The Journal of Machine Learning Research

Volume 10
Print-Archive Edition

Pages 1–1484

Microtome Publishing
Brookline, Massachusetts
www.mtome.com

The Journal of Machine Learning Research
Volume 10
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 2009.

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

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

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

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

JMLR Editorial Board

Journal of Machine Learning Research

Volume 10, 2009

Journal of Machine Learning Research 1 (2009) 1-40 Submitted 12/07; Revised 9/08; Published 1/09

Exploring Strategies for Training Deep Neural Networks

Hugo Larochelle LAROCHEH@IRO.UMONTREAL.CA
Yoshua Bengio BENGIOY@IRO.UMONTREAL.CA
Jérôme Louradour LOURADOJ@IRO.UMONTREAL.CA
Pascal Lamblin LAMBLINP@IRO.UMONTREAL.CA
Département d'informatique et de recherche opérationnelle
Université de Montréal
2920, chemin de la Tour
Montréal, Québec, Canada, H3T 1J8

Editor: Léon Bottou

Abstract

Deep multi-layer neural networks have many levels of non-linearities allowing them to compactly represent highly non-linear and highly-varying functions. However, until recently it was not clear how to train such deep networks, since gradient-based optimization starting from random initialization often appears to get stuck in poor solutions. Hinton et al. recently proposed a greedy layer-wise unsupervised learning procedure relying on the training algorithm of restricted Boltzmann machines (RBM) to initialize the parameters of a deep belief network (DBN), a generative model with many layers of hidden causal variables. This was followed by the proposal of another greedy layer-wise procedure, relying on the usage of autoassociator networks. In the context of the above optimization problem, we study these algorithms empirically to better understand their success. Our experiments confirm the hypothesis that the greedy layer-wise unsupervised training strategy helps the optimization by initializing weights in a region near a good local minimum, but also implicitly acts as a sort of regularization that brings better generalization and encourages internal distributed representations that are high-level abstractions of the input. We also present a series of experiments aimed at evaluating the link between the performance of deep neural networks and practical aspects of their topology, for example, demonstrating cases where the addition of more depth helps. Finally, we empirically explore simple variants of these training algorithms, such as the use of different RBM input unit distributions, a simple way of combining gradient estimators to improve performance, as well as on-line versions of those algorithms.

Keywords: artificial neural networks, deep belief networks, restricted Boltzmann machines, autoassociators, unsupervised learning

1. Introduction

Training deep multi-layered neural networks is known to be hard. The standard learning strategy—consisting of randomly initializing the weights of the network and applying gradient descent using backpropagation—is known empirically to find poor solutions for networks with 3 or more hidden layers. As this is a negative result, it has not been much reported in the machine learning literature. For that reason, artificial neural networks have been limited to one or two hidden layers.

However, complexity theory of circuits strongly suggests that deep architectures can be much more efficient (sometimes exponentially) than shallow architectures, in terms of computational elements and parameters required to represent some functions (Bengio and Le Cun, 2007; Bengio, 2007). Whereas it cannot be claimed that deep architectures are better than shallow ones on every problem (Salakhutdinov and Murray, 2008; Larochelle and Bengio, 2008), there has been evidence of a benefit when the task is complex enough, and there is enough data to capture that complexity (Larochelle et al., 2007). Hence finding better learning algorithms for such deep networks could be beneficial.

An approach that has been explored with some success in the past is based on constructively adding layers. Each layer in a multi-layer neural network can be seen as a representation of the input obtained through a learned transformation. What makes a good internal representation of the data? We believe that it should disentangle the factors of variation that inherently explain the structure of the distribution. When such a representation is going to be used for unsupervised learning, we would like it to preserve information about the input while being easier to model than the input itself. When a representation is going to be used in a supervised prediction or classification task, we would like it to be such that there exists a "simple" (i.e., somehow easy to learn) mapping from the representation to a good prediction. To constructively build such a representation, it has been proposed to use a *supervised* criterion at each stage (Fahlman and Lebiere, 1990; Lengellé and Denoeux, 1996). However, as we discuss here, the use of a supervised criterion at each stage may be too greedy and does not yield as good generalization as using an unsupervised criterion. Aspects of the input may be ignored in a representation tuned to be immediately useful (with a linear classifier) but these aspects might turn out to be important when more layers are available. Combining unsupervised (e.g., learning about $p(x)$) and supervised components (e.g., learning about $p(y|x)$) can be be helpful when both functions $p(x)$ and $p(y|x)$ share some structure.

The idea of using unsupervised learning at each stage of a deep network was recently put forward by Hinton et al. (2006), as part of a training procedure for the deep belief network (DBN), a generative model with many layers of hidden stochastic variables. Upper layers of a DBN are supposed to represent more "abstract" concepts that explain the input observation $\mathbf{x}$, whereas lower layers extract "low-level features" from $\mathbf{x}$. In other words, this model first learns simple concepts, on which it builds more abstract concepts.

This training strategy has inspired a more general approach to help address the problem of training deep networks. Hinton (2006) showed that stacking restricted Boltzmann machines (RBMs)—that is, training upper RBMs on the distribution of activities computed by lower RBMs—provides a good initialization strategy for the weights of a deep artificial neural network. This approach can be extended to non-linear autoencoders or autoassociators (Saund, 1989), as shown by Bengio et al. (2007), and is found in stacked autoassociators network (Larochelle et al., 2007), and in the deep convolutional neural network (Ranzato et al., 2007b) derived from the convolutional neural network (LeCun et al., 1998). Since then, deep networks have been applied with success not only in classification tasks (Bengio et al., 2007; Ranzato et al., 2007b; Larochelle et al., 2007; Ranzato et al., 2008), but also in regression (Salakhutdinov and Hinton, 2008), dimensionality reduction (Hinton and Salakhutdinov, 2006; Salakhutdinov and Hinton, 2007b), modeling textures (Osindero and Hinton, 2008), information retrieval (Salakhutdinov and Hinton, 2007a), robotics (Hadsell et al., 2008), natural language processing (Collobert and Weston, 2008; Weston et al., 2008), and collaborative filtering (Salakhutdinov et al., 2007).

In this paper, we discuss in detail three principles for training deep neural networks and present experimental evidence that highlight the role of each in successfully training deep networks:

1. Pre-training one layer at a time in a greedy way;

2. using unsupervised learning at each layer in a way that preserves information from the input and disentangles factors of variation;

3. fine-tuning the whole network with respect to the ultimate criterion of interest.

The experiments reported here suggest that this strategy improves on the traditional random initialization of supervised multi-layer networks by providing "hints" to each intermediate layer about the kinds of representations that it should learn, and thus initializing the supervised fine-tuning optimization in a region of parameter space from which a better local minimum (or plateau) can be reached. We also present a series of experiments aimed at evaluating the link between the performance of deep neural networks and aspects of their topology such as depth and the size of the layers. In particular, we demonstrate cases where the addition of depth helps classification error, but too much depth hurts. Finally, we explore simple variants of the aforementioned training algorithms, such as a simple way of combining them to improve their performance, RBM variants for continuous-valued inputs, as well as on-line versions of those algorithms.

2. Notations and Conventions

Before describing the learning algorithms that we will study and experiment with in this paper, we first present the mathematical notation we will use for deep networks.

A deep neural network contains an input layer and an output layer, separated by l layers of hidden units. Given an input sample clamped to the input layer, the other units of the network compute their values according to the activity of the units that they are connected to in the layers below. We will consider a particular sort of topology here, where the input layer is fully connected to the first hidden layer, which is fully connected to the second layer and so on up to the output layer.

Given an input $\mathbf{x}$, the value of the j-th unit in the i-th layer is denoted $\widehat{h}^i_j(\mathbf{x})$, with $i = 0$ referring to the input layer, $i = l + 1$ referring to the output layer (the use of "$\widehat{\ }$" will become clearer in Section 4). We refer to the size of a layer as $|\widehat{\mathbf{h}}^i(\mathbf{x})|$. The default activation level is determined by the internal bias b^i_j of that unit. The set of weights W^i_{jk} between $\widehat{h}^{i-1}_k(\mathbf{x})$ in layer $i-1$ and unit $\widehat{h}^i_j(\mathbf{x})$ in layer i determines the activation of unit $\widehat{h}^i_j(\mathbf{x})$ as follows:

$$\widehat{h}^i_j(\mathbf{x}) = \operatorname{sigm}\left(a^i_j\right) \text{ where } a^i_j(\mathbf{x}) = b^i_j + \sum_k W^i_{jk}\widehat{h}^{i-1}_k(\mathbf{x})\ \forall i \in \{1,\ldots,l\},\text{ with } \widehat{h}^0(\mathbf{x}) = \mathbf{x} \tag{1}$$

where $\operatorname{sigm}(\cdot)$ is the sigmoid squashing function: $\operatorname{sigm}(a) = \frac{1}{1+e^{-a}}$ (alternatively, the sigmoid could be replaced by the hyperbolic tangent). Given the last hidden layer, the output layer is computed similarly by

$$\mathbf{o}(\mathbf{x}) = \widehat{\mathbf{h}}^{l+1}(\mathbf{x}) = f\left(\mathbf{a}^{l+1}(\mathbf{x})\right) \text{ where } \mathbf{a}^{l+1}(\mathbf{x}) = \mathbf{b}^{l+1} + \mathbf{W}^{l+1}\widehat{\mathbf{h}}^l(\mathbf{x})$$

where the activation function $f(\cdot)$ depends on the (supervised) task the network must achieve. Typically, it will be the identity function for a regression problem and the softmax function

$$f_j(\mathbf{a}) = \operatorname{softmax}_j(\mathbf{a}) = \frac{e^{a_j}}{\sum_{k=1}^K e^{a_k}} \tag{2}$$

for a classification problem, in order to obtain a distribution over the K classes.

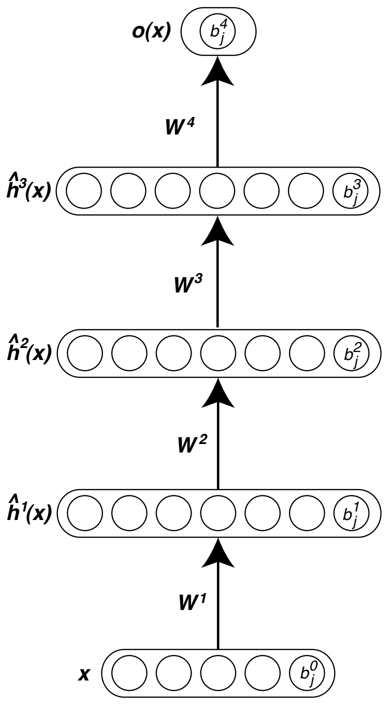


Figure 1: Illustration of a deep network and its parameters.

When an input sample **x** is presented to the network, the application of Equation 1 at each layer will generate a pattern of activity in the different layers of the neural network. Intuitively, we would like the activity of the first layer neurons to correspond to low-level features of the input (e.g., edge orientations for natural images) and to higher level abstractions (e.g., detection of geometrical shapes) in the last hidden layers.

3. Deep Neural Networks

It has been shown that a "shallow" neural network with only one arbitrarily large hidden layer could approximate a function to any level of precision (Hornik et al., 1989). Similarly, any Boolean function can be represented by a two-layer circuit of logic gates. However, most Boolean functions require an exponential number of logic gates (with respect to the input size) to be represented by a two-layer circuit (Wegener, 1987). For example, the parity function, which can be efficiently represented by a circuit of depth $O(\log n)$ (for n input bits) needs $O(2^n)$ gates to be represented by a depth two circuit (Yao, 1985). What about deeper circuits? Some families of functions which can be represented with a depth k circuit are such that they require an exponential number of logic gates to be represented by a depth $k-1$ circuit (Håstad, 1986). Interestingly, an equivalent result has been proved for architectures whose computational elements are not logic gates but linear threshold units (i.e., formal neurons) (Hastad and Goldmann, 1991). The machine learning literature also suggests that shallow architectures can be very inefficient in terms of the number of computational units (e.g., bases, hidden units), and thus in terms of required examples (Bengio and Le Cun, 2007; Bengio et al., 2006). On the other hand, a highly-varying function can sometimes be represented compactly (with fewer parameters) through the composition of many non-linearities, that is, with a deep architecture. When the representation of a concept requires an exponential number of elements (more generally exponential capacity), for example, with a shallow circuit, the number of training examples required to learn the concept may also be impractical. Smoothing the learned function by regularization would not solve the problem here because in these cases the target function itself is complicated and requires exponential capacity just to be represented.

3.1 Difficulty of Training Deep Architectures

Given a particular task, a natural way to train a deep network is to frame it as an optimization problem by specifying a supervised cost function on the output layer with respect to the desired target and use a gradient-based optimization algorithm in order to adjust the weights and biases of the network so that its output has low cost on samples in the training set. Unfortunately, deep networks trained in that manner have generally been found to perform worse than neural networks with one or two hidden layers.

We discuss two hypotheses that may explain this difficulty. The first one is that gradient descent can easily get stuck in poor local minima (Auer et al., 1996) or plateaus of the non-convex training criterion. The number and quality of these local minima and plateaus (Fukumizu and Amari, 2000) clearly also influence the chances for random initialization to be in the basin of attraction (via gradient descent) of a poor solution. It may be that with more layers, the number or the width of such poor basins increases. To reduce the difficulty, it has been suggested to train a neural network in a constructive manner in order to divide the hard optimization problem into several greedy but simpler ones, either by adding one neuron (e.g., see Fahlman and Lebiere, 1990) or one layer (e.g., see Lengellé and Denoeux, 1996) at a time. These two approaches have demonstrated to be very effective for learning particularly complex functions, such as a very non-linear classification problem in 2 dimensions. However, these are exceptionally hard problems, and for learning tasks usually found in practice, this approach commonly overfits.

This observation leads to a second hypothesis. For high capacity and highly flexible deep networks, there actually exists many basins of attraction in its parameter space (i.e., yielding different solutions with gradient descent) that can give low training error but that can have very different generalization errors. So even when gradient descent is able to find a (possibly local) good minimum in terms of training error, there are no guarantees that the associated parameter configuration will provide good generalization. Of course, model selection (e.g., by cross-validation) will partly correct this issue, but if the number of good generalization configurations is very small in comparison to good training configurations, as seems to be the case in practice, then it is likely that the training procedure will not find any of them. But, as we will see in this paper, it appears that the type of unsupervised initialization discussed here can help to select basins of attraction (for the supervised fine-tuning optimization phase) from which learning good solutions is easier both from the point of view of the training set and of a test set.

