is an empirical average of i.i.d. random variables (of a stochastic order of $n^{-1/2}$) and the remainder term $\Lambda_n(s)$ is of a stochastic order of n^{-1}. The nature of the decomposition of $\hat{L}_n(s)$ is certainly unexpected because the leading term contains an additional derivative term given by $K'(s, 1-u_0)\,(v - \mathbb{I}\{s(X_i) \leq Q(s, 1-u_0)\})$. The revelation of this fact is one of the major contributions in the work of Koul (2002).

Now, in order to establish consistency and rates-of-convergence-type results, we need to focus only on the leading term which carries most of the statistical information, while the remainder needs to be controlled uniformly over the candidate class $\mathcal{S}$. As a consequence, the variance control assumption will only concern the variance of the kernel h_s involved in the empirical average $Z_n(s, 1-u_0)$ and defined as follows:

$$h_s(X_i, Y_i) = \big(Y_i - K'(s,v)\big)\, \mathbb{I}\{s(X_i) \leq Q(s,v)\} - K(s,v) + vK'(s,v) \ ,$$

We then have

$$Z_n(s,v) - Z_n(s^*,v) = \frac{1}{n} \sum_{i=1}^{n} \big(h_s(X_i,Y_i) - h_{s^*}(X_i,Y_i)\big) \ .$$

Proposition 16 *Fix $v \in [0,1]$. Assume that condition (iii) holds. Then, we have, for all $s \in \mathcal{S}$:*

$$\mathrm{Var}\big(h_s(X_i,Y_i) - h_{s^*}(X_i,Y_i)\big) \leq c\big(L(s) - L(s^*)\big)^{\alpha} \ ,$$

for some constant c.

PROOF. We first write that:

$$h_s(X_i,Y_i) - h_{s^*}(X_i,Y_i) = I + II + III + IV + V$$

where

$$\begin{aligned}
I &= Y_i \left(\mathbb{I}\{s(X_i) \leq Q(s,v)\} - \mathbb{I}\{s^*(X_i) \leq Q(s^*,v)\}\right), \\
II &= \left(K'(s^*,v) - K'(s,v)\right) \mathbb{I}\{s^*(X_i) \leq Q(s^*,v)\}, \\
III &= K'(s,v) \left(\mathbb{I}\{s^*(X_i) \leq Q(s^*,v)\} - \mathbb{I}\{s(X_i) \leq Q(s,v)\}\right), \\
IV &= K(s^*,v) - K(s,v), \\
V &= v\left(K'(s,v) - K'(s^*,v)\right) \ .
\end{aligned}$$

By Cauchy-Schwarz inequality, we only need to show that the expected value of the square of these quantities is smaller than $(L(s)-L^*)^\alpha$ up to some multiplicative constant.

Note that, by definition of K, we have:

$$\begin{aligned} II &= (L'(s^*,v)-L'(s,v))\,\mathbb{I}\{s^*(X_i)\leq Q(s^*,v)\}, \\ IV &= L(s^*)-L(s), \\ V &= v\,(L'(s,v)-L'(s^*,v)) \end{aligned}$$

where $L'(s,v)$ denotes the derivative of the function $v\mapsto L(s,v)$. It is clear that, for any s, we have $L(s,v)=L(s^*,v)$ implies that $L'(s,v)=L'(s^*,v)$ otherwise s^* would not be an optimal scoring function at some level v' in the vicinity of v. Therefore, since $\mathcal{S}$ is finite, there exists a constant c such that

$$(L'(s,v)-L'(s^*,v))^2\leq c(L(s)-L^*)^\alpha$$

and then $\mathbb{E}\big(II^2\big)$ and $\mathbb{E}\big(V^2\big)$ are bounded accordingly.

Moreover, we have:

$$\begin{aligned} \mathbb{E}\big(I^2\big) &\leq \mathbb{E}\big(\mathbb{I}_{C_s\Delta C_{s^*}}(X)\big) \\ &\leq c(L(s)-L(s^*))^\alpha \end{aligned}$$

for some positive constant c, by assumption (iii).

Eventually, by assumption (ii), we have that $K'(s,v)$ is uniformly bounded and thus, the term $\mathbb{E}\big(III^2\big)$ can be handled similarly. ■

Proof of Theorem 10. Set $v_0=1-u_0$. First notice that $\hat{s}_n=\arg\min_{s\in\mathcal{S}}\hat{K}_n(s,1-u_0)$. We then have

$$\begin{aligned} L(\hat{s}_n)-L(s^*) &= K(\hat{s}_n,v_0)-K(s^*,v_0) \\ &\leq \hat{K}_n(s^*,v_0)-\hat{K}_n(\hat{s}_n,v_0)-(K(s^*,v_0)-K(\hat{s}_n,v_0)) \\ &\leq Z_n(s^*,v_0)-Z_n(\hat{s}_n,v_0))+2\sup_{s\in\mathcal{S}}|\Lambda_n(s)| \end{aligned}$$

where we used the decomposition of the linear signed rank statistic from Proposition 15 to obtain the last inequality.

By Proposition 15, we know that the second term on the right hand side is of stochastic order n^{-1} since the class $\mathcal{S}$ is of finite cardinality. It remains to control the leading term $Z_n(s^*,v_0)-Z_n(\hat{s}_n,v_0)$. At this point, we will use the same argument as in Section 5.2 from Boucheron et al. (2005).

Denote by $C=\sup_{s,x,y}|h_s(x,y)|$ and by $\sigma^2(s)=\mathrm{Var}\big(h_s(X_i,Y_i)-h_{s^*}(X_i,Y_i)\big)$. By Bernstein's inequality for averages of upper bounded and centered random variables (see Devroye et al., 1996) and the union bound, we have, with probability $1-\delta$, for all $s\in\mathcal{S}$:

$$Z_n(s^*,v_0)-Z_n(s,v_0)\leq\sqrt{\frac{2\sigma^2(s)\log(N/\delta)}{n}}+\frac{2C\log(N/\delta)}{3n}$$

$$\leq \sqrt{\frac{2c(L(s)-L^*)^\alpha \log(N/\delta)}{n}} + \frac{2C\log(N/\delta)}{3n}$$

thanks to the variance control obtained in Proposition 16. Since this inequality holds for any s, it holds in particular for $s = \hat{s}_n$. Therefore, we have obtained the following result, with probability $1-\delta$:

$$L(\hat{s}_n) - L(s^*) \leq \sqrt{\frac{2c(L(\hat{s}_n)-L^*)^\alpha \log(N/\delta)}{n}} + \frac{2c'\log(N/\delta)}{3n}$$

for some constants c, c'. At the cost of increasing the multiplicative constant factor, we can get rid of the second term and solve the inequality in the quantity $L(\hat{s}_n) - L(s^*)$ to get

$$L(\hat{s}_n) - L(s^*) \leq c\left(\frac{\log(N/\delta)}{n}\right)^{\frac{1}{2-\alpha}}$$

for some constant c. To end the proof, we plug the value of $\alpha = 1/2$ following from Proposition 9. ∎

3. Performance Measures for Local Ranking

Our main interest here is to develop a setup describing the problem of not only finding but also ranking the best instances. In the sequel, we build on the results from Section 2 and also on our previous work on the (global) ranking problem (Clémençon et al., To appear) in order to capture some of the features of the local ranking problem. The present section is devoted to the construction of performance measures reflecting the quality of ranking rules on a restricted set of instances.

3.1 ROC Curves and Optimality in the Local Ranking Problem

We consider the same statistical model as before with (X,Y) being a pair of random variables over $\mathcal{X} \times \{-1,+1\}$ and we examine ranking rules resulting from real-valued scoring functions $s : \mathcal{X} \to (0,\lambda)$. The reference tool for assessing the performance of a scoring function s in separating the two populations (positive vs. negative labels) is the Receiver Operating Characteristic known as the ROC curve (van Trees, 1968; Egan, 1975). If we take the notations $\bar{G}_s(z) = \mathbb{P}\{s(X) > z \mid Y = 1\}$ (true positive rate) and $\bar{H}_s(z) = \mathbb{P}\{s(X) > z \mid Y = -1\}$ (false positive rate), we can define the ROC curve, for any scoring function s, as the plot of the function:

$$z \mapsto \left(\bar{H}_s(z), \bar{G}_s(z)\right)$$

for thresholds $z \in (0,\lambda)$, or equivalently as the plot of the function:

$$t \mapsto \bar{G}_s \circ H_s^{-1}(1-t)$$

for $t \in (0,1)$. The optimal scoring function is the one whose ROC curve dominates all the others for all $z \in (0,\lambda)$ (or $t \in (0,1)$) and such a function actually exists. Indeed, by recalling the hypothesis testing framework in the classification model (see Remark 2) and using Neyman-Pearson's Lemma, it is easy to check that the ROC curve of the function $\eta(x) = \mathbb{P}\{Y = 1 \mid X = x\}$ dominates the ROC

curve of any other scoring function. We point out that the ROC curve of a scoring function s is invariant to strictly increasing transformations of s.

In our approach, for a given scoring function s, we focus on thresholds z corresponding to the cut-off separating a proportion $u \in (0,1)$ of top scored instances according to s from the rest. Recall from Section 2 that the best instances according to s are the elements of the set $C_{s,u} = \{x \in \mathcal{X} \mid s(x) \geq Q(s,1-u)\}$ where $Q(s,1-u)$ is the $(1-u)$-quantile of $s(X)$. We set the following notations:

$$\alpha(s,u) = \mathbb{P}\{s(X) \geq Q(s,1-u) \mid Y=-1\} = \bar{H}_s \circ F_s^{-1}(1-u),$$
$$\beta(s,u) = \mathbb{P}\{s(X) \geq Q(s,1-u) \mid Y=+1\} = \bar{G}_s \circ F_s^{-1}(1-u)\ .$$

We propose to re-parameterize the ROC curve with the proportion $u \in (0,1)$ and then describe it as the plot of the function:

$$u \mapsto (\alpha(s,u), \beta(s,u))\ ,$$

for each scoring function s. When focusing on the best instances at rate u_0, we only consider the part of the ROC curve for values $u \in (0,u_0)$.

However attractive is the ROC curve as a graphical tool, it is not a practical one for developing learning procedures achieving straightforward optimization. The most natural approach is to consider risk functionals built after the ROC curve such as the Area Under an ROC Curve (known as the AUC or AROC, see Hanley and McNeil, 1982). Our goals in this section are:

1. to extend the AUC criterion in order to focus on restricted parts of the ROC curve,
2. to describe the optimal elements with respect to this extended criterion.

We point out the fact that extending the AUC is not trivial. In order to focus on the best instances, a natural idea is to truncate the AUC (as in the approach by Dodd and Pepe (2003)).

Definition 17 (Partial AUC) *We define the partial AUC for a scoring function s and a rate u_0 of best instances as:*

$$\text{PARTAUC}(s,u_0) = \int_0^{\alpha(s,u_0)} \beta(s,\alpha)\, d\alpha\ .$$

We conjecture that such a criterion is not appropriate for local ranking. If it was, then we should have: $\forall s, \text{PARTAUC}(s,u_0) \leq \text{PARTAUC}(\eta,u_0)$, since the function η would provide the optimal ranking. However, there is strong evidence that this is not true as shown by a simple geometric argument which we describe below.

In order to represent the partial AUC of a scoring function s, we need to locate the cut-off point given the constraint on the rate u_0 of best instances. We notice that $\alpha(s,u)$ and $\beta(s,u)$ are related by a linear relation, for fixed u and p, when s varies:

$$u = p\beta(s,u) + (1-p)\alpha(s,u)$$

where $p = \mathbb{P}\{Y=1\}$. We denote the line plot of this relation by $D(u,p)$ and call it the *control line* when $u = u_0$. Hence, the part of the ROC curve of a scoring function s corresponding to the best instances at rate u_0 is the part going from the origin $(0,0)$ to the intersection with the control line $D(u_0,p)$. The partial AUC is then the area under this part of the ROC curve (it corresponds to the shaded area in the left display of Figure 2).

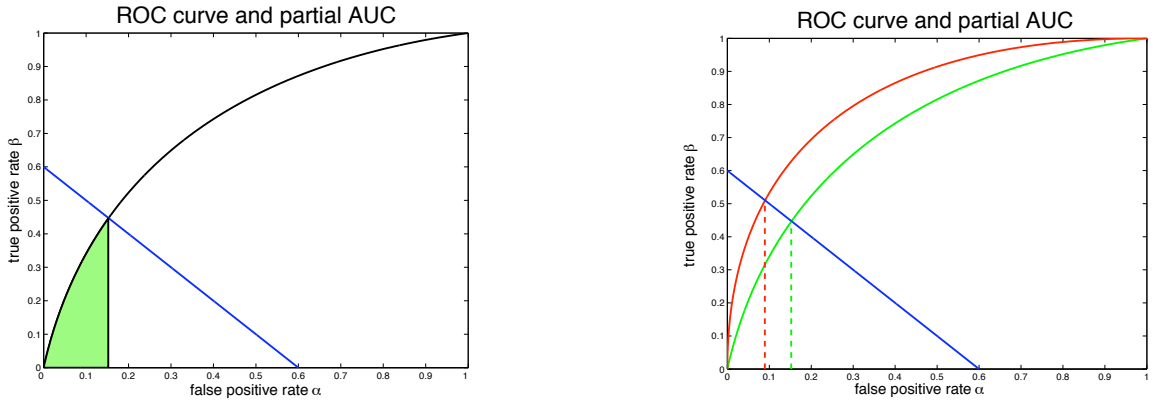


Figure 2: ROC curves, control line $D(u_0, p)$ and partial AUC at rate u_0 of best instances.

The optimality of η with respect to the partial AUC can then be questioned. Indeed, the closer to η the scoring function s is, the higher the ROC curve is, but at the same time the integration domain shrinks (right display of Figure 2) so that the overall impact on the integral is not clear. Let us now put things formally in the following lemma.

Lemma 18 *For any scoring function s, we have for all $u \in (0,1)$,*

$$\begin{aligned}\beta(s,u) &\leq \beta(\eta,u),\\ \alpha(s,u) &\geq \alpha(\eta,u)\ .\end{aligned}$$

Moreover, we have equality only for those s such that $C_{s,u_0} = C^*_{u_0}$.

PROOF. We show the first inequality. By definition, we have:

$$\beta(s,u) = 1 - G_s(Q(s,1-u))\ .$$

Observe that, for any scoring function s,

$$\begin{aligned}p(1-G_s(Q(s,1-u)) &= \mathbb{P}\{Y=1, s(X) > Q(s,1-u)\}\\ &= \mathbb{E}\left(\eta(X)\mathbb{I}\{X \in C_{s,u}\}\right)\ .\end{aligned}$$

We thus have

$$\begin{aligned}p\left(G_s(Q(s,1-u)) - G_\eta(Q(\eta,1-u)\right) &= \mathbb{E}\left(\eta(X)(\mathbb{I}\{X \in C^*_u\} - \mathbb{I}\{X \in C_{s,u}\})\right)\\ &= \mathbb{E}\left(\eta(X)\mathbb{I}\{X \notin C^*_u\}(\mathbb{I}\{X \in C^*_u\} - \mathbb{I}\{X \in C_{s,u}\})\right)\\ &\quad + \mathbb{E}\left(\eta(X)\mathbb{I}\{X \in C^*_u\}(\mathbb{I}\{X \in C^*_u\} - \mathbb{I}\{X \in C_{s,u}\})\right)\\ &\geq -\mathbb{E}\left(Q(\eta,1-u)\mathbb{I}\{X \notin C^*_u\}\,\mathbb{I}\{X \in C_{s,u}\}\right)\\ &\quad + \mathbb{E}\left(Q(\eta,1-u)\mathbb{I}\{X \in C^*_u\}(1 - \mathbb{I}\{X \in C_{s,u}\})\right)\end{aligned}$$

$$= Q(\eta, 1-u)(1-u-1+u) = 0 \; .$$

The second inequality simply follows from the identity below:

$$1-u = pG_s(Q(s,1-u)) + (1-p)H_s(Q(s,1-u)) \; .$$

■

The previous lemma will be important when describing the optimal rules for local ranking criteria. But, at this point, we still do not know any nice criterion for the problem of ranking the best instances. Before considering different heuristics for extending the AUC criterion in the next subsections, we will proceed backwards and define our target, that is to say, the optimal scoring functions for our problem.

Definition 19 (Class $\mathcal{S}^*$ of optimal scoring functions) *The optimal scoring functions for ranking the best instances at the rate u_0 are defined as the members of the equivalence class (functions defined up to the composition with a nondecreasing transformation) of scoring functions s^* such that:*

$$s^*(x) = \begin{cases} \eta(x) & \text{if } x \in C^*_{u_0} \\ < \inf_{z \in C^*_{u_0}} \eta(z) & \text{if } x \notin C^*_{u_0} \; . \end{cases}$$

Such scoring functions fulfill the two properties of locating the best instances (indeed $C_{s^*,u_0} = C^*_{u_0}$) and ranking them as well as the regression function.

Under the light of Lemma 18, we will see that a wide collection of criteria with the set $\mathcal{S}^*$ as the set of optimal elements could naturally be considered, depending on how one wants to weight the two types of error $1-\beta(s,u)$ (type II error in the hypothesis testing framework) and $\alpha(s,u)$ (type I error) according to the rate $u \in [0,u_0]$. However, not all the criteria obtained in this manner can be interpreted as generalizations of the AUC criterion for $u_0 = 1$.

3.2 Generalization of the AUC Criterion

In Clémençon et al. (To appear), we have considered the ranking error of a scoring function s as defined by:

$$R(s) = \mathbb{P}\{(Y-Y')(s(X)-s(X')) < 0\} \; ,$$

where (X',Y') is an i.i.d. copy of the random pair (X,Y).

Interestingly, it can be proved that minimizing the ranking error $R(s)$ is equivalent to maximizing the well-known AUC criterion. This is trivial once we write down the probabilistic interpretation of the AUC:

$$\mathrm{AUC}(s) = \mathbb{P}\left\{s(X) > s(X') \mid Y = 1, Y' = -1\right\} = 1 - \frac{1}{2p(1-p)} R(s) \; .$$

We now propose a local version of the ranking error on a measurable set $C \subset \mathcal{X}$:

$$R(s,C) = \mathbb{P}\left\{(s(X)-s(X'))(Y-Y') < 0, \; (X,X') \in C^2\right\} \; .$$

On sets of the form $C = C_{s,u} = \{x \in \mathcal{X} \mid s(x) \geq Q(s, 1-u)\}$ with mass equal to u, the local ranking error will be denoted by $R(s,u) \stackrel{\circ}{=} R(s, C_{s,u})$.

We will also consider the local analogue of the AUC criterion:

$$\text{LOCAUC}(s,u) = \mathbb{P}\left\{s(X) > s(X'),\ s(X) \geq Q(s, 1-u) \ \mid\ Y = 1, Y' = -1\right\} .$$

This criterion obviously boils down to the standard criterion for $u = 1$. However, in the case where $u < 1$, we will see that there is no equivalence between maximizing the LOCAUC criterion and minimizing the local ranking error $s \mapsto R(s,u)$. Indeed, the local ranking error is not a relevant performance measure for finding the best instances. Minimizing it would solve the problem of finding the instances that are the easiest to rank.

The following theorem states that optimal scoring functions s^* in the set $\mathcal{S}^*$ maximize the LOCAUC criterion and that the latter may be decomposed as a sum of a 'power' term and (the opposite of) a local ranking error term.

Theorem 20 *Let $u_0 \in (0,1)$. We have, for any scoring function s:*

$$\forall s^* \in \mathcal{S}^*, \quad \text{LOCAUC}(s, u_0) \leq \text{LOCAUC}(s^*, u_0) .$$

Moreover, the following relation holds:

$$\forall s, \quad \text{LOCAUC}(s, u_0) = \beta(s, u_0) - \frac{1}{2p(1-p)} R(s, u_0) ,$$

where $R(s, u_0) = R(s, C_{s,u_0})$.

PROOF. We first introduce the notation for the Lebesgue-Stieltjes integral. Whenever φ is a cdf on $\mathbb{R}$ and ψ is integrable, the integral $\int \psi(z)\, d\varphi(z)$ denotes the Lebesgue-Stieltjes integral (integration with respect to the measure ν defined by $\nu[a,b) = \varphi(b) - \varphi(a)$ for any real numbers $a < b$). If φ has a density with respect to the Lebesgue measure, then the integral can be written as a Lebesgue integral: $\int \psi(z)\, d\varphi(z) = \int \psi(z)\varphi'(z)\, dz$. We shall use this convention repeatedly in the sequel. In particular, if Z is a random variable with cdf given by F_Z then we can write: $\mathbb{E}(Z) = \int z\, dF_Z(z)$. Now set $v_0 = 1 - u_0$. Observe first that, by conditioning on X, we have:

$$\begin{aligned}
\text{LOCAUC}(s, u_0) &= \mathbb{E}\left(\mathbb{I}\{s(X) > s(X')\}\, \mathbb{I}\{s(X) \geq Q(s, v_0)\} \ \mid Y = 1, Y' = -1\right) \\
&= \mathbb{E}\left(\ \mathbb{I}\{s(X) \geq Q(s, v_0)\}\, \mathbb{E}\left(\ \mathbb{I}\{s(X) > s(X')\} \mid Y' = -1, X\ \right) \ \mid Y = 1\ \right) \\
&= \mathbb{E}\left(H_s(s(X))\ \mathbb{I}\{s(X) \geq Q(s, v_0)\} \ \mid Y = 1\ \right) \\
&= \int_{Q(s,v_0)}^{+\infty} H_s(z)\ dG_s(z) .
\end{aligned}$$

The last equality is obtained by using the fact that, conditionally on $Y = 1$, the random variable $s(X)$ has cdf G_s. We now use that $pG_s = F_s - (1-p)H_s$ and we obtain:

$$p\text{LOCAUC}(s, u_0) = \int_{Q(s,v_0)}^{+\infty} H_s(z)\ dF_s(z) - (1-p) \int_{Q(s,v_0)}^{+\infty} H_s(z)\ dH_s(z).$$

Recall now that $\alpha(s,v) = \bar{H}_s \circ F_s^{-1}(1-v)$ and make the change of variable $1-v = F_s(z)$

$$\int_{Q(s,v_0)}^{+\infty} H_s(z)\, dF_s(z) = \int_0^{v_0} (1-\alpha(s,v))\, dv\ .$$

The second term is computed by making the change of variable $a = H_s(z)$ which leads to:

$$\int_{Q(s,v_0)}^{+\infty} H_s(z)\, dH_s(z) = \int_{1-\alpha(s,u_0)}^{1} a\, da\ .$$

We have obtained:

$$p\text{LOCAUC}(s,u_0) = \int_0^{v_0} (1-\alpha(s,v))\, dv - \frac{1-p}{2}\left(1-(1-\alpha(s,v_0))^2\right)\ .$$

From Lemma 18, we have that, for any $u \in (0,1)$, the functional $s \mapsto \alpha(s,u))$ is minimized for $s = \eta$. Hence, the first part of Theorem 20 is established.

Besides, integrating by parts, we get:

$$\int_{Q(s,v_0)}^{+\infty} H_s(z)\, dG_s(z) = [H_s(z)G_s(z)]_{Q(s,v_0)}^{+\infty} - \int_{Q(s,v_0)}^{+\infty} G_s(z)\, dH_s(z).$$

The same change of variables as before leads to:

$$\int_{Q(s,v_0)}^{+\infty} G_s(z)\, dH_s(z) = \int_0^{\alpha(s,u_0)} (1-\beta(s,\alpha))\, d\alpha.$$

We then have another expression of the $\text{LOCAUC}(s,u_0)$:

$$\text{LOCAUC}(s,u_0) = \int_0^{\alpha(s,u_0)} \beta(s,\alpha)\, d\alpha + \beta(s,u_0)(1-\alpha(s,u_0))\ .$$

We develop further by expressing the product of α and β in terms of probability. Using the independence of (X,Y) and (X',Y'), we obtain:

$$\begin{aligned}
\alpha(s,u_0)\beta(s,u_0) &= \frac{1}{p(1-p)}\mathbb{P}\left\{s(X) \wedge s(X') > Q(s,v_0),\ Y=1,\ Y'=-1\right\} \\
&= \mathbb{P}\left\{s(X) > s(X'),\ s(X) \wedge s(X) > Q(s,v_0)\ \mid\ Y=1,\ Y'=-1\right\} \\
&\quad + \frac{1}{p(1-p)}\mathbb{P}\left\{s(X) < s(X'),\ (X,X') \in C^2_{s,u_0},\ Y=1,\ Y'=-1\right\} \\
&= \int_0^{\alpha(s,u_0)} \beta(s,\alpha)\, d\alpha + \frac{1}{2p(1-p)} R(s,u_0)\ .
\end{aligned}$$

Combining this with the previous formula leads to the second statement of the theorem. ■

Remark 21 (TRUNCATING THE AUC) *In the theorem, we obviously recover the relation between the standard* AUC *criterion and the (global) ranking error when* $u_0 = 1$. *Besides, by checking the proof, one may relate the generalized* AUC *criterion to the partial* AUC. *As a matter of fact, we have:*

$$\forall s\,, \quad \text{LOCAUC}(s,u_0) = \text{PARTAUC}(s,u_0) + \beta(s,u_0) - \alpha(s,u_0)\beta(s,u_0)\,.$$

The values $\alpha(s,u_0)$ *and* $\beta(s,u_0)$ *are the coordinates of the intersecting point between the ROC curve of the scoring function* s *and the control line* $D(u_0,p)$. *The theorem reveals that evaluating the local performance of a scoring statistic* $s(X)$ *by the truncated AUC as proposed in Dodd and Pepe (2003) is highly arguable since the maximizer of the functional* $s \mapsto \text{PARTAUC}(s,u_0)$ *is usually not in* $\mathcal{S}^*$.

3.3 Generalized Wilcoxon Statistic

We now propose a different extension of the plain AUC criterion. Consider $(X_1,Y_1),\ldots,(X_n,Y_n)$, n i.i.d. copies of the random pair (X,Y). The intuition relies on a well-known relationship between Mann-Whitney and Wilcoxon statistics. Indeed, a natural empirical estimate of the AUC is the *rate of concording pairs*:

$$\widehat{\text{AUC}}(s) = \frac{1}{n_+ n_-} \sum_{1 \le i,j \le n} \mathbb{I}\{Y_i = -1, Y_j = 1, s(X_i) < s(X_j)\}\,,$$

with $n_+ = n - n_- = \sum_{i=1}^n \mathbb{I}\{Y_i = +1\}$.

It will be useful to have in mind the definition of a linear rank statistic.

Definition 22 (linear rank statistic) *Consider* $Z_1,\ldots,Z_n$ *an i.i.d. sample with distribution* F *and a real-valued score generating function* Φ. *Denote by* $R_i = rank(Z_i)$ *the rank of* Z_i *in the sample* $Z_1,\ldots,Z_n$. *Then the statistic*

$$\sum_{i=1}^n c_i \Phi\left(\frac{R_i}{n+1}\right)$$

is a linear rank statistic.

We refer to Hájek and Sidák (1967) and van de Vaart (1998) for basic results related to linear rank statistics. In particular, we recall that, for fixed s, the Wilcoxon statistic $T_n(s)$ is a linear rank statistic for the sample $s(X_1),\ldots,s(X_n)$, with random weights $c_i = \mathbb{I}\{Y_i = 1\}$, score generating function $\Phi(v) = v$:

$$T_n(s) = \sum_{i=1}^n \mathbb{I}\{Y_i = 1\}\,\frac{\text{rank}(s(X_i))}{n+1}\,,$$

where $\text{rank}(s(X_i))$ denotes the rank of $s(X_i)$ in the sample $\{s(X_j), 1 \le j \le n\}$. The following relation is well-known:

$$\frac{n_+ n_-}{n+1}\widehat{\text{AUC}}(s) + \frac{n_+(n_+ + 1)}{2} = T_n(s)\,.$$

Moreover, the statistic $T_n(s)/n_+$ is an asymptotically normal estimate of

$$W(s) = \mathbb{E}\left(F_s(s(X)) \,|\, Y = 1\right)\,.$$

Note the theoretical counterpart of the previous relation may be written as

$$W(s) = (1-p)\text{AUC}(s) + p/2\,.$$

Now, in order to take into account a proportion u_0 of the highest ranks only, we introduce the following quantity:

Definition 23 (W-ranking performance measure) *Consider the criterion related to the score generating function $\Phi_{u_0}(v) = v\,\mathbb{I}\{v > 1-u_0\}$:*

$$W(s,u_0) = \mathbb{E}\left(\Phi_{u_0}(F_s(s(X)))\,|\,Y=1\right).$$

*It will be called the W-*ranking performance measure at rate u_0.

Note that the empirical counterpart of $W(s,u_0)$ is given by $T_n(s,u_0)/n_+$, with

$$T_n(s,u_0) = \sum_{i=1}^{n} \mathbb{I}\{Y_i=1\}\,\Phi_{u_0}\left(\frac{\text{rank}(s(X_i))}{n+1}\right).$$

Using the results from the previous subsection, we can easily check that the following theorem holds.

Theorem 24 *We have, for all s:*

$$\forall s^* \in \mathcal{S}^*, \quad W(s,u_0) \leq W(s^*,u_0)\ .$$

Furthermore, we have:

$$W(s,u_0) = \frac{p}{2}\beta(s,u_0)(2-\beta(s,u_0)) + (1-p)\text{LOCAUC}(s,u_0)\ .$$

PROOF. We start by the definition of W:

$$W(s,u_0) = \mathbb{E}\left(F_s(s(X))\mathbb{I}\{F_s(s(X)) > 1-u_0\}\,|\,Y=1\right)$$

$$= \int_{Q(s,1-u_0)}^{+\infty} F_s(z)\,dG_s(z).$$

We recall that: $F_s = pG_s + (1-p)H_s$ which leads to:

$$W(s,u_0) = p\int_{Q(s,1-u_0)}^{+\infty} G_s(z)\,dG_s(z) + (1-p)\int_{Q(s,1-u_0)}^{+\infty} H_s(z)\,dG_s(z)\ .$$

The second term corresponds exactly to the LOCAUC. The first term is easily computed by a change of variable $b = G_s(z)$:

$$\int_{Q(s,1-u_0)}^{+\infty} G_s(z)\,dG_s(z) = \int_{1-\beta(s,u_0)}^{1} b\,db\ .$$

Elementary computations lead to the formula in the theorem. Moreover the application $t \mapsto t(2-t)$ being nondecreasing for $t \in (0,1)$, we have, from Lemma 18:

$$\forall s^* \in \mathcal{S}^*, \quad \beta(s,u_0)(2-\beta(s,u_0)) \leq \beta(s^*,u_0)(2-\beta(s^*,u_0))\ .$$

We also use the optimality of s^* for LOCAUC established in Theorem 20 to conclude the proof. ■

Remark 25 (EVIDENCE AGAINST 'TWO-STEP' STRATEGIES) *It is noteworthy that not all combinations of $\beta(s,u_0)$ (or $\alpha(s,u_0)$) and $R(s,u_0)$ lead to a criterion with $\mathcal{S}^*$ being the set of optimal scoring functions. We have provided two non-trivial examples for which this is the case (Theorems 20 and 24). But, in general, this remark should prevent from considering 'naive' two-step strategies for solving the local ranking problem. By 'naive' two-step strategies, we refer here to stagewise strategies which would, first, compute an estimate $\hat{C}$ of the set containing the best instances, and then, solve the ranking problem over $\hat{C}$ as described in Clémençon et al. (To appear). However, this idea combined with a certain amount of iterativeness might be the key to the design of efficient algorithms. In any case, we stress here the importance of making use of a global criterion, synthesizing our double goal: finding and ranking the best instances.*

Remark 26 (OTHER RANKING PERFORMANCE MEASURES) *The ideas expressed above suggest that several ranking criteria can be proposed. For instance, one can consider maximization of other linear rank statistics with particular score generating functions Φ and there are many possible choices which would emphasize the importance of the highest ranks. One of these choices is $\Phi(v) = v^p$ which corresponds to the p-norm push proposed by Rudin (2006) although the definition of the ranks in her work is slightly different. The Discounted Cumulative Gain criterion, studied in particular by Cossock and Zhang (2006) and Li et al. (2007), is of different nature and cannot be represented in a similar way. Other extensions can be proposed in the spirit of the tail strength measure from Taylor and Tibshirani (2006). The theoretical study of such criteria is still at an early stage, especially for the last proposal. We also point out that with such extensions, probabilistic interpretations and explicit connection to the AUC criterion seem to be lost.*

4. Empirical Risk Minimization of the Local AUC Criterion

In the previous section, we have seen that there are various performance measures which can be considered for the problem of ranking the best instances. In order to perform the statistical analysis, we will favor the representations of LOCAUC and W which involve the classification error $L(s,u_0)$ and the local ranking error $R(s,u_0)$. By combining Theorems 20 and 24, we can easily get:

$$2p(1-p)\mathrm{LOCAUC}(s,u_0) = (1-p)(p+u_0) - (1-p)L(s,u_0) - R(s,u_0)$$

and

$$2pW(s,u_0) = C(p,u_0) + \left(\frac{p+u_0}{2} - 1\right) L(s,u_0) - \frac{1}{4}L^2(s,u_0) - R(s,u_0)$$

where $C(p,u_0)$ is a constant depending only on p and u_0.

We exploit the first expression and choose to study the minimization of the following criterion for ranking the best instances:

$$M(s) \stackrel{\circ}{=} M(s,u_0) = R(s,u_0) + (1-p)L(s,u_0)\ .$$

It is obvious that the elements of $\mathcal{S}^*$ are the optimal elements of the functional $M(\ \cdot\ ,u_0)$ and we will now consider scoring functions obtained through empirical risk minimization of this criterion.

More precisely, given n i.i.d. copies $(X_1,Y_1),\ldots,(X_n,Y_n)$ of (X,Y), we introduce the empirical counterpart:

$$\hat{M}_n(s) \stackrel{\circ}{=} \hat{M}_n(s,u_0) = \hat{R}_n(s) + \frac{n_-}{n}\hat{L}_n(s),$$

with $n_- = \sum_{i=1}^n \mathbb{I}\{Y_i = -1\}$ and

$$\hat{R}_n(s) = \frac{1}{n(n-1)} \sum_{i \neq j} \mathbb{I}\{(s(X_i) - s(X_j))(Y_i - Y_j) < 0,\ s(X_i) \wedge s(X_j) \geq \hat{Q}(s, 1-u_0)\} .$$

Note that $\hat{R}_n(s)$ is expected to be close to the U-statistic of degree two

$$R_n(s) = \frac{1}{n(n-1)} \sum_{i \neq j} k_s((X_i, Y_i), (X_j, Y_j)),$$

with symmetric kernel

$$k_s((x,y),(x',y')) = \mathbb{I}\{(s(x) - s(x'))(y - y') < 0,\ s(x) \wedge s(x') \geq Q(s, 1-u_0)\} .$$

The statistic $R_n(s)$ corresponds to an unbiased estimate of the local ranking error $R(s, u_0)$. The next result provides a standard error bound for the excess risk of the empirical risk minimizer over a class $\mathcal{S}$ of scoring functions:

$$\hat{s}_n = \arg\min_{s \in \mathcal{S}} \hat{M}_n(s) .$$

Proposition 27 *Assume that conditions (i)-(ii) of Theorem 2 are fulfilled. Then, there exist constants c_1 and c_2 such that, for any $\delta > 0$, we have:*

$$M(\hat{s}_n) - \inf_{s \in \mathcal{S}} M(s) \leq c_1 \sqrt{\frac{V}{n}} + c_2 \sqrt{\frac{\ln(1/\delta)}{n}}$$

with probability larger than $1 - \delta$.

PROOF. (SKETCH) The proof combines the argument used in the proof of Theorem 5 with the techniques used in establishing Proposition 2 in Clémençon et al. (2005).

$$M(\hat{s}_n) - \inf_{s \in \mathcal{S}} M(s) \leq 2 \left(\sup_{s \in \mathcal{S}} \left| \hat{R}_n(s) - R_n(s) \right| + \sup_{s \in \mathcal{S}} |R(s) - R_n(s)| \right)$$
$$+ 2(1-p) \left(\sup_{s \in \mathcal{S}} \left| \hat{L}_n(s) - L_n(s) \right| + \sup_{s \in \mathcal{S}} |L(s) - L_n(s)| \right) + 2 \left| \frac{n_+}{n} - p \right| .$$

The middle term may be bounded by applying the result stated in Theorem 5, while the last one can be handled by using Bernstein's exponential inequality for an average of Bernoulli random variables. By combining Lemma 1 in Clémençon et al. (2005) with the Chernoff method, we can deal with the U-process term $\sup_{s \in \mathcal{S}} |R(s) - R_n(s)|$. Finally, the term $\sup_{s \in \mathcal{S}} \left| \hat{R}_n(s) - R_n(s) \right|$ can also be controlled by repeating the argument in the proof of Theorem 5. The only difference here is that we have to consider the U-process term

$$\sup_{(s,t)} \left| \frac{2}{n(n-1)} \sum_{i \neq j} \{K_{s,t}((X_i, Y_i), (X_j, Y_j)) - \mathbb{E}[K_{s,t}((X,Y),(X',Y'))]\} \right|$$

with

$$K_{s,t}((x,y),(x',y')) = \mathbb{I}\{(s(x) - s(x'))(y - y') > 0,\ s(x) \wedge s(x') \geq t\} .$$

For deriving first-order results with such a process, we refer to the same type of argument as used in Clémençon et al. (2005). ■

Remark 28 (ABOUT THE POSSIBILITY OF DERIVING FAST RATES) *By checking the proof sketch, it turns out that sharper bounds may be achieved for the U-process term. Indeed, it is a simple variation of our previous work in Clémençon et al. (2005) where we have used Hoeffding's decomposition in order to grasp the deep structure of the underlying statistic. Here we will need, in addition, condition (iii) to hold for all $u \in (0, u_0]$. Indeed, if we localize our low-noise assumption from Clémençon et al. (2005), it takes the following form: there exist constants $\alpha \in (0,1)$ and $B > 0$ such that, for all $t \geq 0$, we have*

$$\forall x \in C^*_{u_0}, \qquad \mathbb{P}\{|\eta(X) - \eta(x)| \leq t\} \leq B t^{\frac{\alpha}{1-\alpha}} \ .$$

It is easy to see that this is equivalent to condition (iii) for all $u \in (0, u_0]$: there exist constants $\alpha \in (0,1)$ and $B > 0$ such that, for all $t \geq 0$, we have

$$\forall u \in (0, u_0], \qquad \mathbb{P}\{|\eta(X) - Q(\eta, 1-u)| \leq t\} \leq B t^{\frac{\alpha}{1-\alpha}} \ .$$

However, in the present formulation where p is assumed to be unknown, it looks like this improvement will be spoiled by the 'proportion term' which will still be of the order of a $O(n^{-1/2})$.

Remark 29 (ABOUT THE EXTENSION TO CONVEX RISK MINIMIZATION) *An important topic in classification theory is convex risk minimization. Understanding the connection between classification error and its convex surrogates has permitted to understand the behavior of practical algorithms such as boosting and* SVM *from a statistical perspective (see Boucheron et al., 2005 for an account on this aspect and Bartlett et al., 2006 for state-of-the-art results). A natural question which arises here is whether the consistency results on local ranking can be extended in this spirit. Note that, if we do not focus on best instances and consider the whole* AUC *as a performance criterion, it is straightforward to obtain consistency and universal rates of convergence for convex risk minimization (as explained in Clémençon et al. (To appear)). In the case of local ranking as we introduced it, this extension is less straightforward since the decision rule represented here by the scoring function s appears under the empirical quantile $\hat{Q}(s, v)$ in the criterion. We refer to Rudin (2006), Cossock and Zhang (2006) and Li et al. (2007) where convex risk minimization strategies in the context of ranking are discussed.*

5. Conclusion

In the present work, we have presented theoretical work on local ranking. In the first part of the paper (Section 2), we considered a subproblem that we called the *classification with mass constraint* problem. The scope was to establish and study an empirical risk minimization strategy for only *finding*, and not ranking, the best instances. In this case, one attempts to minimize the classification error over classifiers that contain a fixed proportion u of observations. This constraint leads to empirical risk functionals which involve an empirical quantile indexed by the class of candidate scoring functions and can be seen as linear signed rank statistics. We then provide a consistency result and discuss the noise assumptions required to derive fast rates of convergence in this setup. These assumptions require a limited regularity of the underlying distributions which prevents the fast rate from dropping below the order of $n^{-2/3}$. The second part of the paper (Section 3) is dedicated to the introduction of new performance measures for local ranking related to the ROC curve and the AUC criterion. We show that the AUC can be extended in several ways (partial AUC and local AUC) but not all these extensions are tailored for the local ranking problem. In particular

the naive extension known as the partial AUC is not appropriate and requires a correction term. We also introduce the *optimal scoring functions* which should be considered as the target of any local ranking method. We also discuss other extensions based on Wilcoxon statistics, the W-ranking performance measure, for which optimal rules can also be recovered. In the last section of the paper (Section 4), the problem of ranking the best instances is studied from a statistical perspective. A consistency result is provided for empirical risk minimization of the W-ranking performance measure.

Acknowledgments

We thank Stéphane Boucheron and Gábor Lugosi for their helpful remarks and encouragements. We also gratefully thank both referees for their comments which helped us improve the clarity of the paper.

Appendix A.

In this section, we provide the proof of Proposition 15.

PROOF. First, for all $(s,v) \in \mathcal{S} \times (0,1)$ set

$$V_n(s,v) = \frac{1}{n}\sum_{i=1}^{n} Y_i \mathbb{I}\{s(X_i) \leq Q(s,v)\} - K(s,v) \ .$$

We have the following decomposition:

$$\forall v \in [0,1] \ , \quad \hat{K}_n(s,v) - K(s,v) = V_n(s, F_s \circ \hat{F}_s^{-1}(v)) + K(s, F_s \circ \hat{F}_s^{-1}(v)) - K(s,v) \ .$$

We shall first prove that

$$V_n(s, F_s \circ \hat{F}_s^{-1}(v_0)) = V_n(s,v_0) + O_{\mathbb{P}}(n^{-1}).$$

We denote by $A(s,\varepsilon)$ the event $\left\{ \left| F_s \circ \hat{F}_s^{-1}(v_0) - v_0 \right| < \varepsilon \right\}$. On the event $A(s,\varepsilon)$, we have:

$$\left| V_n(s, F_s \circ \hat{F}_s^{-1}(v_0)) - V_n(s,v_0) \right| \leq \sup_{v \,:\, |v - v_0| < \varepsilon} |V_n(s,v) - V_n(s,v_0)| \ .$$

We bound the right hand side for fixed ε, by making use of an argument from van de Geer (2000). First, we need to put things into the right format. Set:

$$V_n(s,v) - V_n(s,v_0) = \frac{1}{n}\sum_{i=1}^{n} \left(u_i(s,v) - u_i(s,v_0) \right) \ ,$$

where $u_i(s,v) = Y_i \mathbb{I}\{s(X_i) \leq Q(s,v) < 0\} - \mathbb{E}\left(Y \mathbb{I}\{s(X) \leq Q(s,v)\}\right)$ for $s \in \mathcal{S}$ and $v \in (0,1)$. We observe that

$$|u_i(s,v) - u_i(s,v_0)| \leq d_i(v,v_0),$$

where

$$d_i(v,v_0) = \mathbb{I}\{s(X_i) \in [Q(s, v \wedge v_0), Q(s, v \vee v_0)]\} + |v - v_0| \ .$$

Denote by

$$\widehat{d}(v,v_0) = \frac{1}{n}\sum_{i=1}^{n} \mathbb{I}\{s(X_i) \in [Q(s, v\wedge v_0), Q(s, v\vee v_0)]\} + |v - v_0| \,.$$

a distance over $\mathbb{R}$. Set also:

$$\widehat{R}(\varepsilon) = \sup_{v\,:\,|v-v_0|<\varepsilon} \widehat{d}(v,v_0)\,.$$

and observe that

$$\widehat{R}(\varepsilon) = \frac{1}{n}\sum_{i=1}^{n} \mathbb{I}\{s(X_i) \in [Q(s, v_0-\varepsilon), Q(s, v_0+\varepsilon)]\} + \varepsilon\,.$$

We then have, by applying Lemma 8.5 from van de Geer (2000), for $nt^2/\widehat{R}^2(\varepsilon)$ sufficiently large,

$$\mathbb{P}\left\{ \sup_{v\,:\,|v-v_0|\leq\varepsilon} |V_n(s,v) - V_n(s,v_0)| \geq t \;\middle|\; X_1,\ldots,X_n \right\} \leq C\exp\left\{ -\frac{cnt^2}{\widehat{R}^2(\varepsilon)} \right\} \,,$$

for some positive constants c and C. It remains to integrate out and, for this purpose, we introduce the event:

$$\forall x>0\,, \qquad \Delta(x) = \left\{ 3\varepsilon - x \leq \widehat{R}(\varepsilon) \leq 3\varepsilon + x \right\}.$$

We then have:

$$\mathbb{E}\left(\exp\left\{ -\frac{cnt^2}{\widehat{R}^2(\varepsilon)} \right\} \right) \leq \exp\left\{ -\frac{cnt^2}{(3\varepsilon+x)^2} \right\} + \mathbb{P}\left\{ \overline{\Delta(x)} \right\} \,.$$

Now, we have, by Bernstein's inequality:

$$\mathbb{P}\left\{ \overline{\Delta(x)} \right\} = 2\mathbb{P}\left\{ \frac{1}{n}B(n, 2\varepsilon) - 2\varepsilon > x \right\} \leq 2\exp\left\{ -\frac{3nx^2}{16\varepsilon} \right\}$$

where we have used the notation $B(n,2\varepsilon)$ for a binomial $(n,2\varepsilon)$ random variable. We can take $x = O(t/\sqrt{\varepsilon})$ and assume also $x = o(\varepsilon)$ to get, for nt^2/ε^2 large enough,

$$\mathbb{P}\left\{ \sup_{v\,:\,|v-v_0|\leq\varepsilon} |V_n(s,v) - V_n(s,v_0)| \geq t \right\} \leq C\exp\left\{ -\frac{cnt^2}{\varepsilon^2} \right\} \,,$$

for some positive constants c and C. This can be reformulated, by writing that the following bound holds, with probability larger than $1-\delta/2$,

$$\sup_{v\,:\,|v-v_0|\leq\varepsilon} |V_n(s,v) - V_n(s,v_0)| \leq \varepsilon\sqrt{\frac{\log(2C/\delta)}{nc}}\,.$$

We recall that, by the triangle inequality and Dvoretsky-Kiefer-Wolfowitz theorem, if we take $\varepsilon = c\,\sqrt{\frac{\log(2/\delta)}{n}}$, we have $\mathbb{P}\{A(s,\varepsilon)\} \geq 1-\delta/2$. It follows that, with probability larger than $1-\delta$, we have, for some constant κ:

$$\left| V_n(s, F_s\circ\widehat{F}_s^{-1}(v_0)) - V_n(s,v_0) \right| \leq \kappa\,\left(\frac{\log(1/\delta)}{n} \right) \,,$$

for any $s \in \mathcal{S}$. Now it remains to deal with the second term $K(s, F_s \circ \hat{F}_s^{-1}(v_0)) - K(s, v_0)$. Therefore, by the differentiability assumption (iv), we have: $\forall s \in \mathcal{S}$,

$$\sup_{|v-v_0| \leq \delta} \{K(s,v) - K(s,v_0) - (v-v_0)K'(s,v_0)\} = O(\delta^2) \ , \quad \text{as } \delta \to 0 \ .$$

Since $|F_s \circ \hat{F}_s^{-1}(v_0)) - v_0| = O_{\mathbb{P}}(n^{-1/2})$, we get that

$$K(s, F_s \circ \hat{F}_s^{-1}(v_0)) - K(s,v_0) = K'(s,v_0)(F_s \circ \hat{F}_s^{-1}(v_0) - v_0) + O_{\mathbb{P}}(n^{-1}) \ , \quad \text{as } n \to \infty \ .$$

Moreover, as

$$F_s \circ \hat{F}_s^{-1}(v_0) - v_0 = -(\hat{F}_s \circ F_s^{-1}(v_0) - v_0) + O_{\mathbb{P}}(n^{-1}) \ ,$$

we finally obtain that

$$K(s, F_s \circ \hat{F}_s^{-1}(v_0)) - K(s,v_0) = -K'(s,v_0)(\hat{F}_s \circ F_s^{-1}(v_0) - v_0) + O_{\mathbb{P}}(n^{-1}) \ .$$

■

References

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.

P. Bartlett, M. Jordan, and J. McAuliffe. Convexity, classification, and risk bounds. *Journal of the American Statistical Association*, 101(473):138–156, 2006.

S. Boucheron, O. Bousquet, and G. Lugosi. Theory of classification: A survey of some recent advances. *ESAIM: Probability and Statistics*, 9:323–375, 2005.

L. Cavalier. Nonparametric estimation of regression level sets. *Statistics*, 29:131–160, 1997.

S. Clémençon, G. Lugosi, and N. Vayatis. Ranking and scoring using empirical risk minimization. In P. Auer and R. Meir, editors, *Proceedings of COLT 2005*, volume 3559 of *Lecture Notes in Computer Science*, pages 1–15. Springer, 2005.

S. Clémençon, G. Lugosi, and N. Vayatis. Ranking and empirical risk minimization of U-statistics. *The Annals of Statistics*, To appear.

C. Cortes and M. Mohri. Auc optimization vs. error rate minimization. In S. Thrun, L. Saul, and B. Schölkopf, editors, *Advances in Neural Information Processing Systems 16*. MIT Press, Cambridge, MA, 2004.

D. Cossock and T. Zhang. Statistical analysis of Bayes optimal subset ranking. Technical report, Yahoo! Research, 2006.

L. Devroye, L. Györfi, and G. Lugosi. *A Probabilistic Theory of Pattern Recognition*. Springer, 1996.

L. E. Dodd and M. S. Pepe. Partial AUC estimation and regression. *Biometrics*, 59(3):614–623, 2003.

R.M. Dudley. *Uniform Central Limit Theorems*. Cambridge University Press, 1999.

V. Dupac and J. Hájek. Asymptotic normality of simple linear rank statistics under alternatives ii. *The Annals of Mathematical Statistics*, (6):1992–2017, 1969.

J.P. Egan. *Signal Detection Theory and ROC Analysis*. Academic Press, 1975.

Y. Freund, R. D. Iyer, R. E. Schapire, and Y. Singer. An efficient boosting algorithm for combining preferences. *Journal of Machine Learning Research*, 4, 2003.

J. Hájek and Z. Sidák. *Theory of Rank Tests*. Academic Press, 1967.

J.A. Hanley and J. McNeil. The meaning and use of the area under a ROC curve. *Radiology*, (143): 29–36, 1982.

K. Järvelin and J. Kekäläinen. IR evaluation methods for retrieving highly relevant documents. In N.J. Belkin, P . Ingwersen, , and M.-K. Leong, editors, *Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 41–48, 2000.

H.L. Koul. *Weighted Empirical Processes in Dynamic Nonlinear Models*, volume 166 of *Lecture Notes in Statistics*. Springer, 2nd edition, 2002.

H.L. Koul. Some convergence theorems for ranks and weighted empirical cumulatives. *The Annals of Mathematical Statistics*, (41):1768–1773, 1970.

H.L. Koul and Jr. R.G. Staudte. Weak convergence of weighted empirical cumulatives based on ranks. *The Annals of Mathematical Statistics*, (43):823–841, 1972.

P. Li, C. Burges, and Q. Wu. Learning to rank using classification and gradient boosting. Technical report MSR-TR-2007-74, Microsoft Research, 2007.

G. Lugosi. Pattern classification and learning theory. In L. Györfi, editor, *Principles of Nonparametric Learning*, pages 1–56.

E. Mammen and A. B. Tsybakov. Smooth discrimination analysis. *Annals of Statistics*, 27(6): 1808–1829, 1999.

P. Massart. *Concentration Inequalities and Model Selection*. Lecture Notes in Mathematics. Springer, 2006.

P. Massart and E. Nédélec. Risk bounds for statistical learning. *Annals of Statistics*, 34(5), 2006.

A. Papoulis. *Probability, Random Variables, and Stochastic Processes*. McGraw-Hill, 1965.

C. Rudin. Ranking with a P-Norm Push. In H.U. Simon and G. Lugosi, editors, *Proceedings of COLT 2006*, volume 4005 of *Lecture Notes in Computer Science*, pages 589–604, 2006.

C. Rudin, C. Cortes, M. Mohri, and R. E. Schapire. Margin-based ranking and boosting meet in the middle. In P. Auer and R. Meir, editors, *Proceedings of COLT 2005*, volume 3559 of *Lecture Notes in Computer Science*, pages 63–78. Springer, 2005.

C. Scott. Performance measures for Neyman-Pearson classification. Technical report, Department of Statistics, Rice University, 2005.

C. Scott and M. Davenport. Regression level set estimation via cost-sensitive classification. *IEEE Transactions on Signal Processing*, 2006, to appear.

C. Scott and R. Nowak. A Neyman-Pearson approach to statistical learning. *IEEE Transactions on Information Theory*, 51(11):3806–3819, November 2005.

C. Scott and R. Nowak. Learning minimum volume sets. *Journal of Machine Learning Research*, 7:665–704, April 2006.

I. Steinwart, D. Hush, and C. Scovel. A classification framework for anomaly detection. *Journal of Machine Learning Research*, 6:211–232, 2005.

J. Taylor and R. Tibshirani. A tail strength measure for assessing the overall univariate significance in a dataset. *Biostatistics*, 7(2):167–181, 2006.

A. Tsybakov. Optimal aggregation of classifiers in statistical learning. *Annals of Statistics*, 32(1): 135–166, 2004.

A. Tsybakov. On nonparametric estimation of density level sets. *Annals of Statistics*, 25(3):948–969, 1997.

S. van de Geer. *Empirical Processes in M-Estimation*. Cambridge University Press, 2000.

A. van de Vaart. *Asymptotic Statistics*. Cambridge University Press, 1998.

H.L. van Trees. *Detection, Estimation, and Modulation Theory, Part I*. John Wiley, 1968.

R. Vert and J.-P. Vert. Consistency and convergence rates of one-class SVMs and related algorithms. *Journal of Machine Learning Research*, 7:817–854, May 2006.

R. Willett and R. Nowak. Minimax optimal level set estimation. Technical report, Rice University, 2006.

Journal of Machine Learning Research 8 (2007) 2701-2726 Submitted 6/07; Revised 7/07; Published 12/07

Stagewise Lasso

Peng Zhao PENGZHAO@STAT.BERKELEY.EDU
Bin Yu BINYU@STAT.BERKELEY.EDU
Department of Statistics
University of Berkeley
367 Evans Hall
Berkeley, CA 94720-3860, USA

Editor: Saharon Rosset

Abstract

Many statistical machine learning algorithms minimize either an empirical loss function as in AdaBoost, or a penalized empirical loss as in Lasso or SVM. A single regularization tuning parameter controls the trade-off between fidelity to the data and generalizability, or equivalently between bias and variance. When this tuning parameter changes, a regularization "path" of solutions to the minimization problem is generated, and the whole path is needed to select a tuning parameter to optimize the prediction or interpretation performance. Algorithms such as homotopy-Lasso or LARS-Lasso and Forward Stagewise Fitting (FSF) (aka e-Boosting) are of great interest because of their resulted sparse models for interpretation in addition to prediction.

In this paper, we propose the BLasso algorithm that ties the FSF (e-Boosting) algorithm with the Lasso method that minimizes the L_1 penalized L_2 loss. BLasso is derived as a coordinate descent method with a fixed stepsize applied to the general Lasso loss function (L_1 penalized convex loss). It consists of both a forward step and a backward step. The forward step is similar to e-Boosting or FSF, but the backward step is new and revises the FSF (or e-Boosting) path to approximate the Lasso path. In the cases of a finite number of base learners and a bounded Hessian of the loss function, the BLasso path is shown to converge to the Lasso path when the stepsize goes to zero. For cases with a larger number of base learners than the sample size and when the true model is sparse, our simulations indicate that the BLasso model estimates are sparser than those from FSF with comparable or slightly better prediction performance, and that the the discrete stepsize of BLasso and FSF has an additional regularization effect in terms of prediction and sparsity. Moreover, we introduce the Generalized BLasso algorithm to minimize a general convex loss penalized by a general convex function. Since the (Generalized) BLasso relies only on differences not derivatives, we conclude that it provides a class of simple and easy-to-implement algorithms for tracing the regularization or solution paths of penalized minimization problems.

Keywords: backward step, boosting, convexity, Lasso, regularization path

1. Introduction

Many statistical machine learning algorithms minimize either an empirical loss function or a penalized empirical loss, in a regression or a classification setting where covariate or predictor variables are used to predict a response variable and i.i.d. samples of training data are available. For example, in classification, both AdaBoost (Schapire, 1990; Freund, 1995; Freund and Schapire, 1996) and SVM (Vapnik, 1995; Cristianini and Shawe-Taylor, 2002; Schölkopf and Smola, 2002) build linear classification functions of basis functions of the covariates (predictors). Adaboost minimizes an

exponential loss function of the margin, while SVM a penalized hinge loss function of the margin. A single regularization tuning parameter, the number of iterations in AdaBoost and the smoothing parameter in SVM, controls the bias-and-variance trade-off or whether the algorithm overfits or underfits the data. When this tuning parameter changes, a regularization "path" of solutions to the minimization problem is generated. These algorithms have been shown to achieve start-of-the-art prediction performance. A tuning parameter value, or equivalently a point on the path of solutions, is chosen to minimize the estimated prediction error over a proper test set or through cross-validation. Hence it is necessary to have algorithms that can generate the path of solutions in an efficient manner. Path following algorithms have also been devised in other statistical penalized minimization problems such as the problems of information bottleneck Tishby et al. (1999) and information distortion Gedeon et al. (2002) where the loss and penalty functions are different from these discussed in this paper. In fact, following the solution paths of numerical problems is the focus of homotopy, a sub-area in numerical analysis. Interested readers are referred to the book Allgower and Georg (1980) (http://www.math.colostate.edu/emeriti/georg/AllgGeorgHNA.pdf) and references at http://www.math.colostate.edu/emeriti/georg/georg.publications.html.

Among all the machine learning methods, those that produce sparse models are of great interest because sparse models lend themselves more easily to interpretation and therefore preferred in sciences and social sciences. Lasso (Tibshirani, 1996) is such a method. It minimizes the L_2 loss function with an L_1 penalty on the parameters in a linear regression model. In signal processing it is called Basis Pursuit (Chen and Donoho, 1994). The L_1 penalty leads to sparse solutions, that is, there are few predictors or basis functions with nonzero weights (among all possible choices). This statement is proved asymptotically under various conditions by different authors (see, e.g., Knight and Fu, 2000; Osborne et al., 2000a,b; Donoho et al., 2006; Donoho, 2006; Rosset et al., 2004; Tropp, 2006; Zhao and Yu, 2006; Zou, 2006; Meinshausen and Yu, 2006; Zhang and Huang, 2006).

Sparsity has also been observed in the models generated by Forward Stagewise Fitting (FSF) or e-Boosting. FSF is a gradient descent procedure with more cautious steps than L_2Boosting (i.e., the usual coordinatewise gradient descent method applied to the L_2 loss) (Rosset et al., 2004; Hastie et al., 2001; Efron et al., 2004). Moreover, these papers also study the similarities between FSF (e-Boosting) with Lasso. This link between Lasso and e-Boosting or FSF is more formally described for the linear regression case through the LARS algorithm (Least Angle Regression Efron et al., 2004). It is also shown in Efron et al. (2004) that for special cases (such as orthogonal designs) e-Boosting or FSF can approximate Lasso path infinitely close, but in general, it is unclear what regularization criterion e-Boosting or FSF optimizes. As can be seen in our experiments (Figure 1 in Section 6.1), e-Boosting or FSF solutions can be significantly different from the Lasso solutions in the case of strongly correlated predictors which are common in high-dimensional data problems. However, FSF is still used as an approximation to Lasso because it is often computationally prohibitive to solve Lasso with general loss functions for many regularization parameters through Quadratic Programming.

In this paper, we propose a new algorithm **BLasso** that connects Lasso with FSF or e-Boosting (and the B in the name stands for this connection to boosting). BLasso generates approximately the Lasso path in general situations for both regression and classification for L_1 penalized convex loss function. The motivation for BLasso is a critical observation that FSF or e-Boosting only works in a forward fashion. It takes steps that reduce empirical loss the most regardless of the impact on model complexity (or the L_1 penalty). Hence it is not able to adjust earlier steps. Taking a coordinate (difference) descent view point of the Lasso minimization with a fixed stepsize, we introduce an

innovative "backward" step. This step uses the same minimization rule as the forward step to define each fitting stage but with an extra rule to force the model complexity or L_1 penalty to decrease. By combining backward and forward steps, BLasso is able to go back and forth to approximate the Lasso path correctly.

BLasso can be seen as a marriage between two families of successful methods. Computationally, BLasso works similarly to e-Boosting and FSF. It isolates the sub-optimization problem at each step from the whole process, that is, in the language of the Boosting literature, each base learner is learned separately. This way BLasso can deal with different loss functions and large classes of base learners like trees, wavelets and splines by fitting a base learner at each step and aggregating the base learners as the algorithm progresses. Moreover, the solution path of BLasso can be shown to converge to that of the Lasso, which uses explicit global L_1 regularization for cases with a finite number of base learners. In contrast, e-Boosting or FSF can be seen as local regularization in the sense that at any iteration, FSF with a fixed small stepsize only searches over those models which are one small step away from the current one in all possible directions corresponding to the base learners or predictors (cf. Hastie et al., 2006, for a recent interpretation of the $\varepsilon \to 0$ case).

In particular, we make three contributions in this paper via BLasso. First, by introducing the backward step we modify e-Boosting or FSF to follow the Lasso path and consequently generate models that are sparser with equivalent or slightly better prediction performance in our simulations (with true sparse models and more predictors than the sample size). Secondly, by showing convergence of BLasso to the Lasso path, we further tighten the conceptual ties between e-Boosting or FSF and Lasso that have been considered in previous works. Finally, since BLasso can be generalized to deal with other convex penalties and does not use any derivatives of the Loss function or penalty, we provide the Generalized BLasso algorithm as a simple and easy-to-implement off-the-shelf method for approximating the regularization path for a general loss function and a general convex penalty.

We would like to note that, for the original Lasso problem, that is, the least squares problem (L_2 loss) with an L_1 penalty, algorithms that give the entire Lasso path have been established, namely, the homotopy method by Osborne et al. (2000b) and the LARS algorithm by Efron et al. (2004). For parametric least squares problems where the number of predictors is not large, these methods are very efficient as their computational complexity is on the same order as a single Ordinary Least Squares regression. For other problems such as classification and for nonparametric setups like model fitting with trees, FSF or e-Boosting has been used as a tool for approximating the Lasso path (Rosset et al., 2004). For such problems, BLasso operates in a similar fashion as FSF or e-Boosting but, unlike FSF, BLasso can be shown to converge to the Lasso path quite generally when the stepsize goes to zero.

The rest of the paper is organized as follows. In Section 2.1, the gradient view of Boosting is provided and FSF (e-Boosting) is reviewed as a coordinatewise descent method with a fixed stepsize on the L_2 loss. In Section 2.2, the Lasso empirical minimization problem is reviewed. Section 3 introduces BLasso that is a coordinate descent algorithm with a fixed stepsize applied to the Lasso minimization problem. Section 4 discusses the backward step and gives the intuition behind BLasso and explains why FSF is unable to give the Lasso path. Section 5 introduces a Generalized BLasso algorithm which deals with general convex penalties. In Section 6, results of experiments with both simulated and real data are reported to demonstrate the attractiveness of BLasso. BLasso is shown as a learning algorithm that gives sparse models and good prediction and as a simple plug-in method for approximating the regularization path for different convex loss functions and penalties. Moreover, we compare different choices of the stepsize and give evidence for the regularization

effect of using moderate stepsizes. Finally, Section 7 is a discussion and a summary. In particular, it comments on the computational complexity of BLasso, compares with the algorithm in Rosset (2004), explores the possibility of BLasso for nonparametric learning problems, summarizes the paper, and points to future directions.

2. Boosting, Forward Stagewise Fitting and the Lasso

Boosting was originally proposed as an iterative fitting procedure that builds up a model sequentially using a weak or base learner and then carries out a weighted averaging (Schapire, 1990; Freund, 1995; Freund and Schapire, 1996). More recently, boosting has been interpreted as a gradient descent algorithm on an empirical loss function. FSF or e-Boosting can be viewed as a gradient descent with a fixed small stepsize at each stage and it produces solutions that are often close to the Lasso solutions (path). We now give a brief gradient descent view of Boosting and of FSF (e-Boosting), followed by a review of the Lasso minimization problem.

2.1 Boosting and Forward Stagewise Fitting

Given data $Z_i = (Y_i, X_i)(i = 1, ..., n)$, where the univariate Y can be continuous (regression problem) or discrete (classification problem), our task is to estimate the function $F : R^d \to R$ that minimizes an expected loss

$$E[C(Y, F(X))], \;\; C(\cdot,\cdot) : R \times R \to R^+.$$

The most prominent examples of the loss function $C(\cdot,\cdot)$ include exponential loss (AdaBoost), logit loss and L_2 loss.

The family of $F(\cdot)$ being considered is the set of ensembles of "base learners"

$$D = \{F : F(x) = \sum_j \beta_j h_j(x), x \in R^d, \beta_j \in R\},$$

where the family of base learners can be very large or contain infinite members, for example, trees, wavelets and splines.

Let $\beta = (\beta_1, ...\beta_j, ...)^T$, we can re-parametrize the problem using

$$L(Z, \beta) := C(Y, F(X)),$$

where the specification of F is hidden by L to make our notation simpler.

To find an estimate for β, we set up an empirical minimization problem:

$$\hat{\beta} = \arg\min_{\beta} \sum_{i=1}^{n} L(Z_i; \beta).$$

Despite the fact that the empirical loss function is often convex in β, exact minimization is usually a formidable task for a moderately rich function family of base learners and with such function families the exact minimization leads to overfitted models. Because the family of base learners is usually large, Boosting can be viewed as finding approximate solutions by applying functional gradient descent. This gradient descent view has been recognized and studied by various authors including Breiman (1998), Mason et al. (1999), Friedman et al. (2000), Friedman (2001) and Buhlmann and Yu (2003). Precisely, boosting is a progressive procedure that iteratively builds up the solution (and it is often stopped early to avoid overfitting):

$$(\hat{j},\hat{g}) = \arg\min_{j,g} \sum_{i=1}^{n} L(Z_i;\hat{\beta}^t + g1_j), \quad (1)$$

$$\hat{\beta}^{t+1} = \hat{\beta}^t + \hat{g}1_{\hat{j}}, \quad (2)$$

where 1_j is the jth standard basis vector with all 0's except for a 1 in the jth coordinate, and $g \in R$ is stepsize. In other words, Boosting favors the direction $\hat{j}$ that reduces most the empirical loss and $\hat{g}$ is found through a line search. The well-known AdaBoost, LogitBoost and L_2Boosting can all be viewed as implementations of this strategy for different loss functions.

Forward Stagewise Fitting (FSF) (Efron et al., 2004) is a similar method for approximating the minimization problem described by (1) with some additional regularization. FSF has also been called e-Boosting for ε-Boosting as in Rosset et al. (2004). Instead of optimizing the stepsize as in (2), FSF updates $\hat{\beta}^t$ by a fixed stepsize ε as in Friedman (2001).

For general loss functions, FSF can be defined by removing the minimization over g in (1):

$$(\hat{j},\hat{s}) = \arg\min_{j,s=\pm\varepsilon} \sum_{i=1}^{n} L(Z_i;\hat{\beta}^t + s1_j), \quad (3)$$

$$\hat{\beta}^{t+1} = \hat{\beta}^t + \hat{s}1_{\hat{j}}. \quad (4)$$

This description looks different from the FSF described in Efron et al. (2004), but the underlying mechanic of the algorithm remains unchanged (see Section 5). Initially all coefficients are zero. At each successive step, a basis function or predictor or coordinate is selected that reduces most the empirical loss. Its corresponding coefficient $\beta_{\hat{j}}$ is then incremented or decremented by a fixed amount ε, while all other coefficients β_j, $j \neq \hat{j}$ are left unchanged.

By taking small steps, FSF imposes some local regularization or shrinkage. A related approach can be found in Zhang (2003) where a relaxed gradient descent method is used. After $T < \infty$ iterations, many of the estimated coefficients by FSF will be zero, namely those that have yet to be incremented. The others will tend to have absolute values smaller than the unregularized solutions. This shrinkage/sparsity property is reflected in the similarity between the solutions given by FSF and Lasso which is reviewed next.

2.2 General Lasso

Let $T(\beta)$ denote the L_1 penalty of $\beta = (\beta_1,...,\beta_j,...)^T$, that is, $T(\beta) = \|\beta\|_1 = \sum_j |\beta_j|$, and $\Gamma(\beta;\lambda)$ denote the Lasso (least absolute shrinkage and selection operator) loss function

$$\Gamma(\beta;\lambda) = \sum_{i=1}^{n} L(Z_i;\beta) + \lambda T(\beta).$$

The general Lasso estimate $\hat{\beta} = (\hat{\beta}_1,...,\hat{\beta}_j,...)^T$ is defined by

$$\hat{\beta}_\lambda = \min_{\beta} \Gamma(\beta;\lambda).$$

The parameter $\lambda \geq 0$ controls the amount of regularization applied to the estimate. Setting $\lambda = 0$ reverses the Lasso problem to minimizing the unregularized empirical loss. On the other hand, a

very large λ will completely shrink $\hat{\beta}$ to 0 thus leading to the empty or null model. In general, moderate values of λ will cause shrinkage of the solutions towards 0, and some coefficients may end up being exactly 0. This sparsity in Lasso solutions has been researched extensively in recent years (e.g., Osborne et al., 2000a,b; Efron et al., 2004; Donoho et al., 2006; Donoho, 2006; Tropp, 2006; Rosset et al., 2004; Meinshausen and Bühlmann, 2005; Candes and Tao, 2007; Zhao and Yu, 2006; Zou, 2006; Wainwright, 2006; Meinshausen and Yu, 2006; Zhang and Huang, 2006). Sparsity can also result from other penalties as in, for example, Fan and Li (2001).

Computation of the solution to the Lasso problem for a fixed λ has been studied for special cases. Specifically, for least squares regression, it is a quadratic programming problem with linear inequality constraints; for 1-norm SVM, it can be transformed into a linear programming problem. But to get a model that performs well on future data, we need to select an appropriate value for the tuning parameter λ. Very efficient algorithms have been proposed to give the entire regularization path for the squared loss function (the homotopy method by Osborne et al. 2000b and similarly LARS by Efron et al. 2004) and SVM (1-norm SVM by Zhu et al., 2003).

However, it remains open how to give the entire regularization path of the Lasso problem for general convex loss function. FSF exists as a compromise since, like Boosting, it is a nonparametric learning algorithm that works with different loss functions and large numbers of base learners (predictors) but it is local regularization and does not converge to the Lasso path in general. As can be seen in Sec. 6.2, FSF has also less sparse solutions comparing to Lasso in our simulations.

Next we propose the BLasso algorithm which works in a computationally efficient fashion as FSF. In contrast to FSF, BLasso converges to the Lasso path for general convex loss functions when the stepsize goes to 0. This relationship between Lasso and BLasso leads to sparser solutions for BLasso comparing to FSF with similar or slightly better prediction performance in our simulation set-up with different choices of the stepsize.

3. The BLasso Algorithm

We first describe the BLasso algorithm (**Algorithm 1**). This algorithm has two related input parameters, a stepsize ε and a tolerance level ξ.

The tolerance level is needed only to avoid numerical instability when assessing changes of the empirical loss function and should be set as small as possible while accommodating the numerical accuracy of the implementation. (ξ is set to 10^{-6} in the implementation of the algorithm that used in this paper.)

We will discuss forward and backward steps in depth in the next section. Immediately, the following properties can be proved for BLasso (see Appendix for the proof).

Lemma 1.

1. For any $\lambda \geq 0$, if there exist j and s with $|s| = \varepsilon$ such that $\Gamma(s1_j;\lambda) \leq \Gamma(0;\lambda)$, we have $\lambda^0 \geq \lambda$.
2. For any t, we have $\Gamma(\hat{\beta}^{t+1};\lambda^t) \leq \Gamma(\hat{\beta}^t;\lambda^t) - \xi$.
3. For $\xi \geq 0$ and any t such that $\lambda^{t+1} < \lambda^t$, we have $\Gamma(\hat{\beta}^t \pm \varepsilon 1_j;\lambda^t) > \Gamma(\hat{\beta}^t;\lambda^t) - \xi$ for every j and $\|\hat{\beta}^{t+1}\|_1 = \|\hat{\beta}^t\|_1 + \varepsilon$.

Lemma 1 (1) guarantees that it is safe for BLasso to start with an initial λ_0 which is the largest λ that would allow an ε step away from 0 (i.e., larger λ's correspond to $\hat{\beta}_\lambda = 0$). Lemma 1 (2) says

Algorithm 1 BLasso

Step 1 (initialization). Given data $Z_i = (Y_i, X_i)$, $i = 1, \ldots, n$ and a small stepsize constant $\varepsilon > 0$ and a small tolerance parameter $\xi > 0$, take an initial forward step

$$\begin{aligned}(\hat{j}, \hat{s}_{\hat{j}}) &= \arg\min_{j, s=\pm\varepsilon} \sum_{i=1}^{n} L(Z_i; s1_j), \\ \hat{\beta}^0 &= \hat{s}_{\hat{j}} 1_{\hat{j}},\end{aligned}$$

Then calculate the initial regularization parameter

$$\lambda^0 = \frac{1}{\varepsilon}\left(\sum_{i=1}^{n} L(Z_i; 0) - \sum_{i=1}^{n} L(Z_i; \hat{\beta}^0)\right).$$

Set the active index set $I_A^0 = \{\hat{j}\}$. Set $t = 0$.

Step 2 (Backward and Forward steps). Find the "backward" step that leads to the minimal empirical loss:

$$\hat{j} = \arg\min_{j \in I_A^t} \sum_{i=1}^{n} L(Z_i; \hat{\beta}^t + s_j 1_j) \quad \text{where } s_j = -\text{sign}(\hat{\beta}_j^t)\varepsilon. \tag{5}$$

Take the step if it leads to a decrease of moderate size ξ in the Lasso loss, otherwise force a forward step (as (3), (4) in FSF) and relax λ if necessary:

If $\Gamma(\hat{\beta}^t + \hat{s}_{\hat{j}} 1_{\hat{j}}; \lambda^t) - \Gamma(\hat{\beta}^t, \lambda^t) \leq -\xi$, then

$$\hat{\beta}^{t+1} = \hat{\beta}^t + \hat{s}_{\hat{j}} 1_{\hat{j}}, \lambda^{t+1} = \lambda^t.$$

Otherwise,

$$\begin{aligned}(\hat{j}, \hat{s}) &= \arg\min_{j, s=\pm\varepsilon} \sum_{i=1}^{n} L(Z_i; \hat{\beta}^t + s1_j), \qquad (6) \\ \hat{\beta}^{t+1} &= \hat{\beta}^t + \hat{s} 1_{\hat{j}}, \\ \lambda^{t+1} &= \min[\lambda^t, \frac{1}{\varepsilon}(\sum_{i=1}^{n} L(Z_i; \hat{\beta}^t) - \sum_{i=1}^{n} L(Z_i; \hat{\beta}^{t+1}) - \xi)], \\ I_A^{t+1} &= I_A^t \cup \{\hat{j}\}.\end{aligned}$$

Step 3 (iteration). Increase t by one and repeat Step 2 and 3. Stop when $\lambda^t \leq 0$.

that for each value of λ, BLasso performs coordinate descent until there is no descent step. Then, by Lemma 1 (3), the value of λ is reduced and a forward step is forced. The stepsize ε controls fineness of the grid BLasso runs on. The tolerance ξ controls how large a descend need to be made for a backward step to be taken. It is needed to accommodate for numerical error and should be set to be much smaller than ε to have a good approximation (see Proof of Theorem 1). In fact, we have a convergence result for BLasso (detailed proof is included in the Appendix):

Theorem 1. For a finite number of base learners and $\xi = o(\varepsilon)$, if $\sum L(Z_i;\beta)$ is strongly convex with bounded second derivatives in β then as $\varepsilon \to 0$, the BLasso path converges to the Lasso path uniformly.

Note that Conjecture 2 of Rosset et al. (2004) follows from Theorem 1. This is because if all the optimal coefficient paths are monotone, then BLasso will never take a backward step, so it will be equivalent to e-Boosting.

Many popular loss functions, for example, squared loss, logistic loss, and negative log-likelihood functions of exponential families are convex and twice differentiable, and they satisify the conditions in Theorem 1. Moreover, from the proof of this theorem in the appendix, it is easy to see that it suffices to have the conditions in the theorem satisfied over a bounded set of β. For the exponential loss, Lemma 1 implies that there is a finite $\lambda_0 < \infty$ for every data set (Z_i). Thus we can restrict the proof of Theorem 1 to this bounded set of β to show the result for the exponential loss. Other functions like the hinge loss (SVM) is continuous and convex but not differentiable. The differentiability, however, is only necessary for the proof of Theorem 1. BLasso does not use any gradient or higher order derivatives but only the differences of the loss function therefore remains applicable to loss functions that are not differentiable or of which differentiation is too complex or computationally expensive. It is theoretically possible that BLasso's coordinate descent strategy gets stuck at nondifferentiable points for functions like the hinge loss. However, as illustrated in our third experiment, BLasso may still work for cases like 1-norm SVM empirically.

Theorem 1 does not cover nonparametric learning problems with an infinite number of base learners either. In fact, for problems with large or infinite number of base learners, the minimization in (6) is usually done approximately by functional gradient descent and a tolerance $\xi > 0$ needs to be chosen to avoid oscillation between forward and backward steps caused by slow descending. We discuss more on this topic in the discussion (Sec. 7).

4. The Backward Step

We now explain the motivation and working mechanic of BLasso. Observe that FSF only uses "forward" steps, that is, it only takes steps that lead to a direct reduction of the empirical loss. Comparing to classical model selection methods like Forward Selection and Backward Elimination, Growing and Pruning of a classification tree, a "backward" counterpart is missing. Without the backward step, when FSF picks up more irrelevant variables as compared to the Lasso path in some cases (cf. Figure 1 in Section 6.2), it does not have a mechanism to remove them. As seen below, this backward step naturally arises in BLasso because of our coordinate descent view of the minimization of the Lasso loss. (Since ξ exists for numerical purpose only, it is assumed to be 0 thus excluded in the following theoretical discussion.)