3.2 Unsupervised Learning as a Promising Paradigm for Greedy Layer-Wise Training

A common approach to improve the generalization performance of a learning algorithm which is motivated by the Occam's razor principle is the use of regularization (such as weight decay) that will favor "simpler" models over more complicated ones. However, using generic priors such as the ℓ^2 norm of the parameters conveys limited information about what the solution to a particular learning task should be. This has motivated researchers to discover more meaningful, data-dependent regularization procedures, which are usually based on unsupervised learning and normally adapted to specific models.

For example, Ando and Zhang (2005) use "auxiliary tasks" designed from unlabelled data and that are appropriate for a particular learning problem, to learn a better regularization term for linear classifiers. Partial least squares (Frank and Friedman, 1993) can also be seen as combining unsupervised and supervised learning in order to learn a better linear regression model when few training data are available or when the input space is very high dimensional.

Many semi-supervised learning algorithms also involve a combination of unsupervised and supervised learning, where the unsupervised component can be applied to additional unlabelled data. This is the case for Fisher-kernels (Jaakkola and Haussler, 1999) which are based on a generative model trained on unlabelled input data and that can be used to solve a supervised problem defined for that input space. In all these cases, unsupervised learning can be seen as adding more constraints on acceptable configurations for the parameters of a model, by asking that it not only describes well the relationship between the input and the target but also contains relevant statistical information about the structure of the input or how it was generated.

Moreover, there is a growing literature on the distinct advantages of generative and discriminative learning. Ng and Jordan (2001) argue that generative versions of discriminative models can be expected to reach their usually higher asymptotic out-of-sample classification error faster (i.e., with less training data), making them preferable in certain situations. Moreover, successful attempts at exploring the space between discriminative and generative learning have been studied (Lasserre et al., 2006; Jebara, 2003; Bouchard and Triggs, 2004; Holub and Perona, 2005).

The deep network learning algorithms that have been proposed recently and that we study in this paper can be seen as combining the ideas of *greedily learning the network* to break down the learning problem into easier steps, *using unsupervised learning* to provide an *effective hint* about what hidden units should learn, bringing along the way a form of regularization that prevents overfitting even in deep networks with many degrees of freedom (which could otherwise overfit). In addition, one should consider the supervised task the network has to solve. The greedy layer-wise unsupervised strategy provides an initialization procedure, after which the neural network is *fine-tuned to the global supervised objective*. The general paradigm followed by these algorithms (illustrated in Figure 2 and detailed in Appendix A) can be decomposed in two phases:

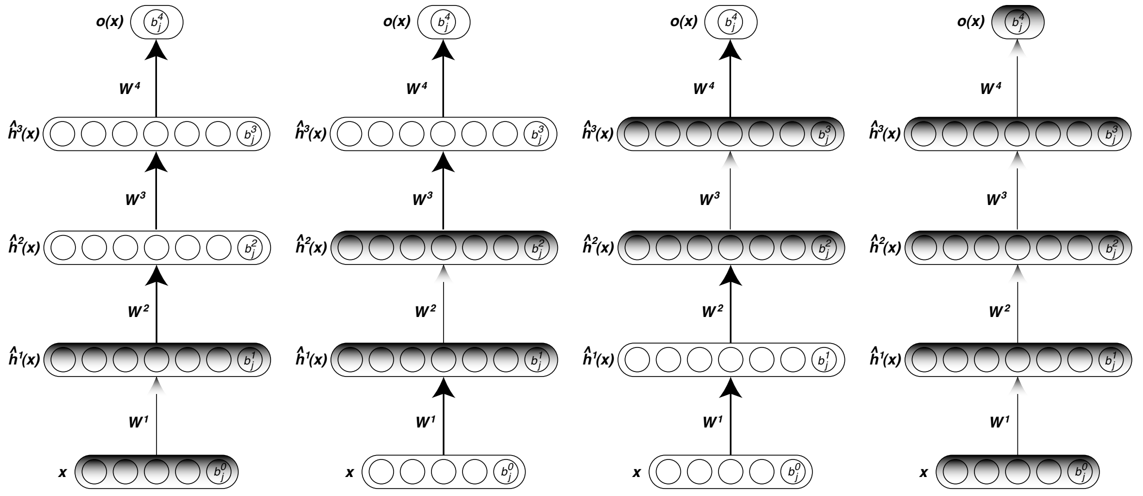
1. In the first phase, greedily train subsets of the parameters of the network using a layer-wise and unsupervised learning criterion, by repeating the following steps for each layer ($i \in \{1, \dots, l\}$)

   Until a stopping criteria is met, iterate through training database by

   (a) mapping input training sample $\mathbf{x}_t$ to representation $\widehat{\mathbf{h}}^{i-1}(\mathbf{x}_t)$ (if $i > 1$) and hidden representation $\widehat{\mathbf{h}}^{i}(\mathbf{x}_t)$,

   (b) updating parameters $\mathbf{b}^{i-1}$, $\mathbf{b}^{i}$ and $\mathbf{W}^{i}$ of layer i using some unsupervised learning algorithm.

   Also, initialize (e.g., randomly) the output layer parameters $\mathbf{b}^{l+1}, \mathbf{W}^{l+1}$.



(a) First hidden layer pre-training (b) Second hidden layer pre-training (c) Third hidden layer pre-training (d) Fine-tuning of whole network

Figure 2: Unsupervised greedy layer-wise training procedure.

2. In the second and final phase, fine-tune all the parameters θ of the network using backpropagation and gradient descent on a global supervised cost function $C(\mathbf{x}_t, y_t, \theta)$, with input $\mathbf{x}_t$ and label y_t, that is, trying to make steps in the direction $E\left[\frac{\partial C(\mathbf{x}_t, y_t, \theta)}{\partial \theta}\right]$.

Regularization is not explicit in this procedure, as it does not come from a weighted term that depends on the complexity of the network and that is added to the global supervised objective. Instead, it is implicit, as the first phase that initializes the parameters of the whole network will ultimately have an impact on the solution found in the second phase (the fine-tuning phase). Indeed, by using an iterative gradual optimization algorithm such as stochastic gradient descent with early-stopping (i.e., training until the error on a validation set reaches a clear minimum), the extent to which the configuration of the network's parameters can be different from the initial configuration given by the first phase is limited. Hence, similarly to using a regularization term on the parameters of the model that constrains them to be close to a particular value (e.g., 0 for weight decay), the first phase here will ensure that the parameter solution for each layer found by fine-tuning will not be far from the solution found by the unsupervised learning algorithm. In addition, the non-convexity of the supervised training criterion means that the choice of initial parameter values can greatly influence the quality of the solution obtained by gradient descent.

In the next two sections, we present a review of the two training algorithms that fall in paradigm presented above and which are empirically studied in this paper, in Section 6.

4. Stacked Restricted Boltzmann Machine Network

Intuitively, a successful learning algorithm for deep networks should be one that discovers a meaningful and possibly complex hidden representation of the data at its top hidden layer. However, learning such non-linear representations is a hard problem. A solution, proposed by Hinton (2006), is based on the learning algorithm of the restricted Boltzmann machine (RBM) (Smolensky, 1986), a generative model that uses a layer of binary variables to explain its input data. In an RBM (see

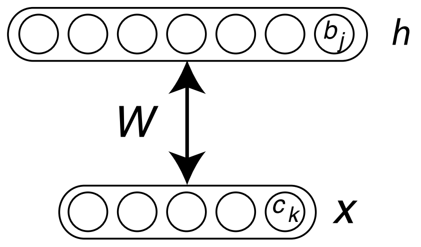


Figure 3: Illustration of a restricted Boltzmann machine and its parameters. **W** is a weight matrix, **b** is a vector of hidden unit biases, and **c** a vector of visible unit biases.

Figure 3 for an illustration), given an input $\mathbf{x}$, it is easy to obtain a hidden representation for that input by computing the posterior $\widehat{\mathbf{h}}(\mathbf{x})$ over the layer of binary hidden variables $\mathbf{h}$ (we use the "$\widehat{\ }$" symbol to emphasize that $\widehat{\mathbf{h}}(\mathbf{x})$ is not a random variable but a deterministic representation of $\mathbf{x}$).

Hinton (2006) argues that this representation can be improved by giving it as input to another RBM, whose posterior over its hidden layer will then provide a more complex representation of the input. This process can be repeated an arbitrary number of times in order to obtain ever more non-linear representations of the input. Finally, the parameters of the RBMs that compute these representations can be used to initialize the parameters of a deep network, which can then be fine-tuned to a particular supervised task. This learning algorithm clearly falls in the paradigm of Section 3.2, where the unsupervised part of the learning algorithm is that of an RBM. We will refer to deep networks trained using this algorithm as stacked restricted Boltzmann machine (SRBM) networks. For more technical details about the SRBM network, and how to train an RBM using the contrastive divergence algorithm (CD-k), see Appendix B.

5. Stacked Autoassociators Network

There are theoretical results about the advantage of stacking many RBMs into a DBN: Hinton et al. (2006) show that this procedure optimizes a bound on the likelihood of the input data when all layers have the same size. An additional hypothesis to explain why this process provides a good initialization for the network is that it makes each hidden layer compute a different, possibly more abstract representation of the input. This is done implicitly, by asking that each layer captures features of the input that help characterize the distribution of values at the layer below. By transitivity, each layer contains some information about the input. However, stacking any unsupervised learning model does not guarantee that the representations learned get increasingly complex or appropriate as we stack more layers. For instance, many layers of linear PCA models could be summarized by only one layer. However, there may be other non-linear, unsupervised learning models that, when stacked, are able to improve the learned representation at the last layer added.

An example of such a non-linear unsupervised learning model is the autoassociator or autoencoder network (Cottrell et al., 1987; Saund, 1989; Hinton, 1989; Baldi and Hornik, 1989; DeMers and Cottrell, 1993). Autoassociators are neural networks that are trained to compute a representation of the input from which it can be reconstructed with as much accuracy as possible. In this paper, we will consider autoassociator networks of only one hidden layer, meaning that the hidden

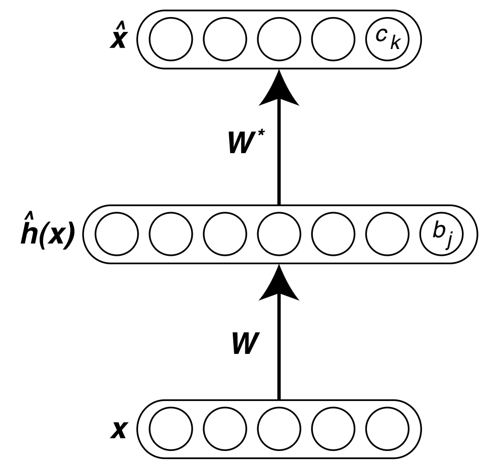


Figure 4: Illustration of an autoassociator and its parameters. $\mathbf{W}$ is the matrix of encoder weights and $\mathbf{W}^*$ the matrix of decoder weights. $\widehat{\mathbf{h}}(\mathbf{x})$ is the code or representation of $\mathbf{x}$.

representation of $\mathbf{x}$ is a code $\widehat{\mathbf{h}}(x)$ obtained from the encoding function

$$\widehat{h}_j(\mathbf{x}) = f(a_j) \text{ where } a_j(\mathbf{x}) = b_j + \sum_k W_{jk} x_k \,. \tag{3}$$

The input's reconstruction is obtained from a decoding function, here a linear transformation of the hidden representation with weight matrix $\mathbf{W}^*$, possibly followed by a non-linear activation function:

$$\widehat{x}_k = g(\widehat{a}_k) \text{ where } \widehat{a}_k = c_k + \sum_j W^*_{jk} \widehat{h}_j(\mathbf{x}) \,.$$

In this work, we used the sigmoid activation function for both $f(\cdot)$ and $g(\cdot)$. Figure 4 shows an illustration of this model.

By noticing the similarity between Equations 3 and 1, we are then able to use the training algorithm for autoassociators as the unsupervised learning algorithm for the greedy layer-wise initialization phase of deep networks. In this paper, stacked autoassociators (SAA) networks will refer to deep networks trained using the procedure of Section 3.2 and the learning algorithm of an autoassociator for each layer, as described in Section 5.1.

Though these neural networks were designed with the goal of dimensionality reduction in mind, the new representation's dimensionality does not necessarily need to be lower than the input's in practice. However, in that particular case, some care must be taken so that the network does not learn a trivial identity function, that is, finds weights that simply "copy" the whole input vector in the hidden layer and then copy it again at the output. For example, a network with small weights W_{jk} between the input and hidden layers (maintaining activations in the linear regime of the activation function f) and large weights W^*_{jk} between the hidden and output layers could encode such an uninteresting identity function. An easy way to avoid such a pathological behavior in the case of continuous inputs is to set the weight matrices $\mathbf{W}^\mathsf{T}$ and $\mathbf{W}^*$ to be the same. This adjustment is motivated by its similarity with the parametrization of the RBM model and by an empirical observation that $\mathbf{W}^\mathsf{T}$ and $\mathbf{W}^*$ tend to be similar up to a multiplicative factor after training. In the case of binary inputs, if the weights are large, the input vector can still be copied (up to a permutation of the elements) to the hidden units, and in turn these used to perfectly reconstruct the input. Weight decay can be useful to prevent such a trivial and uninteresting mapping to be learned, when the inputs are binary. We set $\mathbf{W}^{\mathsf{T}} = \mathbf{W}^*$ in all of our experiments. Vincent et al. (2008) have an improved way of training autoassociators in order to produce interesting, non-trivial features in the hidden layer, by partially corrupting the network's inputs.

The reconstruction error of an autoassociator can be connected to the log-likelihood of an RBM in several ways. Ranzato et al. (2008) connect the log of the numerator of the input likelihood with a form of reconstruction error (where one replaces the sum over hidden unit configurations by a maximization). The denominator is the normalization constant summing over all input configurations the same expression as in the numerator. So whereas maximizing the numerator is similar to minimizing reconstruction error for the training examples, minimizing the denominator means that most input configurations should not be reconstructed well. This can be achieved if the autoassociator is constrained in such a way that it cannot compute the identity function, but only minimizes the reconstruction for training examples.

Another connection between reconstruction error and log-likelihood of the RBM was made in Bengio and Delalleau (2007). They consider a converging series expansion of the log-likelihood gradient and show that whereas CD-k truncates the series by keeping the first $2k$ terms and then approximates expectations by a single sample, reconstruction error is a mean-field approximation of the first term in that series.

5.1 Learning in an Autoassociator Network

Training an autoassociator network is almost identical to training a standard artificial neural network. Given a cost function, backpropagation is used to compute gradients and perform gradient descent. However, autoassociators are "self-supervised", meaning that the target to which the output of the autoassociator is compared is the input that it was fed.