For a given $\beta \neq 0$ and $\lambda > 0$, consider the impact of a small $\varepsilon > 0$ change of β_j to the Lasso loss $\Gamma(\beta;\lambda)$. For an $|s| = \varepsilon$,

$$\begin{aligned}\Delta_j\Gamma(Z;\beta) &= \Big(\sum_{i=1}^n L(Z_i;\beta + s1_j) - \sum_{i=1}^n L(Z_i;\beta)\Big) + \lambda(T(\beta + s1_j) - T(\beta)) \\ &:= \Delta_j\Big(\sum_{i=1}^n L(Z_i;\beta)\Big) + \lambda\Delta_j T(\beta).\end{aligned}$$

Since $T(\beta)$ is simply the L_1 norm of β, $\Delta T(\beta)$ reduces to a simple form:

$$\begin{aligned}\Delta_j T(\beta) &= \|\beta + s1_j\|_1 - \|\beta\|_1 = |\beta_j + s| - |\beta_j| \\ &= \varepsilon \cdot \text{sign}^+(\beta_j, s) \\ &= \varepsilon \cdot \begin{cases} 1 & \text{if} \quad s\beta_j > 0 \text{ or } \beta_j = 0 \\ \text{-1} & \text{if} \quad s\beta_j < 0 \end{cases}.\end{aligned} \tag{7}$$

Equation (7) shows that an ε step changes the penalty by a fixed ε in absolute value for any j. That is, only the sign of the penalty change may vary. In the beginning of BLasso, all j directions are leaving zero and hence changing the L_1 penalty by the same positive amount $\lambda \cdot \varepsilon$. Therefore the first step of BLasso is a forward step because minimizing Lasso loss is equivalent to minimizing the L_2 loss due to the same positive change of the L_1 penalty. As the algorithm proceeds, some of the penalty changes might become negative and minimizing the empirical loss is no longer equivalent to minimizing the Lasso loss. In fact, except for special cases like orthogonal covariates (predictors), the FSF steps might result in negative changes of the L_1 penalty. In some of these situations, a step that goes "backward" reduces the penalty with a small sacrifice in the empirical loss. In general, to minimize the Lasso loss, one needs to go "back and forth" to trade off the penalty with the empirical loss for different regularization parameters.

To be precise, for a given $\hat{\beta}$, a **backward step** is such that:

$$\Delta\hat{\beta} = s_j 1_j, \text{ subject to } \hat{\beta}_j \neq 0, \text{sign}(s) = -\text{sign}(\hat{\beta}_j) \text{ and } |s| = \varepsilon.$$

Making such a step will reduce the penalty by a fixed amount $\lambda \cdot \varepsilon$, but its impact on the empirical loss can be different, therefore as in (5) we want:

$$\hat{j} = \arg\min_j \sum_{i=1}^{n} L(Z_i; \hat{\beta} + s_j 1_j) \text{ subject to } \hat{\beta}_j \neq 0 \text{ and } s_j = -\text{sign}(\hat{\beta}_j)\varepsilon,$$

that is, $\hat{j}$ is selected such that the empirical loss after making the step is as small as possible.

While forward steps try to reduce the Lasso loss through minimizing the empirical loss, the backward steps try to reduce the Lasso loss through minimizing the Lasso penalty. In summary, by allowing the backward steps, we are able to work with the Lasso loss directly and take backward steps to correct earlier forward steps that might have picked up irrelevant variables.

Since much of the discussion on the similarity and difference between FSF and Lasso is focused on Least Squares problems (e.g., Efron et al., 2004; Hastie et al., 2001), we next examine the BLasso algorithm in this case. It is straightforward to see that in LS problems both forward and backward steps in BLasso are based only on the correlations between fitted residuals and the covariates (predictors). It follows that BLasso in this case reduces to finding the best direction in both forward and backward steps by examining the inner-products, and then deciding whether to go forward or backward based on the regularization parameter. This not only simplifies the minimization procedure but also significantly reduces the computation complexity for a large number of observations since the inner-product between η_t and X_j can be updated by

$$(\eta^{t+1})'X_j = (\eta^t - sX_{\hat{j}^t})'X_j = (\eta^t)'X_j - sX'_{\hat{j}^t}X_j, \tag{8}$$

which takes only one operation if $X'_{\hat{j}^t}X_j$ is precalculated. Therefore, when the number of base learners is small, based on precalculated $X'X$ and $Y'X$, BLasso could use (8) to make its computation

complexity independent from the number of observations. This nice property is not surprising as it is also observed in established algorithms like LARS and Osborne's homotopy method which are specialized for LS problems.

In nonparametric situations, the number of base learners is large therefore the aforementioned strategy becomes inefficient. BLasso has a natural extention to this case as follows: similar to boosting, the forward step is carried out by a sub-optimization procedure such as fitting trees, smoothing splines or stumps. For the backward step, only inner-products between base learners that have entered the model need to be calculated. The inner products between these base learners and residuals can be updated by (8). This makes the backward steps' computation complexity proportional to the number of base learners that are already chosen instead of the number of all possible base learners. Therefore BLasso works not only for cases with large sample size but also for cases where a class of large or infinite number of possible base learners is given.

As mentioned earlier, there are already established efficient algorithms for solving the least square (L_2) Lasso problem, for example, the homotopy method by Osborne et al. (2000b) and LARS (Efron et al., 2004). These algorithms are very efficient for giving the exact Lasso paths for parametric settings. For nonparametric learning problems with a large or an infinite number of base learners, we believe BLasso is an attractive strategy for approximating the path of the Lasso, as it shares the same computational strategy as Boosting which has proven itself successful in applications. Also, in cases where the Ordinary Least Square (OLS) method performs well, BLasso can be modified to start from the OLS estimate, go backward and stop in a few iterations.

5. Generalized BLasso

As stated earlier, BLasso not only works for general convex loss functions, but also extends to convex penalties other than the L_1 penalty. For the Lasso problem, BLasso does a fixed stepsize coordinate descent to minimize the penalized loss. Since the penalty has the special L_1 norm and (7) holds, a step's impact on the penalty has a fixed size ε with either a positive or a negative sign, and the coordinate descent takes form of "backward" and "forward" steps. This reduces the minimization of the penalized loss function to unregularized minimizations of the loss function as in (6) and (5). For general convex penalties, since a step on different coordinates does not necessarily have the same impact on the penalty, one is forced to work with the penalized function directly. Assume $T(\beta)$: $R^m \to R$ is a convex penalty function. We next describe the Generalized BLasso algorithm (**Algorithm 2**).

In the Generalized BLasso algorithm, explicit "forward" or "backward" steps are no longer seen. However, the mechanism remains the same—minimize the penalized loss function for each λ, relax the regularization by reducing λ through a "forward" step when the minimum of the loss function for the current λ is reached.

6. Experiments

In this section, three experiments are carried out to illustrate the attractiveness of BLasso. The first experiment runs BLasso under the classical Lasso setting on the diabetes data set (cf. Efron et al., 2004) often used in studies of Lasso with an added artificial covariate variable to highlight the difference between BLasso and FSF. This added covariate is strongly correlated with a couple of the original covariates (predictors). In this case, BLasso is seen to produce a path almost exactly the

Algorithm 2 Generalized BLasso

Step 1 (initialization). Given data $Z_i = (Y_i, X_i)$, $i = 1, ..., n$ and a fixed small stepsize $\varepsilon > 0$ and a small tolerance parameter $\xi \geq 0$, take an initial forward step

$$(\hat{j}, \hat{s}_{\hat{j}}) = \arg\min_{j, s=\pm\varepsilon} \sum_{i=1}^{n} L(Z_i; s1_j), \hat{\beta}^0 = \hat{s}_{\hat{j}} 1_{\hat{j}}.$$

Then calculate the corresponding regularization parameter

$$\lambda^0 = \frac{\sum_{i=1}^n L(Z_i; 0) - \sum_{i=1}^n L(Z_i; \hat{\beta}^0)}{T(\hat{\beta}^0) - T(0)}.$$

Set $t = 0$.

Step 2 (steepest descent on Lasso loss). Find the steepest coordinate descent direction on the penalized loss:

$$(\hat{j}, \hat{s}_{\hat{j}}) = \arg\min_{j, s=\pm\varepsilon} \Gamma(\hat{\beta}^t + s1_j; \lambda^t).$$

Update $\hat{\beta}$ if it reduces Lasso loss by at least a ξ amount, otherwise force $\hat{\beta}$ to minimize L and recalculate the regularization parameter:
If $\Gamma(\hat{\beta}^t + \hat{s}_{\hat{j}} 1_{\hat{j}}; \lambda^t) - \Gamma(\hat{\beta}^t, \lambda^t) < -\xi$, then

$$\hat{\beta}^{t+1} = \hat{\beta}^t + \hat{s}_{\hat{j}} 1_{\hat{j}}, \lambda^{t+1} = \lambda^t.$$

Otherwise,

$$\begin{aligned}
(\hat{j}, \hat{s}_{\hat{j}}) &= \arg\min_{j, |s|=\varepsilon} \sum_{i=1}^{n} L(Z_i; \hat{\beta}^t + s1_j), \\
\hat{\beta}^{t+1} &= \hat{\beta}^t + \hat{s}_{\hat{j}} 1_{\hat{j}}, \\
\lambda^{t+1} &= \min[\lambda^t, \frac{\sum_{i=1}^n L(Z_i; \hat{\beta}^t) - \sum_{i=1}^n L(Z_i; \hat{\beta}^{t+1})}{T(\hat{\beta}^{t+1}) - T(\hat{\beta}^t)}].
\end{aligned}$$

Step 3 (iteration). Increase t by one and repeat Step 2 and 3. Stop when $\lambda^t \leq 0$.

same as the Lasso path which shrinks the added irrelevant variable back to zero, while FSF's path parts drastically from Lasso's due to the added strongly correlated covariate and does not move it back to zero.

In the second experiment, we compare the prediction and variable selection performance of FSF and BLasso in a least squares regression simulation using a large number ($p = 500 >> n = 50$) of randomly correlated base learners to emulate the nonparametric learning scenario and when the true model is sparse. The result shows, overall, BLasso gives sparser solutions than FSF and with similar or slightly better predictions. And this holds for various stepsizes. Moreover, we find that when the stepsize increases, there is a regularization effect in terms of both prediction and sparsity, for both BLasso and FSF.

The last experiment is to illustrate BLasso as an off-the-shelf method for computing the regularization path for general convex loss functions and general convex penalties. Two cases are presented. The first case is bridge regression (Frank and Friedman, 1993) on diabetes data using different L_γ ($\gamma \geq 1$) norms as penalties. The other is a simulated classification problem using 1-norm SVM (Zhu et al., 2003) with the hinge loss.

6.1 L_2 Regression with L_1 Penalty (Classical Lasso)

The data set used in this experiment is the diabetes data set where n=442 diabetes patients were measured on 10 baseline predictor variables $X^1, ..., X^{10}$. A prediction model was desired for the response variable Y, a quantitative measure of disease progression one year after baseline. We add one additional predictor variable to make more visible the difference between FSF and Lasso solutions. This added variable is

$$X^{11} = -X^7 + X^8 + 5X^9 + e,$$

where e is i.i.d. Gaussian noise (mean zero and variance $1/442$). The following vector gives the correlations of X^{11} with $X^1, X^2, ..., X^{10}$:

$$(0.25\ ,\ 0.24\ ,\ 0.47\ ,\ 0.39\ ,\ 0.48\ ,\ 0.38\ ,\ -0.58\ ,\ 0.76\ ,\ 0.94\ ,\ 0.47).$$

The classical Lasso (L_2 regression with L_1 penalty) is applied to this data set with the added covariate. Location and scale transformations are made so that all the covariates or predictors are standardized to have mean 0 and unit length, and the response has mean zero.

The penalized loss function has the form:

$$\Gamma(\beta;\lambda) = \sum_{i=1}^{n} (Y_i - X_i\beta)^2 + \lambda\|\beta\|_1.$$

The middle panel of Figure 1 shows the coefficient path plot for BLasso applied to the modified diabetes data. Left (Lasso) and Middle (BLasso) panels are indistinguishable from each other. Both FSF and BLasso pick up the added artificial and strongly correlated X^{11} (the solid line) in the earlier stages, but due to the greedy nature of FSF, it is not able to remove X^{11} in the later stages thus every parameter estimate is affected leading to significantly different solutions from Lasso.

The BLasso solutions were built up in 8700 steps (making the step size $\varepsilon = 0.5$ small so that the coefficient paths are smooth), 840 of which were backward steps. In comparison, FSF took 7300 pure forward steps. BLasso's backward steps concentrate mainly around the steps where FSF and BLasso tend to differ.

6.2 Comparison of BLasso and Forward Stagewise Fitting by Simulation

In this experiment, we compare the model estimates generated by FSF and BLasso in a large p(=500) and small n(=50) setting to mimic a nonparametric learning scenario where FSF and BLasso are computationally attractive. In this least squares regression simulation, the design is randomly generated as described below to guarantee a fair amount of correlation among the covariates (predictors). Otherwise, if the design is close to orthogonal, the FSF and BLasso paths will be too similar for this simulation to yield interesting results.

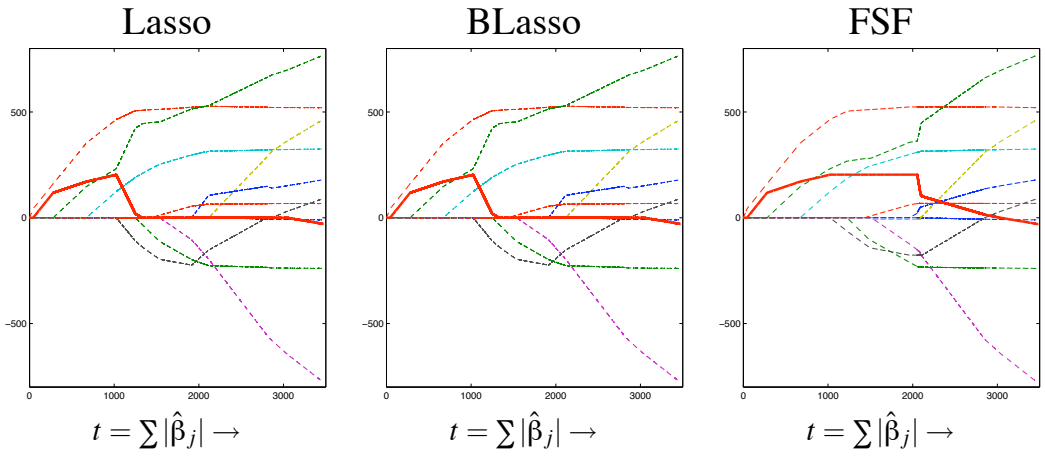


Figure 1: Regularization path plots, for the diabetes data set, of Lasso, BLasso and FSF: the curves (or paths) of estimates $\hat{\beta}_j$ for 10 original and 1 added covariates (predictors), as the regularization is relaxed or t tends to infinity. The thick solid curves correspond to the 11th added covariate. *Left Panel:* Lasso solution paths (produced using simplex search method on the penalized empirical loss function for each λ) as a function of $t = \|\beta\|_1$. *Middle Panel:* BLasso solution paths, which can be seen indistinguishable to the Lasso solutions. *Right Panel:* FSF solution paths, which are different from Lasso and BLasso.

We first draw 5 covariance matrices C_i, $i = 1,..,5$ from $.95 \times D_i + .05 I_{p \times p}$ where D_i is sampled from $Wishart(20, p)$ then normalized to have 1's on diagonal. The Wishart distribution creates a fair amount of correlation in C_i (average absolute value is about 0.18) between the covariates and the added identity matrix guarantees C_i to be full rank. For each of the covariance matrix C_i, the design X is then drawn independently from $N(0, C_i)$ with $n = 50$.

The target variable Y is then computed as

$$Y = X\beta + e,$$

where β_1 to β_q with $q = 7$ are drawn independently from $N(0,1)$ and β_8 to β_{500} are set to zero to create a sparse model. e is the Gaussian noise vector with mean zero and variance 1. For each of the 5 cases with different C_i, both BLasso and FSF are run using stepsizes $\varepsilon = \frac{1}{5}$, $\frac{1}{10}$, $\frac{1}{20}$, $\frac{1}{40}$ and $\frac{1}{80}$. We also run Lasso which is listed as BLasso when $\varepsilon = 0$.

To compare the performances, we examine the solutions on the regularization paths that give the smallest mean squared error $\|X\beta - X\hat{\beta}\|^2$. The mean squared error (on log scale) of these solutions are tabulated together with the number of nonzero estimates in each solution. All cases are run 50 times and the average results are reported in Table 1.

As can be seen from Table 1, since our true model is sparse, in almost all cases the BLasso solutions are sparser and have similar prediction performances comparing to the FSF solutions with the same stepsize. It is also interesting to note that, smaller stepsizes require more computation but often give worse predictions and much less sparsity. We conjecture that there is also a regularization effect caused by the discretization of the solution paths (more discussion in Section 8) and this effect has also been observed by Gao et al. (2006) in a language ranking problem.

Design			$\varepsilon = \frac{1}{5}$	$\varepsilon = \frac{1}{10}$	$\varepsilon = \frac{1}{20}$	$\varepsilon = \frac{1}{40}$	$\varepsilon = \frac{1}{80}$	Lasso ($\varepsilon = 0$)
C_1	MSE	BLasso	18.60	18.27	18.33	18.60	19.42	19.98
		FSF	19.77	19.40	19.60	19.82	19.96	
	$\hat{q}$	BLasso	15.38	20.08	21.76	21.44	20.50	21.86
		FSF	18.32	24.00	27.28	30.48	32.14	
C_2	MSE	BLasso	19.58	19.28	19.65	19.94	20.76	21.12
		FSF	20.67	20.29	20.63	20.94	21.11	
	$\hat{q}$	BLasso	14.80	18.92	20.18	21.22	20.52	21.82
		FSF	18.34	21.90	25.70	28.80	29.38	
C_3	MSE	BLasso	18.83	18.14	18.55	18.90	19.32	20.15
		FSF	19.35	19.11	19.52	19.78	19.93	
	$\hat{q}$	BLasso	15.22	19.10	19.92	20.02	19.52	21.08
		FSF	15.38	19.72	23.30	25.88	27.30	
C_4	MSE	BLasso	20.09	19.88	19.85	20.20	21.84	21.70
		FSF	21.53	21.09	21.13	21.35	21.57	
	$\hat{q}$	BLasso	15.76	20.82	22.20	22.42	21.12	22.24
		FSF	18.90	24.64	30.38	32.02	34.16	
C_5	MSE	BLasso	18.79	18.62	18.70	19.09	19.47	20.12
		FSF	19.99	19.92	19.84	20.19	20.36	
	$\hat{q}$	BLasso	15.58	19.16	21.26	21.92	22.18	22.76
		FSF	17.10	23.24	28.24	30.94	32.84	

Table 1: Comparison of FSF and BLasso in a simulated nonparametric regression setting. The log of MSE and $\hat{q}$ =# of nonzeros are reported for the oracle solutions on the regularization paths. All results are averaged over 50 runs.

Design			$\varepsilon = \frac{1}{5}$	$\varepsilon = \frac{1}{10}$	$\varepsilon = \frac{1}{20}$	$\varepsilon = \frac{1}{40}$	$\varepsilon = \frac{1}{80}$
C_1	MSE	BLasso−Lasso	-1.38 (0.37)	-1.71 (0.23)	-1.65 (0.21)	-1.38 (0.21)	-0.56 (0.35)
		BLasso−FSF	-1.17 (0.27)	-1.13 (0.28)	-1.27 (0.26)	-1.22 (0.26)	-0.54 (0.24)
	$\hat{q}$	BLasso−Lasso	-6.48 (0.64)	-1.78 (0.70)	-0.10 (0.67)	-0.42 (0.63)	-1.36 (0.65)
		BLasso−FSF	-2.94 (0.89)	-3.92 (1.22)	-5.52 (1.26)	-9.04 (1.43)	-11.64 (1.64)
C_2	MSE	BLasso−Lasso	-1.54 (0.37)	-1.84 (0.29)	-1.47 (0.26)	-1.18 (0.25)	-0.36 (0.45)
		BLasso−FSF	-1.09 (0.32)	-1.01 (0.27)	-0.98 (0.23)	-1.00 (0.23)	-0.35 (0.38)
	$\hat{q}$	BLasso−Lasso	-7.02 (0.58)	-2.90 (0.65)	-1.64 (0.52)	-0.60 (0.50)	-1.30 (0.48)
		BLasso−FSF	-3.54 (0.99)	-2.98 (0.88)	-5.52 (1.09)	-7.58 (1.31)	-8.86 (1.41)
C_3	MSE	BLasso−Lasso	-1.32 (0.35)	-2.01 (0.36)	-1.60 (0.33)	-1.25 (0.32)	-0.83 (0.32)
		BLasso−FSF	-0.53 (0.28)	-0.97 (0.22)	-0.97 (0.23)	-0.88 (0.23)	-0.62 (0.24)
	$\hat{q}$	BLasso−Lasso	-5.86 (0.81)	-1.98 (0.72)	-1.16 (0.54)	-1.06 (0.55)	-1.56 (0.56)
		BLasso−FSF	-0.16 (0.78)	-0.62 (0.87)	-3.38 (1.05)	-5.86 (0.97)	-7.78 (1.08)
C_4	MSE	BLasso−Lasso	-1.61 (0.45)	-1.82 (0.33)	-1.85 (0.33)	-1.50 (0.33)	0.14 (0.66)
		BLasso−FSF	-1.44 (0.30)	-1.20 (0.28)	-1.28 (0.24)	-1.15 (0.29)	0.27 (0.67)
	$\hat{q}$	BLasso−Lasso	-6.48 (0.71)	-1.42 (0.85)	-0.04 (0.73)	0.18 (0.52)	-1.12 (0.67)
		BLasso−FSF	-3.14 (0.92)	-3.82 (1.16)	-8.18 (1.12)	-9.60 (1.35)	-13.04 (1.68)
C_5	MSE	BLasso−Lasso	-1.33 (0.38)	-1.50 (0.26)	-1.41 (0.26)	-1.03 (0.22)	-0.65 (0.22)
		BLasso−FSF	-1.20 (0.25)	-1.30 (0.23)	-1.14 (0.28)	-1.10 (0.29)	-0.89 (0.28)
	$\hat{q}$	BLasso−Lasso	-7.18 (0.84)	-3.60 (0.64)	-1.50 (0.58)	-0.84 (0.52)	-0.58 (0.55)
		BLasso−FSF	-1.52 (0.88)	-4.08 (1.10)	-6.98 (1.08)	-9.02 (1.21)	-10.66 (1.50)

Table 2: Means and Standard Errors of the differences of MSE and $\hat{q}$ between BLasso and Lasso, and between Blasso and FSF in Table 1.

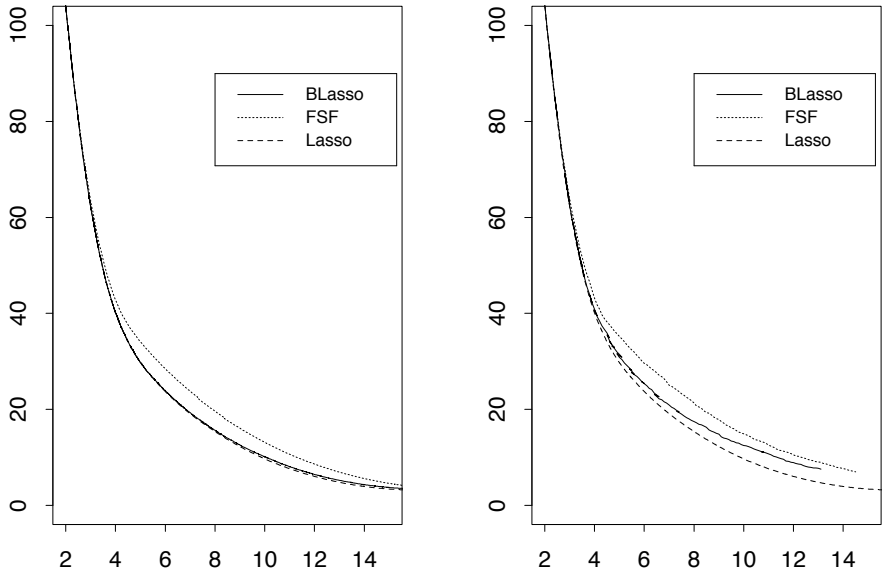


Figure 2: Plots of in-sample Mean Squared Error (y-axis) versus $\|\beta\|_1$ (x-axis) for a typical realization of the experiment (on run under C_2 from Table 1). The step size is set to $\varepsilon = \frac{1}{80}$ in the left plot and $\varepsilon = \frac{1}{5}$ in the right.

Table 2 gives a further analysis of the results in Table 1. It contains means and standard errors of the differences of MSE and $\hat{q}$, between BLasso and Lasso and between BLasso and FSF, for the stepsizes given in Table 1. First of all, all the mean differences are negative and when compared with their SE's, the differences are also significant except for few cells for small stepsizes 1/40 and 1/80 (in the last two columns). This overwhelming pattern of significant negative difference suggests that, for this simulation, BLasso is better than Lasso and FSF in terms of both prediction and sparsity unless the stepsize is very small as in the last two columns. Moreover, for MSE the stepsize $\varepsilon = 1/10$ seems to bring the best improvement of BLasso over Lasso, and the improvement is pretty robust against the choice of stepsize. On the other hand, the improvements of BLasso over FSF on MSE are less then those of BLasso over Lasso because FSF has the same discrete stepsizes. Hence these improvements reflect the gains only by the backward steps since FSF takes also forward steps. In terms of $\hat{q}$, the number of covariates selected, as expected, the larger the stepsize, the sparser the BLasso model is relative to the Lasso model or the FSF model. The sparsity improvements over Lasso are significant for all cells except for the last column with $\varepsilon = 1/80$. When compared with FSF, the sparsity improvements are less and smaller (still significant). In terms of gains on both MSE and sparsity and relative to both Lasso and FSF, stepsizes 1/10 and 1/20, that is, 0.1 or 0.05, seem good overall choices for this simulation study.

As suggested by one referee, we compare the Lasso empirical loss functions induced by BLasso, FSF and Lasso (through LARS). Figure 2 shows plots of in-sample Mean Squared Error versus L_1 norms of the coefficients taken from one typical run of the simulation conducted in this section. As shown by the plots, the in-sample MSE from BLasso approximates the in-sample MSE from the Lasso better than the FSF under both big and small step sizes. In particular, when the step size is small, the BLasso path is almost indiscernible from the Lasso path. A final comment on Figure 2 is in order. Although the in-sample MSE curve for BLasso in the right panel of Figure 2 does seem to go up at the end of the plot, we can not extend the x-axis further to higher $||\beta||_1$ values because at the stepsize $\varepsilon = 1/5$, the BLasso solution has achieved its L_1 norm maximum around $14-15$ – the maximum of the x-axis on the right panel of Figure 2.

6.3 Generalized BLasso for Other Penalties and Nondifferentiable Loss Functions

First, to demonstrate Generalized BLasso for different penalties, we use the Bridge Regression setting with the diabetes data set (without the added covariate in the first experiment). The Bridge Regression (first proposed by Frank and Friedman 1993 and later more carefully discussed and implemented by Fu 2001) is a generalization of the ridge regression (L_2 penalty) and Lasso (L_1 penalty). It considers a linear (L_2) regression problem with L_γ penalty for $\gamma \geq 1$ (to maintain the convexity of the penalty function). The penalized loss function has the form:

$$\Gamma(\beta;\lambda) = \sum_{i=1}^{n}(Y_i - X_i\beta)^2 + \lambda\|\beta\|_\gamma,$$

where γ is the bridge parameter. The data used in this experiment are centered and rescaled as in the first experiment.

Generalized BLasso successfully produced the paths for all 5 cases which are verified by pointwise minimization using simplex method ($\gamma = 1, \gamma = 1.1, \gamma = 4$ and $\gamma = max$) or close form solutions ($\gamma = 2$). It is interesting to notice the phase transition from the near-Lasso to the Lasso as the solution paths are similar but only Lasso has sparsity. Also, as γ grows larger, estimates for different β_j tend to have more similar sizes and in the extreme $\gamma = \infty$ there is a "branching" phenomenon— the estimates stay together in the beginning and branch out into different directions as the path progresses.

To demonstrate the Generalized BLasso algorithm for classification using an nondifferentiable loss function with a L_1 penalty function, we look at binary classification with the hinge loss. As in Zhu et al. (2003), we generate n=50 training data points in each of two classes. The first class has two standard normal independent inputs X^1 and X^2 and class label $Y = -1$. The second class also has two standard normal independent inputs, but conditioned on $4.5 \leq (X^1)^2 + (X^2)^2 \leq 8$ and has class label $Y = 1$. We wish to find a classification rule from the training data. so that when given a new input, we can assign a label from $\{1, -1\}$ to it.

1-norm SVM (Zhu et al., 2003) is used to estimate β:

$$(\hat{\beta}_0, \beta) = \arg\min_{\beta_0,\beta} \sum_{i=1}^{n}(1 - Y_i(\beta_0 + \sum_{j=1}^{m}\beta_j h_j(X_i)))^+ + \lambda\sum_{j=1}^{5}|\beta_j|,$$

where $h_i \in D$ are basis functions and λ is the regularization parameter. The dictionary of basis functions is $D = \{\sqrt{2}X^1, \sqrt{2}X^2, \sqrt{2}X^1X^2, (X^1)^2, (X^2)^2\}$. Notice that β_0 is left unregularized so the penalty function is not the L_1 penalty.

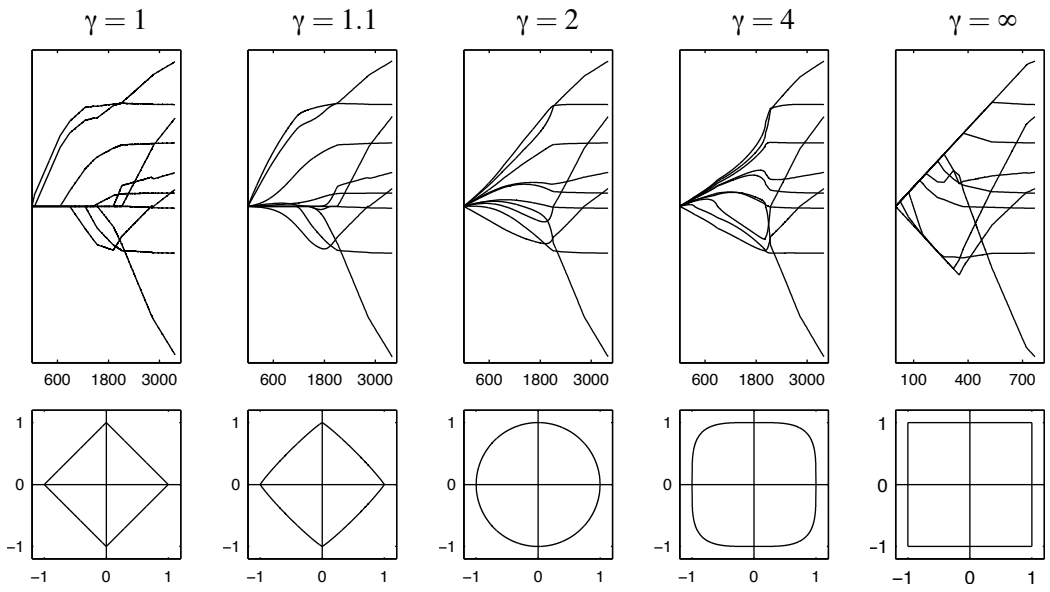


Figure 3: *Upper Panel*: Solution paths produced by BLasso for different bridge parameters, on the diabetes data set. From left to right: Lasso ($\gamma = 1$), near-Lasso ($\gamma = 1.1$), Ridge ($\gamma = 2$), over-Ridge ($\gamma = 4$), max ($\gamma = \infty$). The Y-axis is the parameter estimate and has the range $[-800, 800]$. The X-axis for each of the left 4 plots is $\sum_i |\beta_i|$, the one for the 5th plot is $\max(|\beta_i|)$ because $\sum_i |\beta_i|$ is unsuitable. *Lower Panel*: The corresponding penalty equal contours for $|\beta_1|^\gamma + |\beta_2|^\gamma = 1$.

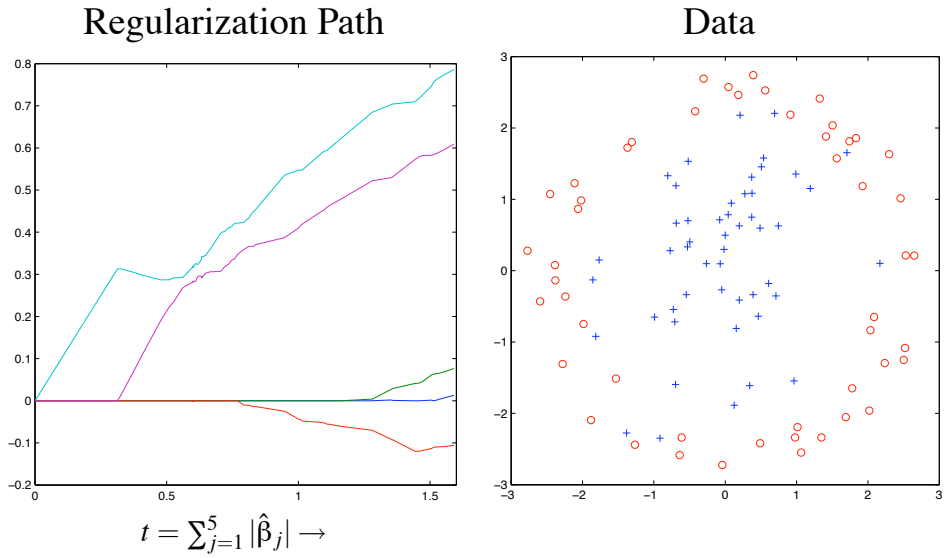


Figure 4: Estimates of 1-norm SVM coefficients $\hat{\beta}_j$, j=1,2,...,5, for the simulated two-class classification data. *Left Panel:* BLasso solutions as a function of $t = \sum_{j=1}^{5} |\hat{\beta}_j|$. *Right Panel:* Scatter plot of the data points with labels: '+' for $y = -1$; 'o' for $y = 1$.

The fitted model is

$$\hat{f}(x) = \hat{\beta}_0 + \sum_{j=1}^{m} \hat{\beta}_j h_j(x),$$

and the classification rule is given by $\text{sign}(\hat{f}(x))$.

Since the loss function is not differentiable, we do not have a theoretical guarantee that BLasso works. Nonetheless the solution path produced by Generalized BLasso has the same sparsity and piecewise linearity as the 1-norm SVM solutions shown in Zhu et al. (2003). It takes Generalized BLasso 490 iterations to generate the solutions. The covariates enter the regression equation sequentially as t increase, in the following order: the two quadratic terms first, followed by the interaction term then the two linear terms. As 1-norm SVM in Zhu et al. (2003), BLasso correctly picked up the quadratic terms early. That come up much later are the interaction term and linear terms that are not in the true model. In other words, BLasso results are in good agreement with Zhu et al.'s 1-norm SVM results and we regard this as a confirmation for BLasso's effectiveness in this nondifferentiable example.

7. Discussion and Concluding Remarks

As seen from our simulations under sparse true models, BLasso generates sparser solutions with similar or slightly better predictions relative to Lasso and FSF. The behavior relative to Lasso is due to the discrete stepsize of BLasso, while the behavior relative to FSF is partially explained by its convergence to the Lasso path as the stepsize goes to 0. We believe that the generalized version

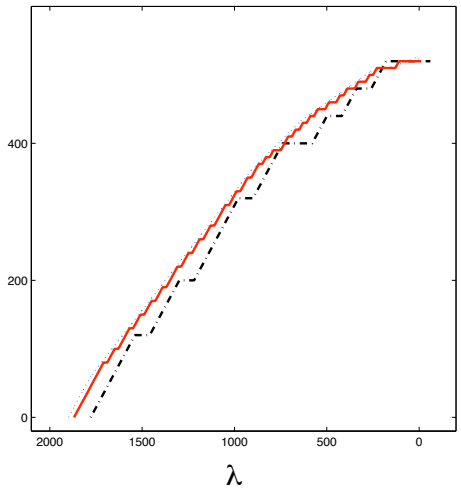


Figure 5: Estimates of regression coefficients $\hat{\beta}_3$ for the diabetes data set. Solutions are plotted as functions of λ. *Dotted Line:* Estimates using stepsize $\varepsilon = 0.05$. *Solid Line:* Estimates using stepsize $\varepsilon = 10$. *Dash-dot Line:* Estimates using stepsize $\varepsilon = 50$.

is also effective as an off-the-shelf algorithm for the general convex penalized loss minimization problems.

Computationally, BLasso takes roughly $O(1/\varepsilon)$ steps to produce the whole path. Depending on the actual loss function, base learners and minimization method used in each step, the actual computation complexity varies. As shown in the simulations, choosing a smaller stepsize gives a smoother solution path but it does not guarantee a better prediction. Actually, for the particular simulation set-up in Sec. 6.2, moderate stepsizes gave better results both in terms of MSE and sparsity. It is worth noting that the BLasso coefficient estimates are pretty close to the Lasso solutions even for relatively large stepsizes.

For the diabetes data, using a moderate stepsize $\varepsilon = 0.05$, the solution path can not be distinguished from the exact regularization path. Moreover, even when the stepsize is as large as $\varepsilon = 10$ and $\varepsilon = 50$, the solutions are still good approximations.

BLasso has only one stepsize parameter (with the exception of the numerical tolerance ξ which is implementation specific but not necessarily a user parameter). This parameter controls both how close BLasso approximates the minimization coefficients for each λ and how close two adjacent λ on the regularization path are placed. As can be seen from Figure 5, a smaller stepsize leads to a closer approximation to the solutions and also finer grids for λ. We argue that, if λ is sampled on a coarse grid we should not spend computational power on finding a much more accurate approximation of the coefficients for each λ. Instead, the available computational power spent on these two coupled tasks should be balanced. BLasso's 1-parameter setup automatically balances these two aspects of the approximation which is graphically expressed by the staircase shape of the solution paths.

Another algorithm similar to Generalized BLasso was developed independently by Rosset (2004). There, starting from $\lambda = 0$, a solution is generated by taking a small Newton-Raphson step for each λ, then λ is increased by a fixed amount. The algorithm assumes twice-differentiability of both

loss function and penalty function and involves calculation of the Hessian matrix which could be heavy-duty computationally when the number p of covariates is not small. In comparison, BLasso uses only the differences of the loss function and involves only basic operations and does not require advanced mathematical knowledge of the loss function or penalty. It can also be used a simple plug-in method for dealing with other convex penalties. Hence BLasso is easy to program and allows testing of different loss and penalty functions. Admittedly, this ease of implementation can cost computation time in large p situations.

BLasso's stepsize is defined in the original parameter space which makes the solutions evenly spread in β's space rather than in λ. In general, since λ is approximately the reciprocal of size of the penalty, as a fitted model grows larger and λ becomes smaller, changing λ by a fixed amount makes the algorithm in Rosset (2004) move too fast in the β space. On the other hand, when the model is close to empty and the penalty function is very small, λ is very large, but the algorithm still uses the same small steps thus computation is spent to generate solutions that are too close to each other.

As we discussed for the least squares problem, BLasso may also be computationally attractive for dealing with nonparametric learning problems with a large or an infinite number of base learners. This is mainly due to two facts. First, the forward step, as in Boosting, is a sub-optimization problem by itself and Boosting's functional gradient descend strategy applies. For example, in the case of classification with trees, one can use the classification margin or the logistic loss function as the loss function and use a reweighting procedure to find the appropriate tree at each step (for details see, e.g., Breiman, 1998; Friedman et al., 2000). In the case of regression with the L^2 loss function, the minimization as in (6) is equivalent to refitting the residuals as we described in the last section. The second fact is that, when using an iterative procedure like BLasso, we usually stop early to avoid overfitting and to get a sparse model. And even if the algorithm is kept running, it usually reaches a close-to-perfect fit without too many iterations. Therefore, the backward step's computation complexity is limited because it only involves base learners that are already included from previous steps.