Previous work on autoassociators minimized the squared reconstruction error:

$$C(\widehat{\mathbf{x}}, \mathbf{x}) = \sum_k (\widehat{x}_k - x_k)^2 \ .$$

However, with squared reconstruction error and linear decoder, the "optimal codes" (the implicit target for the encoder, irrespective of the encoder) are in the span of the principal eigenvectors of the input covariance matrix. When we introduce a saturating non-linearity such as the sigmoid, and we want to reconstruct values $[0,1]$, the binomial KL divergence (also known as cross-entropy) seems more appropriate:

$$C(\widehat{\mathbf{x}}, \mathbf{x}) = -\sum_k \left(x_k \log(\widehat{x}_k) + (1 - x_k) \log(1 - \widehat{x}_k) \right) \ . \tag{4}$$

It corresponds to the assumption that $\widehat{\mathbf{x}}$ and $\mathbf{x}$ can be interpreted as factorized distributions over binary units. It is well known that the cross-entropy $-p\log(q) - (1-p)\log(1-q)$ between two binary distributions parametrized by p and q is minimized when $q = p$ (for a fixed p), making it an appropriate cost function to evaluate the quality of a reconstruction. We used this cost function in all the experiments with SAA networks. Appendix C details the corresponding autoassociator training update.

6. Experiments

In this section, we present several experiments set up to evaluate the deep network learning algorithms that fall in the paradigm presented in the Section 3.2 and highlight some of their properties. Unless otherwise stated, stochastic gradient descent was used for layer-wise unsupervised learning (first phase of the algorithm) and global supervised fine-tuning (second phase of the algorithm). The data sets were separated in disjoint training, validation and testing subsets. Model selection consisted of finding the best values for the learning rates of layer-wise unsupervised and global supervised learning as well as the number of unsupervised updates preceding the fine-tuning phase. The number of epochs of fine-tuning was chosen using early-stopping based on the progression of classification error on the validation set. All experiments correspond to classification problems. Hence, to fine-tune the deep networks, we optimized the negative conditional log-likelihood of the training samples' target class (as given by the softmax output of the neural network).

The experiments are based on the MNIST data set[1] (see Figure 5), a benchmark for handwritten digit recognition, as well as variants of this problem where the input distribution has been made more complex by inserting additional factors of variations, such as rotations and background images. The input images are made of 28×28 pixels giving an input dimensionality of 784, the number of classes is 10 (corresponding to the digits from 0 to 9) and the inputs were scaled between 0 and 1.

Successful applications of deep networks have already been presented on a large variety of data, such as images of faces (Salakhutdinov and Hinton, 2008), real-world objects (Ranzato et al., 2007a) as well as text data (Hinton and Salakhutdinov, 2006; Salakhutdinov and Hinton, 2007a; Collobert and Weston, 2008; Weston et al., 2008), and on different types of problems such as regression (Salakhutdinov and Hinton, 2008), information retrieval (Salakhutdinov and Hinton, 2007a), robotics Hadsell et al. (2008), and collaborative filtering (Salakhutdinov et al., 2007).

In Bengio et al. (2007), we performed experiments on two regression data sets, with non-image continuous inputs (UCI Abalone, and a financial data set), demonstrating the use of unsupervised (or partially supervised) pre-training of deep networks on these tasks. In Larochelle et al. (2007), we studied the performance of several architectures on various data sets, including variations of MNIST (with rotations, random background, and image background), and discrimination tasks between wide and tall rectangles, and between convex and non-convex images. On these tasks, deep networks compared favorably to shallow architectures.

Our focus is hence not on demonstrating their usefulness on a wide range of tasks, but on exploring their properties empirically. Such experimental work required several weeks of cumulative CPU time, which restricted the number of data sets we could explore. However, by concentrating on the original MNIST data set and harder versions of it, we were able not only to confirm the good performance of deep networks, but also to study practical variations, to help understand the algorithms, and to discuss the impact on a deep network's performance of stepping to a more complicated problem.

6.1 Validating the Unsupervised Layer-Wise Strategy for Deep Networks

In this section, we evaluate the advantages brought by the unsupervised layer-wise strategy of Section 3.2. We want to separate the different algorithmic concepts behind it, in order to understand their contribution to the whole strategy. In particular, we pursue the following two questions:

1. This data set can be downloaded from `http://yann.lecun.com/expdb/mnist/`.

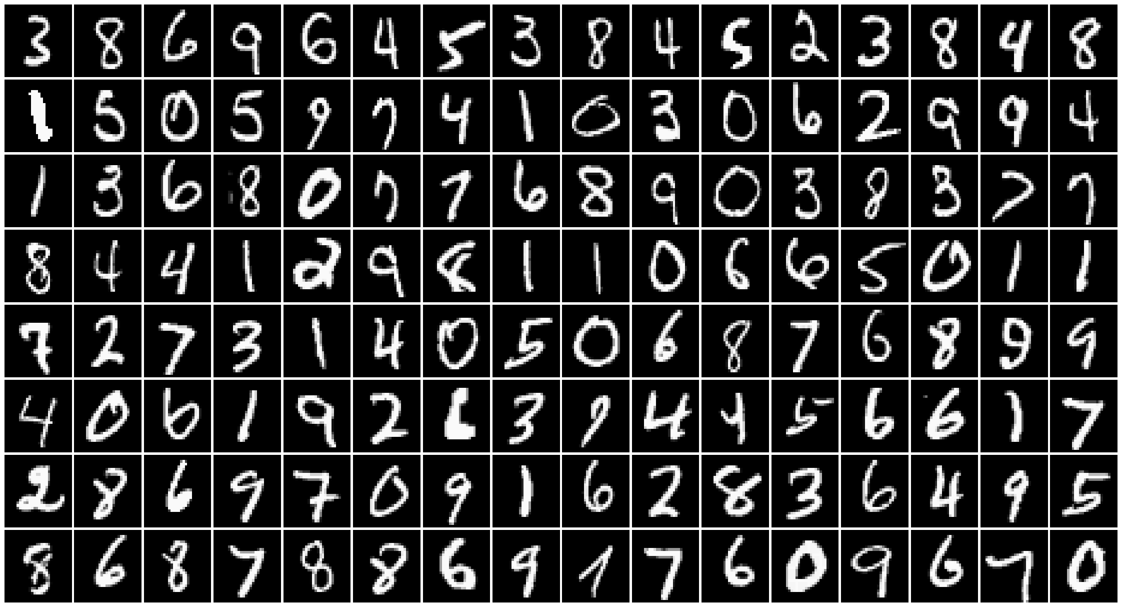


Figure 5: Samples from the MNIST digit recognition data set. Here, a black pixel corresponds to an input value of 0 and a white pixel corresponds to 1 (the inputs are scaled between 0 and 1).

1. To what extent does initializing greedily the parameters of the different layers help?
2. How important is unsupervised learning for this procedure?

To address these two questions, we will compare the learning algorithms for deep networks of Sections 4 and 5 with the following algorithms.

6.1.1 DEEP NETWORK WITHOUT PRE-TRAINING

To address the first question above, we compare the greedy layer-wise algorithm with a more standard way to train neural networks: using standard backpropagation and stochastic gradient descent, and starting at a randomly initialized configuration of the parameters. In other words, this variant simply puts away the pre-training phase of the other deep network learning algorithms.

6.1.2 DEEP NETWORK WITH SUPERVISED PRE-TRAINING

To address the second question, we run an experiment with the following algorithm. We greedily pre-train the layers using a *supervised criterion* (instead of the unsupervised one), before performing as before a final supervised fine-tuning phase. Specifically, when greedily pre-training the parameters $\mathbf{W}^i$ and $\mathbf{b}^i$, we also train another set of weights $\mathbf{V}^i$ and biases $\mathbf{c}^i$ which connect hidden layer $\widehat{\mathbf{h}}^i(\mathbf{x})$ to a temporary output layer as follows:

$$o^i(\mathbf{x}) = f\left(\mathbf{c}^i + \mathbf{V}^i\widehat{\mathbf{h}}^i(\mathbf{x})\right)$$

where $f(\cdot)$ is the softmax function of Equation 2. This output layer can be trained using the same cost as the global supervised cost. However, as this is a greedy procedure, only the parameters $\mathbf{W}^i$, $\mathbf{b}^i$, $\mathbf{V}^i$ and $\mathbf{c}^i$ are updated, that is, the gradient is not propagated to the layers below. When the training of a layer is finished, we can simply discard the parameters $\mathbf{V}^i$ and $\mathbf{c}^i$ and move to pre-training the next hidden layer, having initialized $\mathbf{W}^i$ and $\mathbf{b}^i$.

6.1.3 STACKED LOGISTIC AUTOREGRESSION NETWORK

The second question aims at evaluating to what extent any unsupervised learning can help. We already know that stacking linear PCA models is not expected to help improve generalization. A slightly more complex yet very simple unsupervised model for data in $[0,1]$ is the logistic autoregression model (see also Frey, 1998)

$$\widehat{x}_k = \mathrm{sigm}\left(b_k + \sum_{j \neq k} W_{kj} x_j\right) \tag{5}$$

where the reconstruction $\widehat{\mathbf{x}}$ is log-linear in the input $\mathbf{x}$. The parameters $\mathbf{W}$ and $\mathbf{b}$ can be trained using the same cost used for the autoassociators in Equation 4. This model can be used to initialize the weights $\mathbf{W}^i$ and biases $\mathbf{b}^i$ of the i-th hidden layer of a deep network. However, because $\mathbf{W}$ in Equation 5 is square, the deep network will need to have hidden layers with the same size as the input layer. Also, the weights on the diagonal of $\mathbf{W}$ are not trained in this model, so we initialize them to zero. The stacked logistic autoregression network will refer to deep networks using this unsupervised layer-wise learning algorithm.

6.1.4 RESULTS

The results for all these deep networks are given in Table 1. We also give results for a "shallow", one hidden layer neural network, to validate the utility of deep architectures. Instead of the sigmoid, this network uses hyperbolic tangent squashing functions, which are usually found to work better for one hidden layer neural networks. The MNIST training set was separated into training (50,000) and validation (10,000) sets. The test set has size 10,000. In addition to the hyperparameters mentioned at the beginning of this section, the validation set was used also to select appropriate decrease constants[2] for the learning rates of the greedy and fine-tuning phases. The SRBM and SAA networks had 500, 500 and 2000 hidden units in the first, second and third layers respectively, as in Hinton et al. (2006) and Hinton (2006). In the pre-training phase of the SRBM and SAA networks, when training the parameters of the i-th layer, the down-biases c_k where set to be equal to b_k^{i-1} (although similar results were obtained by using a separate set of biases c_k^{i-1} when the $i-1$-th layer is the down-layer). For the deep networks with supervised or no pre-training, different sizes of hidden layers were compared, including sizes similar to the stacked logistic autoregression network, and to the SRBM and SAA networks. All deep networks had 3 hidden layers.

Overall, the models that use the unsupervised layer-wise procedure of Section 3.2 outperform those that do not. We also observe a slight advantage in the performance of the SRBM network over that of the SAA network (on the MNIST test set, differences of more than 0.1% are statistically significant). The performance difference between the stacked logistic autoregressions network and

2. When using a decrease constant β, the learning rate for the t^{th} update becomes $\frac{\varepsilon_0}{1+t\beta}$, where ε_0 is the initial learning rate.

Models	Train.	Valid.	Test
SRBM (stacked restricted Boltzmann machines) network	0%	1.20%	1.20%
SAA (stacked autoassociators) network	0%	1.31%	1.41%
Stacked logistic autoregressions network	0%	1.65%	1.85%
Deep network with supervised pre-training	0%	1.74%	2.04%
Deep network, no pre-training	0.004%	2.07%	2.40%
Shallow network, no pre-training	0%	1.91%	1.93%

Table 1: Classification error on MNIST training, validation, and test sets, with the best hyperparameters according to validation error.

the deep network with supervised layer-wise pre-training particularly highlights the importance of unsupervised learning. Indeed, even though supervised layer-wise pre-training explicitly trains the hidden layers to capture non-linear information about the input, the overall procedure seems to be too greedy with respect to the supervised task to be learned. On the other hand, even though logistic autoregressions are simple log-linear models and their optimization is blind with respect to the future usage of the weights $\mathbf{W}$ as connections into non-linear hidden layers, the unsupervised nature of training makes them still useful for improving generalization. As a point of comparison, besides the deep networks, the best result on this data set reported for a learning algorithm that does not use any prior knowledge about the task (e.g., image pre-processing like deskewing or subsampling) is that of a support vector machine with a Gaussian kernel,[3] with 1.4% classification error on the test set.

At this point, it is clear that unsupervised layer-wise pre-training improves generalization. However, we could wonder whether it also facilitates the optimization problem of the global fine-tuning. The results of Table 1 do not shed any light on this aspect. Indeed, all the networks, even those without greedy layer-wise pre-training, perform almost perfectly on the training set. The explanatory hypothesis we evaluate here is that, without pre-training, the lower layers are initialized poorly, but still allow the top two layers to learn the training set almost perfectly because the output layer and the last hidden layer form a standard shallow but fat neural network. Consider the top two layers of the deep network *with pre-training*: it presumably takes as input a *better representation*, one that allows for better generalization. Instead, the network *without pre-training* sees a "random" transformation of the input, one that preserves enough information about the input to fit the training set, but that does not help to generalize. To test this hypothesis, we performed a second series of experiments in which we constrain the top hidden layer to be small (20 hidden units).

The results (Table 2) clearly suggest that optimization of the global supervised objective is made easier by greedy layer-wise pre-training. This result for supervised greedy pre-training is also coherent with past experiments on similar greedy strategies (Fahlman and Lebiere, 1990; Lengellé and Denoeux, 1996). Here, we have thus confirmed that it also applies to unsupervised greedy pre-training. With no pre-training, training error degrades significantly when there are only 20 hidden units in the top hidden layer. In addition, the results obtained without pre-training were found to have much larger variance than those with pre-training, indicating high sensitivity to initial

3. See `http://yann.lecun.com/exdb/mnist/` for more details.

Models	Train.	Valid.	Test
SRBM network	0%	1.5%	1.5%
SAA network	0%	1.38%	1.65%
Deep network with supervised pre-training	0%	1.77%	1.89%
Deep network, no pre-training	0.59%	2.10%	2.20%
Shallow network, no pre-training	3.6%	4.77%	5.00%

Table 2: Classification error on MNIST training, validation, and test sets, with the best hyperparameters according to validation error, when the last hidden layer only contains 20 hidden units

conditions: the unsupervised pre-training more consistently puts the parameters in a "good" basin of attraction for the supervised gradient descent procedure.

Figures 6 and 7 show the sorts of first hidden layer features (weights going into different hidden neurons) that are learned by the first (bottom) RBM and autoassociator respectively, before fine-tuning. Both models were trained on the MNIST training set of Section 6.1 for 40 epochs, with 250 hidden units and a learning rate of 0.005. We see that they both learn visual features characterized by local receptive fields, which ought to be useful to recognize more global shapes (though the autoassociator also learns high frequency receptive fields that are spread over the whole image). This is another account of how unsupervised greedy pre-training is able to help the optimization of the network. Even if the supervised fine-tuning gradient at the first hidden layer is weak, we can see that the first hidden layer appears to learn a relevant representation.

6.2 Exploring the Space of Network Architectures

An important practical aspect in using deep network is the choice the architecture or topology of the network. Once we allow ourselves to consider an arbitrary number of hidden layers of arbitrary sizes, some questions naturally arise. First, we would like to know how deep a neural network can be made while still obtaining generalization gains, given a strategy for initializing its parameters (randomly or with unsupervised greedy pre-training). We would also like to know, for a determined depth, what type of architecture is more appropriate. Should the hidden layer's size increase, decrease or stay the same from the first to the last? In this section, we explore those two questions with experiments on the MNIST data set as well as a variant, taken from Larochelle et al. (2007), where the digit images have been randomly rotated. This last data set, noted MNIST-rotation[4] (see Figure 8), contains much more intraclass variability, is much less well described by relatively well separated class-specific clusters and corresponds to a much harder classification problem. The training, validation and test sets contain 10 000, 2 000 and 50 000 examples each. We also generated sets of the same size for the MNIST data set. We refer to this version with a smaller training set by MNIST-small.

4. This data set has been regenerated since Larochelle et al. (2007) and is available here: `http://www.iro.umontreal.ca/~lisa/icml2007`.

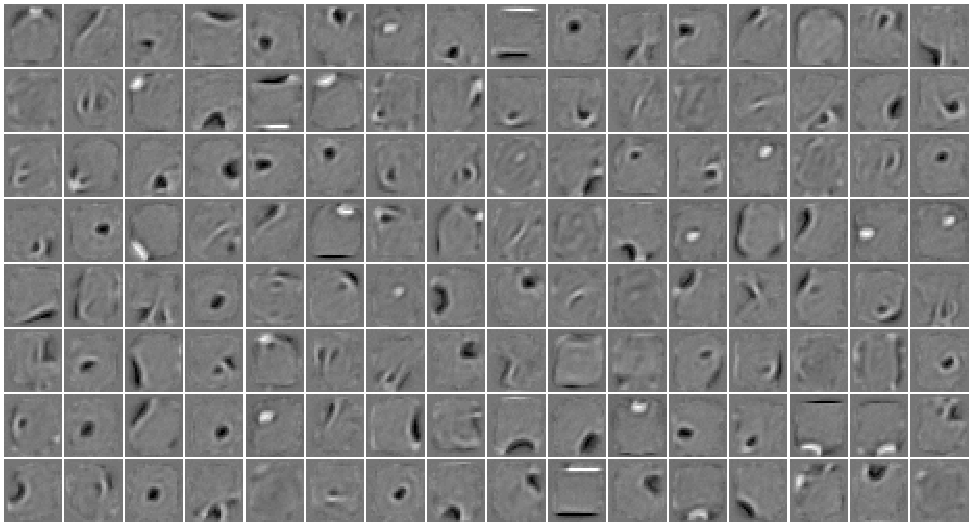


Figure 6: Display of the input weights of a random subset of the hidden units, learned by an RBM when trained on samples from the MNIST data set. The activation of units of the first hidden layer is obtained by a dot product of such a weight "image" with the input image. In these images, a black pixel corresponds to a weight smaller than -3 and a white pixel to a weight larger than 3, with the different shades of gray corresponding to different weight values uniformly between -3 and 3.

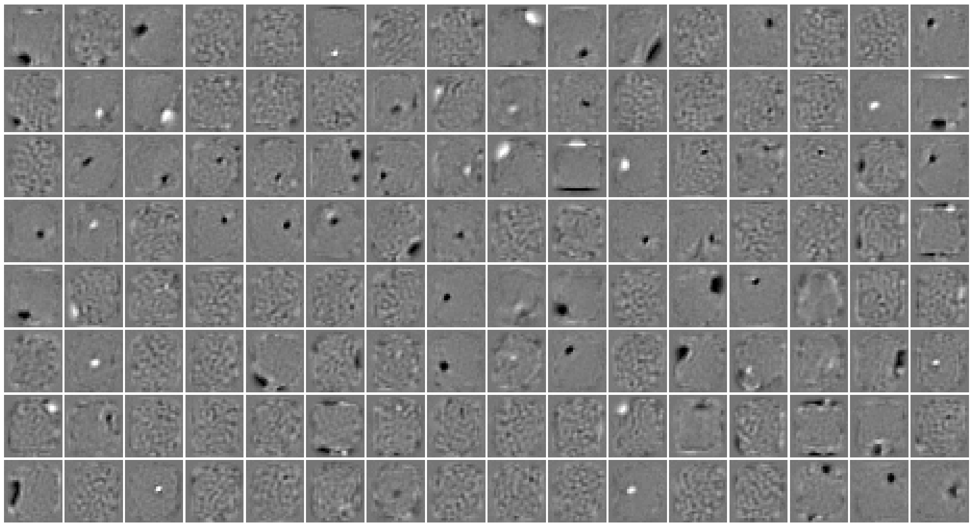


Figure 7: Input weights of a random subset of the hidden units, learned by an autoassociator when trained on samples from the MNIST data set. The display setting is the same as for Figure 6.

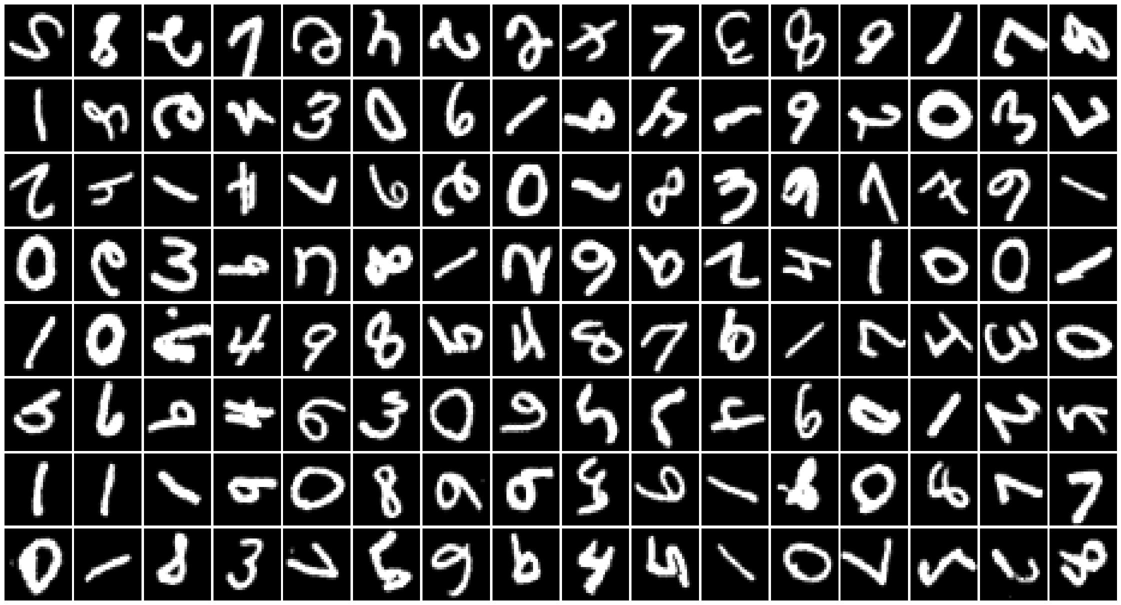


Figure 8: Samples from the MNIST-rotation data set. Here, a black pixel corresponds to an input value of 0 and a white pixel corresponds to 1 (the inputs are scaled between 0 and 1).

6.2.1 NETWORK DEPTH

One can wonder whether a neural network can be made too deep, that is, whether having too many hidden layers can worsen the generalization performance. Of course there are many reasons why this may happen. When a neuron is added, more parameters are inserted in the mathematical formulation of the model, giving it more degrees of freedom to fit the model and hence possibly making it able to overfit. On the other hand, it is less clear to what extent the performance can worsen, since a neuron added at the top layer of a neural network does not increase the capacity the same way a neuron added "in parallel" in a given hidden layer. Also, in the case of an SRBM network, we can imagine that as we stack RBMs, the representation at a hidden layer contains units that correspond to more and more disentangled concepts of the input. Now, consider a hypothetical deep network where the top-level stacked RBM has learned a representation made of units that are mostly independent. An additional RBM stacked on this representation would have no statistical structure to learn. This would initialize the weights of that new RBM to zero, which is particularly troublesome as the representation at this level would then contain no information about the input. It is not clear if this scenario is plausible, unlike in the case of independent component analysis, but if it were approached the result would be detrimental to supervised classification performance. This particular situation is not expected with stacked autoassociators, as it will always learn a representation from which the previous layer can be reconstructed. Another reason why a deeper architecture could produce worse results is simply that our algorithms for training a deep architecture can probably be improved. In particular, note that the only joint training of all layers that we have done in our experiments, if any, is at the supervised fine-tuning stage.

Network		MNIST-small	MNIST-rotation
Type	Depth	classif. test error	classif. test error
Neural network	1	**4.14** % ± 0.17	15.22 % ± 0.31
(random initialization,	2	**4.03** % ± 0.17	**10.63** % ± 0.27
+ fine-tuning)	3	**4.24** % ± 0.18	11.98 % ± 0.28
	4	4.47 % ± 0.18	11.73 % ± 0.29
SAA network	1	3.87 % ± 0.17	11.43% ± 0.28
(autoassociator learning	2	**3.38** % ± 0.16	9.88 % ± 0.26
+ fine-tuning)	3	**3.37** % ± 0.16	**9.22** % ± 0.25
	4	**3.39** % ± 0.16	**9.20** % ± 0.25
SRBM network	1	3.17 % ± 0.15	10.47 % ± 0.27
(CD-1 learning	2	**2.74** % ± 0.14	9.54 % ± 0.26
+ fine-tuning)	3	**2.71** % ± 0.14	**8.80** % ± 0.25
	4	**2.72** % ± 0.14	**8.83** % ± 0.24

Table 3: Classification performance on MNIST-small and MNIST-rotation of different networks for different strategies to initialize parameters, and different depths (number of layers).

Table 3 presents the classification performance obtained by the different deep networks with up to 4 hidden layers on MNIST-small and MNIST-rotation. The hyperparameters of each layer were *separately* selected with the validation set for all hidden layers, using the following greedy strategy: for a network with l hidden layers, only the hyperparameters for the top layer were optimized, the hyperparameters for the layers below being set to those of the best $l-1$ layers deep network according to the validation performance. We settled for this strategy because of the exponential number of possible configurations of hyperparameters. For standard neural networks, we also tested several random initializations of the weights. For SRBM as well as SAA networks, we tuned the unsupervised learning rates and the number of updates. For MNIST-small, we used hidden layers of 500 neurons, since the experiments by Hinton (2006) suggest that it is an appropriate choice. As for MNIST-rotation, the size of each hidden layer had to be validated separately for each layer, and we tested values among 500, 1000, 2000 and 4000.

Table 3 show that there is indeed an optimal number of hidden layers for the deep networks, and that this optimum tends to be larger when unsupervised greedy layer-wise learning is used. For the MNIST-small data set (Table 3), the gain in performance between 2 and 3 hidden layers for SRBM and SAA networks is not statistically significant. However, for the MNIST-rotation data set, the improvement from 2 to 3 hidden layers is clear. This observation is consistent with the increased complexity of the input distribution and classification problem of MNIST-rotation, which should require a more complex model. The improvement remains significant when fixing the network's hidden layers to the same size as in the experiments on MNIST-small, as showed in the results of Table 4 where the number of units per hidden layer was set to 1000. We also compared the performance of shallow and deep SRBM networks with roughly the same number of parameters. With a shallow SRBM network, the best classification error achieved was 10.47%, with 4000 hidden units (around 3.2×10^6 free parameters). With a 3-layers deep SRBM network, we reached 9.38%

Network			**MNIST-rotation**
Type	Depth	Layers width	**classif. test error**
SRBM network	1	1k	12.44 % $\pm$ 0.29
(CD-1 learning	2	1k, 1k	9.98 % $\pm$ 0.26
+ fine-tuning)	3	1k, 1k, 1k	**9.38** % $\pm$ 0.25

Table 4: Classification performance on MNIST-rotation of different networks for different strategies to initialize parameters, and different depths (number of layers). All hidden layers have 1000 units.

classification error with 1000 units in each layer (around 2.8×10^6 parameters): better generalization was achieved with deeper nets having less parameters.

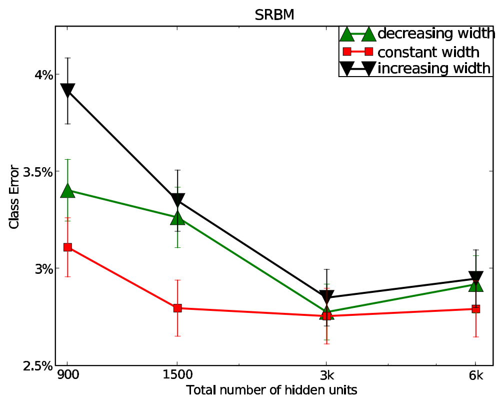
6.2.2 TYPE OF NETWORK ARCHITECTURE

The model selection procedure of Section 6.2.1 works well, but is rather expensive. Every time one wants to train a 4 hidden layer network, networks with 1, 2 and 3 hidden layers effectively have to be trained as well, in order to determine appropriate hyperparameters for the lower hidden layers. These networks can't even be trained in parallel, adding to the computational burden of this model selection procedure. Moreover, the optimal hidden layer size for a 1-hidden layer network could be much bigger than necessary for a 4 hidden layer network, since a shallow network cannot rely on other upper layers to increase its capacity.