There is, however, a difference in the BLasso algorithm between the case with a small number of base learners and that with a large or an infinite number of base learners. For the finite case, BLasso avoids oscillation by requiring a backward step to be strictly descending and relax λ whenever no descending step is available. Hence BLasso never reaches the same solution more than once and the tolerance constant ξ can be set to 0 or a very small number to accommodate the program's numerical accuracy. In the nonparametric learning case, a different kind of oscillation can occur when BLasso keeps going back and force in different directions but only improving the penalized loss function by a diminishing amount, therefore a positive tolerance ξ is mandatory. As suggested by the proof of Theorem 1, we suggest choosing $\xi = o(\varepsilon)$ to warrant a good approximation to the Lasso path.

One direction for future research is to apply BLasso in an online or time series setting. Since BLasso has both forward and backward steps, we believe that an adaptive online learning algorithm can be devised based BLasso so that it goes back and forth to track the best regularization parameter and the corresponding model.

We end with a summary of our main contributions:

1. By combining both forward and backward steps, the BLasso algorithm is constructed to minimize an L_1 penalized convex loss function. While it maintains the simplicity and flexibility of e-Boosting (or Forward Stagewise Fitting), BLasso efficiently approximate the Lasso so-

lutions for general loss functions and large classes of base learners. This can be proven rigorously for a finite number of base learners under some assumptions.

2. The backward steps introduced in this paper are critical for producing the Lasso path. Without them, the FSF algorithm in general does not produce Lasso solutions, especially when the base learners are strongly correlated as in cases where the number of base learners is larger than the number of observations. As a result, FSF loses some of the sparsity provided by Lasso and might also suffer in prediction performance as suggested by our simulations.

3. We generalized BLasso as a simple, easy-to-implement, plug-in method for approximating the regularization path for other convex penalties.

4. Discussions based on intuition and simulation results are made on the regularization effect of using stepsizes that are not very small.

Last but not least, matlab codes by Guilherme V. Rocha for BLasso in the case of L_2 loss and L_1 penalty can be downloaded at
http://www.stat.berkeley.edu/twiki/Research/YuGroup/Software.

Acknowledgments

Yu would like to gratefully acknowledge the partial supports from NSF grants FD01-12731 and CCR-0106656 and ARO grant DAAD19-01-1-0643, and the Miller Research Professorship in Spring 2004 from the Miller Institute at University of California at Berkeley. We thank Dr. Chris Holmes and Mr. Guilherme V. Rocha for their very helpful comments and discussions on the paper. Finally, we would like to thank three referees and the action editor for their thoughtful and detailed comments on an earlier version of the paper.

Appendix A. Proofs

Proof (Lemma 1)

1. It is assumed that there exist λ and j with $|s| = \varepsilon$ such that

$$\Gamma(s1_j;\lambda) \leq \Gamma(0;\lambda).$$

Then we have

$$\sum_{i=1}^{n} L(Z_i;0) - \sum_{i=1}^{n} L(Z_i;s1_j) \geq \lambda T(s1_j) - \lambda T(0).$$

Therefore

$$\begin{aligned}
\lambda &\leq \frac{1}{\varepsilon}\{\sum_{i=1}^{n} L(Z_i;0) - \sum_{i=1}^{n} L(Z_i;s1_j)\} \\
&\leq \frac{1}{\varepsilon}\{\sum_{i=1}^{n} L(Z_i;0) - \min_{j',|s|=\varepsilon} \sum_{i=1}^{n} L(Z_i;s1_{j'})\} \\
&= \frac{1}{\varepsilon}\{\sum_{i=1}^{n} L(Z_i;0) - \sum_{i=1}^{n} L(Z_i;\hat{\beta}^0)\} \\
&= \lambda^0.
\end{aligned}$$

2. Since a backward step is only taken when $\Gamma(\hat{\beta}^{t+1};\lambda^t) < \Gamma(\hat{\beta}^t;\lambda^t) - \xi$ and $\lambda^{t+1} = \lambda^t$, so we only need to consider forward steps. When a forward step is forced, if $\Gamma(\hat{\beta}^{t+1};\lambda^{t+1}) > \Gamma(\hat{\beta}^t;\lambda^{t+1}) - \xi$, then

$$\sum_{i=1}^{n} L(Z_i;\hat{\beta}^t) - \sum_{i=1}^{n} L(Z_i;\hat{\beta}^{t+1}) - \xi < \lambda^{t+1} T(\hat{\beta}^{t+1}) - \lambda^{t+1} T(\hat{\beta}^t).$$

Hence

$$\frac{1}{\varepsilon}\{\sum_{i=1}^{n} L(Z_i;\hat{\beta}^t) - \sum_{i=1}^{n} L(Z_i;\hat{\beta}^{t+1}) - \xi\} < \lambda^{t+1},$$

which contradicts the algorithm.

3. Since $\lambda^{t+1} < \lambda^t$ and λ can not be relaxed by a backward step, we immediately have $\|\hat{\beta}^{t+1}\|_1 = \|\hat{\beta}^t\|_1 + \varepsilon$. Then from

$$\lambda^{t+1} = \frac{1}{\varepsilon}\{\sum_{i=1}^{n} L(Z_i;\hat{\beta}^t) - \sum_{i=1}^{n} L(Z_i;\hat{\beta}^{t+1}) - \xi\},$$

we get

$$\Gamma(\hat{\beta}^t;\lambda^{t+1}) - \xi = \Gamma(\hat{\beta}^{t+1};\lambda^{t+1}).$$

Add $(\lambda^t - \lambda^{t+1})\|\hat{\beta}^t\|_1$ to both sides, and recall $T(\hat{\beta}^{t+1}) = \|\hat{\beta}^{t+1}\|_1 > |\hat{\beta}^t\|_1 = T(\hat{\beta}^t)$, we get

$$\begin{aligned} \Gamma(\hat{\beta}^t;\lambda^t) - \xi &< \Gamma(\hat{\beta}^{t+1};\lambda^t) \\ &= \min_{j',|s|=\varepsilon} \Gamma(\hat{\beta}^t + s1_{j'};\lambda^t) \\ &\leq \Gamma(\hat{\beta}^t \pm \varepsilon 1_j;\lambda^t) \end{aligned}$$

for all j.

■

Proof (Theorem 1)

Theorem 3.1 claims that "the BLasso path converges to the Lasso path uniformly" for $\sum L(Z;\beta)$ that is strongly convex with bounded second derivatives in β. The strong convexity and bounded second derivatives imply the Hessian w.r.t. β satisfies

$$mI \preceq \nabla^2 \sum L \preceq MI,$$

for positive constants $M \geq m > 0$. Using these notations, we will show that for any t s.t. $\lambda^{t+1} > \lambda^t$, we have

$$\|\hat{\beta}^t - \beta^*(\lambda^t)\|_2 \leq (\frac{M}{m}\varepsilon + \frac{\xi}{\varepsilon}\frac{2}{m})\sqrt{p}, \tag{9}$$

where $\beta^*(\lambda^t) \in R^p$ is the Lasso estimate with a regularization parameter λ^t.

The proof of (9) relies on the following inequalities for strongly convex functions, some of which can be found in Boyd and Vandenberghe (2004). First, because of the strong convexity, we have

$$\sum L(Z;\beta^*(\lambda^t)) \geq \sum L(Z;\hat{\beta}^t) + \nabla \sum L(Z;\hat{\beta}^t)^T(\beta^*(\lambda^t) - \hat{\beta}^t) + \frac{m}{2}\|\beta^*(\lambda^t) - \hat{\beta}^t\|_2^2.$$

The L_1 penalty function is also convex although not strictly convex nor differentiable at 0, but we have

$$\|\beta^*(\lambda^t)\|_1 \geq \|\hat{\beta}^t\|_1 + \delta^T(\beta^*(\lambda^t) - \hat{\beta}^t)$$

hold for any p-dimensional vector δ with δ_i the i'th entry of $\text{sign}(\hat{\beta}^t)^T$ for the nonzero entries and $|\delta_i| \leq 1$ otherwise.

Putting both inequalities together, we have

$$\Gamma(\beta^*(\lambda^t);\lambda^t) \geq \Gamma(\hat{\beta}^t;\lambda^t) + (\nabla \sum L(Z;\hat{\beta}^t) + \lambda_t \delta)^T(\beta^*(\lambda^t) - \hat{\beta}^t) + \frac{m}{2}\|\beta^*(\lambda^t) - \hat{\beta}^t\|_2^2. \tag{10}$$

Using Equation (10), we can bound the L^2 distance between $\beta^*(\lambda_t)$ and $\hat{\beta}^t$ by applying Cauchy-Schwartz to get

$$\Gamma(\beta^*(\lambda^t);\lambda^t) \geq \Gamma(\hat{\beta}^t;\lambda^t) - \|\nabla \sum L(Z;\hat{\beta}^t) + \lambda_t \delta\|_2 \|\beta^*(\lambda^t) - \hat{\beta}^t\|_2 + \frac{m}{2}\|\beta^*(\lambda^t) - \hat{\beta}^t\|_2^2.$$

Since $\Gamma(\beta^*(\lambda^t);\lambda^t) \leq \Gamma(\hat{\beta}^t;\lambda^t)$, we have

$$\|\beta^*(\lambda^t) - \hat{\beta}^t\|_2 \leq \frac{2}{m}\|\nabla \sum L(Z;\hat{\beta}^t) + \lambda_t \delta\|_2. \tag{11}$$

By statement (3) of Lemma 1, for $\hat{\beta}^t_j \neq 0$, we have

$$\sum L(Z;\hat{\beta}^t \pm \varepsilon \text{sign}(\hat{\beta}^t_j) 1_j) \pm \lambda_t \varepsilon \geq \sum L(Z;\hat{\beta}^t) - \xi. \tag{12}$$

At the same time, by the bounded Hessian assumption, we have

$$\sum L(Z;\hat{\beta}^t \pm \varepsilon \text{sign}(\hat{\beta}^t_j) 1_j) \leq \sum L(Z;\hat{\beta}^t) \pm \varepsilon \nabla \sum L(Z;\hat{\beta}^t)^T \text{sign}(\hat{\beta}^t_j) 1_j + \frac{M}{2}\varepsilon^2. \tag{13}$$

Connect these two inequalities, we have

$$\mp \varepsilon \times (\nabla \sum L(Z;\hat{\beta}^t)^T 1_j \text{sign}(\hat{\beta}^t_j) + \lambda_t) \leq \frac{M}{2}\varepsilon^2 + \xi,$$

therefore

$$|(\nabla \sum L(Z;\hat{\beta}^t)^T 1_j \text{sign}(\hat{\beta}^t_j) + \lambda_t)| \leq \frac{M}{2}\varepsilon + \frac{\xi}{\varepsilon}. \tag{14}$$

Similarly, for $\hat{\beta}^t_j = 0$, instead of (12), we have

$$\sum L(Z;\hat{\beta}^t \pm \varepsilon \text{sign}(\hat{\beta}^t_j) 1_j) + \lambda_t \varepsilon \geq \sum L(Z;\hat{\beta}^t) - \xi.$$

Combine with (13), we have

$$|\nabla \sum L(Z;\hat{\beta}^t)^T 1_j| - \lambda_t \leq \frac{M}{2}\varepsilon + \frac{\xi}{\varepsilon}.$$

For j such that $\hat{\beta}^t_j = 0$, we choose δ_j appropriately and combine with (14) so that the right hand side of (11) is controlled by $\sqrt{p} \times \frac{2}{m} \times (\frac{M}{2}\varepsilon + \frac{\xi}{\varepsilon})$. This way we obtain (9). ■

References

E.L. Allgower and K. Georg. Homotopy methods for approximating several solutions to nonlinear systems of equations. In W. Forster, editor, *Numerical solution of highly nonlinear problems*, pages 253–270. North-Holland, 1980.

S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

L. Breiman. Arcing classifiers. *The Annals of Statistics*, 26:801–824, 1998.

P. Buhlmann and B. Yu. Boosting with the l2 loss: regression and classification. *Journal of American Statistical Association*, 98, 2003.

E. Candes and T. Tao. The danzig selector: Statistical estimation when p is much larger than n. *Annals of Statistics (to appear)*, 2007.

S. Chen and D. Donoho. Basis pursuit. Technical report, Department of Statistics, Stanford University, 1994.

N. Cristianini and J. Shawe-Taylor. *An introduction to support vector machines and other kernel-based learning methods*. Cambridge University Press, 2002.

D. Donoho. For most large undetermined system of linear equatnions the minimal l_1-norm near-solution approximates the sparsest solution. *Communications on Pure and Applied Mathematics*, 59(6):797–829, 2006.

D. Donoho, M. Elad, and V. Temlyakov. Stable recovery of sparse overcomplete representations in the presence of noise. *IEEE Trans. Information Theory*, 52(1):6–18, 2006.

B. Efron, T. Hastie, and R. Tibshirani. Least angle regression. *Annals of Statistics*, 32:407–499, 2004.

J. Fan and R.Z. Li. Variable selection via nonconcave penalized likelihood and its oracle properties. *Journal of American Statistical Association*, 96(456):1348–1360, 2001.

I. Frank and J. Friedman. A statistical view od some chemometrics regression tools. *Technometrics*, 35:109–148, 1993.

Y. Freund. Boosting a weak learning algorithm by majority. *Information and Computation*, 121: 256–285, 1995.

Y. Freund and R.E. Schapire. Experiments with a new boosting algorithm. In *Machine Learning: Proc. Thirteenth International Conference*, pages 148–156. Morgan Kauffman, San Francisco, 1996.

J.H. Friedman. Greedy function approximation: a gradient boosting machine. *Annal of Statistics*, 29:1189–1232, 2001.

J.H. Friedman, T. Hastie, and R. Tibshirani. Additive logistic regression: a statistical view of boosting. *Annal of Statistics*, 28:337–407, 2000.

W.J. Fu. Penalized regression: The bridge versus the lasso. *Journal of Computational and Graphical Statistics*, 7(3):397–416, 2001.

J. Gao, H. Suzuki, and B. Yu. Approximate lasso methods for language modeling. *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the ACL, Sydney*, pages 225–232, 2006.

T. Gedeon, A. E. Parker, and A. G. Dimitrov. Information distortion and neural coding. *Canadian Applied Mathematics Quarterly*, 2002.

T. Hastie, Tibshirani, R., and J.H. Friedman. *The Elements of Statistical Learning: Data Mining, Inference and Prediction*. Springer Verlag, 2001.

T. Hastie, J. Taylor, R. Tibshirani, and G. Walther. Forward stagewise regression and the monotone lasso. Technical report, Department of Statistics, Stanford University, 2006.

K. Knight and W. J. Fu. Asymptotics for lasso-type estimators. *Annals of Statistics*, 28:1356–1378, 2000.

L. Mason, J. Baxter, P. Bartlett, and M. Frean. Functional gradient techniques for combining hypotheses. *Advance in Large Margin Classifiers*, 1999.

N. Meinshausen and P. Bühlmann. High-dimensional graphs and variable selection with the lasso. *Annals of Statistics*, 34:1436–1462, 2005.

N. Meinshausen and B. Yu. Lasso-type recovery of sparse representations for high-dimensional data. *Annals of Statistics (to appear)*, 2006.

M.R. Osborne, B. Presnell, and B.A. Turlach. A new approach to variable selection in least squares problems. *Journal of Numerical Analysis*, 20(3):389–403, 2000a.

M.R. Osborne, B. Presnell, and B.A. Turlach. On the lasso and its dual. *Journal of Computational and Graphical Statistics*, 9(2):319–337, 2000b.

S. Rosset. Tracking curved regularized optimization solution paths. *NIPS*, 2004.

S. Rosset, J. Zhu, and T. Hastie. Boosting as a regularized path to a maximum margin classifier. *Journal of Machine Learning Research*, 5:941–973, 2004.

R.E. Schapire. The strength of weak learnability. *Journal of Machine Learning*, 5(2):1997–2027, 1990.

B. Schölkopf and A. J. Smola. *Learning with kernels: support vector machines, regularization, optimization and beyond*. MIT Press, 2002.

R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society, Series B*, 58(1):267–288, 1996.

N. Tishby, F. C. Pereira, and W. Bialek. The information bottleneck method. In *The 37th annual Allerton Conference on Communication, Control and Computing*, 1999.

J.A. Tropp. Just relax: Convex programming methods for identifying sparse signals in noise. *IEEE Trans. Information Theory*, 52(3):1030 –1051, 2006.

V. N. Vapnik. *The Nature of Statistical Learning Theory*. Springer-Verlag, New York, 1995.

M. J. Wainwright. Sharp thresholds for noisy and high-dimensional recovery of sparsity using ℓ_1-constrained quadratic programming. *Technical Report 709, Statistics Department, UC Berkeley*, 2006.

C.-H. Zhang and J. Huang. The sparsity and bias of the lasso selection in high dimensional linear regression. *Annals of Statistics (to appear)*, 2006.

T. Zhang. Sequentiall greedy approximation for certain convex optimization problems. *IEEE Trans. on Information Theory*, 49(3):682–691, 2003.

P. Zhao and B. Yu. On model selection consistency of lasso. *Journal of Machine Learning Research*, 7 (Nov):2541–2563, 2006.

J. Zhu, S. Rosset, T. Hastie, and R. Tibshirani. 1-norm support vector machines. *Advances in Neural Information Processing Systems*, 16, 2003.

H. Zou. The adaptive lasso and its oracle properties. *Journal of American Statistical Association*, 101:1418–1429, 2006.

Journal of Machine Learning Research 8 (2007) 2727-2754 Submitted 1/07; Revised 7/07; Published 12/07

A New Probabilistic Approach in Rank Regression with Optimal Bayesian Partitioning

Carine Hue CARINE.HUE@GMAIL.COM
Marc Boullé MARC.BOULLE@ORANGE-FTGROUP.COM
France Telecom R&D
2, avenue Pierre Marzin
22307 Lannion cedex, France

Editors: Isabelle Guyon and Amir Saffari

Abstract

In this paper, we consider the supervised learning task which consists in predicting the normalized rank of a numerical variable. We introduce a novel probabilistic approach to estimate the posterior distribution of the target rank conditionally to the predictors. We turn this learning task into a model selection problem. For that, we define a 2D partitioning family obtained by discretizing numerical variables and grouping categorical ones and we derive an analytical criterion to select the partition with the highest posterior probability. We show how these partitions can be used to build univariate predictors and multivariate ones under a naive Bayes assumption.

We also propose a new evaluation criterion for probabilistic rank estimators. Based on the logarithmic score, we show that such criterion presents the advantage to be minored, which is not the case of the logarithmic score computed for probabilistic value estimator.

A first set of experimentations on synthetic data shows the good properties of the proposed criterion and of our partitioning approach. A second set of experimentations on real data shows competitive performance of the univariate and selective naive Bayes rank estimators projected on the value range compared to methods submitted to a recent challenge on probabilistic metric regression tasks.

Our approach is applicable for all regression problems with categorical or numerical predictors. It is particularly interesting for those with a high number of predictors as it automatically detects the variables which contain predictive information. It builds pertinent predictors of the normalized rank of the numerical target from one or several predictors. As the criteria selection is regularized by the presence of a prior and a posterior term, it does not suffer from overfitting.

Keywords: rank regression, probabilistic approach, 2D partitioning, non parametric estimation, Bayesian model selection

1. Introduction

In this introduction, we precise the supervised learning task we address in this paper, that is the rank regression. We then show the interest of probabilistic learning approaches compared to deterministic ones. Finally, we outline our contribution which aims at selecting a probabilistic predictive model for the rank of a numerical target with a nonparametric Bayesian approach.

1.1 Value, Ordinal and Rank Regression

In supervised learning, classification tasks, where the target variable is categorical, are usually distinguished from regression tasks, where it is numerical. A less known task is the case where the target variable is ordinal, usually called ordinal regression (see Chu and Ghahramani, 2005, for a state of the art). In this case, there is a total order between the target values but no distance information. The practical problems studied in the machine learning community consider a low number of distinct integer ranks, roughly 5 or 10, fixed before the learning. The aim is then to predict the right quintile or decile an example belongs to. The algorithms are generally evaluated with the mean zero-one error or with the mean absolute error obtained by considering the ordinal scales as consecutive integers. Among the proposed methods, the principle of empirical risk minimization with a loss function measuring the probability of misclassification is applied by Herbrich et al. (2000), an online algorithm based on the perceptron algorithm is proposed in the work of Crammer and Singer (2001), support vector machines are used by Shashua and Levin (2002) and Chu and Keerthi (2005) and Gaussian processes by Chu and Ghahramani (2005).

The choice of a low number of distinct ranks is generally motivated by simplicity reasons. However, this predefined target discretization can separate values which form a pertinent prediction interval.

In this paper, we address rank regression tasks. More precisely, for a given numerical target variable, we aim at computing an estimator of its rank. During the estimation procedure we never take into account the distance between instances but only their order. Several reasons have guided that choice. First, considering ranks rather than values is a classical way to obtain models more robust to outliers and to heteroscedasticity. In linear regression for example, an estimator based on the centered ranks in the minimization of the least squared equation is proposed in the approach of Hettmansperger and McKean (1998). Secondly in some applications, predicting the rank of a target variable is more interesting than predicting its intrinsic value. For instance in information retrieval, some search engines use numerical scores to rank web pages but the score value has no other usefulness.

1.2 Deterministic and Probabilistic Regression

Whatever the learning task, the simpler approach is determinist in so far as its outputs is deterministic: the majority class in classification, the mean rank in ordinal regression and the conditional mean in metric regression. These punctual predictors turn out to be inefficient as soon as confidence intervals or prediction of extreme values are needed. In this context, quantile regression or density estimation aims at estimating the predictive density more accurately. Such a probabilistic approach is very useful as soon as the predictive model is used for decision-making. For instance, modelling predictive uncertainty is still an active research domain and has been the subject of two recent challenges: the evaluating predictive uncertainty challenge supported by the PASCAL network of excellence in 2004-2005 and the predictive uncertainty in environmental modelling competition (organized by Cawley et al., 2006).

Quantile regression consists in estimating some quantiles of the predictive law. For a real α in $[0,1]$, the conditional quantile $q_\alpha(x)$ is defined as the lowest real value such that the conditional cumulative distribution is higher than α. Quantile regression can be formulated as a minimization problem. Starting from that, different methods have been proposed according to the form assumed for the quantile function: the minimization problem is solved with splines in the approach

of Koenker (2005), with kernel functions in the work of Takeuchi et al. (2006) and neural networks in the work of White (1991). The approach proposed by Chaudhuri et al. (1994); Chaudhuri and Loh (2002) mixes a tree partitioning of the predictors space and a local polynomial assumption. Random forests have been extended to conditional quantile estimation in the work of Meinshausen (2006). For all these approaches, the reals α are known in advance and usually in a small number, and the estimation of each quantile is done and evaluated independently from the others.

Conditional density estimation aims at giving for any couple (x,y) an estimator of the predictive density $p(y|x)$. The parametric approaches assume that the predictive density belongs to a fixed parametric family and then reduce the density estimation problem to the estimation of a parameter vector. The non parametric approaches do not assume any fixed parametric form for the predictive density and are generally based on two ingredients: first, the estimator is computed on each point by using data contained in a point neighbourhood; then, an assumption is done on the local form of the estimator. Very popular, kernel methods weight the contribution of the data by convoluting the empirical law with a kernel density. The form and the width of the kernel remain tuning parameters. Once a neighbourhood is defined, methods differ on the estimator form: the local polynomial approach includes constant, linear and polynomial estimator (see Fan et al., 1996). Splines can also be used. Such a probabilistic approach has already been proposed in ordinal regression in the parametric context of the Gaussian processes with a Bayesian approach (see Chu and Keerthi, 2005).

1.3 Our Contribution

In this paper, we consider regression tasks, where the unknown target variable is numerical. Instead of looking for an estimator of the *value* of the target variable, we aim at building an estimator of the normalized *rank* (between 0 and 1) of this variable. Our second objective is to propose an evaluation criterion for these rank probabilistic estimators.

First, we propose a non parametric Bayesian method to build a probabilistic estimator specified by a set of quantiles of the rank cumulative distribution function. Unlike quantile regression approach, the choice of the quantiles is not made before the learning but is determined during this step. Moreover, in the case of several predictor variables, the multivariate estimator is obtained as a combination of univariate estimators under a naive Bayes assumption. Each univariate estimator is obtained from a 2D partition of the space (predictor, target) assuming that the rank density is constant on each cell of the partition. The optimal 2D partition is searched according to a model selection approach.

Secondly, we propose and evaluate a new criterion to evaluate probabilistic regression methods by comparing the rank predictive density and the true insertion rank of all test instances.

This paper is organized as follows: in the second section, we first describe the 2D-partitioning for numerical predictors. Compared to our previous work introduced in a conference paper (see Boullé and Hue, 2006), the approach is much more detailed, the selection criteria is completely explicited and we propose an estimator of the rank predictive density. We also propose the 2D-partitioning for categorical predictors that has never been published before.

In the third section we first expose how we can build a univariate estimator of the rank predictive density from each 2D partition. Using the naive Bayes assumption of conditional independence of the predictors, we then describe how to obtain multivariate predictors, with and without variable selection.

The fourth section focuses on the important topic of the evaluation of such rank regression models. We first give a brief overview of classical scores used in supervised learning. We then propose to use one of them, the logarithmic score, for rank probabilistic estimators without the shortcomings noted for probabilistic estimators based on values.

The last section is devoted to experimental evaluation. We begin with experiments on synthetic data to demonstrate on the one hand the relevance of the proposed evaluation criterion and on the other hand the performance of the 2D partitionings. We pursue with experimentations on real data sets to show the performance of the univariate and multivariate predictors. We end by a comparison of our approach with alternative methods proposed in a recent challenge dedicated to probabilistic metric regression.

2. A 2D-Partitioning Method for Probabilistic Regression

The 2D-partitioning method we present here comes from the extension of the so-called MODL approach to regression tasks. This approach has first been proposed for classification tasks in the work of Boullé (2005) and Boullé (2006).

For regression tasks, we present in the sequel two 2D partitioning methods depending on whether the considered predictor is numerical or categorical.

For numerical predictors, the 2D partition is the grid resulting from the discretization of both target and predictor. For categorical predictors, the 2D partition is the grid resulting from the discretization of the numerical target and from the grouping of the categorical predictor.

2.1 The 2D Discretization for Probabilistic Regression with Numerical Predictor

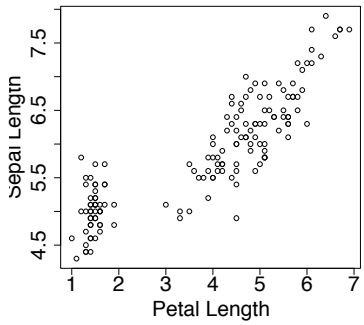


Figure 1: Scatter-plot of the iris data set of Fisher (1936) considered for a *regression* problem with the petal length variable as predictor and the sepal length variable as target.

In order to illustrate the regression problem with numerical predictor, we present in Figure 1 the scatter-plot of the iris data set considered for a *regression* problem with the petal length variable as predictor and the sepal length variable as target. The figure shows that iris plants with petal length below 2 cm always have a sepal length below 6 cm. We propose to exhibit the predictive information of the petal length variable by discretizing both the predictor and the target variables. For instance, the grid with six cells presented on the left of Figure 2 indicates that:

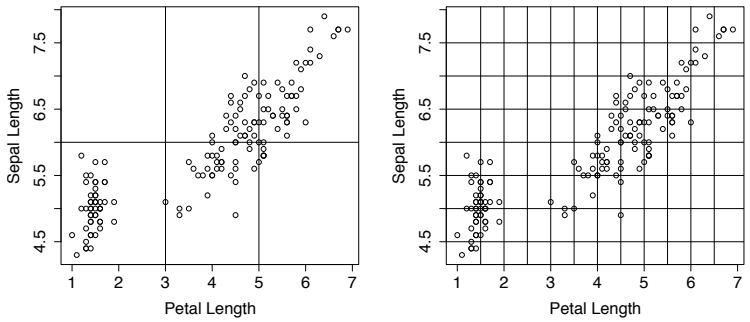


Figure 2: Two discretization grids with 6 or 96 cells, describing the correlation between the petal length and sepal length variables of the Iris data set.

- for a petal length lower than 3 cm, 100% of the training instances have a petal length higher than 6 cm;

- for a petal length between 3 and 5 cm, the two sepal length intervals are equiprobable (53% and 47%);

- for a petal length higher than 5 cm, 90% of the instances have a sepal length higher than 6 cm.

The 96 cells grid presented on the right of Figure 2 seems more accurate but may be less robust. These two examples illustrate that a compromise has to be found between the quality of the correlation information and the generalization ability, on the basis of the grain level of the discretization grid.

The issue is to describe the predictive distribution of the rank of the target value given the rank of the predictor value.

Let us now formalize this approach using a Bayesian model selection approach.

Definition 1 *A regression 2D discretization model is defined by:*

1. *a number of intervals for the target and predictor variables;*
2. *a partition of the predictor variable specified on the ranks of the predictor values;*
3. *for each predictor interval, the repartition of the instances among the target intervals specified by the instance counts locally to each predictor interval.*

Notations

N : the number of training instances

I : the number of predictor intervals

J : the number of target intervals

$N_{i.}$: the number of instances in the predictor interval i

$N_{.j}$: the number of instances in the target interval j

N_{ij} : the number of instances in the grid cell associated to predictor interval i and the target interval j.

A regression 2D discretization model is then entirely characterized by the parameters $\left\{I, J, \{N_{i.}\}_{1\leq i\leq I}, \left\{N_{ij}\right\}_{1\leq i\leq I, 1\leq j\leq J}\right\}$. The number of instances $N_{.j}$ can be deduced by adding the N_{ij} for each predictor interval, according to $N_{.j} = \sum_{i=1}^{I} N_{ij}$.

We now want to select the best model M given the available data, that is the most likely model given the data. Adopting a Bayesian approach, it comes to maximize :

$$p(M|D) = \frac{p(M)p(D|M)}{p(D)}.$$

The data distribution $p(D)$ being constant whatever the model M, it comes to maximize $p(M)p(D|M)$ which can be written:

$$p(M)p(D|M) = p(I,J)p(\{N_{i.}\}|I,J)p(\{N_{ij}\}|I,J,\{N_{i.}\})p(D|M).$$

We then add the restriction that the searched model is such that the conditional target distributions are independent. This assumption is first consistent with the objective to obtain a partition that discriminates distinct conditional distributions. Moreover, mathematically speaking, it enables to write the last factors in the precedent equation as products over the predictor intervals. It also reduces the complexity of the associated optimization algorithm.

Denoting by D_i the subset of D restricted to the interval i, one obtains:

$$p(M)p(D|M) = p(I,J)p(\{N_{i.}\}|I,J)\prod_{i=1}^{I} p(\{N_{ij}\}|I,J,\{N_{i.}\})\prod_{i=1}^{I} p(D_i|M).$$

To be able to evaluate a given model, we have to choose a prior distribution for the model parameters and a likelihood function. In Definition 2, we formalize our choices by using the assumption independence and proposing a uniform distribution at each stage of the prior parameter structure and of the likelihood function.

Definition 2 *The prior for the parameters of a regression 2D discretization model and the likelihood function of the data given a model are chosen hierarchically and uniformly at each level:*

1. *the numbers of intervals I and J are independent from each other, and uniformly distributed between* 1 *and N,*
2. *for a given number of predictor intervals I, every set of intervals is equiprobable,*
3. *for a given predictor interval, every distribution of the instances on the target intervals is equiprobable,*
4. *the distributions of the target intervals on each predictor interval are independent from each other,*
5. *for a given target interval, every distribution of the rank of the target values is equiprobable.*

Taking the negative log of the probabilities, this provides the evaluation criterion given in the following theorem:

Theorem 3 *A 2D-discretization model distributed according to a uniform hierarchical prior is Bayes optimal if its evaluation according to the following the criteria is minimal*

$$\begin{aligned} c(M) &= -\log(p(M)) - \log(p(D/M)) \\ &= 2\log N + \log\binom{N+I-1}{I-1} + \sum_{i=1}^{I} \log\binom{N_{i.}+J-1}{J-1} \\ &+ \sum_{i=1}^{I} \log \frac{N_{i.}!}{N_{i1}!N_{i2}!\ldots N_{iJ}!} + \sum_{j=1}^{J} \log N_{.j}!. \end{aligned} \tag{1}$$

The first hypothesis introduced in Definition 2 gives that $p(I,J) = p(I)p(J) = \frac{1}{N}\frac{1}{N}$.

The second hypothesis is that all the divisions into I intervals are equiprobable for a given I. Computing the probability of one set of intervals turns into the combinatorial evaluation of the number of possible interval sets. Dividing the predictor values into I intervals is equivalent to decompose the number N as the sum of the $N_{i.}$ frequencies of the intervals. Using combinatorics, we can prove that this number of choices is equal to $\binom{N+I-1}{I-1}$. Using the equiprobability assumption, one finally obtains:

$$P(\{N_{i.}\}|I) = \frac{1}{\binom{N+I-1}{I-1}}.$$

The third hypothesis assumes that, for a given interval i of size $N_{i.}$, every distribution of the instances on the J target intervals are equiprobable. It remains to specify the parameters of a multinomial distribution of $N_{i.}$ instance over J values. Using combinatorics again, one obtains

$$P\big(\{N_{ij}\}|I,\{N_{i.}\}\big) = \frac{1}{\binom{N_{i.}+J-1}{J-1}}.$$

The prior terms being explicited, it remains to evaluate the likelihood on each predictor interval, that is the probability to observe the data restricted to each interval knowing the multinomial distribution model on each interval. The number of ways to observe $N_{i.}$ instances distributed according to such multinomial law is given by $\frac{N_{i.}!}{N_{i1}!N_{i2}!\ldots N_{iJ}!}$. To finish, according to the last hypothesis, for a given target interval, every distribution of the ranks of the target values are equiprobable which leads to the last terms.

By taking negative logarithms, one obtains the above Formula (1).□

To provide a first intuition, we can compute that for $I = J = 1$ the criterion value is $2\log(N) + \log(N!)$ (about 615 for $N = 150$) and for $I = J = N$ it gives $2\log(N) + \log\begin{pmatrix} 2N-1 \\ N-1 \end{pmatrix} + N\log(N)$ (about 966 for $N = 150$). This means that a discretization with one cell is always more likely that a 2D discretization with N^2 elementary cells.

We adopt a simple heuristic to optimize this criterion. We start with an initial random model and alternate the optimization on the predictor and target variables. For a given target distribution with fixed $J < \sqrt{(N)}$ and $N_{.j}$, we optimize the discretization of the predictor variable to determine the values of I, N_i and N_{ij}. Then, for this predictor discretization, we optimize the discretization of the target variable to determine new values of J, $N_{.j}$ and N_{ij}. The process is iterated until convergence, which usually takes between two and three steps in practice. The univariate discretization optimizations are performed using the MODL discretization algorithm. This process is repeated

several times, starting from different random initial solutions. The best solution is returned by the algorithm as described in Algorithm 1.

Each 1D-discretization is implemented according to a bottom-up greedy heuristic followed by a post-optimisation whose time complexity is in $N\log(N)$ times the size of the fixed partition. This algorithm complexity is mainly obtained by using the criteria additivity for 1D discretization (see Boullé, 2006). Imposing a maximum iteration number $P = 10$ for instance, the worst case complexity is bounded by $PN\sqrt(N)log(N)$ without decreasing the quality of the search algorithm. Despite

Algorithm 1 Optimization of a MODL 2D-discretization for regression tasks

Ensure: $M^*; c(M^*) \leq c(M)$ {Final solution with minimal cost}
1: **for** $m = 1, \ldots, 10$ **do**
2: {Initialize with a random partition}
3: $M \leftarrow$ a random partition of size $\sqrt(N)$
4: **while** improved **do**
5: {Univariate optimal 1D-discretization of predictor variable X:}
6: freeze the univariate partition of target variable Y
7: $M \leftarrow$ call *univariate optimal 1D-discretization* (M) for predictor variable X
8: {Univariate optimal 1D-discretization of target variable Y:}
9: freeze the univariate partition of predictor variable X
10: $M \leftarrow$ call *univariate optimal 1D-discretization* (M) for target variable Y
11: **end while**
12: **if** $c(M) \leq c(M^*)$ **then**
13: $M^* \leftarrow M$
14: **end if**
15: **end for**

the fact that this optimization algorithm discretizes alternatively the predictor and target variables, it is important to notice that the criterion in (1) is not symmetrical in I and J. In other words, for a given target discretization, the criterion to minimize is not identical to the criterion to minimize given a predictor discretization.

The evaluation criterion $c(M)$ given in Formula (1) is related to the probability that a regression 2D discretization model M explains the target variable. In previous work (see Boullé and Hue, 2006), we propose to use it to build a relevance criterion for the predictor variables in a regression problem. The predictor variables can be sorted by decreasing probability of explaining the target variable. In order to provide a normalized indicator, we consider the following transformation of c:

$$g(M) = 1 - \frac{c(M)}{c(M_\emptyset)}, \tag{2}$$

where $M_\emptyset$ is the null model with only one interval for the predictor and target variables. This can be interpreted as a compression gain, since negative log of probabilities are no other than coding lengths (see Shannon, 1948). The compression gain $g(M)$ holds its values between 0 and 1, since the null model is always considered in our optimization algorithm. It has value 0 for the null model and is maximal when the best possible explanation of the target ranks conditionally to the predictor ranks is achieved.

Our method is non parametric both in the statistical and algorithmic sense: no statistical hypothesis needs to be done on the data distribution (like Gaussianity for instance) and, as the criterion is

regularized, there is no parameter to tune before minimizing it. This strong point enables to consider large data sets.

To our knowledge, few other works address the problem of discretization for regression problems. Nevertheless, we can cite a three-step approach proposed in the approach of Ludl and Widmer (2000): they first propose an equal width pre-discretization of the continuous predictors. These pre-discretizations are next projected onto the target values. A postprocessing consists in merging the split points found according to an algorithm inspired by edge detection concepts. The drawbacks of such an algorithm are the unsupervised way the predictor pre-discretizations are led and the tuning of the merging parameter needed in the postprocessing step.

2.2 The 2D Discretization-Grouping for Probabilistic Regression with Categorical Predictor

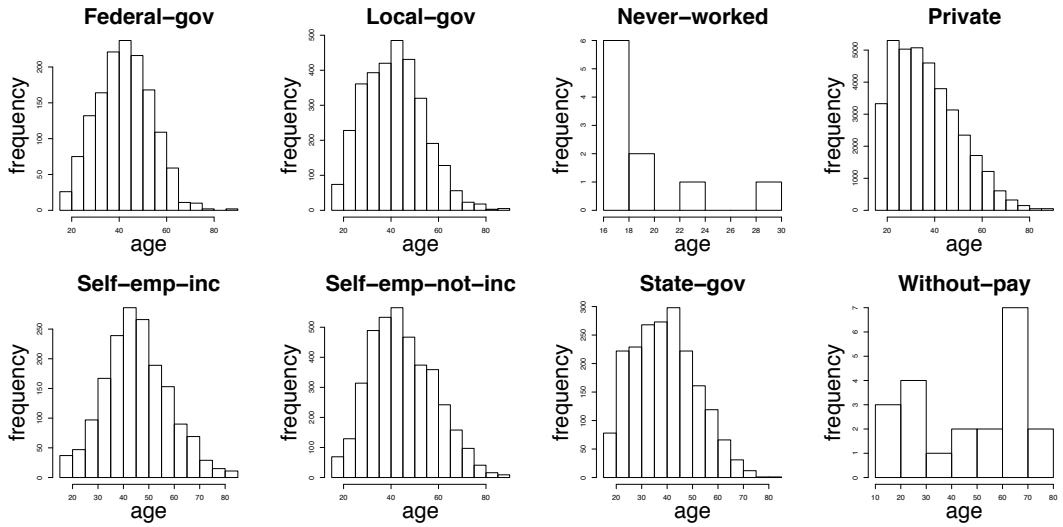


Figure 3: Equal-width histograms of the age variable according to each level of the workclass factor

In order to illustrate the regression problem with a categorical predictor, we present in Figure 3 the equal-width empirical histograms of the target age variable for each value of the predictor workclass variable from the 48842 instances of the adult data set from the UCI repository (see D.J. Newman and Merz, 1998). The workclass variable clearly influences the distribution of the age variable. The four values Federal-gov, Local-gov, Self-emp-inc and Self-emp-not-inc lead to similar histograms with a maximum for the $40-45$ interval. The Private value gives a distinct histogram with a maximum for the $20-25$ interval and the Stat-gov value leads to an histogram between theses two groups. The frequencies of the Never-worked and Without-pay values seems too low to constitute significant groups.