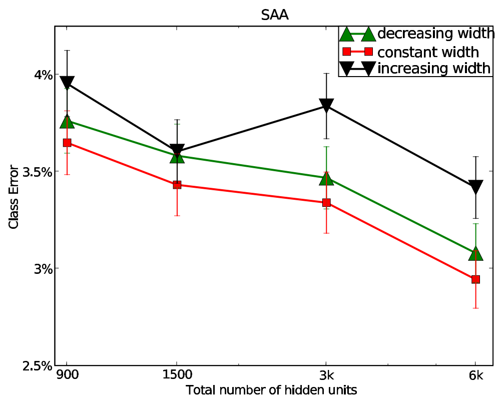
Let us consider the situation where the number of hidden layers of a deep network has already been chosen and good sizes of the different layers must be found. Because the space of such possible choices is exponential in the number of layers, we consider here only three general cases where, as the layer index increases, their sizes either increases (doubles), decreases (halves) or does not change. We conducted experiments for all three cases and varied the total number of hidden neurons in the network. The same hyperparameters as in the experiment of Table 3 had to be selected for each network topologies, however a single unsupervised learning rate and number of updates were chosen for all layers.[5]

We observe in Figures 9 and 10 that the architecture that most often is among the best performing ones across the different sizes of network is the one with equal sizes of hidden layers. It should be noted that this might be a consequence of using the same unsupervised learning hyperparameters for each layer. It might be that the size of a hidden layer has a significant influence on the optimum value for these hyperparameters, and that tying them for all hidden layers induces a bias towards networks with equally-sized hidden layers. However, having untied hyperparameters would make model selection too computationally demanding. Actually, even with tied unsupervised learning hyperparameters, the model selection problem is already complex enough (and prone to overfitting with small data sets), as is indicated by the differences in the validation and test classification errors of Table 3.

5. We imposed this restriction because of the large number of experiments that would otherwise had been required.

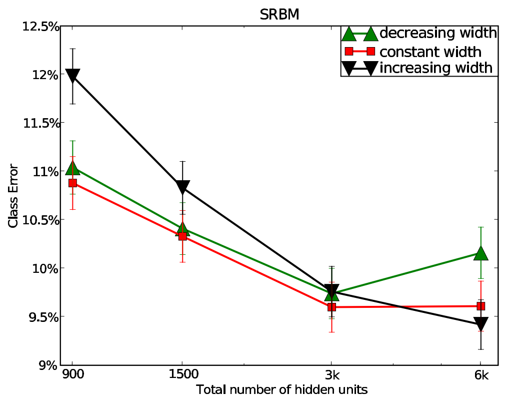


(a) SRBM network.

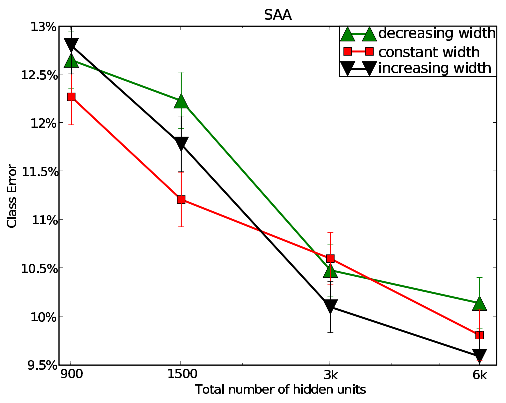


(b) SAA network.

Figure 9: Classification performance on MNIST-small of 3-layer deep networks for three kinds of architectures, as a function of the total number of hidden units. The three architectures have increasing / constant / decreasing layer sizes from the bottom to the top layers. Error-bars represent 95% confidence intervals.



(a) SRBM network.



(b) SAA network.

Figure 10: Classification performance on MNIST-rotation of 3-layer deep networks for three kinds of architectures. Same conventions as in Figure 9.

7. Continuous-Valued Inputs

In this section, we wish to emphasize the importance of adapting the unsupervised learning algorithms to the nature of the inputs. We will focus on the SRBM network because they rely on RBMs, which are less simple to work with and adapt to the sorts of visible data we want to model. With the binary units introduced for RBMs and DBNs in Hinton et al. (2006) one can "cheat" and handle continuous-valued inputs by scaling them to the $[0,1]$ interval and considering each input continuous value as the probability for a binary random variable to take the value 1. This has worked well for pixel gray levels, but it may be inappropriate for other kinds of input variables. Previous work on continuous-valued input in RBMs include Chen and Murray (2003), in which noise is added to sigmoidal units, and the RBM forms a special form of diffusion network (Movellan et al., 2002). Welling et al. (2005) also show how to derive RBMs from arbitrary choices of exponential distributions for the visible and hidden layers of an RBM. We show here simple extensions of the RBM framework in which only the energy function and the allowed range of values are changed. As can be seen in Figures 11 and 12 and in the experiment of Section 7.3, such extensions have a very significant impact on nature of the solution learned for the RBM's weights and hence on the initialization of a deep network and its performance.

7.1 Linear Energy: Exponential or Truncated Exponential

Consider a unit with value x_k in an RBM, connected to units $\mathbf{h}$ of the layer above. $p(x_k|\mathbf{h})$ can be obtained by considering the terms in the energy function that contain x_k. These terms can be grouped in $x_k(\mathbf{W}_{\cdot k}^{\mathsf{T}}\mathbf{h}+c_k)$ when the energy function is linear in x_k (as in Equation 7, appendix B), where $\mathbf{W}_{\cdot k}$ is the k-th column of $\mathbf{W}$. If we allow x_k to take any value in interval I, the conditional density of x_k becomes

$$p(x_k|\mathbf{h}) = \frac{e^{x_k(\mathbf{W}_{\cdot j}^{\mathsf{T}}\mathbf{h}+c_k)}\mathbf{1}_{x_k\in I}}{\int_v e^{v(\mathbf{W}_{\cdot j}^{\mathsf{T}}\mathbf{h}+c_k)}\mathbf{1}_{v\in I}dv}\,.$$

When $I=[0,\infty)$, this is an exponential density with parameter $a(\mathbf{h})=\mathbf{W}_{\cdot j}^{\mathsf{T}}\mathbf{h}+c_k$, and the normalizing integral, equal to $\frac{-1}{a(\mathbf{h})}$, only exists if $a(\mathbf{h})<0\ \forall\mathbf{h}$. Computing the density, the expected value ($\frac{-1}{a(\mathbf{h})}$) and sampling would all be easy, but since the density does not always exist it seems more appropriate to let I be a closed interval, yielding a *truncated exponential* density. For simplicity we consider the case $I=[0,1]$ here, for which the normalizing integral, which always exists, is

$$\frac{e^{-a(\mathbf{h})}-1}{a(\mathbf{h})}\,.$$

The conditional expectation of x_k given $\mathbf{h}$ is interesting because it has a sigmoidal-like saturating and monotone non-linearity:

$$E\left[x_k|\mathbf{h}\right] = \frac{1}{1-e^{-a(\mathbf{h})}} - \frac{1}{a(\mathbf{h})}\,.$$

Note that $E\left[x_k|\mathbf{h}\right]$ does not explode for $a(\mathbf{h})$ near 0, but is instead smooth and in the interval $[0,1]$. A sample from the truncated exponential is easily obtained from a uniform sample U, using the inverse cumulative F^{-1} of the conditional density $p(x_k|\mathbf{h})$:

$$F^{-1}(U) = \frac{\log(1-U\times(1-e^{a(\mathbf{h})}))}{a(\mathbf{h})}\,.$$

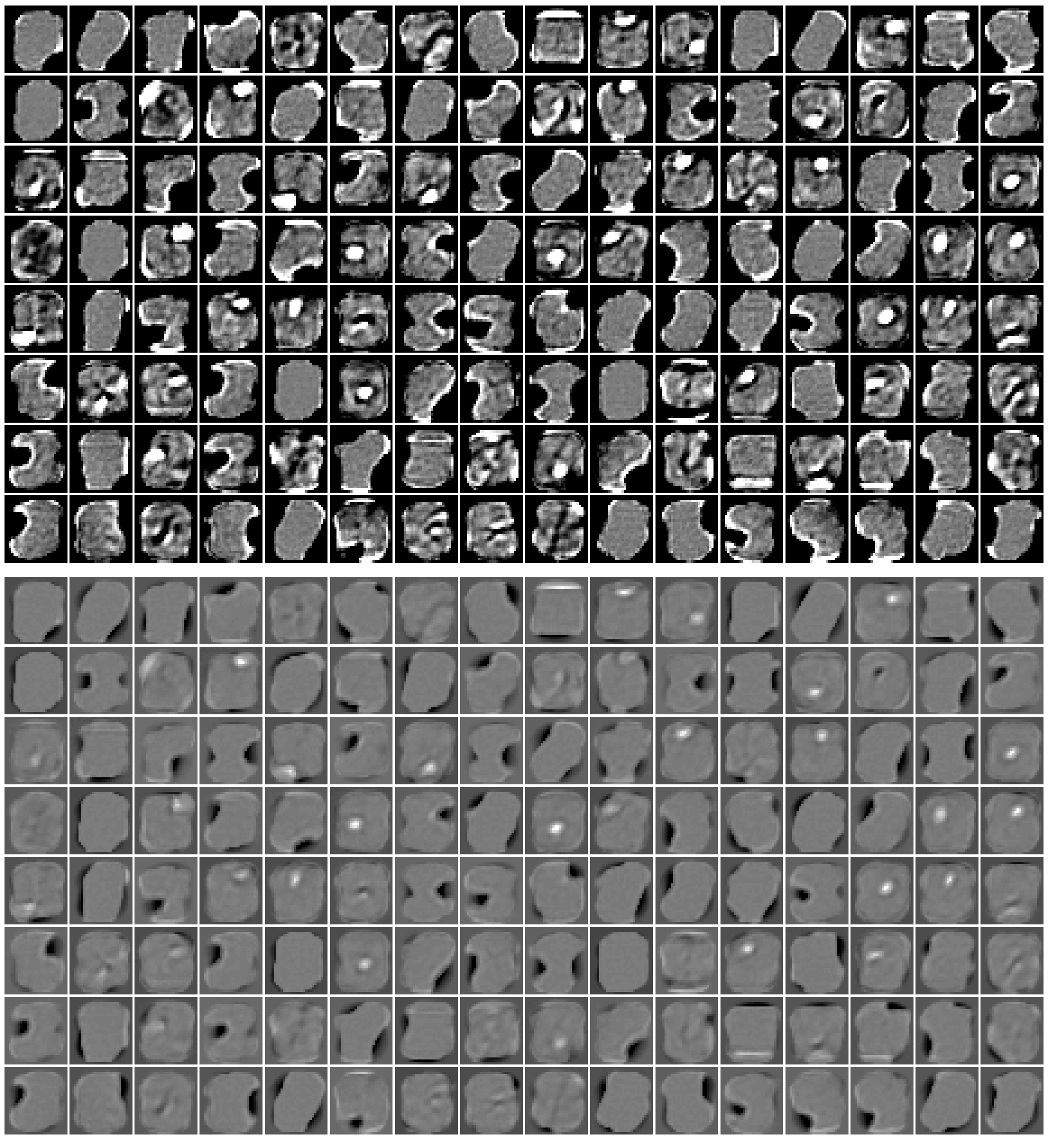


Figure 11: Input weights of a random subset of the hidden units, learned by an RBM with truncated exponential visible units, when trained on samples from the MNIST data set. The top and bottom images correspond to the same filters but with different color scale. On the top, the display setup is the same as for Figures 6 and 7 and, on the bottom, a black and white pixel correspond to weights smaller than -30 and larger than 30 respectively.

The contrastive divergence updates have the same form as for binary units of Equation 11, since the updates only depend on the derivative of the energy with respect to the parameters. Only sampling is changed, according to the unit's conditional density. Figure 11 shows the filters learned by an RBM with truncated exponential visible units, when trained on MNIST samples. Note how these are strikingly different from those obtained with binomial units.

7.2 Quadratic Energy: Gaussian Units

To obtain Gaussian-distributed units, one only needs to add quadratic terms to the energy. Adding $\sum_k d_k^2 x_k^2$ gives rise to a diagonal covariance matrix between units of the same layer, where x_k is the continuous value of a Gaussian unit and d_k^2 is a positive parameter that is equal to the inverse of the variance of x_k. In this case the variance is unconditional, whereas the mean depends on the inputs of the unit: for a visible unit x_k with hidden layer $\mathbf{h}$ and inverse variance d_k^2,

$$E\left[x_k|\mathbf{h}\right] = \frac{a(\mathbf{h})}{2d_k^2} .$$

The contrastive divergence updates are easily obtained by computing the derivative of the energy with respect to the parameters. For the parameters in the linear terms of the energy function $\mathbf{b}$, $\mathbf{c}$ and $\mathbf{W}$, the derivatives have the same form as for the case of binary units. For quadratic parameter $d_k > 0$, the derivative is simply $2d_k x_k^2$. Figure 12 shows the filters learned by an RBM with Gaussian visible units, when trained on MNIST samples.

Gaussian units were previously used as hidden units of an RBM (with multinomial inputs) applied to an information retrieval task (Welling et al., 2005). That same paper also shows how to generalize RBMs to units whose marginal distribution is from any member of the exponential family.

7.3 Impact on Classification Performance

In order to assess the impact of the choice for the visible layer distribution on the ultimate performance of an SRBM network, we trained and compared different deep networks whose first level RBM had binary, truncated exponential or Gaussian input units. These networks all had 3 hidden layers, with 2000 hidden units for each of these layers. The hyperparameters that were optimized are the unsupervised learning rates and number of updates as well as the fine-tuning learning rate. Because the assumption of binary inputs is not unreasonable for the MNIST images, we conducted this experiment on a modified and more challenging version of the data set where the background contains patches of images downloaded from the Internet. Samples from this data set are shown in Figure 13. This data set is part of a benchmark[6] designed by Larochelle et al. (2007). The results are given in Table 5, where we can see that the choice of the input distribution has a significant impact on the classification performance of the deep network. As a comparison, a support vector machine with Gaussian kernel achieves 22.61% error on this data set (Larochelle et al., 2007). Other experimental results with truncated exponential and Gaussian input units are found in Bengio et al. (2007).

8. Generating vs Encoding

Though the SRBM and SAA networks are similar in their motivation, there is a fundamental difference in the type of unsupervised learning used during training. Indeed, the RBM is based on the learning algorithm of a *generative model*, which is trained to be able to generate data similar to those found in the training set. On the other hand, the autoassociator is based on the learning algorithm of an *encoding model* which tries to learn a new representation or code from which the input can be reconstructed without too much loss of information.