An example of discretization/grouping is shown for this data set on Figure 4 with 3 groups and 7 age brackets. Let us now formalize this approach using the MODL approach to explain how optimal Discretization/Grouping can be obtained.

Definition 4 *A regression discretization/grouping model is defined by:*

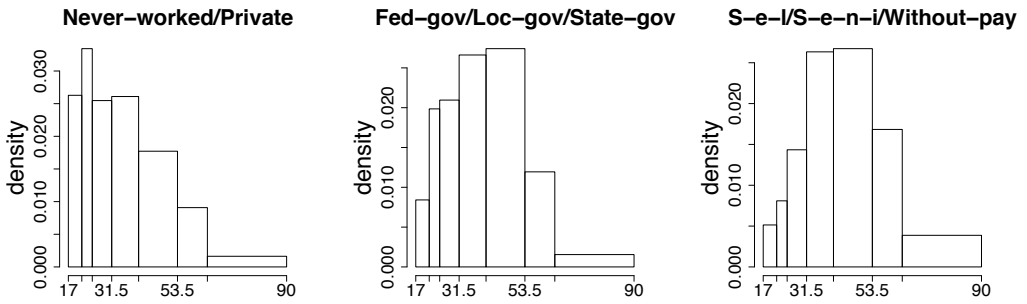


Figure 4: Histograms of the age variable for three groups of the workclass factor

1. *a number of intervals for the target variable and a number of groups for the predictor variable;*

2. *a partition of the predictor variable in a finite number of groups;*

3. *for each predictor group, the repartition of the instances among the target intervals specified by the instance counts locally to each predictor group*.

Notations

N : the number of training instances

V : the number of predictor values

I : the number of predictor groups

J : the number of target intervals

$\iota(\nu)$: the group index the ν value belongs to

$N_{i.}$: the number of instances in the predictor group i

$N_{.j}$: the number of instances in the target interval j

N_{ij} : the number of instances in the grid cell associated to predictor group i and the target interval j.

A regression discretization/grouping model is then entirely characterized by the parameters $\left\{I, J, \{\iota(\nu)\}_{1\leq i\leq V}, \{N_{ij}\}_{1\leq i\leq I, 1\leq j\leq J}\right\}$. The number of instances $N_{.j}$ can be deduced by adding the N_{ij} for each predictor group.

We adopt the following uniform hierarchical prior for the parameters of regression discretization/grouping models:

Definition 5 *The prior for the parameters of a regression discretization/grouping model is chosen hierarchically and uniformly at each level:*

1. *the number of groups I is uniformly distributed between* 1 *and* V,
2. *the numbers of intervals J is independent from the number of groups, and uniformly distributed between* 1 *and* N,
3. *for a given number of groups I, every partition of the predictor values into I groups is equiprobable,*
4. *for a given predictor group, every distribution of the instances on the target intervals is equiprobable,*
5. *the distributions of the target intervals on each predictor group are independent from each other,*
6. *for a given target interval, every distribution of the rank of the target values is equiprobable.*

The definition of the regression discretization/grouping model space and its prior distribution leads to the evaluation criterion given in Formula (3) for a discretization/grouping model M:

$$\begin{aligned} c(M) = \log(V) + \log(N) + \log B(V,I) + \sum_{i=1}^{I} \log \left(\begin{array}{c} N_{i.}+J-1 \\ J-1 \end{array} \right) \\ + \sum_{i=1}^{I} \log \frac{N_{i.}!}{N_{i,1}!N_{i,2}!\ldots N_{i,J}!} + \sum_{j=1}^{J} \log N_{.j}!, \end{aligned} \tag{3}$$

where $B(V,I)$ is the number of ways to partition V values into I groups (possibly empty). For $I = V$, $B(V,I)$ corresponds to the Bell number. In general, $B(V,I)$ can be written as a sum of Stirling numbers of the second kind $S(V,i)$ (number of ways to partition a set of V values into i nonempty subsets) (see Abramowitz and Stegun, 1970):

$$B(V,I) = \sum_{i=1}^{I} S(V,i).$$

This criterion can be deduced from the grouping criterion in classification (see Boullé, 2005) and the 2D discretization criterion in regression presented in the previous section.

3. From 2D Partitioning to Rank Predictive Cumulative Distribution Estimate

In this section we expose how we can build a univariate estimator of the rank predictive density from each 2D partition and how to obtain multivariate predictors under the naive Bayes assumption.

3.1 From Values to Normalized Training Ranks and Vice-Versa

As seen in the precedent section, the MODL partitions are defined only with the ranks of the training instances and not with their values. Given N_T numerical training values $D^T = (y^T_1, \ldots, y^T_{N_T})$, the N_T ranked values are noted $y^T_{(1)}, \ldots, y^T_{(N_T)}$ once the training values have been sorted.

A partition of the N_T ranked instances defined by J numbers $N_1, \ldots, N_J$ such that $\sum_{j=1}^{J} N_j = N_T$ is associated to a partition of the values as follows: we define $J-1$ boundaries $b_1, \ldots, b_{J-1}$ by $b_j =$

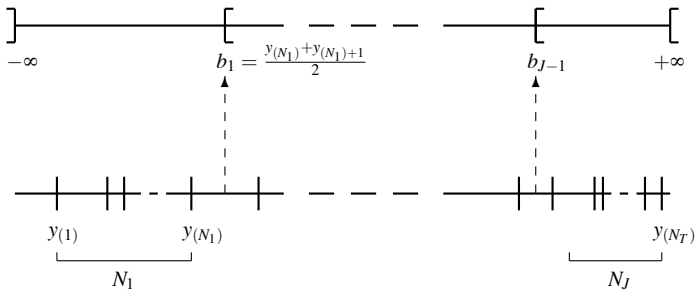


Figure 5: Value partition from the partition frequencies: for example, the upper bound of the first interval containing N_1 instances is the mean of the last value of this interval and of the first value of the second interval.

$\frac{y^T_{(\beta_j)}+y^T_{(\beta_{j+1})}}{2}$ where $\beta_j = \sum_{l=1}^{j} N_l$ and the J value partition intervals are $]-\infty, b_1[, [b_1, b_2[, \ldots, [b_{J-1}, +\infty[$. For a numerical value, its rank interval index is equal to its value interval index. This value partition from the partition frequencies is illustrated in Fig 5.

As presented in the introduction, ordinal regression aims at predicting an ordinal variable which takes a finite number of ordered values, most of the time already known in advance. In our case, we aim at giving a finer grain prediction by considering the set of the N_T possible ranks of a training data set of size N_T. In order to manipulate normalized values in $[0,1]$, we consider the N_T elementary equal-width rank intervals of $[0,1]$ denoted by Te_n for $n = 1, \ldots, N_T$ and equal to $Te_1 = [0, \frac{1}{N_T}[, Te_2 = [\frac{1}{N_T}, \frac{2}{N_T}[, \ldots, Te_{N_T} = [\frac{N_T-1}{N_T}, 1]$. These intervals are centered on the normalized ranks $R_{D^T}(y^T_{(n)}) = \frac{1}{2N_T} + \frac{n}{N_T}$ of the training instances obtained by projection on $[0,1]$ of the rank of $y^T_{(n)}$ among D^T. This normalization is illustrated in Fig 6.

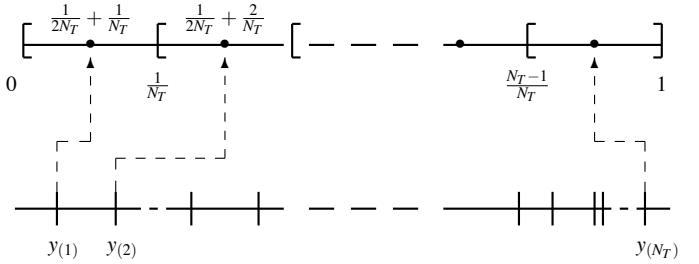


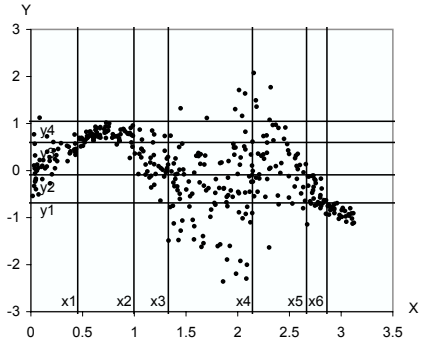
Figure 6: From sorted values to normalized ranks: for example, the normalized rank related to the first value $y^T_{(1)}$ is $R_{D^T}(y^T_{(1)}) = \frac{1}{2N_T} + \frac{1}{N_T}$, which is the center of the first elementary interval $Te_1 = [0, \frac{1}{N_T}[$.

In practice, there may be equal values in D^T. In this case, we affect the averaged rank to the concerned instances. For a new value y unseen during training, we define its rank $R_{D^T}(y)$ as the average of the normalized ranks of $y^T_{(n_1)}$ and $y^T_{(n_2)}$ such that $y^T_{(n_1)} \leq y < y^T_{(n_2)}$. The integers n_1 and n_2 may not be consecutive if one of them is associated with several equal values. In the rest of the paper, we use either ranks or values depending on the context.

We now detail how to build an estimate of the predictive cumulative distribution function of the target standardized rank from a univariate MODL 2D discretization in a first section and from multiple univariate partitionings in a second section.

3.2 Univariate Case

We illustrate the construction of the univariate estimator from the 2D partitioning on a synthetic data set proposed during the recent *predictive uncertainty in environmental modelling* competition (see Cawley et al., 2006). This data set, called synthetic, contains $N_T = 384$ training instances and one numerical predictor. The scatter plot and the optimal MODL partition are presented in Figure 7. The optimal MODL rank intervals are denoted P_i for $i = 1,\ldots,7$ for predictor and T_j, $j = 1,\ldots,5$ for the target. The first and the last rank intervals are of the form $[0, k_f/N_T[$ and $[k_l/N_T, 1]$ respectively and the other intervals are of the form $[k_1/N_T, k_2/N_T[$. The *value* interval bounds $x_1,\ldots,x_6$ and $y_1,\ldots,y_4$ are obtained by projecting the frequencies partition on the value partition as described in 3.1. For the predictor component x of a new instance whose rank range is $P_i(x)$, the number of



	P1	P2	P3	P4	P5	P6	P7	Total
T5	1	0	0	6	5	0	0	12
T4	4	68	5	2	9	0	0	88
T3	38	5	26	32	24	0	0	125
T2	9	0	6	33	15	21	0	84
T1	0	0	0	34	4	9	28	75
Total	52	73	37	107	57	30	28	384

Figure 7: Scatter plot, 2D partitioning and numbers of the MODL grid for the synthetic data set.

examples in each grid cell give us an estimator of the probability that the target standardized rank belongs to a given range T_j:

$$P_{Modl}\left(R_{D^T}(y) \in T_j \mid R_{D^T}(x) \in P_i(x)\right) = \frac{N_{ij}}{N_{i.}}.$$

Assuming that the conditional rank density is constant over each rank interval, the probabilities of the elementary intervals Te_n for $n = 1,\ldots,N_T$ are given by:

$$P_{Modl}\left(R_{D^T}(y) \in Te_n \mid R_{D^T}(x) \in P_i(x)\right) = \frac{N_{ij}}{N_{i.}N_{.j}} \text{ for } j \text{ such that } Te_n \subset T_j. \tag{4}$$

We obtain an estimate of the n^{th} N_T-quantile n/N_T of the conditional cumulative distribution for $n = 1,\ldots,N_T$ by summing these elementary probabilities.

The MODL estimators are plotted for each of the seven rank predictor ranges on Figure 8 for the synthetic data set. The marks on the $x-$axis correspond to the normalized target ranks of the boundaries training instances exhibited by the target partition: $\frac{75}{384} \approx 0.19$, $\frac{75+84}{384} \approx 0.41$, $\frac{75+84+125}{384} \approx 0.74$ and $\frac{75+84+125+88}{384} \approx 0.97$ which correspond to the projection of the value bounds y_1, y_2, y_3 and y_4

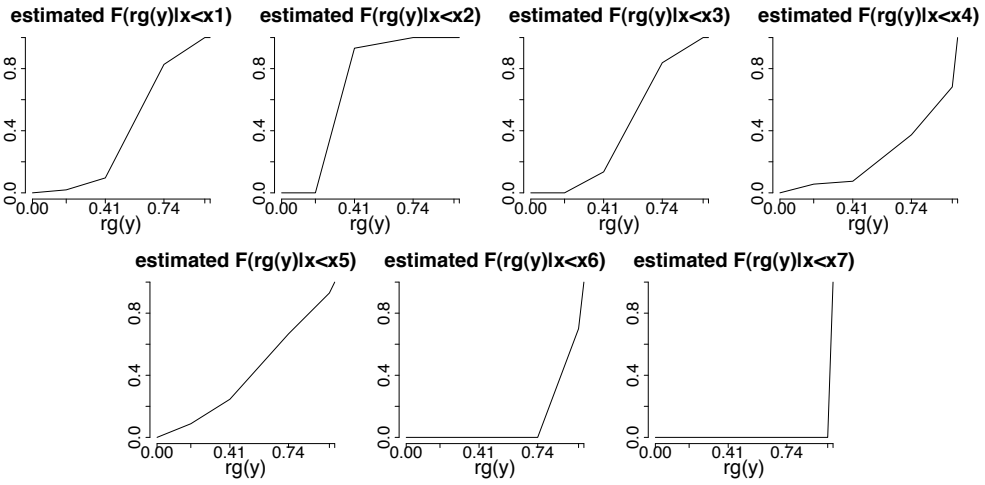


Figure 8: MODL estimators of the conditional univariate standardized rank cumulative distribution for the seven predictor rank ranges.

on the normalized ranks. The shape differences illustrate the seven distinct zones characterized by the MODL optimal partition.

In the case of categorical predictors, the rank predictive cumulative distribution estimator can be obtained in the same way by replacing the predictor intervals by predictor groups.

3.3 Multivariate Case

In the case of several predictors (K with $K > 1$), a first approach is to build an estimator under the naive Bayesian assumption that the predictors are independent given the target. Let $x = (x^1, \ldots, x^K)$ be the coordinates of a new instance in the predictors space and $P^k_{i_k}(x)$ the discretization interval (or group of values) to which belongs each component x^k and $R^k_{D^T}(x)$ its rank. Under the naive Bayesian assumption, the elementary probability can be written :

$$\begin{aligned} & P\left(R_{D^T}(y) \in Te_n \mid (R^1_{D^T}(x), \ldots, R^K_{D^T}(x)) \in (P^1_{i_1}(x), \ldots, P^K_{i_K}(x))\right) \\ & \propto P\left(R_{D^T}(y) \in Te_n\right) \prod_{k=1}^{K} P\left(R^k_{D^T}(x) \in P^k_{i_k}(x) | R_{D^T}(y) \in Te_n\right) \\ & = P\left(R_{D^T}(y) \in Te_n\right) \prod_{k=1}^{K} \frac{P(R_{D^T}(y) \in Te_n | R^k_{D^T}(x) \in P^k_{i_k}(x)) P(R^k_{D^T}(x) \in P^k_{i_k}(x))}{P(R_{D^T}(y) \in Te_n)}. \end{aligned} \tag{5}$$

This last expression can be estimated with the instance numbers in the 2D grid cells. The first factor $P(R_{D^T}(y) \in Te_n)$ can be estimated by the empirical probability $1/N_T$. Each factor of the product can be computed from the numbers of the 2D partitioning of the target and of the k^{th} predictor, denoted $(I_k, J_k, N^k_{i_k.}, N^k_{i_k j_k})$:

$$P(R_{D^T}(y) \in Te_n | R^k_{D^T}(x) \in P^k_{i_k}(x)) = \frac{N^k_{i_k j_k}}{N^k_{i_k.} N^k_{.j_k}} \text{ according to (4).}$$

$$P(R^k_{D^T}(x) \in P^k_{i_k}(x)) = \frac{N^k_{i_k\cdot}}{N_T} \text{ and } P(R_{D^T}(y) \in Te_n) = \frac{1}{N_T}.$$

Each factor reduces to the fraction $\frac{N^k_{i_k j_k}}{N^k_{\cdot j_k}}$ and the elementary probabilities in Formula 5 reduce to $\frac{1}{N_T}\prod_{k=1}^{K}\frac{N^k_{i_k j_k}}{N^k_{\cdot j_k}}$.

However, the independence hypothesis assumed in the naive Bayes predictor is usually violated for real data sets. In this case, estimates of the conditional probabilities are deteriorated as already noticed in the work of Frank et al. (1998). For classification tasks, variable selection has been employed to build selective naive Bayes classifiers. This procedure reduces the strong bias of the naive independence assumption. The objective is to search among all the subsets of variables, in order to find the best possible classifier, compliant with the naive Bayes assumption. Several selection criteria have been tested, such as the accuracy criterion (see Langley and Sage, 1994), the area under receiver operating characteristic (ROC) curve (see Provost et al., 1998) or the posterior probability of the model given the data proposed in the work of Boullé (2007). In this last case, the posterior probability is written as the sum of the prior probability of the model and of the likelihood of the data given the model. The prior is chosen such that each specific small subset of variables has a greater probability than each specific large subset of variables in order to favour small models. For a given subset of variables, the likelihood is computed using the naive Bayes assumption.

We propose here to build selective naive Bayes rank predictors using a MAP approach as in Boullé (2007). The extension to rank regression tasks is straightforward: the prior law remains unchanged and, for a given subset of variables, the likelihood of the ranks of the instances are computed assuming the naive Bayes assumption according to (5).

To summarize this section, we have exposed how the 2D partitionings give us estimators of quantiles of the univariate or multivariate rank cdf. The choice of the estimated quantiles are given by the univariate partitionings. Let us see, in the next section, on which criterion such probabilistic rank models can be evaluated.

4. Performance Evaluation for Probabilistic Rank Regression

For deterministic predictive models, performance evaluation consists in evaluating the distance between the predicted class or value and the true class or value. Depending on the metric used, several performance measures can be used such as the mean absolute error or the mean squared error.

For probabilistic predictive models, performance evaluation consists in comparing the true value or class with an estimate of its conditional cumulative distribution function (cdf) or an estimate of its conditional probability density function (pdf). It is measured through a score function which can be negatively oriented (large values imply poor performance) or positively oriented (large values imply high performance). A score function is said to be proper if its expectation is maximized (or minimized) for the true predictive distribution. It is said strictly proper if this optimum is unique.

Among the strictly proper scoring functions, the two more commonly used are the logarithmic and the quadratic scores (see Gneiting and Raftery, 2004). They take different forms depending on the learning task. In the sequel, we first describe the quadratic and the logarithmic scores and their use in classification and regression. We then present an interesting way to build the logarithmic score function from ranks rather than from values.

4.1 Performance Evaluation for Probabilistic Classification and Regression

The quadratic score is called the Brier score for binary classification (see Brier, 1950). For classification with a finite number of ordered classes $j = 1,\ldots,J$ (ordinal regression), the discrete ranked probability score, DRPS, has been proposed in the work of Epstein (1969):

$$S_{DRPS}(\hat{p},(x,y)) = 1 - \frac{1}{I-1}\sum_{j=1}^{J}(\hat{P}(Y \leq j \mid x) - 1_{\{Y \geq j\}}(y))^2.$$

Its extension to the continuous case (metric regression) is the continuous ranked probability score, CRPS (see Matheson and Winkler, 1976):

$$S_{CRPS}(\hat{p},(x,y)) = -\int_{-\infty}^{+\infty}(\hat{P}(Y \leq u \mid x) - 1_{\{Y \geq u\}}(y))^2 du.$$

The CRPS is a bounded global score which manipulates the cumulative distribution function. Its main disadvantage is that it is generally not a closed form. Nevertheless, a closed form has been derived in the Gaussian case and in the case of ensemble prediction systems where the cdf is piecewise constant (see Hersbach, 2000).

The logarithmic score is commonly called the ignorance score for classification and has been introduced in the work of Good (1952). For value regression it is called the negative log-likelihood (NLL) or negative log predictive density (NLPD). It takes the negative logarithm of the posterior class probabilities for classification and of the predictive density for regression:

$$S_{NLPD}(\hat{p},(x,y)) = -\log(\hat{p}(y|x)). \tag{6}$$

The NLPD is a local score as it only depends on the predictive density on the example. It is clearly negatively oriented.

Practically, for regression, this criterion requires the definition of the probability density function on any point. If the target distribution is described by a sample, its pdf is not defined. A common choice to specify such predictive distribution is to describe the cdf by a set of N quantiles of the form:

$$\alpha^x(n) = \hat{P}(y < q^x(n) \mid x) \text{ for } n = 1,\ldots,N.$$

Moreover, assuming that the cdf is constant on each interval $[q^x(n);q^x(n+1)[$, one obtains the following expression for the logarithmic score:

$$S_{NLPD}(\hat{p}(\{q^x(n)\}),(x,y)) = -\log\left(\frac{\alpha^x(n_y) - \alpha^x(n_y-1)}{q^x(n_y) - q^x(n_y-1)}\right), \tag{7}$$

where $q^x(n_y-1) \leq y < q^x(n_y)$ and $\hat{p}(\{q^x(n)\})$ designs the estimator specified by the quantiles $\{q^x(n)\}$.

Such as defined in Equation (6), the NLPD presents the main drawback *not to be minored*: the more the density is peaked around the sample values, the smaller the score values. It is not a problem in classification as the posterior probabilities are bounded but, for regression, it encourages dishonest predictions concentrated around some specific values, as noticed in recent challenges (see Kohonen and Suomela, 2005). In order to achieve a fair evaluation, if the predictive density is expressed as a set of quantiles as exposed before, a last resort is to impose a minimum width for

scoring rule	quadratic score	logarithmic score
binary classification	Brier score	Ignorance score
ordinal regression	DRPS	
value regression	CRPS	NLPD

Table 1: Quadratic and logarithmic scores for performance evaluation of probabilistic predictive models

the intervals $[q^x(n-1);q^x(n)[$. Anyway, such a score function contributes to confusing performance prediction since arbitrary small values can be obtained fortunately.

The two scores mentioned above are presented in Table 1.

Let now study what happens if we compute the logarithmic score for the rank predictive density rather than for the value predictive density.

4.2 A Robust Logarithmic Score Defined on the Rank Predictive Density

Given a training data set set $D^T = (x_n^T, y_n^T)_{n=1,\ldots,N_T}$, we assume that we have at our disposal a rank probabilistic estimator specified for any given x by the N_T estimated quantiles $\alpha^x(n)$ for $n = 1,\ldots,N_T$ of the standardized rank cumulative distribution function:

$$\alpha^x(n) = \hat{P}(R_{D^T}(y) < \frac{n}{N_T}|x) \text{ for } n = 1,\ldots,N_T.$$

We can immediately see that this assumption is not restrictive: knowing the values associated to the ranks, each *value* estimator gives us an estimator of the cdf on the N_T target normalized *ranks* of the training data set. If we denote by $y_{(n)}$ the n^{th} target value of D_T, we have :

$$\alpha^x(n) = \hat{P}(R_{D^T}(y) < \frac{n}{N_T}|x) = \hat{P}(y < \frac{y_{(n)} + y_{(n+1)}}{2}|x). \tag{8}$$

By appropriate integration of the cdf, each *value* estimator specified by N given quantiles gives us the N_T quantiles estimates defined in (8).

Let us come back to the evaluation of such a rank probabilistic estimator with the logarithmic score function. It consists in comparing it with the standardized insertion rank $R_{D^T}(y)$ of the true value y among the training data set D^T. We propose to estimate the predictive density on the insertion rank by the ratio:

$$\hat{p}(R_{D^T}(y)|x) \approx \frac{\hat{P}(R_{D^T}(y) \in Te_{n_y} \mid x)}{1/N_T},$$

where Te_{n_y} is the elementary interval to which the insertion rank $R_{D^T}(y)$ belongs.

This approximation enables us to define the Negative Log Rank Predictive Density as follows:

Definition 6 *Let a training data set set $D^T = (x_n^T, y_n^T)_{n=1,\ldots,N_T}$ and $\alpha^x(n)$ for $n = 1,\ldots,N_T$ some estimates of the N_T quantiles of the standardized rank cumulative distribution function:*

$$\alpha^x(n) = \hat{P}(R_{D^T}(y) < \frac{n}{N_T}|x) \textit{ for } n = 1,\ldots,N_T.$$

Rank predictive distribution specification: Let $D^T = (x_n^T, y_n^T)_{n=1,\ldots,N_T}$ be a training data set with Y a numerical target and $X = (X^1, \ldots, X^K)$ K numerical or categorical predictors. For any given x, the rank predictive distribution is specified by the N_T quantiles of the rank cdf:

$$\alpha^x(n) = \hat{P}(R_{D^T}(y) < \frac{n}{N_T} \mid x) \text{ for } n = 1, \ldots, N_T.$$

Rank predictive distribution evaluation: Let $D^V = (x_n^V, y_n^V)_{n=1,\ldots,N_V}$ be a validation data set. Compute the logarithmic score of the rank predictive distribution on data set D^V as

$$NLRPD = -\frac{1}{N_V} \sum_{n=1}^{N_V} \log \frac{\alpha^{x_n^V}(n_y) - \alpha^{x_n^V}(n_y - 1)}{1/N_T},$$

where $R_{D^T}(y_v)$, the standardized insertion rank of the true value y_n^V among the training data set D^T, is included in the elementary interval Te_{n_y}.

Table 2: Summary of the proposed approach for probabilistic standardized rank regression and its evaluation.

The score function for the negative log rank predictive density is then defined by:

$$\begin{aligned} S_{NLRPD}(\hat{P}(N_T), (x,y)) &= -\log \frac{\hat{P}(R_{D^T}(y) \in Te_{n_y} \mid x)}{1/N_T} \\ &= -\log(\alpha^x(n_y) - \alpha^x(n_y - 1)) - \log(N_T), \end{aligned} \tag{9}$$

where $\hat{P}(N_T)$ is the rank predictive density estimator specified by the quantiles $\{\frac{n}{N_T}\}$. We now present two interesting properties of the NLRPD.

Theorem 7 *In absence of predictive information the NLRPD is equal to zero.*

Without any predictive information, the probability that the insertion rank of a new instance belongs to a given interval is simply equal to $1/N_T$. By construction, we have then for the uniform predictor $\hat{P}^{unif}(N_T)$:

$$S_{NLRPD}(\hat{P}^{unif}(N_T), (x,y)) = -\log 1 = 0.$$

□

Moreover, unlike the NLPD score, this score function on ranks has the great advantage to be minored as it is precised in the following property:

Theorem 8 *Given a training data set set D^T of size N_T, the NLRPD score function is minored by $-\log(N_T)$.*

This bound is directly obtained by considering that the difference $\alpha^x(n_y) - \alpha^x(n_y - 1)$ belongs to $]0;1]$.□

By construction, the proposed NLRPD score is then bounded. However, it depends on the training data set through its size. We will see in next section its relative insensitivity to this size.

Let us now examine the link between the score function on ranks and the score function on values. Using the expression (7) with the $N_T - 1$ quantiles $b_1 = \frac{y_{(1)}+y_{(2)}}{2}, b_2 = \frac{y_{(2)}+y_{(3)}}{2}, \ldots, b_{N_T-1} = \frac{y_{(N_T-1)}+y_{(N_T)}}{2}$, we obtain the relation :

$$S_{NLRPD}(\hat{P}(N_T),(x,y)) = S_{NLPD}(\hat{p}(\{b_{n_y}\}),(x,y)) - \log(N_T) - \log(b_{n_y} - b_{n_y-1}).$$

As precised at the beginning of the section, the NLRPD score function can be used by converting any value predictive density estimator to rank predictive density estimator specified or using the above relation between NLRPD and NLPD score functions.

The framework of our approach is summarized in Table 2.

5. Experimental Evaluation

We first present experiments about the criteria proposed in the precedent section. Then, we focus on the quality of the 2D-partitioning with experiments on synthetic data. We finish by experiments with the univariate and multivariate predictors presented on five real data sets.

5.1 Experiments on the NLRPD

We focus here on the properties of the proposed criterion, the NLRPD. We consider the data generated according to the following heteroscedastic model (used for the synthetic data set of the *predictive uncertainty in environmental modeling competition*):

$$\begin{cases} x_n \sim U_{[0\pi]} \\ y_n \sim \mathcal{N}\left(\sin\left(\frac{5x}{2}\right)\sin\left(\frac{3x}{2}\right), \frac{1}{100} + \frac{1}{4}\left(1-\sin\left(\frac{5x}{2}\right)\right)^2\right) \end{cases}$$

This data set has been used in Section 3 to describe the building of the rank conditional densities. Knowing the true cdf of the synthetic data set, we can compute the true NLRPD by using the true probabilities instead of their estimates in (9).

The contentious aspect of the NLRPD is that it depends on the training data set size. The objective of this experiment is then to study the sensitivity of the NLRPD with respect to the training size in a first time and to the validation data set size in a second time. For that, we have generated $m = 100$ training data sets of size $N_T = 2^n$ for $n = 1, \ldots, 12$. The true NLRPD has been computed given each of the $m * 12$ training quantile vectors for a test data set of size $N_V = 1000$ and another one of size $N_V = 10000$. The mean NLRPD over the m training data sets is plotted versus the training data set size N_T on left of Figure 9 for $N_V = 1000$ and on right for $N_V = 10000$.

First, this plot shows a threshold around a training data set size of $N_T = 100$ instances. Below this threshold, the true NLRPD decreases when the training set size increases. Above this threshold, the optimal NLRPD seems insensitive to the training set size. Secondly, we can notice that both curves for $N_V = 1000$ and $N_V = 10000$ are very similar. This allows us to think the NLRPD is not very sensitive to the test data set size. To confirm this fact, we have fixed a training data set of size 384 and we have computed the true NLRPD for $m = 100$ test data sets of size $N_V = 1024$. The standard-deviation obtained is around 3%. Our criterion looks robust with respect to the training and validation data set size.

5.2 Experiments on the 2D-Partitioning

In this section, we focus on the quality of the 2D-partitioning with three experiments.

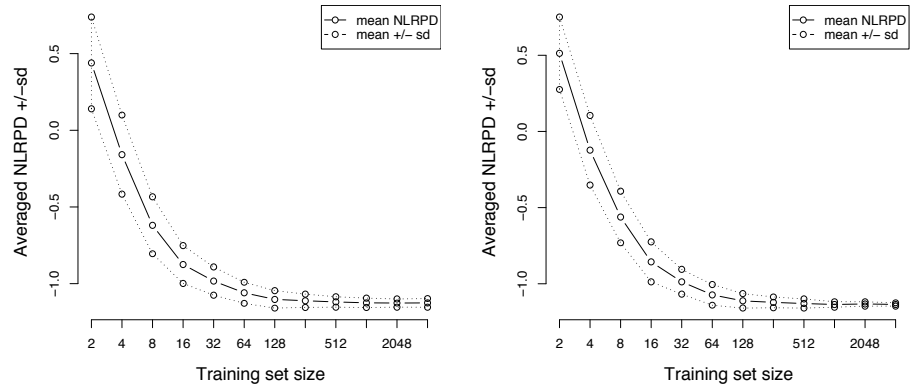


Figure 9: Mean NLRPD over $m = 100$ training data sets of size $N_T = 2^n$, $n = 1, \ldots, 12$ and for a validation data set of size $N_V = 1000$ on the left and $N_V = 10000$ on the right.

A strong point of the MODL approach is that it does not presuppose the existence of a relation between the predictor and the target variables. In the case of numerical predictors, the absence of relevant information should conduct to elementary 2D-partitions consisting of one single cell. To check the quality of the MODL 2D discretizations, we test it on several *noise pattern* data sets: we generate 10^5 training data sets of size equal to 2^n for $n = 2, \ldots, 10$, for which the predictor and target variables are generated independently according to a uniform law in $[0, 1]$. As the predictor variable contains in fact no relevant information to predict the target variable, we expect that the number of predictor intervals I and of target intervals J would be equal to one. Mean target interval number versus the data set size is plotted in Figure 10. For the two smallest sizes 4 and 8, the number of target intervals is always equal to one as there are not enough instances to constitute any pattern. For larger sample sizes, the number of intervals is sometimes equal to two with an exponential decreasing frequency after a peakly increasing for sizes 16 and 32. In the worst case for $N_T = 32$, some predictive information is wrongly detected in 0.8% of the realizations and it falls to 0.2% for $N_T = 128$ and 0.006% for $N_T = 1024$. The absence of predictive information is then almost always detected.

Secondly we test the capacity of our method to detect predictive information contained in a noisy XOR pattern. A XOR pattern is obtained by generating a predictor variable uniformly in $[0, 1]$ and the target variable uniformly in $[0, 0.5]$ if the predictor variable is less than 0.5 and uniformly in $[0.5, 1]$ otherwise. In this case, the expected partition is the one with $I = J = 2$ intervals which splits the predictor and the target variable on 0.5. Some noise instances generated as in the previous noise pattern are added to constitute what we call a noisy XOR pattern. We can study the robustness of the discretization to noise. In Figure 11, the mean number of target intervals on 100 samples are plotted versus the data set size for noise rates varying from 0 to 1 by 0.2.

Each curve shows a sharp threshold above which the pattern is correctly detected. First, a XOR pattern without noise is correctly detected from $N_T = 16$. The resultat is similar for low noise rate

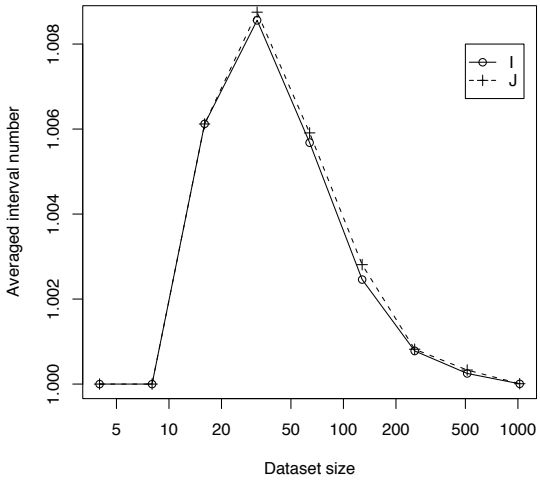


Figure 10: Mean target and predictor interval numbers for 100000 noise pattern data sets of size 4 to 1024

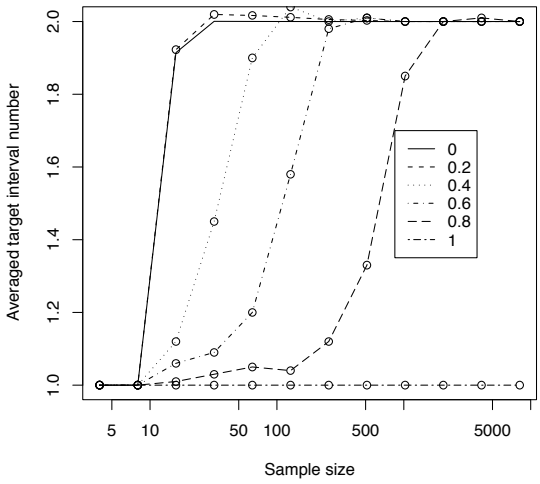


Figure 11: Mean target interval number for 100 noisy XOR pattern data sets of size 4 to 5096 for noise rate $= 0, 0.2, 0.4, 0.6, 0.8, 1$.

equal to 0.2. This threshold increases with the noise rate and reachs respectively 128, 256 and 1024 for a rate equal to 0.4, 0.6 and 0.8. Our discretization is then robust to noise rate.

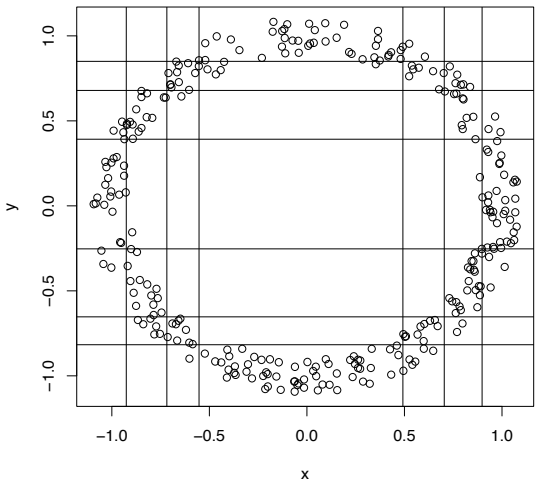


Figure 12: Optimal MODL 2D-partition for data on a noisy circle.

Thirdly, we test the capacity of our method to detect multimodality. For that we generate 300 instances on a noisy circle as showed in Figure 12. The density of y conditionally to x is bimodal for most values of x. As there is no assumption in the MODL approach about the form of the conditional distribution, the 2D-partitioning can produce multimodal conditional densities. For the *noisy circle* data set, the optimal MODL 2D-partition plotted in Figure 12 clearly shows the two modes of the law.

5.3 Experiments on Real Data Sets

In this section, we test the estimators proposed in Section 3 on real data . We have chosen the following five regression data sets, detailed in Table 3:

- SO2, precip and temp, available from the predictive uncertainty competition website (http://theoval.cmp.uea.ac.uk/competition/).
- Adult and housing (Boston), available from the UCI machine learning repository (http://www.ics.uci.edu/ mlearn/MLSummary.html);

Data Set	SO2	Precip	Temp	Adult	Housing
Numerical predictors	27	106	106	6	13
Categorical predictors	0	0	0	7	0
Number of patterns	22956	10546	10675	48842	506

Table 3: Summary of the dimensions of five data sets chosen to evaluate the probabilistic rank predictive density estimators.

For each data set, we have performed a five-fold cross validation for the three following estimators:

SO2	Level	I	J	Precip	Level	I	J	Temp	Level	I	J
V7	0.01467	9	7	V3	0.01986	5	6	V102	0.1234	13	14
V25	0.00993	7	6	V35	0.01858	5	6	V104	0.1095	12	13
V27	0.009658	7	7	V4	0.01823	5	6	V101	0.1069	13	12
V2	0.009483	6	28	V81	0.01729	4	6	V103	0.1039	12	13
V26	0.008302	6	7	V69	0.01716	4	7	V100	0.06891	9	12
V1	0.003991	5	127	V36	0.01674	4	7	V98	0.06288	9	10
V11	0.001366	5	5	V82	0.01653	4	6	V106	0.05691	8	8
V8	0.0009511	3	4	V70	0.01633	4	6	V99	0.05614	8	9
V12	0.0008027	4	3	V57	0.01632	4	6	V97	0.05566	9	10
V18	0.0007759	3	4	V45	0.01629	4	6	V6	0.03347	7	7

Adult	Level	I	J	Housing	Level	I	J
marital-status	0.0244	11	6	LSTAT	0.0945	5	5
relationship	0.01936	12	5	RM	0.06378	5	5
hours-per-week	0.009682	10	6	NOX	0.04466	5	5
education	0.009164	10	9	INDUS	0.04350	3	5
education-num	0.009016	9	9	PTRATIO	0.03779	5	4
class	0.006555	10	2	CRIM	0.03451	4	5
occupation	0.003556	7	7	TAX	0.03352	4	5
workclass	0.002672	6	5	AGE	0.03208	4	3
capital-gain	0.002257	4	18	DIS	0.02929	5	3
sex	0.0007349	3	2	RAD	0.01977	3	2

Table 4: Compression gains (levels) and size (I,J) of the 2D partitions for the ten most informative variables of the 1-fold of the five data sets.