6. The benchmark's data sets can be downloaded from `http://www.iro.umontreal.ca/~lisa/icml2007`.

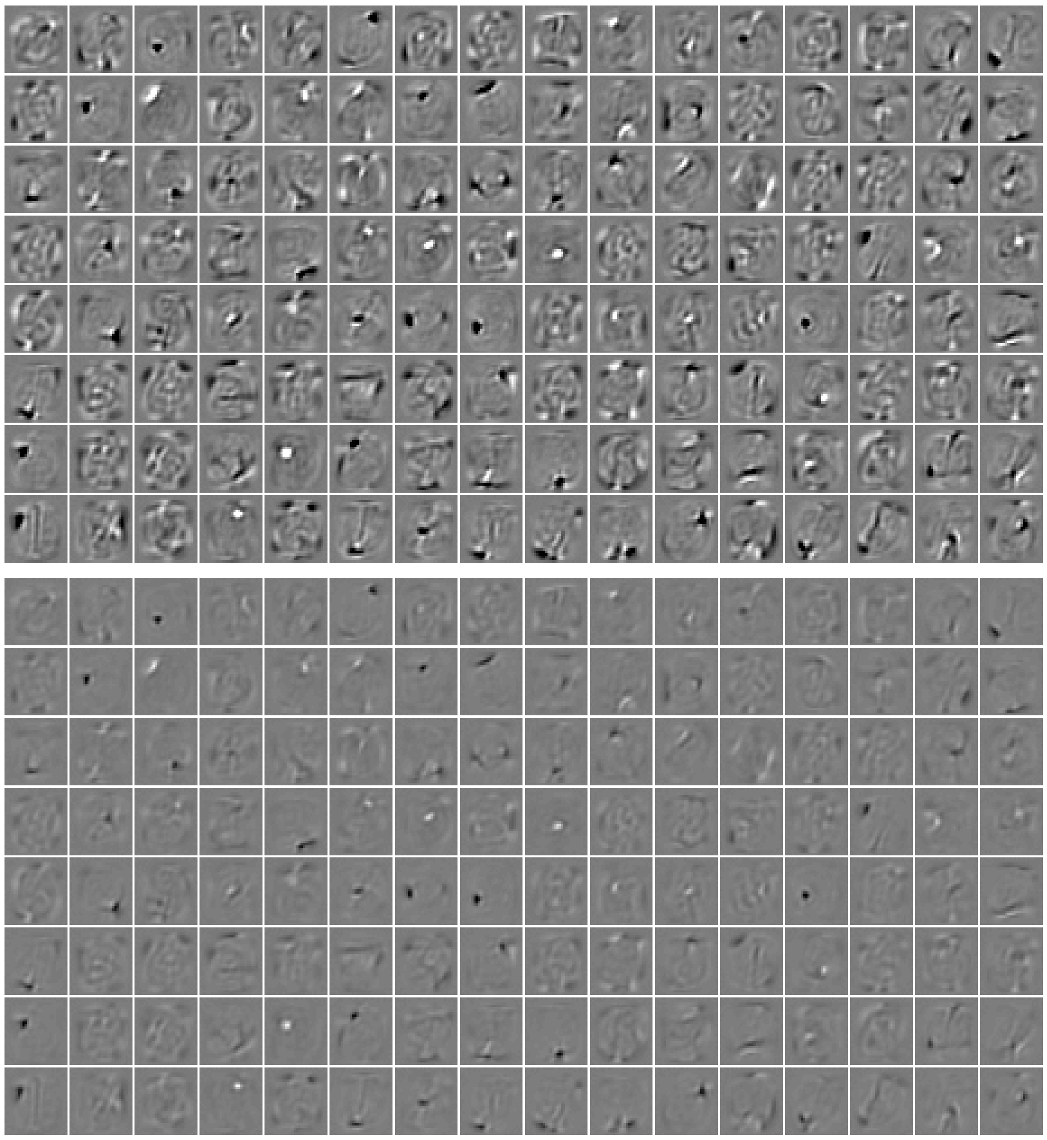


Figure 12: Input weights of a random subset of the hidden units, learned by an RBM with Gaussian visible units, when trained on samples from the MNIST data set. The top and bottom images correspond to the same filters but with different color scale. On top, the display setup is the same as for Figures 6 and 7 and, on the bottom, a black and white pixel correspond to weights smaller than -10 and larger than 10 respectively.

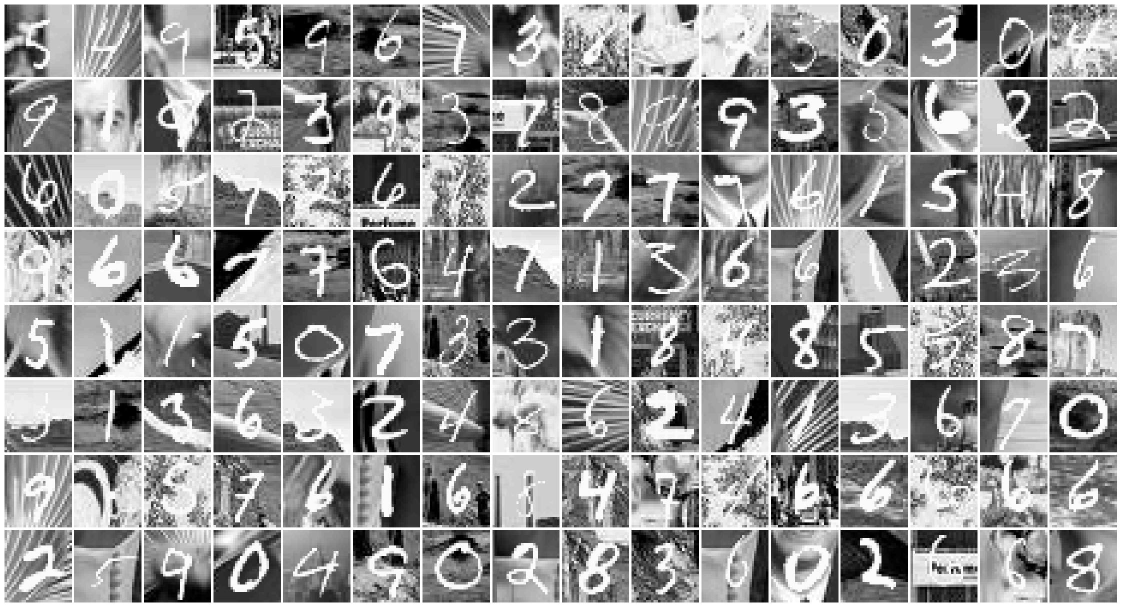


Figure 13: Samples from the modified MNIST digit recognition data set with a background containing image patches. Here, a black pixel corresponds to an input value of 0 and a white pixel corresponds to 1 (the inputs are scaled between 0 and 1).

SRBM input type	Train.	Valid.	Test
Bernoulli	10.50%	18.10%	20.29%
Gaussian	0%	20.50%	21.36%
Truncated exponential	0%	14.30%	14.34%

Table 5: Classification error on MNIST with background containing patches of images (see Figure 13) on the training, validation, and test sets, for different distributions of the input layer for the bottom RBM. The best hyperparameters were selected according to the validation error.

It is not clear which of the two approaches (generating or encoding) is the most appropriate. The advantage of a generative model is that the assumptions that are made are usually clear. However, it is possible that the problem it is trying to solve is harder than it needs to be, since ultimately we are only interested in coming up with good representations or features of the input. For instance, if one is interested in finding appropriate clusters in a very high dimensional space, using a mixture of Gaussians with full covariance matrix can quickly become too computationally intensive, whereas using the simple k-means algorithm might do a good enough job. As for encoding models, they do not require to be interpretable as a generative model and they can be more flexible because any parametric or non-parametric form can be chosen for the encoder and decoder, as long as they are differentiable.

Another interesting connection between reconstruction error in autoassociators and CD in RBMs was mentioned earlier: the reconstruction error can be seen as an estimator of the log-likelihood gradient of the RBM which has more bias but less variance than the CD update rule (Bengio and Delalleau, 2007). In that paper it is shown how to write the RBM log-likelihood gradient as a series expansion where each term is associated with a sample of the contrastive divergence Gibbs chain. Because the terms become smaller and converge to zero, this justifies taking a truncation of the series as an estimator of the gradient. The reconstruction error gradient is a mean-field (i.e., biased) approximation of the first term, whereas CD-1 is a sampling (i.e., high-variance) approximation of the first two terms, and similarly CD-k involves the first $2k$ terms.

This suggests combining the reconstruction error and contrastive divergence for training RBMs. During unsupervised pre-training, we can use the updates given by both algorithms and combine them by associating a coefficient to each of them. This is actually equivalent to applying the updates one after the other but using different learning rates for both. We tested this idea in the MNIST data set split of Section 6.1, where we had to validate separately the learning rates for the RBM and the autoassociator updates. This combination improved on the results of the SRBM and the SAA networks, obtaining 1.02% and 1.09% on the validation and test set respectively. This improvement was confirmed in a more complete experiment on 6 other folds with mutually exclusive test sets of 10 000 examples, where the mixed gradient variant gave on average a statistically significant improvement of 0.1% on a SRBM network. One possible explanation for the improvement brought by this combination is that it uses a better trade-off between bias and variance in estimating the log-likelihood gradient.

Another deterministic alternative to CD is mean-field CD (MF-CD) of Welling and Hinton (2002), and is equivalent to the pseudocode code in Appendix B, with the statements $\mathbf{h}^0 \sim p(\mathbf{h}|\mathbf{x}^0)$ and $\mathbf{v}^1 \sim p(\mathbf{x}|\mathbf{h}^0)$ changed to $\mathbf{h}^0 \leftarrow \mathrm{sigm}(\mathbf{b}+\mathbf{W}\mathbf{v}^0)$ and $\mathbf{v}^1 \leftarrow \mathrm{sigm}(\mathbf{c}+\mathbf{W}^\mathsf{T}\mathbf{h}^0)$ respectively. MF-CD can be used to test another way to change the bias/variance trade-off, either as a gradient estimator alone, or by combining it to the CD-1 gradient estimate (in the same way we combined the autoassociator gradient with CD-1, previous paragraph). On the MNIST split of Section 6.1, SRBM networks with MF-CD and combined CD-1/MF-CD[7] achieved 1.26% and 1.17% on the test set respectively. The improvement brought by combining MF-CD with CD-1 was not found to be statistically significant, based on similar experiments on the 6 other folds.

This suggests that something else than the bias/variance trade-off is at play in the improvements observed when combining CD-1 with the autoassociator gradient. A hypothesis that should be explored is that whereas there is no guarantee that an RBM will encode in its hidden representation all the information in the input vector, an autoassociator is trying to achieve this. In fact an RBM trained by maximum likelihood would be glad to completely ignore the inputs if these were independent of each other. Minimizing the reconstruction error would prevent this, and may be useful in the context where the representations are later used for supervised classification (which is the case here).

9. Continuous Training of all Layers of a Deep Network

The layer-wise training algorithm for networks of depth l actually has $l+1$ separate training phases: first the l phases for the unsupervised training of each layer, and then the final supervised fine-tuning phase to adjust all the parameters simultaneously. One element that we would like to dispense with

7. The weight of the CD-1 and MF-CD gradient estimates was considered as a hyperparameter.

is having to decide the number of unsupervised training iterations for each layer before starting the fine-tuning. One possibility is then to execute all phases simultaneously, that is, train all layers based on both their greedy unsupervised and global supervised gradients. The advantage is that we can now have a single stopping criterion (for the whole network). However, computation time is slightly greater, since we do more computations initially on the upper layers, which might be wasted before the lower layers converge to a decent representation, but time is saved on optimizing hyper-parameters. When this continuous training variant is used on the MNIST data set with the same experimental setup as in Section 6.1, we reach 1.6% and 1.5% error on the test set respectively for the SRBM network and the SAA network, so unsupervised learning still brings better generalization in this setting. This variant may be more appealing for on-line training on very large data sets, where one would never cycle back on the training data.

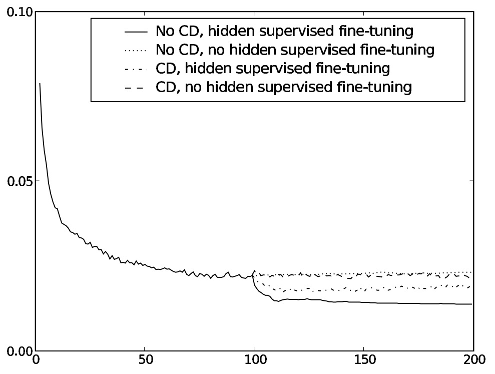
However, there seems to be a price to pay in terms of classification error, with this online variant. In order to investigate what could be the cause, we experimented with a 2-phase algorithm designed to shed some light on the contribution of different factors to this decrease. In the first phase, all layers of networks were simultaneously trained according to their unsupervised criterion *without* fine-tuning. The output layer is still trained according to the supervised criterion, however, unlike in Section 6.1, the gradient is not backpropagated into the rest of the network. This allows us to monitor the discriminative capacity of the top hidden layer. This first phase also enables us to verify whether the use of the supervised gradient too early during training explains the decrease in performance (recall the poor results obtained with purely supervised greedy layer-wise training). Then, in the second phase, 2 options were considered:

1. fine-tune the whole network according to the supervised criterion and stop layer-wise unsupervised learning;
2. fine-tune the whole network and maintain layer-wise unsupervised learning (as in the previous experiment).

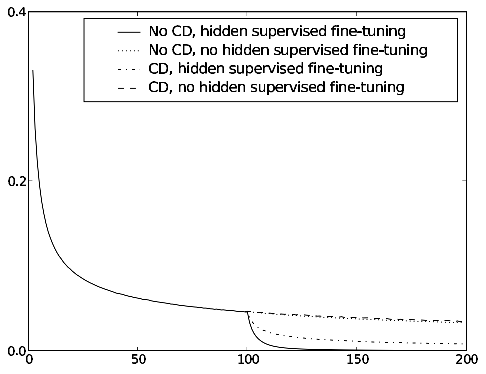
Figures 14(a) and 15(a) show examples of the progression of the test classification error for such an experiment with the SRBM and SAA networks respectively. As a baseline for the second phase, we also give the performance of the networks when unsupervised learning is stopped and only the parameters of the output layer are trained. These specific curves do not correspond to the best values of the hyperparameters, but are representative of the global picture we observed on several runs with different hyperparameter values.

We observe that the best option is to perform fine-tuning without layer-wise unsupervised learning, even when supervised learning is not introduced at the beginning. Also, though performing unsupervised and supervised learning at the same time outperforms unsupervised learning without fine-tuning, it appears to yield over-regularized networks, as indicated by the associated curves of the training negative log-likelihood of the target classes for both networks (see Figures 14(b) and 15(b)). Indeed, we see that by maintaining some unsupervised learning, the networks are not able to optimize as well their supervised training objective. From other runs with different learning rates, we have observed that this effect becomes less visible when the supervised learning rate gets larger, which reduces the relative importance of the unsupervised updates. But then the unsupervised updates usually bring significant instabilities in the learning process, making even the training cost oscillate.

Another interesting observation is that, when layer-wise unsupervised learning is performed, the classification error is less stable in an SRBM network than in an SAA network, as indicated by

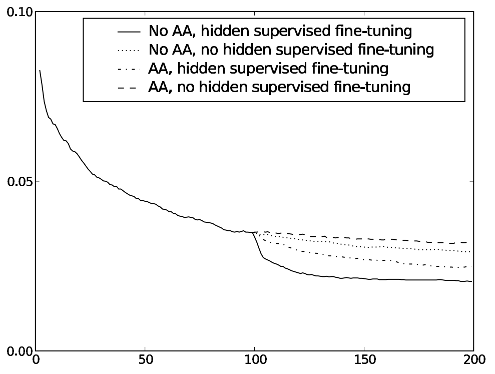


(a) SRBM network, test classification error curves

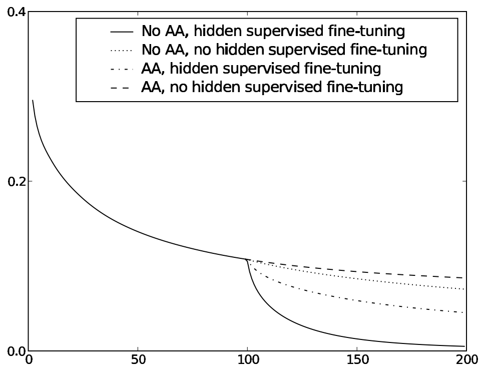


(b) SRBM network, train NLL error curves.

Figure 14: Example of learning curves of the 2-phase experiment of Section 9. During the first half of training, all hidden layers are trained according to CD and the output layer is trained according to the supervised objective, for all curves. In the second phase, all combinations of two possibilities are displayed: CD training is performed at all hidden layers ("CD") or not ("No CD"), and all hidden layers are fine-tuned according to the supervised objective ("hidden supervised fine-tuning") or not ("no hidden supervised fine-tuning").



(a) SAA network, test classification error curves



(b) SAA network, train NLL error curves.

Figure 15: Same as Figure 14, but with autoassociators ("AA") used for layer-wise unsupervised learning.

the dented learning curves in Figure 14(b), whereas the curves in Figure 15(b) are smoother. This may be related to the better performance of the SAA network (1.5%) versus the SRBM network (1.6%) when combining unsupervised and supervised gradients, in the experiment reported at the beginning of this section. Autoassociator learning might hence be more appropriate here, possibly because its training objective, that is, the discovery of a representation that preserves the information in the input, is more compatible with the supervised training objective, which asks that the network discovers a representation that is predictive of the input's class. This hypothesis is related to the one presented at the end of Section 8 regarding the apparent improvement brought by minimizing the reconstruction error in addition to CD-1 updates.

These experiments show that one can eliminate the multiple unsupervised phases: each layer can be pre-trained in a way that simply ignores what the layer above are doing. However, it appears that a final phase involving only supervised gradient yields the best performance. A plausible explanation of these results, and in particular the quick improvement when the unsupervised updates are removed, is that the unsupervised pre-training brings the parameters near a good solution for the supervised criterion, but far enough from that solution to yield a significantly higher classification error. Note that in a setting where there is little labeled data but a lot of unlabelled examples, the additional regularization introduced by maintaining some unsupervised learning might be beneficial (Salakhutdinov and Hinton, 2007b).

10. Conclusion

In this paper, we discussed in detail three principles for training deep neural networks, which are (1) pre-training one layer at a time in a greedy way (2) using unsupervised learning at each layer in a way that preserves information from the input and disentangles factors of variation and (3) fine-tuning the whole network with respect to the ultimate criterion of interest. We also presented experimental evidence that supports the claim that they are key ingredients for reaching good results. Moreover, we presented a series of experimental results that shed some light on many aspects of deep networks: confirming that the unsupervised procedure helps the optimization of the deep architecture, while initializing the parameters in a region near which a good solution of the supervised task can be found. Our experiments showed cases where greater depth clearly helps, but too much depth could be slightly detrimental. We found that CD-1 can be improved by combining it with the gradient of reconstruction error, and that this is not just due to the use of a lower-variance update. We showed that the choice of input distribution in RBMs could be important for continuous-valued input and yielded different types of filters at the first layer. Finally we studied variants more amenable to online learning in which we show that if different training phases can be combined, the best results were obtained with a final fine-tuning phase involving only the supervised gradient.

There are many questions and issues that remain to be addressed and that we intend to investigate in future work. As pointed out in Section 8, the most successful unsupervised learning approach seems to fall in between generative and encoding approaches. This raises questions about what are the properties of a learning algorithm that learns good representations for deep networks. Finding good answers to these questions would have a direct positive impact on the performance of deep networks. Finally, better model selection techniques that would permit to reduce the number of hyperparameters would be beneficial and will need to be developed for deep network learning algorithms to become easier to use.

Acknowledgments

The author are particularly grateful for the inspiration from and constructive discussions with Dan Popovici, Aaron Courville, Olivier Delalleau, James Bergstra, and Dumitru Erhan. The authors also want to thank the editor and reviewers for their helpful comments and suggestions. This research was performed thanks to funding from NSERC, MITACS, and the Canada Research Chairs.

Appendix A. Pseudocode for Greedy Layer-Wise Training Paradigm

```
**Input:** training set $\mathcal{D} = \{(\mathbf{x}_t, y_t)\}_{t=1}^{T}$, pre-training learning rate $\varepsilon_{\text{pre-train}}$ and fine-tuning learning rate $\varepsilon_{\text{fine-tune}}$

Initialize weights $\mathbf{W}^i_{jk} \sim U(-a^{-0.5}, a^{-0.5})$ with $a = \max(|\widehat{\mathbf{h}}^{i-1}|, |\widehat{\mathbf{h}}^i|)$ and set biases $\mathbf{b}^i$ to 0

% Pre-training phase
for $i \in \{1,\ldots,l\}$ do
  while Pre-training stopping criterion is not met do
    Pick input example $\mathbf{x}_t$ from training set
    $\widehat{\mathbf{h}}^0(\mathbf{x}_t) \leftarrow \mathbf{x}_t$
    for $j \in \{1,\ldots,i-1\}$ do
      $\mathbf{a}^j(\mathbf{x}_t) = \mathbf{b}^j + \mathbf{W}^j \widehat{\mathbf{h}}^{j-1}(\mathbf{x}_t)$
      $\widehat{\mathbf{h}}^j(\mathbf{x}_t) = \text{sigm}\left(\mathbf{a}^j(\mathbf{x}_t)\right)$
    end for
    Using $\widehat{\mathbf{h}}^{i-1}(\mathbf{x}_t)$ as input example, update weights $\mathbf{W}^i$ and biases $\mathbf{b}^{i-1}$, $\mathbf{b}^i$ with learning rate $\varepsilon_{\text{pre-train}}$ according to a layer-wise unsupervised criterion (see pseudocodes in appendices B and C)
  end while
end for

% Fine-tuning phase
while Fine-tuning stopping criterion is not met do
  Pick input example $(\mathbf{x}_t, y_t)$ from training set

  % Forward propagation
  $\widehat{\mathbf{h}}^0(\mathbf{x}_t) \leftarrow \mathbf{x}_t$
  for $i \in \{1,\ldots,l\}$ do
    $\mathbf{a}^i(\mathbf{x}_t) = \mathbf{b}^i + \mathbf{W}^i \widehat{\mathbf{h}}^{i-1}(\mathbf{x}_t)$
    $\widehat{\mathbf{h}}^i(\mathbf{x}_t) = \text{sigm}\left(\mathbf{a}^i(\mathbf{x}_t)\right)$
  end for
  $\mathbf{a}^{l+1}(\mathbf{x}_t) = \mathbf{b}^{l+1} + \mathbf{W}^{l+1} \widehat{\mathbf{h}}^l(\mathbf{x}_t)$
  $\mathbf{o}(\mathbf{x}_t) = \widehat{\mathbf{h}}^{l+1}(\mathbf{x}_t) = \text{softmax}\left(\mathbf{a}^{l+1}(\mathbf{x}_t)\right)$

  % Backward gradient propagation and parameter update
  $\frac{\partial \log o_{y_t}(\mathbf{x}_t)}{\partial a^{l+1}_j(\mathbf{x}_t)} \leftarrow \mathbf{1}_{y_t=j} - o_j(\mathbf{x}_t)$ for $j \in \{1,\ldots,K\}$
```

```
    $\mathbf{b}^{l+1} \leftarrow \mathbf{b}^{l+1} + \varepsilon_{\text{fine-tune}} \frac{\partial \log o_{y_t}(\mathbf{x}_t)}{\partial \mathbf{a}^{l+1}(\mathbf{x}_t)}$
    $\mathbf{W}^{l+1} \leftarrow \mathbf{W}^{l+1} + \varepsilon_{\text{fine-tune}} \frac{\partial \log o_{y_t}(\mathbf{x}_t)}{\partial \mathbf{a}^{l+1}(\mathbf{x}_t)} \widehat{\mathbf{h}}^{l}(\mathbf{x}_t)^{\mathsf{T}}$
    for $i \in \{1,\dots,l\}$, in decreasing order do
        $\frac{\partial \log o_{y_t}(\mathbf{x}_t)}{\partial \widehat{\mathbf{h}}^{i}(\mathbf{x}_t)} \leftarrow \left(\mathbf{W}^{i+1}\right)^{\mathsf{T}} \frac{\partial \log o_{y_t}(\mathbf{x}_t)}{\partial \mathbf{a}^{i+1}(\mathbf{x}_t)}$
        $\frac{\partial \log o_{y_t}(\mathbf{x}_t)}{\partial a^{i}_{j}(\mathbf{x}_t)} \leftarrow \frac{\partial \log o_{y_t}(\mathbf{x}_t)}{\partial \widehat{h}^{i}_{j}(\mathbf{x}_t)} \widehat{h}^{i}_{j}(\mathbf{x}_t)\left(1 - \widehat{h}^{i}_{j}(\mathbf{x}_t)\right)$   for $j \in \{1,\dots,|\widehat{\mathbf{h}}^{i}|\}$
        $\mathbf{b}^{i} \leftarrow \mathbf{b}^{i} + \varepsilon_{\text{fine-tune}} \frac{\partial \log o_{y_t}(\mathbf{x}_t)}{\partial \mathbf{a}^{i}}$
        $\mathbf{W}^{i} \leftarrow \mathbf{W}^{i} + \varepsilon_{\text{fine-tune}} \frac{\partial \log o_{y_t}(\mathbf{x}_t)}{\partial \mathbf{a}^{i}} \widehat{\mathbf{h}}^{i-1}(\mathbf{x}_t)^{\mathsf{T}}$
    end for
end while
```

In the first step of the gradient computation, one has to be careful to compute the gradient of the cost with respect to $\mathbf{a}^{l+1}(\mathbf{x}_t)$ at once, in order not to lose numerical precision during the computation. In particular, computing $\frac{\partial \log o_{y_t}(\mathbf{x}_t)}{\partial \mathbf{o}(\mathbf{x}_t)}$ first, then $\frac{\partial \mathbf{o}(\mathbf{x}_t)}{\partial \mathbf{a}^{l+1}(\mathbf{x}_t)}$ and applying chain-rule, leads to numerical instability and sometimes parameter value explosion (NaN).

Appendix B. Restricted Boltzmann Machines and Deep Belief Networks

In this section, we give a brief overview of restricted Boltzmann machines and deep belief networks.

B.1 Restricted Boltzmann Machine

A restricted Boltzmann machine is an energy-based generative model defined over a visible layer $\mathbf{v}$ (sometimes called input) and a hidden layer $\mathbf{h}$ (sometimes called hidden factors or representation). Given an energy function $\text{energy}(\mathbf{v},\mathbf{h})$ on the whole set of visible and hidden units, the joint probability is given by

$$p(\mathbf{v},\mathbf{h}) = \frac{e^{-\text{energy}(\mathbf{v},\mathbf{h})}}{Z} \tag{6}$$

where Z ensures that $p(\mathbf{v},\mathbf{h})$ is a valid distribution and sums to one. See Figure 3 for an illustration of an RBM.

Typically we take $h_i \in \{0,1\}$, but other choices are possible. For now, we consider only binary units, that is, $v_i \in \{0,1\}$ (the continuous case will be discussed in Section 7), where the energy function has the form

$$\text{energy}(\mathbf{v},\mathbf{h}) = -\mathbf{h}^{\mathsf{T}} W \mathbf{v} - c^{\mathsf{T}}\mathbf{v} - b^{\mathsf{T}}\mathbf{h} = -\sum_k c_k v_k - \sum_j b_j h_j - \sum_{jk} W_{jk} v_k h_j \,. \tag{7}$$

When considering the marginal distribution over $\mathbf{v}$, we obtain a mixture distribution

$$p(\mathbf{v}) = \sum_{\mathbf{h}} p(\mathbf{v},\mathbf{h}) = \sum_{\mathbf{h}} p(\mathbf{v}|\mathbf{h}) p(\mathbf{h})$$

with a number of parameters linear in the number of hidden units H, while having a number of components exponential in H. This is because $\mathbf{h}$ can take as many as 2^H possible values. The 2^H distributions $p(\mathbf{v}|\mathbf{h})$ will in general be different, but they are tied. Though computing exactly the marginal $p(\mathbf{v})$ for large values of H is impractical, a good estimator of the log-likelihood gradient have been found with the contrastive divergence algorithm described in the next section. An important property of RBMs is that the posterior distribution over one layer given the other is tractable and fast to compute, as opposed to mixture models with very many components in general. Indeed one can show that

$$p(\mathbf{v}|\mathbf{h}) = \prod_k p(v_k|\mathbf{h}) \quad \text{where} \quad p(v_k = 1|\mathbf{h}) = \text{sigm}(c_k + \sum_j W_{jk}h_j)\ , \tag{8}$$

$$p(\mathbf{h}|\mathbf{v}) = \prod_j p(h_j|\mathbf{v}) \quad \text{where} \quad p(h_j = 1|\mathbf{v}) = \text{sigm}(b_j + \sum_k W_{jk}v_k)\ . \tag{9}$$

Because of the particular parametrization of RBMs, inference of the "hidden factors" **h** given the observed input vector **v** is very easy because those factors are conditionally independent given **v**. On the other hand, unlike in many factor models (such as ICA Jutten and Herault, 1991; Comon, 1994; Bell and Sejnowski, 1995 and sigmoidal belief networks Dayan et al., 1995; Hinton et al., 1995; Saul et al., 1996), these factors are generally not marginally independent (when we integrate **v** out). Notice the similarity between Equations 9 and 1, which makes it possible to relate the weights and biases of an RBM with those of a deep neural network.