- the univariate estimator built with the most MODL informative variable;
- the multivariate estimator built under the naive Bayes assumption with all the predictor variables (NB);
- the multivariate estimator built under the naive Bayes assumption with the best selected subset of predictor variables (SNB).

For each fold, the estimators are built from the optimal MODL 2D-discretizations for each couple (predictor,target). The compression gain (or level) defined in (2) enables us to rank the predictors. For illustration, Table 4 presents the level and the size (I,J) of the 2D partitions for the best predictors, for the first training set of each data set.

Each estimator is evaluated with the NLRPD criteria proposed in the previous section. Table 5 presents the mean and standard-deviation of the NLRPD for each data set and each estimator.

First, we can see the poor performance of the naive Bayes estimator which exploits all the univariate predictors: for all data sets except for the adult data set, the NLRPD for the NB estimator is positive that is to say it performs not as good as the predictor built without any predictive information. This phenomenon is due to the violation of the naive Bayes assumption. For example, in

	Univariate	SNB	NB
SO2	-0.136 +/- 0.003	-0.162 +/- 0.0076	0.24 +/- 0.036
Precip	-0.165 +/- 0.005	-0.220 +/- 0.023	9.0 +/- 1.36
Temp	-1.018 +/- 0.012	-1.046 +/- 0.0173	5.46 +/- 0.99
Adult	-0.237+/- 0.0048	-0.378+/- 0.03	-0.287 +/- 0.029
Housing	-0.393+/- 0.128	-0.26 +/- 0.26	0.849+/- 0.49

Table 5: Mean and standard-deviation of the NLRPD for the 3 predictors and the five real data sets.

the case of the temp data set, Figure 13 presents the scatter plot of the two most informative variables. The very high linear correlation clearly deteriorates the naive Bayes predictor based on these variables.

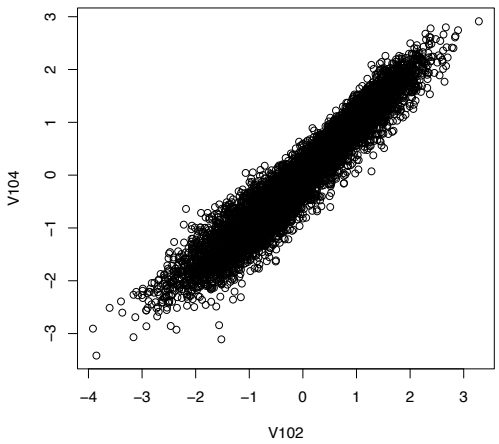


Figure 13: Scatter plot of the two most informative variables for the temp data set.

The second point from these results is the relative good performance of the univariate predictor. For the SO2, precip and temp data sets, it performs nearly as well as the SNB, which selects around five predictors, even if the NLRPD mean for the SNB is significantly lower than for the univariate predictor according to a Student's t-test. These three data sets seem a bit specific in the sense that the most informative variable contains a lot of the predictive information. The good 2D partitioning quality enables to build a very performant univariate predictor. For the adult data set, the SNB, which selects around 9 predictors, performs better than the NB which performs better than the Univariate predictor. The predictive information is shared by several variables and the selection procedure enables to eliminate redundant predictors like education and education-num, or marital-status and relationship. For the housing data set, the univariate predictor performs as well as the SNB which selects around five predictors. The variance of the results is important which certainly explains that the equality hypothesis of the means is not rejected according to a Student t-test. It

NLPD on test set	SO2	Precip	Temp
Organizer's method	4.25 (1st)	-0.509 (1st)	0.053 (2nd)
Best submitted method	4.37 (3th)	-0.279 (3th)	0.034 (1st)
MODL SNB	4.31 (2nd)	-0.437 (2nd)	0.259 (8th)
MODL Univariate	4.33 (2nd)	-0.361 (2nd)	0.284 (8th)
Reference method	4.5 (4th)	-0.177 (4th)	1.30 (9th)

Table 6: NLPD values for each *test* data set for the univariate estimator using the best MODL predictor (MODL univariate), the selective naive Bayes predictor (MODL SNB), the best method in competition, the organizer's submission and the reference method. The ranking of each method is between brackets after each NLPD value.

may seem surprising that the univariate sometimes performs better than the SNB. The reason may be that the selection procedure in the SNB assumes that the univariate predictors are perfect and focuses on the choice of the number of variables. In other words, the uncertainty of the univariate predictive model is not taken into account at this stage. It also suggests that the SNB predictor could be improved. Contrary to the classification case, the target partition is different for each predictor considered. This aspect could be taken into account in the selection of the predictors. Model averaging could also improve our multivariate predictors.

The objective of our last experiment is to compare our approach with other regression methods. To our knowledge, there is no alternative rank regression method available in the literature. We therefore compare it to *value* predictive density estimators. Such estimators being still an active subject of research, we decide to compare our approach to the methods proposed very recently in the predictive uncertainty in environmental modelling competition organized in 2006 by Gavin Cawley. Since these methods are hard to re-implement and tune, we project our rank estimator to a value estimator and we compare them with the NLPD criteria. Knowing the values associated to the ranks, each *rank* estimator gives us an estimator of the cdf on the N_T target values of the training data set using (8). To compute the predictive pdf on any point from the conditional quantiles, we adopt the same assumptions as those used in the challenge, that is that the pdf is assumed uniform between two successive values and that the distribution tails are exponential.[1]

As our approach is implicitly regularized and needs no tuning parameter, we use the training and validation data sets to compute the optimal 2D partitionings. Given the poor performance of the NB in the previous experiments, we only train the univariate and the SNB predictors. Table 6 indicates the NLPD on the test data set for these two MODL estimators, the best method in competition and the reference method which computes the empirical estimator of the marginal law $p(y)$. For the three data sets, the MODL estimators are better than the reference method, that is far from being the case for all submitted methods. Secondly, we observe good performance for the MODL estimators, in particular for the SO2 and Precip data sets where the SNB estimator is at the front after the organizer's method. The good performance of the univariate predictor demonstrates the 2D partitioning quality despite the use of the ranks and not of the values for this step. Moreover, the SNB estimator is always better than the univariate estimator. This proves the presence of additional information and the interest of the selection procedure.

1. For that we affect an $\varepsilon = 1/2N$ probability mass at each tail.

6. Conclusion

We have first proposed a non parametric Bayesian approach for the estimation of the conditional distribution of the normalized rank of a numerical target. Our approach is based on an optimal 2D partitioning of each couple (target, predictor). These partitionings are used to build univariate estimators and multivariate ones under the naive Bayesian assumption of predictors conditional independence, with and without variable selection.

Our approach is applicable for all regression problems with categorical or numerical predictors. It is particularly interesting for those with a high number of predictors as it automatically detects the variables which contain predictive information. As the criteria selection is regularized by the presence of a prior and a posterior term, it does not suffer from overfitting.

Secondly we have proposed a new criterion to evaluate a probabilistic estimator of the rank predictive density. It uses the logarithmic score and presents the main advantage to be minored contrary to the logarithmic score computed for probabilistic estimators of the target *value*. As a value estimator can be projected on a rank estimator, this criterion provides a reliable evaluation criterion for all probabilistic regression estimators on values or on ranks.

Experiments on synthetic data sets show the validity of the proposed evaluation criterion and the quality of the 2D partitioning. Experiments on real data sets show the failure of the naive Bayes but the potential of the selective naive Bayes estimator. A comparison with methods proposed in a recent challenge dedicated to probabilistic metric regression methods evaluates the competitivity of our approach after the projection of our rank estimators on the value range. The very good performance of our best univariate and selective naive Bayes estimators encourages us to work in the future to improve the SNB approach and to evaluate the potential benefit of model averaging.

Acknowledgments

We are grateful to the editor and the anonymous reviewers for their useful comments.

References

M. Abramowitz and I. Stegun. *Handbook of Mathematical Functions*. Dover Publications Inc., New York, 1970.

M. Boullé. A Bayes optimal approach for partitioning the values of categorical attributes. *Journal of Machine Learning Research*, 2005.

M. Boullé. MODL: A Bayes optimal discretization method for continuous attributes. *Machine Learning*, 65(1):131–165, 2006.

M. Boullé. Compression-based averaging of selective naive Bayes classifiers. *Journal of Machine Learning Research*, To appear, 2007.

M. Boullé and C. Hue. Optimal Bayesian 2d-discretization for variable ranking in regression. In *Ninth International Conference on Discovery Science (DS 2006)*, 2006.

G. W. Brier. Verification of forecasts expressed in terms of probability. *Monthly Weather Review*, 78(1):1–3, 1950.

G.C. Cawley, M.R. Haylock, and S.R. Dorling. Predictive uncertainty in environmental modelling. In *2006 International Joint Conference on Neural Networks*, pages 11096–11103, 2006.

P. Chaudhuri and W.-Y. Loh. Nonparametric estimation of conditional quantiles using quantile regression trees. *Bernouilli*, 8, 2002.

P. Chaudhuri, M.-C. Huang, W.-Y. Loh, and R. Yao. Piecewise-polynomial regression trees. *Statistica Sinica*, 4, 1994.

W. Chu and Z. Ghahramani. Gaussian processes for ordinal regression. *Journal of Machine Learning Research*, 6:1019–1041, 2005.

W. Chu and S. Keerthi. New approaches to support vector ordinal regression. In *ICML '05: Proceedings of the 22nd international conference on Machine Learning*, 2005.

K. Crammer and Y. Singer. Pranking with ranking. In *Proceedings of the Fourteenth Annual Conference on Neural Information Processing Systems (NIPS)*, 2001.

C.L. Blake D.J. Newman, S. Hettich and C.J. Merz. UCI repository of machine learning databases, 1998. URL `http://www.ics.uci.edu/~mlearn/MLRepository.html`.

E. S. Epstein. A scoring system for probability forecasts of ranked categories. *Journal of Applied Meteorology*, 8:985–987, December 1969.

J. Fan, Q. Yao, and H. Tong. Estimation of conditional densities and sensitivity measures in nonlinear dynamical systems. *Biometrika*, 83:189–196, 1996.

R.A. Fisher. The use of multiple measurements in taxonomic problems. *Annual Eugenics*, 7, 1936.

E. Frank, L. Trigg, G. Holmes, and I. Witten. Naive Bayes for regression, 1998. URL `citeseer.ist.psu.edu/article/frank98naive.html`. Working Paper 98/15. Hamilton, NZ: Waikato University, Department of Computer Science.

T. Gneiting and A. Raftery. Strictly proper scoring rules, prediction and estimation. Technical report, Department of Statistics, University of Washington, 2004.

I. Good. Rational decisions. *Journal of the Royal Statistical Society*, 14(1):107–114, 1952.

R. Herbrich, T. Graepel, and K. Obermayer. *Large margin rank boundaries for ordinal regression*, chapter 7, pages 115–132. 2000.

H. Hersbach. Decomposition of the Continuous Ranked Probability Score for ensemble prediction systems. *Weather and Forecasting*, 15(5):559–570, 2000.

T. P. Hettmansperger and J. W. McKean. *Robust Nonparametric Statistical Methods*. Arnold, London, 1998.

R. Koenker. *Quantile Regression*. Econometric Society Monograph Series. Cambridge University Press, 2005.

J. Kohonen and J. Suomela. Lessons learned in the challenge: making predictions and scoring them. In *Revised Selected Papers of the 1st PASCAL Machine Learning Challenges Workshop (MLCW, Southampton, UK, April 2005)*, Lecture Notes in Artificial Intelligence 3944, pages 95–116, 2005.

P. Langley and S. Sage. Induction of selective Bayesian classifiers. In *In Proceedings of the Tenth Conference on Uncertainty in Artificial Intelligence*, pages 399–406, 1994. URL `citeseer.ist.psu.edu/langley94induction.html`.

M.-C. Ludl and G. Widmer. Relative unsupervised discretization for regression problems. In *Eleventh European Conference on Machine Learning (ECML-2000)*, pages 246–254, 2000.

J. Matheson and R. Winkler. Scoring rules for continuous probability distributions. *Management Sci.*, 22:1087–1096, 1976.

N. Meinshausen. Quantile regression forests. *Journal of Machine Learning Research*, 7:983–999, 2006.

F. Provost, T. Fawcett, and R. Kohavi. The case against accuracy estimation for comparing induction algorithms. In *In Proc. Fifteenth Intl. Conf. Machine Learning*, pages 445–453, 1998. URL `citeseer.ist.psu.edu/provost97case.html`.

C.E. Shannon. A mathematical theory of communication. *Bell Systems Technical Journal*, 1948.

A. Shashua and A. Levin. Ranking with large margin principles : two approaches. In *Proceedings of the Fiveteenth Annual Conference on Neural Information Processing Systems (NIPS)*, 2002.

I. Takeuchi, Q.V. Le, T.D. Sears, and Smola A.J. Nonparametric quantile estimation. *Journal of Machine Learning Research*, 7:1231–1264, 2006.

H White. Nonparametric estimation of conditional quantiles using neural networks. In *Proceedings of the 1991 Interface Conference*, 1991.

Journal of Machine Learning Research 8 (2007) 2755-2790 Submitted 11/05; Revised 6/07; Published 12/07

Dynamic Weighted Majority: An Ensemble Method for Drifting Concepts*

J. Zico Kolter KOLTER@CS.STANFORD.EDU
Department of Computer Science
Stanford University
Stanford, CA 94305-9025, USA

Marcus A. Maloof† MALOOF@CS.GEORGETOWN.EDU
Department of Computer Science
Georgetown University
Washington, DC 20057-1232, USA

Editor: Dale Schuurmans

Abstract

We present an ensemble method for concept drift that dynamically creates and removes weighted experts in response to changes in performance. The method, dynamic weighted majority (DWM), uses four mechanisms to cope with concept drift: It trains online learners of the ensemble, it weights those learners based on their performance, it removes them, also based on their performance, and it adds new experts based on the global performance of the ensemble. After an extensive evaluation—consisting of five experiments, eight learners, and thirty data sets that varied in type of target concept, size, presence of noise, and the like—we concluded that DWM outperformed other learners that only incrementally learn concept descriptions, that maintain and use previously encountered examples, and that employ an unweighted, fixed-size ensemble of experts.

Keywords: concept learning, online learning, ensemble methods, concept drift

1. Introduction

In this paper, we describe an ensemble method designed expressly for tracking concept drift (Kolter and Maloof, 2003). Ours is an extension of the weighted majority algorithm (Littlestone and Warmuth, 1994), which also tracks drifting concepts (Blum, 1997), but our algorithm, dynamic weighted majority (DWM), adds and removes *base learners* or *experts* in response to global and local performance. As a result, DWM is better able to respond in non-stationary environments.

Informally, concept drift occurs when a set of examples has legitimate class labels at one time and has different legitimate labels at another time. Naturally, over some time scale, the set of examples and the change they undergo must produce a measurable effect on a learner's performance. Concept drift is present in many applications, such as intrusion detection (Lane and Brodley, 1998)

∗. Based on "Dynamic Weighted Majority: A new ensemble method for tracking concept drift", by Jeremy Z. Kolter and Marcus A. Maloof, which appeared in the Proceedings of the Third IEEE International Conference on Data Mining. © 2003 IEEE.

†. Corresponding author

and credit card fraud detection (Wang et al., 2003). Indeed, tracking concept drift is important for any application involving models of human behavior.

In previous work, we evaluated DWM using two synthetic data sets, the STAGGER concepts (Schlimmer and Granger, 1986) and the SEA concepts (Street and Kim, 2001), achieving the best published results (Kolter and Maloof, 2003). Using the STAGGER concepts, we also made a direct comparison to Blum's (1997) implementation of weighted majority (Littlestone and Warmuth, 1994).

Here, we present additional results for the STAGGER data set by making a direct comparison to our implementation of the STAGGER learning system (Schlimmer and Granger, 1986) and to a series of rule learners for concept drift that use the AQ algorithm: AQ-PM (Maloof and Michalski, 2000), AQ11-PM (Maloof and Michalski, 2004), and AQ11-PM+WAH (Maloof, 2003). We also present results for two real-world data sets. The first is the CAP data set (Mitchell et al., 1994), which Blum (1997) used to evaluate his implementation of weighted majority. The second is a data set for electricity pricing, which Harries (1999) used to evaluate Splice2. Finally, we include results for twenty-six UCI data sets (Asuncion and Newman, 2007) because we were interested in evaluating DWM's performance on static concepts.

Overall, results suggest that a weighted ensemble of incremental learners tracks drifting concepts better than learners that simply modify concept descriptions, that store and learn from examples encountered previously, and that use an unweighted ensemble of experts. Although DWM has no advantage over a single base learner for static concepts, results from 26 UCI data sets show that, overall, DWM performs no worse than a single base learner.

We cite two main contributions of this work. First, we present a general algorithm for using any online learning algorithm for problems with concept drift. Second, we present results of an extensive empirical study characterizing DWM's performance along several dimensions: with different base learners, with well-studied data sets, with class noise, with static concepts, and with respect to learners appearing previously in the literature.

The organization of the paper is as follows. In the next section, we survey work on the problem of concept drift, on ensemble methods, and at the intersection: work on ensemble methods for concept drift. In Section 3, we describe dynamic weighted majority, an ensemble method for tracking concept drift. In Section 4, we present the results of our empirical study. Section 5 concludes the paper, and it is here that we discuss directions for future research.

2. Background and Related Work

Dynamic weighted majority is an ensemble method designed expressly for concept drift. For many years, research on ensemble methods and research on methods for concept drift have intermingled little. However, within the last few years, researchers have proposed several ensemble methods for tracking concept drift. In the next three sections, we survey relevant work on the problem of tracking concept drift, on ensemble methods, and on ensemble methods for tracking concept drift.

2.1 Concept Drift

Concept drift (Schlimmer and Granger, 1986; Maloof, 2005) is an online learning task in which concepts change or drift over time. For example, with a professor's e-mail classification system, the concept of an important e-mail will change with semesters and with conference deadlines. Concepts may change suddenly or gradually, and if we view concepts as shapes in a representation space, then

they can change their shape, size, and location. In concrete terms, concept drift occurs when the class labels of a set of examples change over time. Researchers have used both real (e.g., Harries and Horn, 1995; Blum, 1997; Lane and Brodley, 1998; Harries, 1999; Black and Hickey, 2002; Gramacy et al., 2003; Wang et al., 2003; Gama et al., 2004; Delany et al., 2005; Gama et al., 2005; Scholz and Klinkenberg, 2005; Tsymbal et al., 2005) and synthetic (e.g., Schlimmer and Granger, 1986; Widmer and Kubat, 1996; Maloof and Michalski, 2000; Hulten et al., 2001; Street and Kim, 2001; Kolter and Maloof, 2003; Klinkenberg, 2004; Maloof and Michalski, 2004; Gama et al., 2005; Kolter and Maloof, 2005; Scholz and Klinkenberg, 2005) data sets as inspiration for and the evaluation of a variety of methods for tracking concept drift. There has also been theoretical work on this problem (e.g., Helmbold and Long, 1991; Kuh et al., 1991; Helmbold and Long, 1994; Auer and Warmuth, 1998; Mesterharm, 2003; Monteleoni and Jaakkola, 2004; Kolter and Maloof, 2005), but a thorough survey of this work is beyond the scope of the paper.

STAGGER (Schlimmer and Granger, 1986) was the first system designed to cope with concept drift. It uses a distributed concept description consisting of class nodes linked to attribute-value nodes by probabilistic arcs. Two probabilities associated with each arc represent necessity and sufficiency, and are updated based on a psychological theory of association learning (Rescorla, 1968). In addition to adjusting these probabilities when new training examples arrive, STAGGER can also add nodes corresponding to new classes and new features. To better cope with concept drift, STAGGER may decay its probabilities over time. Empirical results on a synthetic data set, now known as the STAGGER concepts, show the method acquiring three changing target concepts in succession. The STAGGER concepts have been a centerpiece of numerous evaluations (Widmer and Kubat, 1996; Widmer, 1997; Maloof and Michalski, 2000; Kolter and Maloof, 2003; Maloof and Michalski, 2004; Kolter and Maloof, 2005), and we discuss them further in Section 4.1.

Concept versioning (Klenner and Hahn, 1994) is a method for coping with gradual or *evolutionary* concept drift. It uses a frame representation and copes with such drift by either adjusting current concept descriptions or creating a new version of these descriptions. The system creates a new version when an instance's attribute values and ranges present in current concept descriptions are dissimilar, based on a measure that accounts for both quantitative (e.g., value differences) and qualitative (e.g., increasing trend) information. Empirical results, based on 124 examples of computers sold between 1987 and 1993, suggest that the system acquired three versions of concepts based on machines' clock frequency and memory size.

The FLORA systems (Widmer and Kubat, 1996) track concept drift by maintaining a sequence of examples over a dynamically adjusted window of time and using them to induce and refine three sets of rules: rules covering the positive examples, rules covering the negative examples, and potential rules that are, at present, too general. Examples entering and leaving the window cause FLORA to refine rules, to move them from one set to another, or to delete them. FLORA2 uses just these basic mechanisms, whereas FLORA3 is an extension that handles recurring contexts. FLORA4 extends FLORA3 with mechanisms for coping with noise, similar to those present in IB3 (Aha et al., 1991). On the STAGGER concepts, the method generally performed well in terms of slope and asymptote after concepts changed. (We further analyze these results in Section 4.1.)

Although not expressly designed for concept drift, the MetaL(B) and MetaL(IB) systems (Widmer, 1997) use meta-learning with naive Bayes and instance-based learning, respectively, to cope with *reoccurring contexts*. For example, the concept of warm is different in the summer than in the winter. In this scenario, *season* or *average temperature* is a contextual attribute that identifies the

relevant concept of warm. As an agent moves from one season to the next, it uses the contextual variable to better focus the base algorithm on those features relevant for learning and prediction.

Meta-learning mechanisms identify contextual attributes by maintaining co-occurrence and frequency counts over the learner's entire history and over a fixed window of time. Using a χ^2 test, the meta-learning algorithm identifies contextual features and those predictive features relevant for the given context. The base learner then uses the predictive features of the contextually relevant examples in the current window to form new concept descriptions. By adding a contextual variable to the STAGGER concepts (Schlimmer and Granger, 1986), experimental results using predictive accuracy suggest that the meta-learners were able use contextual features to acquire target concepts better than and more quickly than did the base learners alone. When compared to FLORA4 (Widmer and Kubat, 1996), which retrieves previously learned concept descriptions when contexts reoccur, MetaL(B) often performed better in terms of slope and asymptote, even though it relearned new concept descriptions rather than retrieving and modifying old ones.

Lane and Brodley (1998) investigated methods for concept drift in one-class learning problems (i.e., anomaly detection). For an intrusion detection domain, they used instance-based learning in which instances were fixed-length sequences of UNIX commands. As a user enters new commands, the system calculates a measure of similarity between the current instance and past instances. They investigated two methods for coping with concept drift. One fits a line to the sequence of similarity measures in a window and uses its direction and magnitude to adjust the decision threshold. A second uses the slope of the line to determine if there is a stable or changing trend and inserts new examples into a user profile only if the trend is changing. Both methods proved useful in striking the delicate balance between learning a user's changing behavior and detecting anomalous behavior.

CD3 (Black and Hickey, 1999) uses ID3 (Quinlan, 1986) to learn online from batches of training data. The method maintains a collection of examples from the stream, all annotated with a time stamp of *current*. It annotates examples in a new training batch with a stamp of *new*. For static concepts, the time stamp is an irrelevant attribute, but if drift occurs, then the time stamp will appear in induced trees. Moreover, one can use its position in the tree as an indicator of how much drift may have occurred—the more relevant the time stamp, the higher it will appear in the tree, and the more drift that has occurred. After pruning, the method produces a set of valid and invalid rules by enumerating paths in the tree with new and current time stamps, respectively. Rule conditions containing a time stamp are then removed.

The performance element uses the valid rules for prediction, and the method uses the invalid rules to remove outdated examples from its store. However, depending on parameter settings, if both an invalid and a valid rule cover a stored example, then the method removes the example. Results from an empirical study involving synthetic data and varying amounts of class noise suggest that CD3 copes with revolutionary and evolutionary concept drift better than—in terms of slope and asymptote—ID3 learning from only the most recent batch and better than ID3 learning from all previously encountered batches.

CD4 (Hickey and Black, 2000) extends CD3 by letting time stamps for a batch be numeric, although the method no longer uses invalid rules to remove stored examples. CD5 (Hickey and Black, 2000) extends CD4 by assigning numeric time stamps to individual examples, rather than to batches. In an empirical study with synthetic data and evolutionary and revolutionary changes in target concepts, all three systems performed comparably.

In subsequent work, Black and Hickey (2002) applied CD3 to twenty-seven months of customer-call data and were able to detect concept drift based on the number and height of time stamps in the

induced decision trees. Indeed, during one period, the time stamp was the most relevant attribute and at the root of the decision tree.

Syed et al. (1999) present an online algorithm for training support vector machines (Boser et al., 1992). The method adds previously obtained support vectors to the new training set and then builds another machine. When compared to a batch algorithm, the online method performed comparable on several UCI data sets (Asuncion and Newman, 2007). The authors pointed to drops and recoveries in predictive accuracy during the incremental runs as evidence that SVMs can handle drift.

The presence of drops and recoveries are necessary for detecting concept drift, but they are not sufficient. The drops in predictive accuracy could have been due to sampling, and in general, it may be difficult to determine if a learner is coping with concept drift or is simply acquiring more information about a static concept. This type of concept drift has been called *virtual concept drift*, as opposed to *real concept drift* (Klinkenberg and Joachims, 2000). Indeed, the data sets included in the evaluation (e.g., monks, sonar, and mushroom) are not typically regarded as having concepts that drift.

Kelly et al. (1999) define *population drift* as being "related to" concept drift, noting that the term *concept drift* has no single meaning. They define population drift as change in the problem's probability distribution. Such changes can occur in the prior, conditional, and posterior distributions, and using artificial data, they analyze the impact on performance of changes to each of these distributions.

The term *concept drift* has been applied to different phenomena, such as drops and recoveries in performance during online learning (Syed et al., 1999). However, in concrete terms, the term has historically meant that the class labels of instances change over time, established in the first paper on concept drift (Schlimmer and Granger, 1986). Such change corresponds to change in the posterior distribution and in the conditional distribution. However, the prior distribution may not change, and changes in the conditional distribution will not necessarily mean that concept drift has occurred.

Klinkenberg and Joachims (2000) used a support vector machine to size windows for concept drift. Rather than tuning parameters for an adaptive windowing heuristic (c.f. Widmer and Kubat, 1996), the algorithm sizes the window by minimizing generalization error on new examples. Upon receiving a new batch of examples, the method generates support vector machines with various sizes of windows using previously encountered batches of training examples, and selects the window size that minimizes the $\xi\alpha$-estimate, an approximation to the leave-one-out or jackknife estimator (Hinkley, 1983). Experimental results on a manually constructed data set derived from 2,608 news documents suggest that exhaustively searching for a window's size was better, in terms of slope and asymptote of predictive accuracy, than a window of fixed size. Results from subsequent experiments indicate that selecting examples in batches or using an adaptive window yielded higher precision and recall than did weighting examples based on their age or their fit to the current model (Klinkenberg, 2004).

The AQ-PM systems (Maloof and Michalski, 2000, 2004; Maloof, 2003), like the FLORA systems (Widmer and Kubat, 1996), induce rules and use partial instance memory (PM) to maintain a subset of the examples from the input stream. However, rather than selecting a sequence, the AQ-PM systems select those examples that lie on the boundaries of concept descriptions, thus they store examples that are important but that do not necessarily reoccur in the stream. AQ-PM (Maloof and Michalski, 2000) uses the AQ algorithm (Michalski, 1969), a batch learning algorithm, to form rules by simply reapplying the algorithm when new examples arrive. It stores examples for a fixed period of time. AQ11-PM and GEM-PM are incremental versions that use the AQ11 (Michalski

and Larson, 1983) and GEM (Reinke and Michalski, 1988) algorithms, respectively, to form rules. Finally, AQ11-PM+WAH is an extension of AQ11-PM that uses Widmer and Kubat's (1996) window adjustment heuristic (WAH) to dynamically size the window of time over which examples are kept. The STAGGER concepts were the centerpiece in the evaluation of all of these systems.

The Concept-adapting Very Fast Decision Tree (CVFDT) learner (Hulten et al., 2001) extends VFDT (Domingos and Hulten, 2000) by adding mechanisms for handling concept drift. VFDT grows a decision tree only from its leaf nodes, so there is no restructuring of the tree (cf. sc iti, Utgoff et al., 1997). The method maintains a decision tree and maintains counts of attribute values for classes in each leaf node. New examples propagate through the tree to a leaf node with the algorithm updating the counts. If the propagated examples and the examples used to form a leaf node are not of the same class, then the method uses a splitting criterion and the Hoeffding (1963) bound to determine if the node should be split.

CVFDT extends VFDT by maintaining at each node in the tree a list of alternate subtrees and attribute-value counts for each class. Like VFDT, new examples propagate through the tree to a leaf node, but they also propagate through all of the alternate subtrees along this path. Periodically, the method forgets examples, and if an alternate subtree is more accurate than the current one, it swaps them. Empirical results on a synthetic data set consisting of a rotating hyperplane, five million training examples, and drift every 50,000 time steps, suggest that CVFDT produced smaller, more accurate trees than did VFDT.

2.2 Ensemble Methods

Ensemble methods maintain a collection of learners and combine their decisions to make an overall decision. Generally, an algorithm applied multiple times to the same data set will produce identical classifiers that make the same decisions, so for ensemble methods to work, there must be some mechanism to produce different classifiers. This is accomplished by either altering the training data or the learners in the collection.

Bagging (Breiman, 1996) is one of the simplest ensemble methods. It entails producing a set of bootstrapped data sets from the original training data. This involves sampling with replacement from the training data and producing multiple data sets of the same size. Once formed, a learning method produces a classifier from each bootstrapped data set. During performance, the method predicts based on a majority vote of the predictions of the individual classifiers.

Weighted majority (Littlestone and Warmuth, 1994), instead of using altered training sets, relies on a collection of different classifiers, often referred to as "experts." Each expert begins with a weight of one, which is decreased (e.g., halved) whenever an expert predicts incorrectly. To make an overall prediction, the method takes a weighted vote of the expert predictions, and predicts the class with the most weight. Winnow is a similar algorithm, but also increases the weights of experts that predict correctly (Littlestone, 1991).

Boosting (Freund and Schapire, 1996) is a method that generates a series of weighted classifiers. It iteratively weights training examples based on how well a classifier predicts them, and in turn, weighting the classifier based on the weights of the examples used to construct it. During performance, each classifier in the ensemble returns a prediction for an instance, and then the global method predicts based on a weighted-majority vote. This method constructs classifiers specialized at predicting examples in specific parts of the representation space. Arcing (Breiman, 1998) is similar to boosting, but uses a different weighting scheme and predicts with a majority vote.

Stacked generalization or *stacking* (Wolpert, 1992), like weighted majority, is an ensemble method for combining the decisions of different types of learners. However, stacking uses the predictions of the base learners to form training examples for a meta-learner. Stacking begins by partitioning the original examples into three sets: training, validation, and testing. The method trains the base learners using the training data and then applies the resulting classifiers to the examples in the validation set. Using the predictions of the examples as features and their original class labels, it forms a new training set, which it uses to train the meta-learner. After training the meta-learner with these examples, performance entails presenting an instance to the base classifiers, using their predictions as input to the meta-classifier. The method's overall prediction is that of the meta-classifier.

There have been numerous studies of ensemble methods (e.g., Hansen and Salamon, 1990; Woods et al., 1997; Quinlan, 1996; Zheng, 1998; Opitz and Maclin, 1999; Bauer and Kohavi, 1999; Maclin and Opitz, 1997; Dietterich, 2000; Zhou et al., 2002; Chawla et al., 2004), and we cannot survey them all. Generally, this research suggests that ensemble methods outperform single classifiers on many standard data sets (Opitz and Maclin, 1999). Boosting is generally better than bagging (Bauer and Kohavi, 1999; Opitz and Maclin, 1999; Dietterich, 2000), although bagging seems more robust to noise than is boosting (Dietterich, 2000). Analysis suggests that ensemble methods work by reducing bias, variance, or both (Breiman, 1998; Bauer and Kohavi, 1999; Zhou et al., 2002).

Popular ensemble methods, such as bagging and boosting, are off-line algorithms, but researchers have developed online ensemble methods. Winnow (Littlestone, 1988) and weighted majority (Littlestone and Warmuth, 1994) fall into this category, as do Blum's (1997) versions of these algorithms.

Online AdaBoost (Fan et al., 1999), when a new batch of examples arrives, weights the examples and then re-weights the ensemble's classifiers based on their performance on the new examples. The method then builds a new classifier from the new, weighted examples. For efficiency, the method retains only the k most recent classifiers.

Online arcing (Fern and Givan, 2003) uses incremental base learners and incrementally updates their voting weights. When a new training instance arrives, for each classifier in the ensemble, the method first increases the classifier's voting weight by one if the classifier correctly classifies the new instance. The method then computes an instance weight for the classifier based on the number of other classifiers in the ensemble that misclassify the instance. Finally, an incremental learning function uses the instance and the weight to refine the classifier. The method may use a weighted or unweighted voting scheme to classify an instance.

2.3 Ensemble Methods for Concept Drift

Ensemble methods for concept drift share many similarities with the online and off-line methods discussed in the previous section. However, methods for concept drift must take into account the temporal nature of the data stream, for a set of examples may have certain class labels at time t and others at time t'.

Clearly, ensemble methods for concept drift must process a stream of data. Some researchers have used repeated applications of off-line learning algorithms to process batches of training examples (Maloof and Michalski, 2000; Street and Kim, 2001; Wang et al., 2003; Scholz and Klinkenberg, 2005). Others have used incremental algorithms (Blum, 1997; Kolter and Maloof, 2003; Maloof and Michalski, 2004; Kolter and Maloof, 2005).

Any method for coping with changing concepts must have mechanisms for refining or removing knowledge of past target concepts. To achieve these effects, one ensemble method for concept drift builds two ensembles and selects the best performing one for subsequent processing (Scholz and Klinkenberg, 2006). Other methods replace poorly performing members of the ensemble (Street and Kim, 2001; Fan, 2004), decrease the effect these members have on the overall prediction (Blum, 1997; Scholz and Klinkenberg, 2006), or both (Kolter and Maloof, 2003; Wang et al., 2003; Kolter and Maloof, 2005).

Blum's (1997) implementation of the weighted majority algorithm (Littlestone and Warmuth, 1994) uses as experts pairs of features coupled with a history of the most recent class labels from the training set appearing with those features. When a new instance arrives, the expert for a given pair of features predicts based on a majority vote of the labels of past observations. The global algorithm predicts based on a weighted-majority vote of the expert predictions and decreases the weight of any expert that predicts incorrectly. Each expert then stores the correct prediction in its history.

Blum (1997) also investigated triples of features as experts and a variant of winnow (Littlestone, 1988) that lets experts abstain if their features are not present in an instance. On a calendar scheduling task, which we describe further in Section 4.3, these methods were able to track a professor's preferences for scheduling meetings across semester boundaries.

The Streaming Ensemble Algorithm (SEA) maintains a fixed-size collection of classifiers, each built from a batch of training examples (Street and Kim, 2001). When a new batch of examples arrives, SEA uses C4.5 (Quinlan, 1993) to build a decision tree. If there is space, SEA adds the new classifier to the ensemble. Otherwise, if the new classifier outperforms a classifier in the ensemble, SEA replaces it with the new one. Performance is measured on the current batch of examples. Overall, SEA predicts based on a majority vote of the predictions of the classifiers in the ensemble. An evaluation using a synthetic data set generated from shifting a hyperplane revealed that the method was able to acquire a series of four changing target concepts with accuracy of about 92%. We discuss this data set further in Section 4.2.

Herbster and Warmuth (1998) consider a setting in which an online learner is trained over several concepts and has access to n experts (which are fixed prediction strategies). They present an algorithm that performs almost as well as the best expert on each concept individually, paying an additional penalty of $\log n$ on each concept. Bousquet and Warmuth (2002) extend this setting to one where the best expert always comes from a smaller pool of $m \ll n$ experts. Here they show that the learner can pay a $\log m$ rather than a $\log n$ penalty on each concept, plus a one-time cost of $\log \binom{n}{m}$ to identify the best m experts.

The Accuracy-weighted Ensemble (AWE) also maintains a fixed-size collection of classifiers built from batches of training examples, but this method weights each classifier based on its performance on the most recent batch (Wang et al., 2003). If there is space in the ensemble, then AWE adds the new weighted classifier. Otherwise, it keeps only the top k weighted classifiers. AWE predicts based on a weighted-majority vote of the predictions of the ensemble's classifiers. The evaluation of this method involved a synthetic data set generated from a rotating hyperplane and a data set for credit card fraud detection. Although the evaluation consisted of many experimental conditions, such as how the problem's dimension affects error rate of the ensemble classifier, it did not include measuring the system's performance over time and over changing target concepts.

Kolter and Maloof (2005) present AddExp, an algorithm for additive expert ensembles. It is similar to dynamic weighted majority, but unlike DWM, AddExp submits to formal analysis due

to differences in their weighting schemes. DWM sets the weight of a new expert to one, whereas AddExp sets a new expert's weight to the total weight of the ensemble times some parameter $\gamma \in (0,1)$. In addition to empirical results, the authors present worst-case bounds for the algorithm's loss and number of mistakes, proving that AddExp performs almost as well as the best-performing expert. They also describe two pruning methods for limiting the number of experts maintained; one is useful in practice, the other gives formal guarantees on the number of experts that AddExp will create.

3. DWM: An Ensemble Method for Concept Drift

Dynamic weighted majority maintains a weighted pool of experts or base learners. It adds and removes experts based on the global algorithm's performance. If the global algorithm makes a mistake, then DWM adds an expert. If an expert makes a mistake, then DWM reduces its weight. If, in spite of multiple training episodes, an expert performs poorly, as indicated by a sufficiently low weight, then DWM removes it from the ensemble. This method is general, and in principle, one could use any online learning algorithm as a base learner. One could also use different types of base learners, although one would also have to implement control policies to determine what base learner to add.

The formal algorithm for DWM appears in Figure 1. The algorithm maintains a set of m experts, E, each with a weight, w_i for $i = 1,\ldots,m$. Input to the algorithm is n training examples, each consisting of a feature vector and a class label. The parameters also include the number of classes (c) and β, a multiplicative factor that DWM uses to decrease an expert's weight when it predicts incorrectly. A typical value for β is 0.5. The parameter θ is a threshold for removing poorly performing experts. If an expert's weight falls below this threshold, then DWM removes it from the ensemble. Finally, the parameter p determines how often DWM creates and removes experts. We found this parameter useful and necessary for large or noisy problems, which we discuss further in Section 4.2. In the following discussion, we assume $p = 1$.

DWM begins by creating an ensemble containing a single learner with a weight of one (lines 1–3 of Figure 1). Initially, this learner could predict a default class, or it could predict using previous experience, background knowledge, or both. DWM then takes a single example (or perhaps a set of examples) from the stream and presents it to the single learner to classify (line 7). If the learner's prediction is wrong (line 8), then DWM decreases the learner's weight by multiplying it by β (line 9). Since there is one expert in the ensemble, its prediction is DWM's global prediction (lines 12 and 24). If DWM's global prediction is incorrect (line 16), then it creates a new learner with a weight of one (lines 17–19). DWM then trains the experts in the ensemble on the new example (line 23). After training, DWM outputs its global prediction (line 24).

When there are multiple learners, DWM obtains a classification from each member of the ensemble (lines 6 and 7). If one's prediction is incorrect, then DWM decreases its weight (lines 8 and 9). Regardless of the correctness of the prediction, DWM uses each learner's prediction and its weight to compute a weighted sum for each class (line 10). The class with the most weight is set as the global prediction (line 12).

Since DWM always decreases the weights of experts, it normalizes the weights by scaling them uniformly so that, after the transformation, the maximum weight is one (line 14). This prevents newly added experts from dominating predictions. DWM also removes poorly performing experts by removing those with a weight less than the threshold θ (line 15), although it will not remove the

Dynamic-Weighted-Majority ($\{\vec{x},y\}_n^1, c, \beta, \theta, p$)

$\{\vec{x},y\}_n^1$: training data, feature vector and class label
$c \in \mathbb{N}^*$: number of classes, $c \geq 2$
β: factor for decreasing weights, $0 \leq \beta < 1$
θ: threshold for deleting experts
p: period between expert removal, creation, and weight update
$\{e,w\}_m^1$: set of experts and their weights
$\Lambda, \lambda \in \{1,\ldots,c\}$: global and local predictions
$\vec{\sigma} \in \mathbb{R}^c$: sum of weighted predictions for each class

```
1.  m ← 1
2.  e_m ← Create-New-Expert()
3.  w_m ← 1
4.  for i ← 1,...,n                          // Loop over examples
5.      σ⃗ ← 0
6.      for j ← 1,...,m                      // Loop over experts
7.          λ ← Classify(e_j, x⃗_i)
8.          if (λ ≠ y_i and i mod p = 0)
9.              w_j ← β w_j
10.         σ_λ ← σ_λ + w_j
11.     end;
12.     Λ ← argmax_j σ_j
13.     if (i mod p = 0)
14.         w ← Normalize-Weights(w)
15.         {e,w} ← Remove-Experts({e,w}, θ)
16.         if (Λ ≠ y_i)
17.             m ← m + 1
18.             e_m ← Create-New-Expert()
19.             w_m ← 1
20.         end;
21.     end;
22.     for j ← 1,...,m
23.         e_j ← Train(e_j, x⃗_i, y_i)
24.     output Λ
    end;
end.
```

Figure 1: Algorithm for dynamic weighted majority (DWM), after Kolter and Maloof (2005).

last expert in the ensemble. As mentioned previously, if the global prediction is incorrect (line 16), DWM adds a new expert to the ensemble with a weight of one (lines 17–19). Finally, after using the new example to train each learner in the ensemble (lines 22 and 23), DWM outputs the global prediction, which is the weighted vote of the expert predictions (line 24).

As mentioned previously, the parameter p lets DWM better cope with many or noisy examples. p defines the period over which DWM will not update learners' weights (line 8) and will not remove or create experts (line 13). During this period, however, DWM still trains the learners (lines 22 and 23).

DWM is a general algorithm for coping with concept drift. One can use any online learning algorithm as the base learner. To date, we have evaluated two such algorithms, naive Bayes and Incremental Tree Inducer, and we describe these versions in the next two sections.

3.1 DWM-NB

DWM-NB uses an incremental version of naive Bayes as the base learner. For this study, we used the implementation from WEKA (Witten and Frank, 2005), which stores for nominal attributes a count for each class and for each attribute value given the class. The count is simply the number of times each class or attribute value appears in the training set, and so the learning element increments the appropriate counts when processing a new example. The performance element uses these counts to compute estimates of the prior probability of each class, $P(C_i)$, and the conditional probability of each attribute value given the class, $P(v_j|C_j)$. It then operates under the assumption that attributes are conditionally independent and uses Bayes' rule to predict the most probable class:

$$C = \operatorname*{argmax}_{C_i} P(C_i) \prod_j P(v_j|C_i) \; .$$

For numeric attributes, it stores the sum of an attribute's values and the sum of the squared values. Given a value, v_j,

$$P(v_j|C_i) = \frac{1}{\sigma_{ij}\sqrt{2\pi}} e^{-(v_j-\mu_{ij})^2/2\sigma_{ij}^2} \; ,$$

where μ_{ij} is the average of the jth attribute's values for the ith class, and σ_{ij} is their standard deviation. The performance element computes these values from the stored sums.

3.2 DWM-ITI

DWM-ITI uses as a base learner an incremental algorithm for inducing decision trees, called Incremental Tree Inducer, or ITI (Utgoff et al., 1997). ITI uses as its concept description a decision tree with only binary tests. (Note that we can represent multi-valued attributes with a binary tree by treating each value as a binary attribute.) A *decision tree* is a rooted tree with internal nodes corresponding to attributes. Edges correspond to attribute values. For numeric attributes, there are two edges corresponding to a *cut-point*: one edge for values above the cut-point, the other for values less than or equal to the cut-point. The external nodes of the decision tree correspond to class labels. To facilitate incremental learning, external nodes also store examples, while the internal nodes of ITI's decision trees store frequency counts for symbolic attributes and a list of observed values for numeric attributes.

Like standard decision tree algorithms, given an observation, the performance element uses the observation's attributes and their values to traverse from the root node to an external node. It predicts the class label stored in the node.

ITI updates a tree by propagating a new example to a leaf node. During the descent, the algorithm updates the information stored at each node—counts or values—and upon reaching a leaf node, determines if the tree should be extended by converting the leaf node to a decision node. A secondary process examines whether the tests at each node are most appropriate, and if not, restructures the tree accordingly.

As one can imagine, for large problems, storing all examples at leaf nodes and restructuring decision trees can be costly, especially when maintaining an ensemble of such trees. While we were judicious when applying DWM-ITI to large problems, as we see in the next section, where we discuss our experimental study, the learner performed well in spite of these potential costs.

4. Empirical Study and Results

In this section, we present experimental results for DWM-NB and DWM-ITI. We conducted five evaluations. The first, most extensive evaluation involved the STAGGER concepts (Schlimmer and Granger, 1986), a standard benchmark for evaluating how learners cope with drifting concepts. In this evaluation, we compared the performance of DWM-NB and DWM-ITI

1. to best- and worst-case base learners,
2. to our implementation of STAGGER (Schlimmer, 1987),
3. to AQ-PM (Maloof and Michalski, 2000), AQ11-PM (Maloof and Michalski, 2004), and AQ11-PM+WAH (Maloof, 2003), and
4. to Blum's (1997) implementation of weighted majority.

To the best of our knowledge, ours are the only results for Blum's algorithm on the STAGGER concepts.

In an effort to determine how our method scales to larger problems involving concept drift, our second evaluation consisted of testing DWM-NB using the SEA concepts (Street and Kim, 2001), a problem recently proposed in the data mining community. For the third evaluation, we applied DWM-NB to the CAP data set, a calendar scheduling task (Mitchell et al., 1994). Blum (1997) used this problem to evaluate weighted majority and winnow. For the fourth, we evaluated DWM-NB on the task of predicting the price of electricity in New South Wales, Australia, between May 1997 and December 1999, a problem originally introduced by Harries (1999).

Finally, although it is clear that DWM should have no advantage over a single learner when acquiring static concepts, for the sake of completeness, we evaluated DWM-NB on twenty-six data sets from the UCI Repository (Asuncion and Newman, 2007). The intent of this evaluation was to show that, on static concepts, DWM performs no worse than a single base learner.

4.1 The STAGGER Concepts

The STAGGER concepts (Schlimmer and Granger, 1986) comprise a standard benchmark for evaluating a learner's performance in the presence of concept drift. Each example consists of three attribute values: *color* $\in$ {*green, blue, red*}, *shape* $\in$ {*triangle, circle, rectangle*}, and *size* $\in$ {*small,*

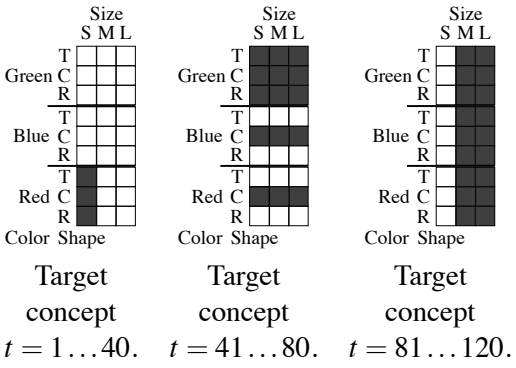


Figure 2: Visualization of the STAGGER Concepts (Maloof and Michalski, 2000). © 2000 Kluwer Academic Publishers. Used with permission.

medium, large$\}$. The presentation of training examples lasts for 120 time steps, and at each time step, the learner receives one example. For the first 40 time steps, the target concept is $color = red \wedge size = small$. During the next 40 time steps, the target concept is $color = green \vee shape = circle$. Finally, during the last 40 time steps, the target concept is $size = medium \vee size = large$. A visualization of these concepts appears in Figure 2.

To evaluate the learner, at each time step, one randomly generates 100 examples of the current target concept, presents these to the performance element, and computes the percent correctly predicted. In our experiments, we repeated this procedure 50 times and averaged the accuracies over these runs. We also computed 95% confidence intervals.

We set the weighted-majority learners—DWM-NB, DWM-ITI, and Blum's (1997) with pairs of features as experts—to halve an expert's weight when it made a mistake (i.e., $\beta = 0.5$). For Blum's weighted majority, each expert maintained a history of only its last prediction (i.e., $k = 1$), under the assumption that this setting would provide the most reactivity to concept drift. For DWM, we set it to update its weights and create and remove experts every time step (i.e., $p = 1$). The algorithm removed experts when their weights fell below 0.01 (i.e., $\theta = 0.01$). Pilot studies indicated that these were the near-optimal settings for p and k; varying β affected performance little; the selected value for θ did not affect accuracy, but did reduce considerably the number of experts.

For the sake of comparison, in addition to these algorithms, we also evaluated naive Bayes, ITI, naive Bayes with perfect forgetting, and ITI with perfect forgetting. The "standard" or "traditional" implementations of naive Bayes and ITI provided a worst-case evaluation, since these systems have not been designed to cope with concept drift and learn from all examples in the stream regardless of changes to the target concept. The implementations with perfect forgetting, which is the same as training the methods on each target concept individually, provided a best-case evaluation, since the systems were never burdened with examples or concept descriptions from previous target concepts.

The left graph of Figure 3 shows the results for DWM-NB on the STAGGER concepts. As expected, naive Bayes with perfect forgetting performed the best on all three concepts, while naive Bayes without forgetting performed the worst. DWM-NB performed almost as well as naive Bayes with perfect forgetting, which converged more quickly to the target concept. Nonetheless, by time

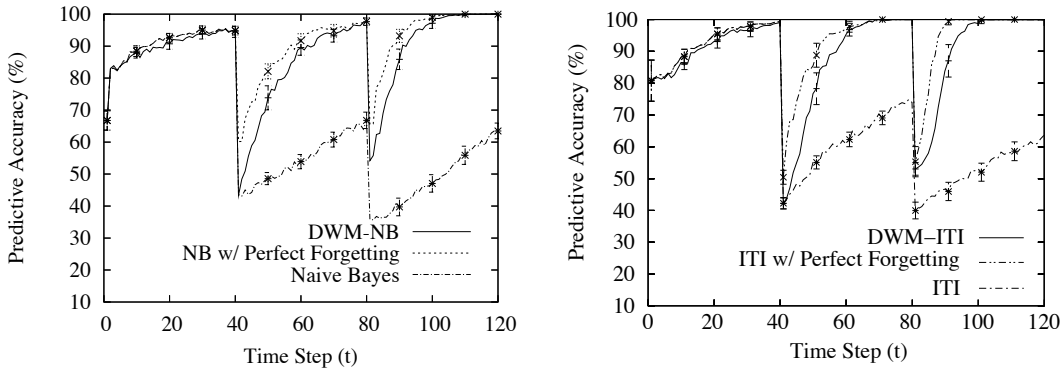


Figure 3: Predictive accuracy with 95% confidence intervals for DWM on the STAGGER concepts (Kolter and Maloof, 2003). Left: DWM-NB. Right: DWM-ITI. © 2003 IEEE Press. Used with permission.

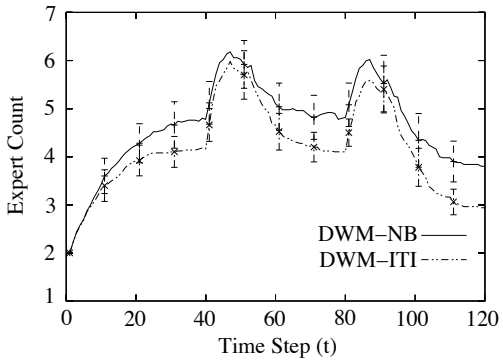


Figure 4: Number of experts maintained with 95% confidence intervals for DWM-NB and DWM-ITI on the STAGGER concepts (Kolter and Maloof, 2003). © 2003 IEEE Press. Used with permission.

step 40 for all three target concepts, DWM-NB performed almost as well as naive Bayes with perfect forgetting.

DWM-ITI performed similarly, as shown in the right graph of Figure 3, achieving accuracies nearly as high as ITI with perfect forgetting. DWM-ITI converged more quickly than did DWM-NB to the second and third target concepts, but if we compare the plots for naive Bayes and ITI with perfect forgetting, we see that ITI converged more quickly to these target concepts than did naive Bayes. Thus, the faster convergence is due to differences in the base learners rather than to something inherent to DWM.

In Figure 4, we present the average number of experts each system maintained over the fifty runs. On average, DWM-ITI maintained fewer experts than did DWM-NB, and we attribute this to the fact that ITI performed better on the individual concepts than did naive Bayes. Since naive Bayes

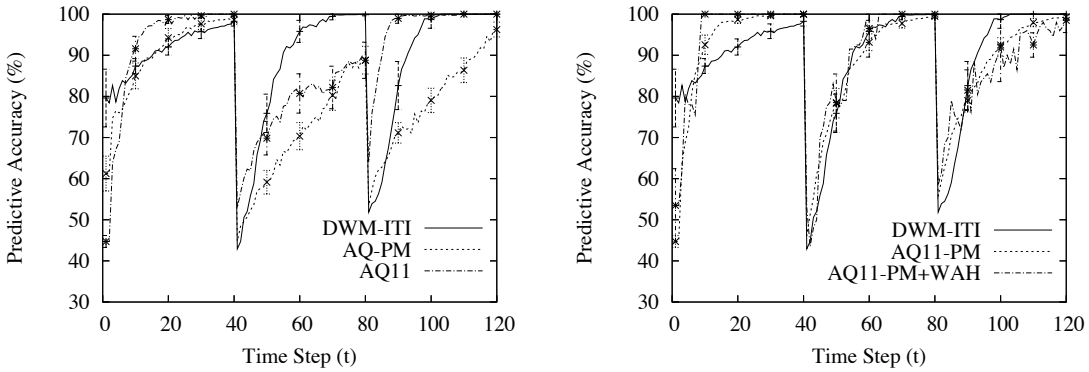


Figure 5: Predictive accuracy with 95% confidence intervals on the STAGGER concepts. Left: DWM-ITI, AQ-PM, and AQ11. Right: DWM-ITI, AQ11-PM, and AQ11-PM+WAH.

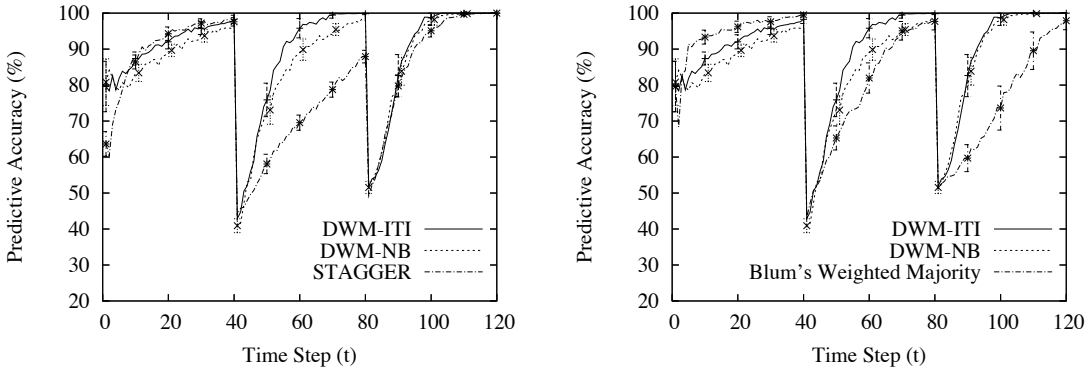


Figure 6: Predictive accuracy with 95% confidence intervals on the STAGGER concepts. Left: DWM-ITI, DWM-NB, and STAGGER. Right: DWM-ITI, DWM-NB, and Blum's weighted majority (Kolter and Maloof, 2003). © 2003 IEEE Press. Used with permission.

made more mistakes than did ITI, DWM-NB created more experts than did DWM-ITI. We can also see in the figure that the rates of removing experts were roughly the same for both learners.

The left graph of Figure 5 compares the performance of DWM-ITI to that of AQ-PM (Maloof and Michalski, 2000) and AQ11 (Michalski and Larson, 1983). DWM-ITI and AQ-PM performed similarly on the first target concept, but DWM-ITI significantly outperformed AQ-PM on the second and third concepts, again, in terms of asymptote and slope. AQ11, although not designed to cope with concept drift, outperformed DWM-ITI in terms of asymptote on the first concept and in terms of slope on the third, but on the second concept, performed significantly worse than did DWM-ITI.

The right graph of Figure 5 compares the performance of DWM-ITI to that of AQ11-PM (Maloof and Michalski, 2004) and AQ11-PM+WAH (Maloof, 2003). DWM-ITI did not perform as well as these other learners on the first target concept, performed comparably on the second, and converged more quickly on the third.

The left graph of Figure 6 compares the performance of our learners to that of our implementation of STAGGER (Schlimmer, 1987). Comparing to both DWM learners, STAGGER's performance was only slightly better on the first target concept, notably worse on the second, and comparable on the third.

Finally, the right diagram of Figure 6 shows the results from the experiment involving Blum's (1997) implementation of weighted majority, the implementation that uses pairs of features as experts. This learner outperformed DWM-NB and DWM-ITI on the first target concept, performed slightly worse on the second, and performed considerably worse on the third.

With respect to complexity of the experts themselves, the STAGGER concepts consist of three attributes, each taking one of three possible values. Therefore, this implementation of weighted majority maintained 27 experts throughout the presentation of examples, as compared to the maximum of six that DWM-NB maintained. Granted, pairs of features and their recent predictions are much simpler than the decision trees that ITI produced, but naive Bayes was quite efficient, maintaining twenty-one integers for each expert. There were occasions when Blum's weighted majority used less memory than did DWM-NB, but we anticipate that using more sophisticated classifiers, such as naive Bayes, instead of all combinations of pairs of features, will lead to scalable algorithms.

We focus our analysis on DWM-ITI, since it performed better than did DWM-NB on this problem. Researchers have built several systems for coping with concept drift and have evaluated many of them on the STAGGER concepts. FLORA2 (Widmer and Kubat, 1996) is a prime example, and on the first target concept, DWM-ITI did not perform as well as did FLORA2. However, on the second and third target concepts, DWM-ITI performed notably better than did FLORA2, not only in terms of asymptote, but also in terms of slope.

DWM-ITI outperformed AQ-PM (Maloof and Michalski, 2000), which had difficulty acquiring the second and third concepts (see Figure 5). AQ-PM maintains examples over a fixed window of time and, at each time step, relearns concepts from these and new examples. In this experiment, when the concept changed from the first to the second, the examples of the first concept remaining in this window prevented AQ-PM from adjusting quickly to the second. Decreasing the size of this window improved accuracy on the second concept, but negatively affected performance on the third.

Compared to DWM-ITI, AQ11 performed poorly on the second concept, but performed exceptionally well on the third concept. Our analysis suggests that its poor performance on the second concept was because of AQ11's expressive language for representing concepts. Notice that all of its rules achieved 100% on the first concept (see Figure 5, left). However, AQ11 produced twelve different rules for the first concept over the fifty runs. Some rules were similar, but others were quite different. AQ11 modifies its rules using only new examples, and in some cases, was able to transform rules for the first concept into rules with high accuracy on the second. Indeed, when we trained AQ11 only on examples of the second concept, it achieved 97%. However, in some trials, AQ11 was unable to make the required transformation and failed to learn adequately the second concept. We contend that if AQ11 produced simpler rules, provided that they were the right ones, it would have performed much better on the second concept. Indeed, it is well known that simpler models often perform better than do more complex ones.

AQ11-PM (Maloof and Michalski, 2004) and AQ11-PM+WAH (Maloof, 2003) performed comparably to DWM-ITI (see Figure 5), although DWM-ITI did appear to outperform the AQ learners on the third concept, especially in the later time steps. Although tangential, it is interesting to contrast the performances of AQ11 and AQ11-PM. The only difference between these two learners is that AQ11-PM maintains a fixed window of thirty examples that have appeared on the boundaries of

concept descriptions. When new examples arrive, AQ11-PM learns incrementally from these new examples and those present in the window. Although the presence of these examples may have decreased AQ11's (i.e., AQ11-PM's) performance on the third concept, their presence notably improved its performance on the second concept. If our hypothesis is correct—that AQ11 would have performed better had it produced simpler models—the examples held in the window may have constrained AQ11 such that it produced simpler models. Put another way, the examples in the window reduced the instability of the learner. We found this discovery intriguing and plan to investigate it in future work.

STAGGER (Schlimmer and Granger, 1986) performed comparably to DWM-ITI on the first and third target concepts, but did not perform as well as DWM-ITI on the second target concept. (See Figure 6, left.) Generally, acquiring the second concept after learning the first is the hardest task, as the second concept is almost a reversal of the first; the two concepts share only one positive example. Acquiring the third concept after acquiring the second is easier because the two concepts share a greater number of positive examples. Performing well on the second concept therefore requires quickly disposing of knowledge and perhaps examples of the first target concept. A learning method, such as STAGGER, that only refines concept descriptions will have more difficulty responding to concept drift, as compared to an ensemble method, such as DWM, that both refines existing concept descriptions and creates new ones.

Comparing DWM-ITI to Blum's weighted majority, DWM-ITI outperformed it on the STAGGER concepts. However, our analysis suggests that the difference in performance is due to the experts, rather than to the global algorithms. Recall that Blum's weighted majority uses as experts pairs of features with a brief history of past predictions. For the STAGGER concepts, pairs of features are useful for acquiring the first target concept (see Figure 2), which is conjunctive. Indeed, pairs of features are two-term conjunctions. However, the second and third concepts are disjunctive, and these are difficult to represent using only weighted pairs of features. As a result, on the STAGGER concepts, Blum's weighted majority did not perform as well as DWM.

Overall, we concluded that DWM-ITI outperformed these other learners in terms of accuracy, both in slope and asymptote. In reaching this conclusion, we gave little weight to performance on the first concept, since most learners can acquire it easily and doing so requires no mechanisms for coping with drift. On the second and third concepts, with the exception of AQ11, DWM-ITI performed as well or better than did the other learners. And while AQ11 outperformed DWM-ITI in terms of slope on the third concept, this does not mitigate AQ11's poor performance on the second.

We attribute the performance of DWM-ITI to the training of multiple experts on different sequences of examples. (Weighting experts also contributed, and we will discuss this topic in detail shortly.) Assume a learner incrementally modifies its concept descriptions as new examples arrive. When the target concept changes, if the new one is disjoint, then the best policy to learn new descriptions, rather than modifying existing ones. This makes intuitive sense, since the learner does not have to first unlearn the old concept, and results from this and other empirical studies support this assertion (Maloof and Michalski, 2000, 2004). Unfortunately, target concepts are not always disjoint, it is difficult to determine precisely when concepts change, and it is challenging to identify which concept descriptions (or parts of concept descriptions) apply to new target concepts. DWM addresses these problems by incrementally updating existing descriptions and, in parallel, by learning new concept descriptions.

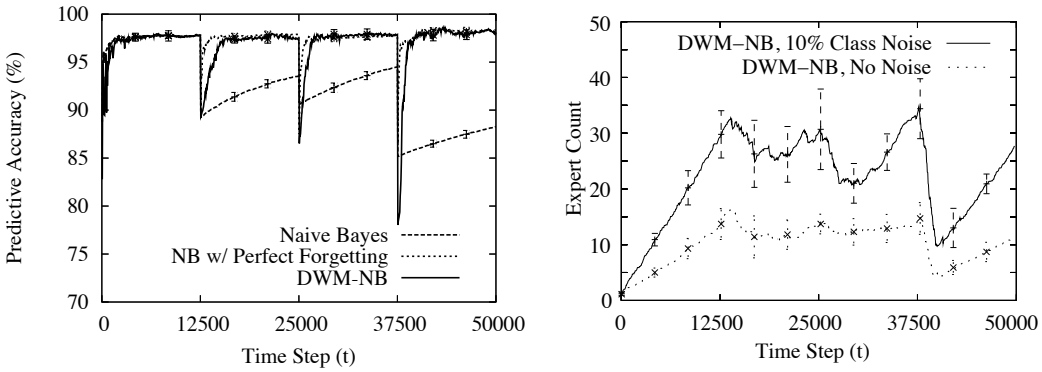


Figure 7: Performance with 95% confidence intervals of DWM-NB on the SEA concepts with 10% class noise (Kolter and Maloof, 2003). Left: Predictive accuracy. Right: Number of experts maintained. © 2003 IEEE Press. Used with permission.

4.2 Performance on a Larger Data Set with Concept Drift

To determine how well DWM-NB performs on larger problems involving concept drift, we evaluated it using a synthetic problem recently proposed in the data mining community (Street and Kim, 2001). This problem, which we call the "SEA concepts", consists of three attributes, $x_i \in \mathbb{R}$ such that $0.0 \leq x_i \leq 10.0$. The target concept is $x_1 + x_2 \leq b$, where $b \in \{7, 8, 9, 9.5\}$. Thus, x_3 is an irrelevant attribute.

The presentation of training examples lasts for $50,000$ time steps. For the first fourth (i.e., $12,500$ time steps), the target concept is with $b = 8$. For the second, $b = 9$; the third, $b = 7$; and the fourth, $b = 9.5$. For each of these four periods, we randomly generated a training set consisting of $12,500$ examples. In one experimental condition, we added 10% class noise; in another, we did not, and this latter condition served as our control. We also randomly generated 2,500 examples for testing. At each time step, we presented each method with one example, tested the resulting concept descriptions using the examples in the test set, and computed the percent correct. We repeated this procedure ten times, averaging accuracy over these runs. We also computed 95% confidence intervals.

On this problem, we evaluated DWM-NB, naive Bayes, and naive Bayes with perfect forgetting. We set DWM-NB to halve the expert weights (i.e., $\beta = 0.5$) and to update these weights and to create and remove experts every fifty time steps (i.e., $p = 50$). We set the algorithm to remove experts with weights less than 0.01 (i.e., $\theta = 0.01$).

In the left graph of Figure 7, we see the predictive accuracies for DWM-NB, naive Bayes, and naive Bayes with perfect forgetting on the SEA concepts with 10% class noise. As with the STAGGER concepts, naive Bayes performed the worst, since it had no direct method of removing outdated concept descriptions. Naive Bayes with perfect forgetting performed the best and represents the best possible performance for this implementation on this problem. Crucially, DWM-NB achieved accuracies nearly equal to those achieved by naive Bayes with perfect forgetting.

Finally, the right graph of Figure 7 shows the number of experts that DWM-NB maintained during the runs with and without class noise. Recall that DWM creates an expert when it misclassifies an

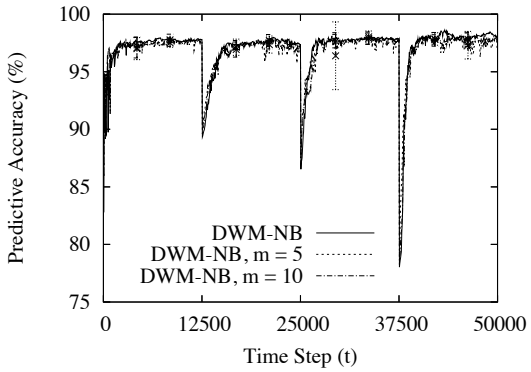


Figure 8: Predictive accuracy with 95% confidence intervals for DWM-NB on the SEA concepts with 10% class noise. The number of experts was capped at five and at ten and compared to DWM-NB with no limit on the number of experts created.

example. In the noisy condition, since 10% of the examples had been relabeled, DWM-NB made more mistakes and therefore created more experts than it did in the condition without noise.

As mentioned previously, DWM has the potential for creating a large number of experts, especially in noisy domains, since it creates a new expert every time the global prediction is incorrect. (The parameter p can help mitigate this effect.) A large number of experts obviously impacts memory utilization, and, depending on the complexity of the base learners, could also affect learning and performance time, since DWM trains and queries each expert in the ensemble when a new example arrives. An obvious scheme when resources are constrained is to limit the number of experts that DWM maintains.

To investigate this strategy's effect on DWM's performance, we produced a version of the algorithm that always keeps the k best performing experts. That is, after reaching the point where there are k experts in the ensemble, when DWM adds a new expert, it removes the expert with the lowest weight. In this version of DWM, we set the threshold for removing experts to zero, which guaranteed that experts were removed only if they were the weakest member of the ensemble of k experts.

We ran this modified version on the SEA concepts, capping the number of experts at five and at ten. We present these results in Figure 8. As one can see, for this problem, restricting the number of experts did not appreciably impact performance. Indeed, DWM-NB with five experts performed almost as well as the original algorithm that placed no limit on the number of experts created.

Comparing our results for the SEA concepts to those reported by Street and Kim (2001), DWM-NB outperformed SEA on all four target concepts. On the first concept, performance was similar in terms of slope, but not in terms of asymptote, and on subsequent concepts, DWM-NB converged more quickly to the target concepts and did so with higher accuracy. For example, on concepts 2–4, just prior to the point at which concepts changed, SEA achieved accuracies in the 90–94% range, while DWM-NB's were in the 96–98% range.

We suspect this is most likely due to SEA's unweighted voting procedure and its method of creating and removing new classifiers. Recall that the method trains a new classifier on a fixed number of examples. If the new classifier improves the global performance of the ensemble, then it

is added, provided the ensemble does not contain a maximum number of classifiers; otherwise, SEA replaces a poorly performing classifier in the ensemble with the new classifier.

However, if every classifier in the ensemble has been trained on a given target concept, and the concept changes to one that is disjoint, then SEA must replace at least half of the classifiers in the ensemble before accuracy on the new target concept will surpass that on the old. For instance, if the ensemble consists of 20 classifiers, and each learns from a fixed set of 500 examples, then it would take at least $5,000$ additional training examples before the ensemble contained a majority number of classifiers trained on the new concept.

In contrast, DWM under similar circumstances requires only $1,500$ examples. Assume $p = 500$, the ensemble consists of 20 fully trained classifiers, all with a weight of one, and the new concept is disjoint from the previous one. When an example of this new concept arrives, all 20 classifiers will predict incorrectly, DWM will reduce their weights to 0.5—since the global prediction is also incorrect—and it will create a new classifier with a weight of one. It will then process the next 499 examples.

Assume another example arrives. The original 20 experts will again misclassify the example, and the new expert will predict correctly. Since the weighted prediction of the twenty will be greater than that of the one, the global prediction will be incorrect, the algorithm will reduce the weights of the twenty to 0.25, and it will again create a new expert with a weight of one. DWM will again process 499 examples.

Assume a similar sequence of events occurs: another example arrives, the original twenty misclassify it, and the two new ones predict correctly. The weighted-majority vote of the original twenty will still be greater than that of the new experts (i.e., $20(0.25) > 2(1)$), so DWM will decrease the weight of the original twenty to 0.125, create a new expert, and process the next 499 examples. However, at this point, the three new classifiers trained on the target concept will be able to overrule the predictions of the original twenty, since $3(1) > 20(0.125)$. Crucially, DWM will reach this state after processing only $1,500$ examples.

Granted, this analysis of SEA and DWM does not take into account the convergence of the base learners, and as such, it is a best-case analysis. The actual number of examples required may be greater for both to converge to a new target concept, but the relative proportion of examples should be similar. This analysis also holds if we assume that DWM replaces experts, rather than creating new ones. Generally, ensemble methods with weighting mechanisms, like those present in DWM, will converge more quickly to target concepts (i.e., require fewer examples) than will methods that replace unweighted learners in the ensemble.

Regarding the number of experts that DWM maintained, we used a simple heuristic that added a new expert whenever the global prediction was incorrect, which intuitively, should be problematic for noisy domains. However, on the SEA concepts, while DWM-NB maintained as many as 40 experts at, say, time step $37,500$, it maintained only 22 experts on average over the 10 runs, which is similar to the 20–25 that SEA reportedly stored (Street and Kim, 2001).

If the number of experts were to reach impractical levels, then DWM could simply stop creating experts after obtaining acceptable accuracy; training would continue. Plus, we could easily distribute the training of experts to processors of a network or of a course-grained parallel machine. And as the results pictured in Figure 8 demonstrate, we can limit the number of experts that DWM maintains with potentially little effect on accuracy.

One could argue that better performance of DWM-NB is due to differences between the base learners. SEA was an ensemble of C4.5 classifiers (Quinlan, 1993), while DWM-NB, of course, used

naive Bayes as the base learner. We refuted this hypothesis by running both base learners on each of the four target concepts. Both achieved comparable accuracies on each concept. For example, on the first target concept, C4.5 achieved 99% accuracy and naive Bayes achieved 98%. Since these learners performed similarly, we concluded that our positive results on this problem were due not to the superiority of the base learner, but to the mechanisms that create, weight, and remove experts.

We did not evaluate DWM-ITI on the SEA concepts, since ITI maintains all training examples and all observed values for continuous attributes, and this would have led to impractical memory requirements. However, this does not exclude the possibility of using DWM on large data sets with a decision-tree learner as the base algorithm. For instance, we could use ITI, but implement schemes to index stored training examples, which would reduce memory requirements. We could also use a decision-tree learner that does not store examples, such as ID4 (Schlimmer and Fisher, 1986) or VFDT (Domingos and Hulten, 2000).

4.3 Calendar Scheduling Domain

The Calendar Apprentice (CAP) predicts user preferences for scheduling meetings in an academic institution (Mitchell et al., 1994). The task is to predict a user's preference for a meeting's location, duration, starting time, and day of the week. The data set consists of 34 features—such as the type of meeting, the purpose of the meeting, the type of attendees, and whether the meeting occurs during lunchtime—with intersecting subsets of these features for each prediction task. There are 12 features for location, 11 for duration, 15 for start time, and 16 for day of week. Although data are available for two users, we used the 1,685 examples of the preferences for User 1 (Tom Mitchell).

For this experiment, we evaluated naive Bayes and DWM-NB. However, unlike previous designs, in which we tested the resulting classifiers using a test set, in this design, we measured performance on the next example (i.e., the next meeting to be scheduled). For this application, when another user proposes a meeting, the Calendar Apprentice predicts, say, the meeting's location, and the user either accepts or rejects the recommendation. The learner then uses this feedback to update its model of the user's preferences.

Table 1 shows the average performance of naive Bayes and DWM-NB for the calendar scheduling task. With the exception of predicting the preferred day of week for meetings, DWM-NB outperformed naive Bayes. Over the four prediction tasks, DWM-NB outperformed naive Bayes. As one can see, increasing the parameter p, which governs how often DWM updates its experts, generally decreased DWM's performance. Figure 9 shows accuracy versus the number of examples for the two learners on the four prediction tasks. According to Blum (1997), the sharp decreases in accuracy roughly correspond to the boundaries of semesters.

On this problem, Blum (1997) reported that, over the four prediction tasks, the CAP system averaged 53% and weighted majority with pairs of features averaged 57%. (Other algorithms in this study, such as winnow, performed even better.) DWM-NB averaged about 55%, which was better than the original CAP system, which used a decision tree to predict, but it was not better than Blum's weighted majority.

However, our analysis of this data set suggests that, again, these differences in performance were due to the base learners rather than to the global algorithms. Because of their complexity, we were not able to analyze the CAP concepts in the same manner that we analyzed the STAGGER concepts. We were nonetheless able to determine that pairs (and triples) of features were better suited to the prediction tasks than were all of the features. It is well known that naive Bayes can be sensitive to

Prediction Task	Naive Bayes	DWM-NB $p=1$	DWM-NB $p=10$	DWM-NB $p=50$
Location	62.14	65.69	65.16	62.43
Duration	62.37	64.44	64.62	63.03
Start Time	32.40	38.10	37.39	34.96
Day of Week	51.22	51.16	49.13	51.34
Average	52.03	54.85	54.07	52.84

Table 1: Percent correct of naive Bayes and DWM-NB on the CAP data set, using 1,685 examples for User 1. The variance of 1,685 Bernoulli trials is 0.0144%.

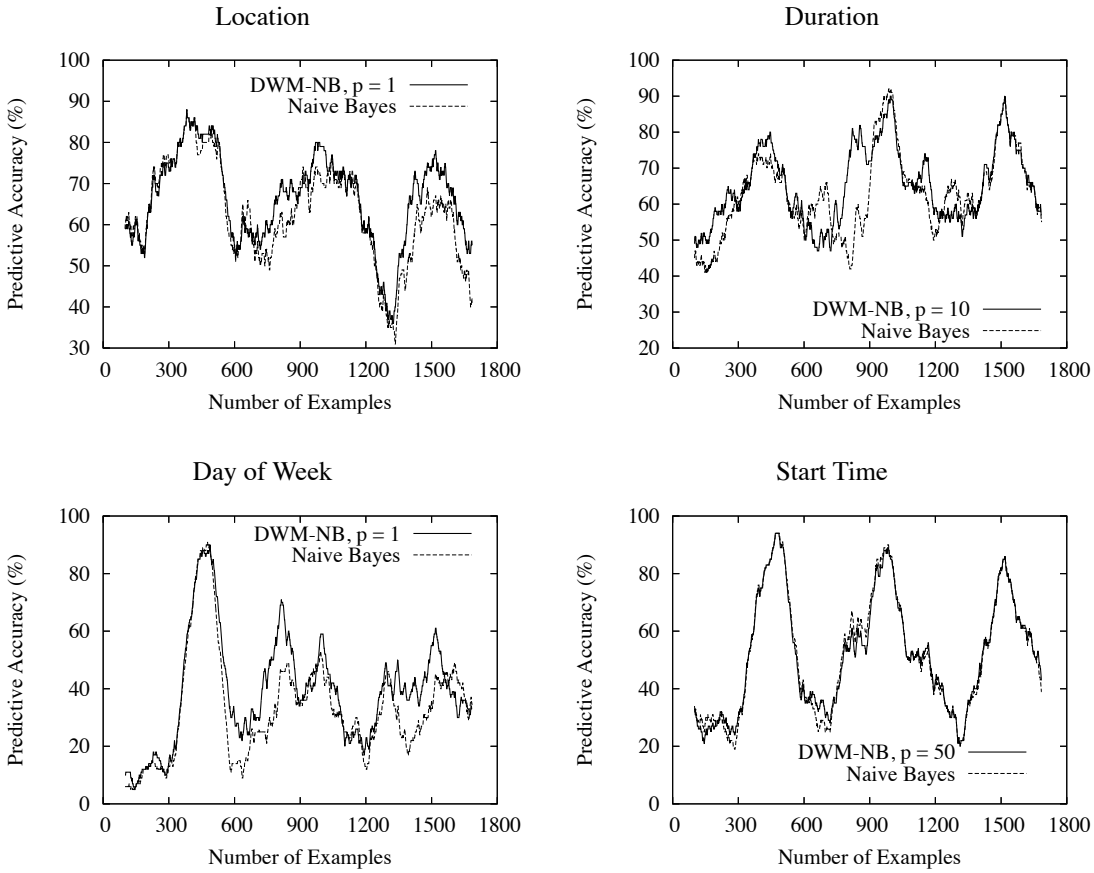


Figure 9: Accuracy on four calendar scheduling tasks for DWM-NB and naive Bayes. Measures are averages of the previous 100 predictions.

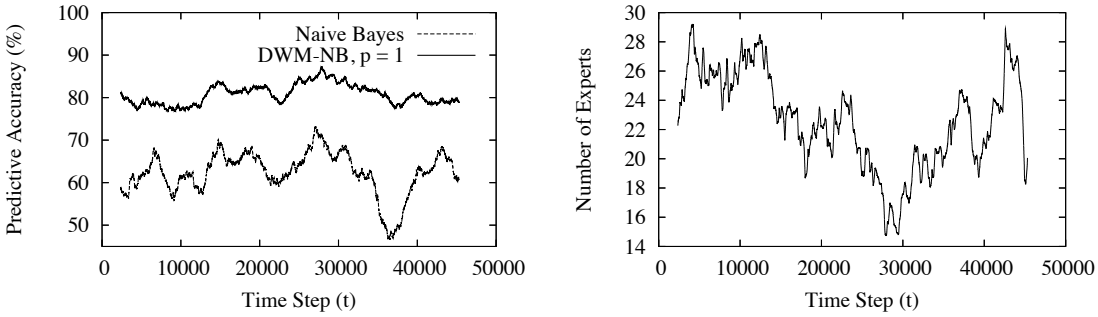


Figure 10: Performance of DWM-NB on the electricity pricing task. Measures are averages of the previous 2,352 predictions. Left: Predictive accuracy. Right: Number of experts maintained.

conditionally-dependent attributes, and we suspect that this is why Blum's (1997) implementation of weighted majority outperformed DWM-NB on this problem.

Since Blum's implementation forms concept descriptions consisting of weighted pairs of attribute values, we reasoned that conditionally-dependent attributes would have less affect on its performance than they would on naive Bayes'. Indeed, experts that predict based on pairs of attribute values should benefit from conditionally-dependent attributes. Furthermore, since the weighting scheme identifies sets of predictive pairs of attribute values, it should do so regardless of conditional dependence among those attributes.

4.4 Electricity Pricing Domain

As a second evaluation on a real-world problem, we selected the domain of electricity pricing (Harries, 1999; Gama et al., 2004). Harries (1999) obtained this data set from TransGrid, the electricity supplier in New South Wales, Australia. It consists of 45,312 instances collected at 30-minute intervals between 7 May 1996 and 5 December 1998.[1] Each instance consists of five attributes and a class label of either `up` or `down`. There are two attributes for time: day of week, which is an integer in $[1,7]$, and period of day, which is an integer in $[1,48]$ (since there are 48 thirty-minute periods in one day). The remaining three attributes are numeric and measure current demand: the demand in New South Wales, the demand in Victoria, and the amount of electricity scheduled for transfer between the two states. The task is to use the attribute values to predict whether the price of electricity will go up or down.

Since predicting the price of electricity is an online task, we processed the examples in temporal order—the order they appeared in the data set. For each example, we first obtained predictions from DWM-NB and from naive Bayes, and then trained each learner on the example. (This is the same experimental design we followed for the calendar scheduling task.) As before, we set DWM to halve expert weights ($\beta = 0.5$), to update each time step ($p = 1$), and to remove an expert if its weight falls below 0.01 ($\theta = 0.01$).

Overall, naive Bayes averaged 62.32% and DWM-NB averaged 80.75%, maintaining an average of 22 experts. For reference, Harries (1999) used an online version of C4.5 (Quinlan, 1993) that

1. Our data set corresponds to the Elec2-3 data set (Harries, 1999).

built a decision tree from examples in a sliding window and then classified examples in the following week, reporting accuracies for various window sizes between 66% and 67.7%.

In Figure 10, we present performance curves for accuracy and for the number of experts that DWM maintained. To produce these curves, we averaged the raw performance metrics over a one-week period (i.e., 2,352 observations).

As a data set derived from real-world phenomenon, we cannot know definitively if or when concept drift occurred. Nonetheless, DWM does appear to have been more robust to change present in the samples than was naive Bayes. One example is the period surrounding time step 10,000. Naive Bayes' predictive accuracy fluctuates during this period, but DWM's remains nearly constant, on average.

Most illustrative is naive Bayes' sudden drop of roughly 12% in accuracy between time steps 35,000 and 37,000. DWM's accuracy decreased slightly and steadily during this period, but there was no comparable sudden decrease in performance.

The number of experts that DWM maintained fluctuated considerably between 1 and 90, but with an overall average of 22 experts. In terms of the weekly average, shown in the right graph of Figure 10, the average size of the ensemble never exceeded 29 experts, which we found encouraging for a problem with 45,312 examples.

4.5 Evaluation on Static Concepts

Finally, we evaluated DWM-NB on 26 data sets from the UCI Repository (Asuncion and Newman, 2007). It is clear that, for static concepts, there is nothing inherent to the DWM algorithm that would make it more advantageous than a single learner. However, it is important to establish that DWM performs *no worse* than a single learner. Therefore, we selected data sets that varied in the number of classes, the number of examples, the types of attributes, and the number of missing values.

For each data set, we randomly selected 10% of the examples for testing.[2] We presented each example of the remaining 90% to naive Bayes and to DWM-NB with three settings of the parameter p: 1, 10, and 50. After each presentation, we evaluated the resulting concept descriptions on the examples in the testing set. We repeated this procedure ten times, averaging percent correct over these runs and computing 95% confidence intervals.

The results, presented in Appendix A, demonstrate that DWM-NB performed no worse than naive Bayes, the single base learner. As one can see in Table 2, naive Bayes outperformed DWM-NB on most of the tasks (16 of 26), but many of these differences are within 0.5%. For the tasks on which DWM-NB outperformed naive Bayes, many of the differences in performance were also within this range. Overall, the average difference in performance is +0.35%—in DWM's favor—and so we concluded that, on these data sets, DWM-NB performed no worse than a single instance of naive Bayes.

In Figures 11–13, we present the learning curves for this experiment. For legibility, we did not include the curves for DWM-NB with $p = 1$. Limiting DWM-NB to ten experts reduced performance only slightly, and we omitted these results for brevity.

2. For data sets with predetermined training and testing sets, we used all available examples to create a single file of examples and proceeded as described.

5. Concluding Remarks

Tracking concept drift is important for many applications. Clearly, with learners for drifting concepts, there is a balance between learning that is completely reactive (e.g., predicting based only on the last example) and learning that is completely unreactive (e.g., predicting based on all encountered examples). If a learner knew when concepts had changed, it could discard its old descriptions and start learning anew. However, this may not always lead to optimum performance on a task because there may be knowledge of the old concept useful for acquiring the new concept. If the learner could appropriately leverage its relevant old knowledge, then performance on the new concept may be better, if not in terms of asymptote, then perhaps in terms of slope.

In this paper, we presented an ensemble method based on the weighted majority algorithm (Littlestone and Warmuth, 1994). Our method, dynamic weighted majority, creates and removes base algorithms in response to changes in performance, which makes it well suited for problems involving concept drift. We described two implementations of DWM, one with naive Bayes as the base learner, the other with ITI (Utgoff et al., 1997). On the problems we considered, a weighted ensemble of learners with mechanisms to add and remove experts in response to changes in performance provided a better response to concept drift than did other learners, especially those that relied on only incremental learning (i.e., STAGGER and AQ11), on the maintenance of previously encountered examples (i.e., FLORA2 and the AQ-PM systems), or on an ensemble of unweighted learners (i.e., SEA).

Using the STAGGER concepts, we evaluated DWM-NB and DWM-ITI our implementation of STAGGER, Blum's implementation of weighted majority, and four rule learners based on the AQ algorithm. To determine performance on a larger problem, we evaluated DWM-NB on the SEA concepts. Results on these problems, when compared to other methods, suggest that DWM maintained a comparable number of experts, but achieved higher predictive accuracies and converged to those accuracies more quickly. Indeed, to the best of our knowledge, these are the best overall results reported for these problems.

DWM, on a calendar scheduling task, outperformed the CAP system and performed comparably to Blum's weighted majority. On an electricity pricing domain, DWM outperformed a single base learner and an online version of C4.5 (Harries, 1999). Finally, on several problems with static concepts, DWM performed as well as a single base learner.

In future work, we plan to investigate more sophisticated heuristics for creating new experts: Rather than creating an expert when the global prediction is wrong, perhaps DWM should take into account the expert's age or its history of predictions. We would also like to investigate another decision-tree learner as the base algorithm, one that does not maintain encountered examples and that does not periodically restructure its tree; VFDT (Domingos and Hulten, 2000) is a likely candidate.

Although removing experts of low weight yielded positive results for the problems we considered in this study, it would be beneficial to investigate mechanisms for explicitly handling noise, such as those present in IB3 (Aha et al., 1991), or for determining when examples are likely to be from a different target concept, such as those based on the Hoeffding (1963) bounds present in VFDT (Domingos and Hulten, 2000) and CVFDT (Hulten et al., 2001).

Finally, the most intriguing prospect for future work is to explore the relationship between a learner's stability and its skill at coping with drifting concepts. Equally interesting is how certain mechanisms, such as learning from previously encountered examples in addition to new ones, af-

fects a learner's stability and its ability to cope in non-stationary environments. We anticipate that these investigations will lead to general, robust, and scalable ensemble methods for tracking concept drift.

Acknowledgments

The authors thank Dale Schuurmans, William Headden, and the anonymous reviewers for helpful comments on earlier drafts of the manuscript. We thank Avrim Blum and Paul Utgoff for releasing their respective systems to the research community. We also thank João Gama for providing the data set for electricity pricing, and Michael Harries for its original distribution. This research was conducted in the Department of Computer Science at Georgetown University. The work was supported in part by the National Institute of Standards and Technology under grant 60NANB2D0013 and by GUROP, the Georgetown Undergraduate Research Opportunities Program. The authors are listed in alphabetical order.

Appendix A. Performance of DWM-NB on Selected UCI Data Sets

To establish DWM's performance on static concepts, we applied DWM-NB to twenty-six data sets from the UCI Repository (Asuncion and Newman, 2007). For each data set, we randomly selected 10% of the examples as a testing set, and used the remaining 90% for training. We presented a single training example to naive Bayes and to DWM-NB, and evaluated the resulting classifiers on the examples in the testing set, measuring percent correct. We proceeded in this manner until processing all of the training examples. We then repeated this procedure nine more times for a total of ten applications, after which we averaged the measures of performance and computed 95% confidence intervals. Table 2 lists these results. Although DWM-NB did not outperform naive Bayes, as expected, it also performed no worse. Over these twenty-six data sets, the average difference in performance is +0.35%. Figures 11–13 contain learning curves for these data sets.

		DWM-NB			
Data Set	NB	$p = 1$	$p = 10$	$p = 50$	$\Delta\%$
balance-scale	**90.27±0.13**	90.19±0.18	89.75±0.43	89.82±0.36	−0.08
breast-cancer	72.60±0.37	70.62±1.42	72.60±0.62	**72.88±0.30**	+0.28
breast-w	**95.99±0.04**	95.98±0.07	95.93±0.25	95.98±0.05	−0.01
colic	78.33±0.24	76.24±3.06	**79.29±1.05**	79.23±0.69	+0.96
credit-a	77.73±0.17	76.52±3.63	**78.70±0.67**	77.99±0.24	+0.97
credit-g	**74.97±0.30**	67.56±2.46	73.62±0.96	74.18±0.41	−0.79
diabetes	75.37±0.21	69.94±2.34	74.71±0.84	**75.51±0.38**	+0.14
glass	46.18±1.36	45.56±3.46	47.05±2.51	**47.97±0.85**	+1.79
heart-h	**83.89±0.42**	82.97±1.06	83.17±0.74	83.51±0.62	−0.38
heart-statlog	**84.11±0.46**	82.89±1.20	83.63±0.57	**84.11±0.48**	0.00
hepatitis	83.07±0.55	82.60±2.33	**84.07±0.76**	82.68±0.75	+1.00
hypothyroid	95.37±0.06	**95.64±0.33**	95.62±0.23	95.38±0.28	+0.27
ionosphere	**88.29±0.41**	87.60±3.24	88.26±0.72	87.81±0.62	−0.03
kr-vs-kp	**87.78±0.09**	78.50±7.69	86.65±1.03	87.15±0.41	−0.63
labor	**90.67±0.61**	89.40±1.33	89.27±0.96	90.33±1.00	−0.34
lymph	**77.34±1.87**	74.81±1.87	75.50±2.18	76.74±2.65	−0.60
mushroom	**95.75±0.03**	94.81±1.18	95.00±0.33	95.25±0.19	−0.50
primary-tumor	**48.62±0.63**	36.76±6.45	44.25±1.94	46.81±1.52	−1.81
segment	79.72±0.13	73.94±11.46	80.32±0.60	**80.38±0.37**	+0.66
sick	92.71±0.07	**94.44±0.32**	93.16±0.69	92.32±0.84	+1.73
soybean	**92.78±0.15**	92.76±0.15	92.50±0.27	90.03±2.14	−0.02
splice	**95.37±0.06**	**95.37±0.06**	95.31±0.14	95.23±0.24	0.00
vehicle	45.51±0.28	42.51±2.87	**46.33±1.82**	44.44±0.81	+0.82
vote	**90.09±0.11**	89.83±0.25	89.88±0.36	89.91±0.26	−0.18
vowel	**63.04±0.81**	49.07±10.16	58.60±3.05	61.01±1.92	−2.03
zoo	**88.27±0.20**	87.97±0.36	87.07±0.90	87.56±0.86	−0.30

Table 2: Performance of naive Bayes (NB) and DWM-NB on twenty-six selected UCI data sets. DWM-NB created experts every p examples. Measures are percent correct with 95% confidence intervals. Maximum accuracies are typeset in boldface; however, note that many of the differences are not statistically significant. The final column shows the difference in percentage between naive Bayes and the best performing DWM classifier.

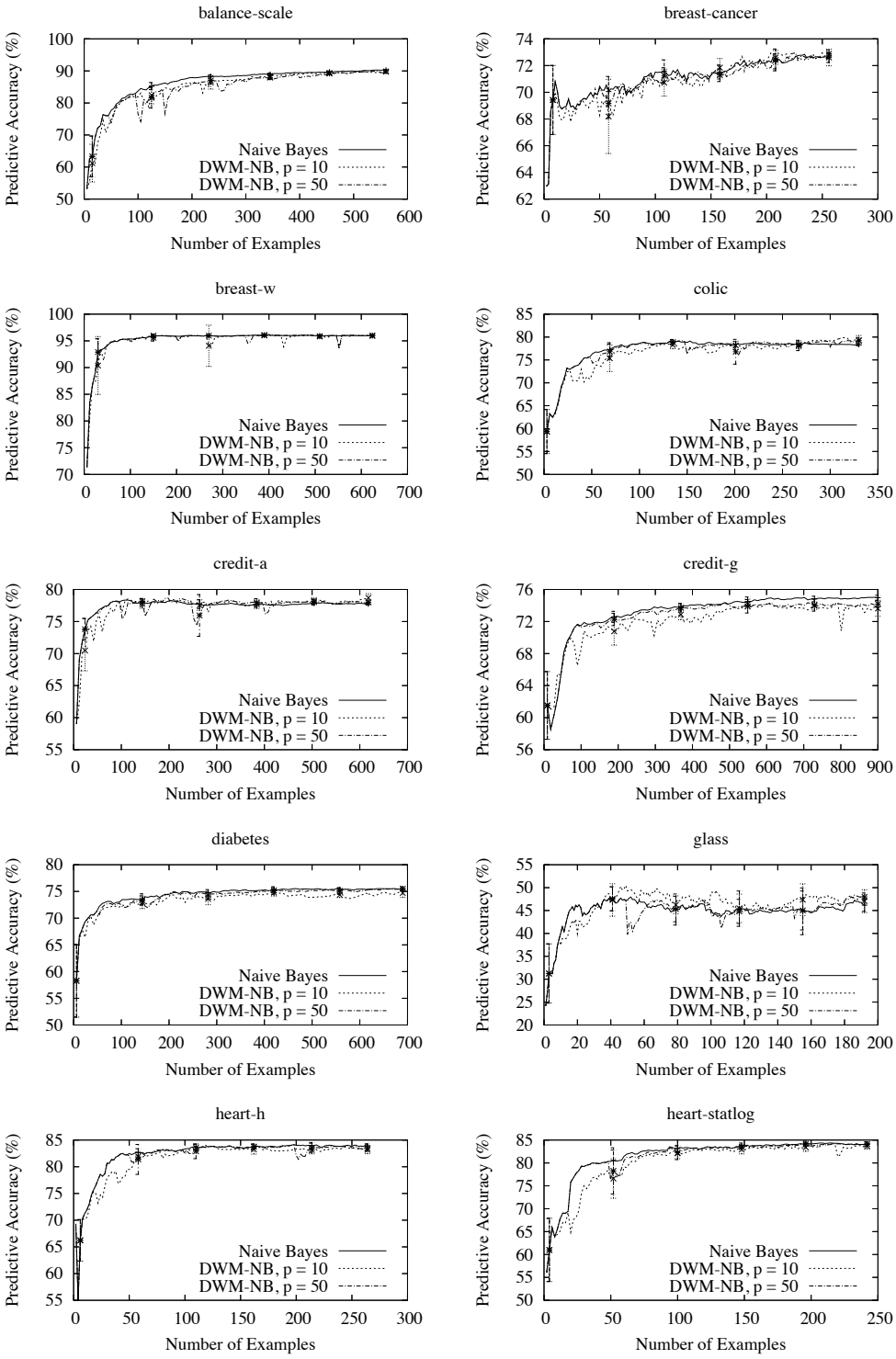


Figure 11: Predictive accuracy with 95% confidence intervals of naive Bayes and DWM-NB on selected UCI data sets, balance-scale to heart-statlog.

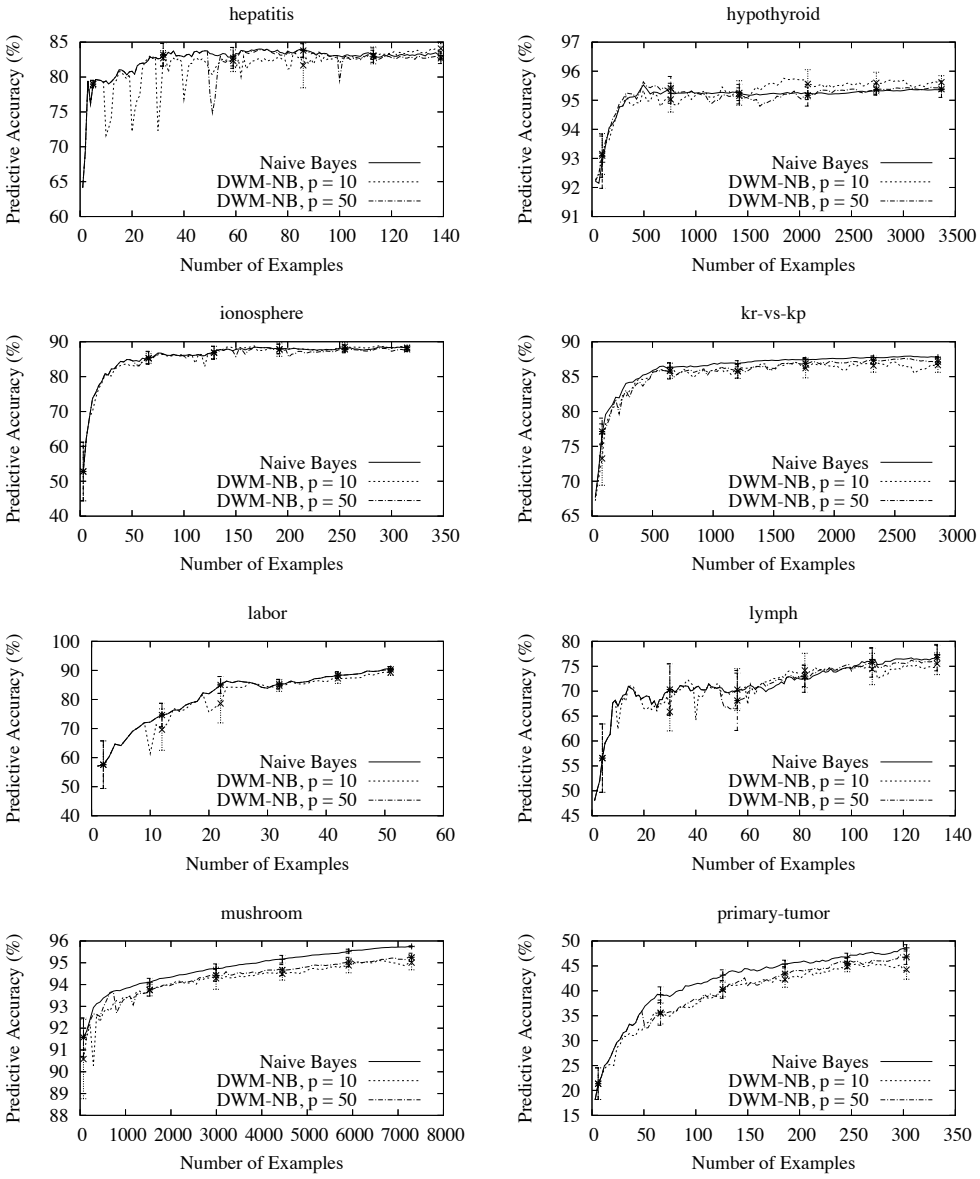


Figure 12: Predictive accuracy with 95% confidence intervals of naive Bayes and DWM-NB on selected UCI data sets, hepatitis to primary-tumor.

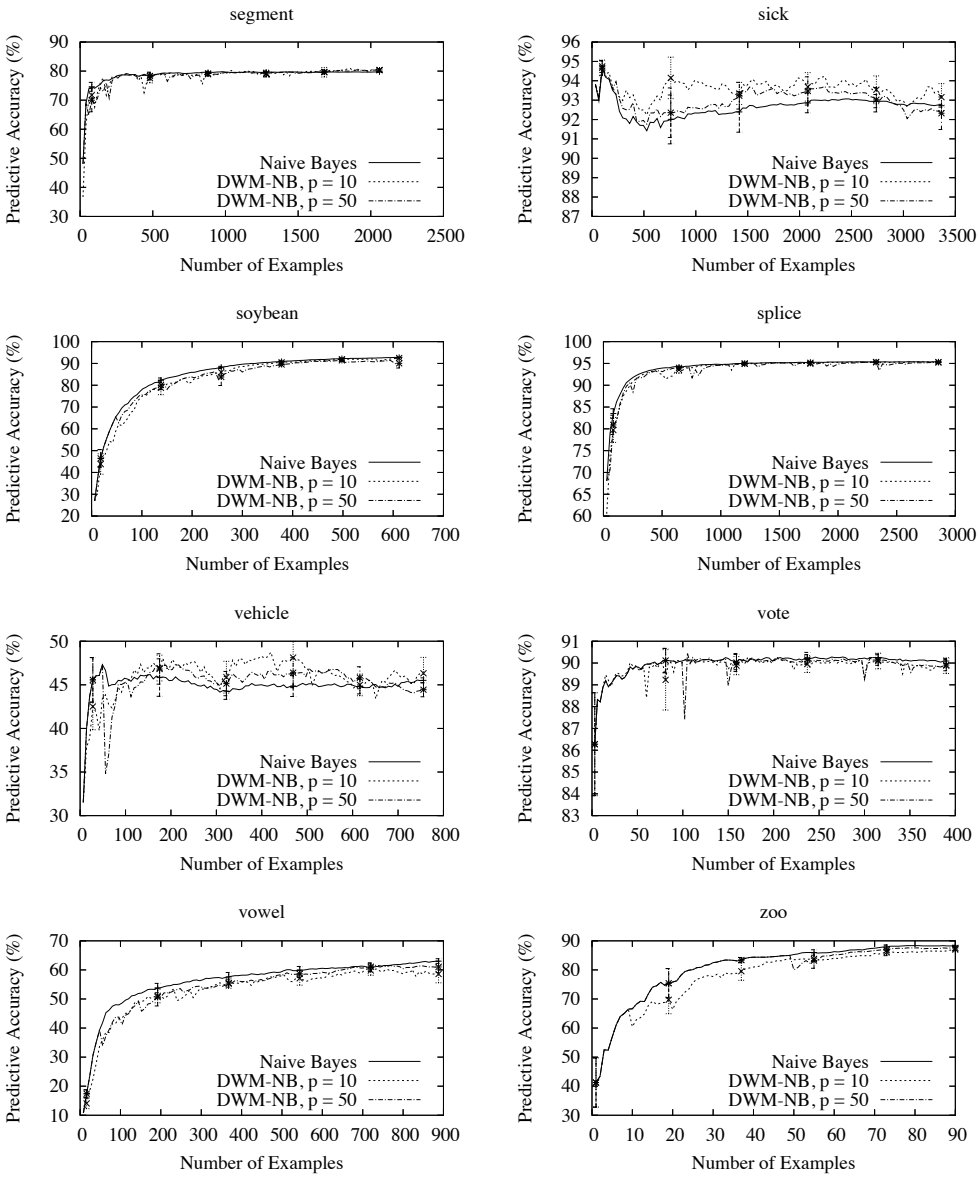


Figure 13: Predictive accuracy with 95% confidence intervals of naive Bayes and DWM-NB on selected UCI data sets, segment to zoo.

References

D. W. Aha, D. Kibler, and M. K. Albert. Instance-based learning algorithms. *Machine Learning*, 6: 37–66, 1991.

A. Asuncion and D. J. Newman. UCI Machine Learning Repository. Web site, Department of Information and Computer Sciences, University of California, Irvine, `http://www.ics.uci.edu/~mlearn/MLRepository.html`, 2007.

P. Auer and M. K. Warmuth. Tracking the best disjunction. *Machine Learning*, 32(2):127–150, 1998.

B. Bauer and R. Kohavi. An empirical comparison of voting classification algorithms: Bagging, boosting, and variants. *Machine Learning*, 36(1–2):105–139, 1999.

M. M. Black and R. J. Hickey. Maintaining the performance of a learned classifier under concept drift. *Intelligent Data Analysis*, 3(6):453–474, 1999.

M. M. Black and R. J. Hickey. Classification of customer call data in the presence of concept drift and noise. In *Soft-Ware 2002: Computing in an Imperfect World*, volume 2311 of *Lecture Notes in Computer Science*, pages 74–87. Springer, Berlin, 2002. First International Conference, Soft-Ware 2002, Belfast, Northern Ireland, April 8–10, 2002. Proceedings.

A. Blum. Empirical support for Winnow and Weighted-Majority algorithms: Results on a calendar scheduling domain. *Machine Learning*, 26:5–23, 1997.

B. E. Boser, I. Guyon, and V. Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the Fourth Workshop on Computational Learning Theory*, pages 144–152. ACM Press, New York, NY, 1992.

O. Bousquet and M. K. Warmuth. Tracking a small set of experts by mixing past posteriors. *Journal of Machine Learning Research*, 3:363–396, 2002.

L. Breiman. Arcing classifiers. *The Annals of Statistics*, 26(3):801–849, 1998.

L. Breiman. Bagging predictors. *Machine Learning*, 24:123–140, 1996.

N. V. Chawla, L. O. Hall, K. W. Bowyer, and W. P. Kegelmeyer. Learning ensembles from bites: A scalable and accurate approach. *Journal of Machine Learning Research*, 5:421–451, 2004.

S. J. Delany, P. Cunningham, A. Tsymbal, and L. Coyle. A case-based technique for tracking concept drift in spam filtering. *Knowledge-Based Systems*, 18(4–5):187–195, 2005.

T. G. Dietterich. An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. *Machine Learning*, 40(2):139–158, 2000.

P. Domingos and G. Hulten. Mining high-speed data streams. In *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 71–80. ACM Press, New York, NY, 2000.

W. Fan. StreamMiner: A classifier ensemble-based engine to mine concept-drifting data streams. In *Proceedings of the Thirtieth International Conference on Very Large Data Bases*, pages 1257–1260. Morgan Kaufmann, San Francisco, CA, 2004.

W. Fan, S. J. Stolfo, and J. Zhang. The application of AdaBoost for distributed, scalable and on-line learning. In *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 362–366. ACM Press, New York, NY, 1999.

A. Fern and R. Givan. Online ensemble learning: An empirical study. *Machine Learning*, 53: 71–109, 2003.

Y. Freund and R. E. Schapire. Experiments with a new boosting algorithm. In *Proceedings of the Thirteenth International Conference on Machine Learning*, pages 148–156. Morgan Kaufmann, San Francisco, CA, 1996.

J. Gama, P. Medas, G. Castillo, and P. Rodrigues. Learning with drift detection. In *Advances in Artificial Intelligence*, volume 3171 of *Lecture Notes in Computer Science*, pages 286–295. Springer, Berlin, 2004. Seventeenth Brazilian Symposium on Artificial Intelligence, SBIA-2004, São Luis, Maranhõ, Brazil, September 29–October 1, 2004, Proceedings.

J. Gama, P. Medas, and P. Rodrigues. Learning decision trees from dynamic data streams. In *Proceedings of the 2005 ACM Symposium on Applied Computing (SAC-2005)*, pages 573–577. ACM Press, New York, NY, 2005.

R. B. Gramacy, M. K. Warmuth, S. A. Brandt, and I. Ari. Adaptive caching by refetching. In *Advances in Neural Information Processing Systems 15*, pages 1465–1472. MIT Press, Cambridge, MA, 2003.

L. K. Hansen and P. Salamon. Neural network ensembles. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(10):993–1001, 1990.

M. Harries. Splice-2 Comparative Evaluation: Electricity Pricing. Technical Report UNSW-CSE-TR-9905, Artificial Intelligence Group, School of Computer Science and Engineering, The University of New South Wales, Sidney, Australia, July 1999.

M. Harries and K. Horn. Detecting concept drift in financial time series prediction using symbolic machine learning. In *Proceedings of the Eighth Australian Joint Conference on Artificial Intelligence*, pages 91–98. World Scientific, Singapore, 1995.

D. P. Helmbold and P. M. Long. Tracking drifting concepts using random examples. In L. G. Valiant and M. K. Warmuth, editors, *Proceedings of the Fourth Annual Workshop on Computational Learning Theory (COLT'91)*, pages 13–23. Morgan Kaufmann, San Francisco, CA, 1991.

D. P. Helmbold and P. M. Long. Tracking drifting concepts by minimising disagreements. *Machine Learning*, 14:27–45, 1994.

M. Herbster and M. K. Warmuth. Tracking the best expert. *Machine Learning*, 32:151–178, 1998.

R. J. Hickey and M. M. Black. Refined time stamps for concept drift detection during mining for classification rules. In *Temporal, Spatial, and Spatio-Temporal Data Mining*, volume 2007 of *Lecture Notes in Artificial Intelligence*, pages 20–30. Springer, Berlin, 2000. First International Workshop, TSDM 2000, Lyon, France, September 2000, Revised papers.

D. Hinkley. Jackknife methods. In S. Kotz, N. L. Johnson, and C. B. Read, editors, *Encyclopedia of Statistical Sciences*, volume 4, pages 280–287. John Wiley & Sons, New York, NY, 1983.

W. Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301):13–30, 1963.

G. Hulten, L. Spencer, and P. Domingos. Mining time-changing data streams. In *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 97–106. ACM Press, New York, NY, 2001.

M. G. Kelly, D. J. Hand, and N. M. Adams. The impact of changing populations on classifier performance. In *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 367–371. ACM Press, New York, NY, 1999.

M. Klenner and U. Hahn. Concept versioning: A methodology for tracking evolutionary concept drift in dynamic concept systems. In *Proceedings of the Eleventh European Conference on Artificial Intelligence*, pages 473–477. John Wiley & Sons, London, 1994.

R. Klinkenberg and T. Joachims. Detecting concept drift with support vector machines. In *Proceedings of the Seventeenth International Conference on Machine Learning*, pages 487–494. Morgan Kaufmann, San Francisco, CA, 2000.

Ralf Klinkenberg. Learning drifting concepts: Example selection vs. example weighting. *Intelligent Data Analysis*, 8(3):281–300, 2004. Special Issue on Incremental Learning Systems Capable of Dealing with Concept Drift.

J. Z. Kolter and M. A. Maloof. Dynamic weighted majority: A new ensemble method for tracking concept drift. In *Proceedings of the Third IEEE International Conference on Data Mining*, pages 123–130. IEEE Press, Los Alamitos, CA, 2003.

J. Z. Kolter and M. A. Maloof. Using additive expert ensembles to cope with concept drift. In *Proceedings of the Twenty-second International Conference on Machine Learning*, pages 449–456. ACM Press, New York, NY, 2005.

A. Kuh, T. Petsche, and R. L. Rivest. Learning time-varying concepts. In *Advances in Neural Information Processing Systems 3*, pages 183–189. Morgan Kaufmann, San Francisco, CA, 1991.

T. Lane and C. E. Brodley. Approaches to online learning and concept drift for user identification in computer security. In *Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining*, pages 259–263. AAAI Press, Menlo Park, CA, 1998.

N. Littlestone. Redundant noisy attributes, attribute errors, and linear-threshold learning using Winnow. In *Proceedings of the Fourth Annual Workshop on Computational Learning Theory*, pages 147–156. Morgan Kaufmann, San Francisco, CA, 1991.

N. Littlestone. Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm. *Machine Learning*, 2:285–318, 1988.

N. Littlestone and M. K. Warmuth. The Weighted Majority algorithm. *Information and Computation*, 108:212–261, 1994.

R. Maclin and D. Opitz. An empirical evaluation of bagging and boosting. In *Proceedings of the Fourteenth National Conference on Artificial Intelligence*, pages 546–551. AAAI Press, Menlo Park, CA, 1997.

M. A. Maloof. Concept drift. In J. Wang, editor, *Encyclopedia of Data Warehousing and Mining*, pages 202–206. Information Science Publishing, Hershey, PA, 2005.

M. A. Maloof. Incremental rule learning with partial instance memory for changing concepts. In *Proceedings of the International Joint Conference on Neural Networks*, pages 2764–2769. IEEE Press, Los Alamitos, CA, 2003.

M. A. Maloof and R. S. Michalski. Incremental learning with partial instance memory. *Artificial Intelligence*, 154:95–126, 2004.

M. A. Maloof and R. S. Michalski. Selecting examples for partial memory learning. *Machine Learning*, 41:27–52, 2000.

C. Mesterharm. Tracking linear-threshold concepts with Winnow. *Journal of Machine Learning Research*, 4:819–838, 2003.

R. S. Michalski. On the quasi-minimal solution of the general covering problem. In *Proceedings of the Fifth International Symposium on Information Processing*, volume A3, pages 125–128. 1969.

R. S. Michalski and J. B. Larson. Incremental Generation of VL_1 Hypotheses: The Underlying Methodology and the Description of Program AQ11. Technical Report UIUCDCS-F-83-905, Department of Computer Science, University of Illinois, Urbana, 1983.

T. M. Mitchell, R. Caruana, D. Freitag, J. McDermott, and D. Zabowski. Experience with a learning personal assistant. *Communications of the ACM*, 37(7):80–91, July 1994.

C. Monteleoni and T. S. Jaakkola. Online learning of non-stationary sequences. In *Advances in Neural Information Processing Systems 16*. MIT Press, Cambridge, MA, 2004.

D. Opitz and R. Maclin. Popular ensemble methods: An empirical study. *Journal of Artificial Intelligence Research*, 11:169–198, 1999.

J. R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, San Francisco, CA, 1993.

J. R. Quinlan. Bagging, boosting, and C4.5. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, pages 725–730. AAAI Press, Menlo Park, CA, 1996.

J. R. Quinlan. Induction of decision trees. *Machine Learning*, 1:81–106, 1986.

R. E. Reinke and R. S. Michalski. Incremental learning of concept descriptions: A method and experimental results. In J. E. Hayes, D. Michie, and J. Richards, editors, *Machine Intelligence 11*, pages 263–288. Clarendon Press, Oxford, 1988.

R. A. Rescorla. Probability of shock in the presence and absence of CS in fear conditioning. *Journal of Comparative and Physiological Psychology*, 66:1–5, 1968.

J. C. Schlimmer. *Concept Acquisition through Representational Adjustment*. PhD thesis, Department of Information and Computer Science, University of California, Irvine, 1987.

J. C. Schlimmer and D. Fisher. A case study of incremental concept induction. In *Proceedings of the Fifth National Conference on Artificial Intelligence*, pages 496–501. AAAI Press, Menlo Park, CA, 1986.

J. C. Schlimmer and R. H. Granger. Beyond incremental processing: Tracking concept drift. In *Proceedings of the Fifth National Conference on Artificial Intelligence*, pages 502–507. AAAI Press, Menlo Park, CA, 1986.

M. Scholz and R. Klinkenberg. Boosting Classifiers for Drifting Concepts. Technical report, Collaborative Research Center on the Reduction of Complexity for Multivariate Data Structures (SFB 475), University of Dortmund, Dortmund, Germany, January 2006.

M. Scholz and R. Klinkenberg. An ensemble classifier for drifting concepts. In J. Aguilar and J. Gama, editors, *Proceedings of the Second International Workshop on Knowledge Discovery in Data Streams*, pages 53–64. `http://www.liacc.up.pt/~jgama/IWKDDS/`, 2005. Held at the 16th European Conference on Machine Learning (ECML) and European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD), ECML/PKDD-2005.

W. N. Street and Y. Kim. A streaming ensemble algorithm (SEA) for large-scale classification. In *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 377–382. ACM Press, New York, NY, 2001.

N. A. Syed, H. Liu, and K. K. Sung. Handling concept drifts in incremental learning with support vector machines. In *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 272–276. ACM Press, New York, NY, 1999.

A. Tsymbal, M. Pechenizkiy, P. Cunningham, and S. Puuronen. Dynamic Integration of Classifiers for Tracking Concept Drift in Antibiotic Resistance Data. Technical Report TCD-CS-2005-26, Trinity College Dublin, Dublin, Ireland, February 2005.

P. E. Utgoff, N. C. Berkman, and J. A. Clouse. Decision tree induction based on efficient tree restructuring. *Machine Learning*, 29:5–44, 1997.

H. Wang, W. Fan, P. S. Yu, and J. Han. Mining concept-drifting data streams using ensemble classifiers. In *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 226–235. ACM Press, New York, NY, 2003.

G. Widmer. Tracking context changes through meta-learning. *Machine Learning*, 27:259–286, 1997.

G. Widmer and M. Kubat. Learning in the presence of concept drift and hidden contexts. *Machine Learning*, 23:69–101, 1996.

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

D. H. Wolpert. Stacked generalization. *Neural Networks*, 5(2):241–259, 1992.

K. Woods, W. P. Kegelmeyer, and K. Bowyer. Combination of multiple classifiers using local accuracy estimates. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(4): 405–410, 1997.

Z. Zheng. Naive Bayesian classifier committees. In *Proceedings of the Tenth European Conference on Machine Learning*, pages 196–207. Springer, Berlin, 1998.

Z.-H. Zhou, J. Wu, and W. Tang. Ensembling neural networks: Many could be better than all. *Artificial Intelligence*, 137(1–2):239–263, 2002.