B.2 Learning in a Restricted Boltzmann Machine

To train an RBM, we would like to compute the gradient of the negative log-likelihood of the data with respect to the RBM's parameters. However, given an input example $\mathbf{v}_0$, the gradient with respect to a parameter θ in an energy-based model

$$\frac{\partial}{\partial\theta}(-\log p(\mathbf{v}_0)) = E_{p(\mathbf{h}|\mathbf{v}_0)}\left[\frac{\partial \text{energy}(\mathbf{v}_0,\mathbf{h})}{\partial\theta}\right] - E_{p(\mathbf{v},\mathbf{h})}\left[\frac{\partial \text{energy}(\mathbf{v},\mathbf{h})}{\partial\theta}\right] \tag{10}$$

necessitates a sum over all possible assignments for **h** (first expectation of Equation 10) and another sum over all assignments for **v** and **h** (second expectation). The first expectation is not problematic in an RBM because the posterior $p(\mathbf{h}|\mathbf{v}_0)$ and $\frac{\partial \text{energy}(\mathbf{v}_0,\mathbf{h})}{\partial\theta}$ factorize. However, the second expectation requires a prohibitive exponential sum over the possible configurations for **v** or **h**.

Fortunately, there exists an approximation for this gradient given by the contrastive divergence (CD) algorithm (Hinton, 2002), which has been shown to work well empirically (Carreira-Perpiñan and Hinton, 2005). There are two key elements in this approximation. First, consider that in order to estimate the second term of Equation 10, we could replace the expectation by a unique evaluation of the gradient $\frac{\partial \text{energy}(\mathbf{v},\mathbf{h})}{\partial\theta}$ at a particular pair of values $(\mathbf{v},\mathbf{h})$. This pair should ideally be sampled from the distribution $p(\mathbf{v},\mathbf{h})$, which would make the estimation of the gradient unbiased. However, sampling exactly from an RBM distribution is not as easy as in a directed graphical model. Instead, we have to rely on sampling methods such as Markov Chain Monte Carlo methods. For an RBM, we can use Gibbs sampling based on the conditional distributions of Equations 8 and 9, but this method can be costly if the Markov chain mixes slowly. So the second key idea is to run only a few iterations of Gibbs sampling and use the data sample $\mathbf{v}_0$ as the initial state for the chain at the visible layer. It turns out that applying only one iteration of the Markov chain works well in practice. This corresponds to the following sampling procedure:

$$\mathbf{v}_0 \overset{p(\mathbf{h}_0|\mathbf{v}_0)}{\longrightarrow} \mathbf{h}_0 \overset{p(\mathbf{v}_1|\mathbf{h}_0)}{\longrightarrow} \mathbf{v}_1 \overset{p(\mathbf{h}_1|\mathbf{x}_1)}{\longrightarrow} \mathbf{h}_1$$

where $\overset{p(\mathbf{h}_i|\mathbf{v}_i)}{\longrightarrow}$ and $\overset{p(\mathbf{v}_{i+1}|\mathbf{h}_i)}{\longrightarrow}$ represent the operations of sampling from $p(\mathbf{h}_i|\mathbf{v}_i)$ and $p(\mathbf{v}_{i+1}|\mathbf{h}_i)$ respectively. Estimation of the gradient using the above sampling procedure is noted CD-1, with CD-k referring to the contrastive divergence algorithm, performing k iterations of the Markov chain up to $\mathbf{v}_k$. Training with CD-k has been shown to empirically approximate well training with the exact log-likelihood gradient (Carreira-Perpiñan and Hinton, 2005). Furthermore, it can be shown that the CD-k update is an unbiased estimator of the truncation of a series expansion of the log-likelihood gradient (Bengio and Delalleau, 2007), where the truncated part of the series converges to 0 as k increases.

Now let us consider the estimation of the gradient on a weight W_{jk}. We have

$$\frac{\partial \text{energy}(\mathbf{v},\mathbf{h})}{\partial W_{jk}} = -h_j v_k$$

which means that the CD-1 estimate of the gradient becomes

$$-E_{p(\mathbf{h}|\mathbf{v}_0)}\left[h_j v_{0k}\right] + E_{p(\mathbf{h}|\mathbf{v}_1)}\left[h_{1j} v_{1k}\right] = -p(h_j|\mathbf{v}_0)v_{0k} + p(h_j|\mathbf{v}_1)v_{1k}\ . \tag{11}$$

This is similar to what was presented in Hinton et al. (2006) except that we clarify here that we take the expected value of $\mathbf{h}$ given $\mathbf{v}$ instead of averaging over samples. These estimators have the same expected value because

$$E_{p(\mathbf{v},\mathbf{h})}\left[h_j v_k\right] = E_{p(\mathbf{v})}\left[E_{p(\mathbf{h}|\mathbf{v})}\left[h_j v_k\right]\right] = E_{p(\mathbf{v})}\left[p(h_j|\mathbf{v})v_k\right]\ .$$

Using $p(\mathbf{h}|\mathbf{v}_k)$ instead of $\mathbf{h}_k$ is also what is found in the Matlab code distributed with Hinton and Salakhutdinov (2006). Note that it is still necessary to sample $\mathbf{h}_0 \sim p(\mathbf{h}|\mathbf{v}_0)$ in order to sample $\mathbf{v}_1$, but it is not necessary to sample $\mathbf{h}_1$. The above gradient estimator can then be used to perform stochastic gradient descent by iterating through all vectors $\mathbf{v}_0$ of the training set and performing a parameter update using that gradient estimator in an on-line fashion. Gradient estimators for the biases b_k and c_j can as easily be derived from Equation 10.

Notice also that, even if $\mathbf{v}_0$ is not binary, the formula for the CD-1 estimate of the gradient does not change and is still computed essentially in the same way: only the sampling procedure for $p(\mathbf{v}|\mathbf{h})$ changes (see Section 7 for more details about dealing with continuous-valued inputs). The CD-1 training update for a given training input is detailed by the pseudocode in the next section.

In our implementation of the greedy layer-wise initialization phase, we use the deterministic sigmoidal outputs of the previous level as training vector for the next level RBM. By interpreting these real-valued components as probabilities, learning such a distribution for binary inputs can be seen as a crude "mean-field" way of dealing with probabilistic binary inputs (instead of summing or sampling across input configurations).

B.3 Pseudocode for Contrastive Divergence (CD-1) Training Update

```
**Input:** training input $\mathbf{x}$, RBM weights $\mathbf{W}^i$ and biases $\mathbf{b}^{i-1}, \mathbf{b}^i$ and learning rate $\epsilon$
**Notation:** $a \sim p(\cdot)$ means set $a$ equal to a random sample from $p(\cdot)$

% Set RBM parameters
$\mathbf{W} \leftarrow \mathbf{W}^i, \mathbf{b} \leftarrow \mathbf{b}^i, \mathbf{c} \leftarrow \mathbf{b}^{i-1}$
```

```
% Positive phase
$\mathbf{v}^0 \leftarrow \mathbf{x}$
$\widehat{\mathbf{h}}^0 \leftarrow \mathrm{sigm}(\mathbf{b} + \mathbf{W}\mathbf{v}^0)$

% Negative phase
$\mathbf{h}^0 \sim p(\mathbf{h}|\mathbf{v}^0)$ according to Equation 9
$\mathbf{v}^1 \sim p(\mathbf{v}|\mathbf{h}^0)$ according to Equation 8
$\widehat{\mathbf{h}}^1 \leftarrow \mathrm{sigm}(\mathbf{b} + \mathbf{W}\mathbf{v}^1)$

% Update
$\mathbf{W}^i \leftarrow \mathbf{W}^i + \varepsilon \left(\widehat{\mathbf{h}}^0 \left(\mathbf{v}^0\right)^\mathsf{T} - \widehat{\mathbf{h}}^1 \left(\mathbf{v}^1\right)^\mathsf{T}\right)$
$\mathbf{b}^i \leftarrow \mathbf{b}^i + \varepsilon \left(\widehat{\mathbf{h}}^0 - \widehat{\mathbf{h}}^1\right)$
$\mathbf{b}^{i-1} \leftarrow \mathbf{b}^{i-1} + \varepsilon \left(\mathbf{v}^0 - \mathbf{v}^1\right)$
```

B.4 Deep Belief Network

We wish to make a quick remark on the distinction between the SRBM network and the more widely known deep belief network (DBN) (Hinton et al., 2006), which is not a feed-forward neural network but a multi-layer generative model. The SRBM network was initially derived (Hinton et al., 2006) from the DBN, for which stacking RBMs also provides a good initialization.

A DBN is a generative model with several layers of stochastic units. It actually corresponds to a sigmoid belief network (Neal, 1992) of $l-1$ hidden layers, where the prior over its top hidden layer $\mathbf{h}^{l-1}$ (second factor of Equation 12) is an RBM, which itself has a hidden layer $\mathbf{h}^l$. More precisely, it defines a distribution over an input layer $\mathbf{x}$ and l layers of binary stochastic units $\mathbf{h}^i$ as follows:

$$p(\mathbf{x}, \mathbf{h}^1, \dots, \mathbf{h}^l) = \left(\prod_{i=1}^{l-1} p(\mathbf{h}^{i-1}|\mathbf{h}^i)\right) p(\mathbf{h}^{l-1}, \mathbf{h}^l) \tag{12}$$

where hidden units are conditionally independent given the units in the above layer

$$p(\mathbf{h}^{i-1}|\mathbf{h}^i) = \prod_k p(h_k^{i-1}|\mathbf{h}^i)\ .$$

To process binary values, Bernoulli layers can be used, which correspond to equations

$$p(h_k^{i-1} = 1|\mathbf{h}^i) = \mathrm{sigm}\left(b_k^{i-1} + \sum_j W_{jk}^i h_j^i\right)$$

where $\mathbf{h}^0 = \mathbf{x}$ is the input. We also have

$$p(\mathbf{h}^{l-1}, \mathbf{h}^l) \propto e^{\sum_j c_j^{l-1} h_j^{l-1} + \sum_k b_k^l h_k^l + \sum_{jk} W_{jk}^l h_j^{l-1} h_k^l} \tag{13}$$

for the top RBM. Note that Equation 13 can be obtained from Equations 6 and 7, by naming $\mathbf{v}$ as $\mathbf{h}^{l-1}$, and $\mathbf{h}$ as $\mathbf{h}^l$.

We emphasize the distinction between $\mathbf{h}^i$ and $\widehat{\mathbf{h}}^i(\mathbf{x})$, where the former is a random variable and the latter is the representation of an input $\mathbf{x}$ at the i-th hidden layer of the network obtained from the repeated application of Equation 1.

To train such a generative model, Hinton et al. (2006) proposed the pre-training phase of the SRBM network. When all layers of a DBN have the same size, it was actually shown that this initialization improves a lower bound on the likelihood of the data as the DBN is made deeper. After this pre-training phase is over, Hinton et al. (2006) propose a variant of the Wake-Sleep algorithm for sigmoid belief networks (Hinton et al., 1995) to fine-tune the generative model.

By noticing the similarity between the process of approximating the posterior $p(\mathbf{h}^i|\mathbf{x})$ in a deep belief network and computing the hidden layer representation of an input $\mathbf{x}$ in a deep network, Hinton (2006) then proposed the use of the greedy layer-wise pre-training procedure for deep belief networks to initialize a deep feed-forward neural network, which corresponds to the SRBM network described in this paper.

Appendix C. Pseudocode of Autoassociator Training Update

```
**Input:** training input $\mathbf{x}$, autoassociator weights $\mathbf{W}^i$ and biases $\mathbf{b}^{i-1}, \mathbf{b}^i$ and learning rate $\varepsilon$

% Set autoassociator parameters
$\mathbf{W} \leftarrow \mathbf{W}^i, \mathbf{b} \leftarrow \mathbf{b}^i, \mathbf{c} \leftarrow \mathbf{b}^{i-1}$

% Forward propagation
$\mathbf{a}(\mathbf{x}) \leftarrow \mathbf{b} + \mathbf{W}\mathbf{x}$
$\mathbf{h}(\mathbf{x}) \leftarrow \mathrm{sigm}(\mathbf{a}(\mathbf{x}))$
$\widehat{\mathbf{a}}(\mathbf{x}) \leftarrow \mathbf{c} + \mathbf{W}^{\mathsf{T}}\mathbf{h}(\mathbf{x})$
$\widehat{\mathbf{x}} \leftarrow \mathrm{sigm}(\widehat{\mathbf{a}}(\mathbf{x}))$

% Backward gradient propagation
$\frac{\partial C(\widehat{\mathbf{x}},\mathbf{x})}{\partial \widehat{\mathbf{a}}(\mathbf{x})} \leftarrow \widehat{\mathbf{x}} - \mathbf{x}$
$\frac{\partial C(\widehat{\mathbf{x}},\mathbf{x})}{\partial \widehat{h}(\mathbf{x})} \leftarrow \mathbf{W}\frac{\partial C(\widehat{\mathbf{x}},\mathbf{x})}{\partial \widehat{\mathbf{a}}(\mathbf{x})}$
$\frac{\partial C(\widehat{\mathbf{x}},\mathbf{x})}{\partial a_j(\mathbf{x})} \leftarrow \frac{\partial C(\widehat{\mathbf{x}},\mathbf{x})}{\partial \widehat{h}_j(\mathbf{x})}\widehat{h}_j(\mathbf{x})\left(1 - \widehat{h}_j(\mathbf{x})\right)$ **for** $j \in \{1, \ldots, |\widehat{\mathbf{h}}(\mathbf{x})|\}$

% Update
$\mathbf{W}^i \leftarrow \mathbf{W}^i - \varepsilon\left(\frac{\partial C(\widehat{\mathbf{x}},\mathbf{x})}{\partial \mathbf{a}(\mathbf{x})}\mathbf{x}^{\mathsf{T}} + \widehat{\mathbf{h}}(\mathbf{x})\frac{\partial C(\widehat{\mathbf{x}},\mathbf{x})}{\partial \widehat{\mathbf{a}}(\mathbf{x})}^{\mathsf{T}}\right)$
$\mathbf{b}^i \leftarrow \mathbf{b}^i - \varepsilon\frac{\partial C(\widehat{\mathbf{x}},\mathbf{x})}{\partial \mathbf{a}(\mathbf{x})}$
$\mathbf{b}^{i-1} \leftarrow \mathbf{b}^{i-1} - \varepsilon\frac{\partial C(\widehat{\mathbf{x}},\mathbf{x})}{\partial \widehat{\mathbf{a}}(\mathbf{x})}$
```

References

Rie Kubota Ando and Tong Zhang. A framework for learning predictive structures from multiple tasks and unlabeled data. *Journal of Machine Learning Research*, 6:1817–1853, 2005.

Peter Auer, Mark Herbster, and Manfred K. Warmuth. Exponentially many local minima for single neurons. In M. Mozer, D. S. Touretzky, and M. Perrone, editors, *Advances in Neural Information Processing System 8*, pages 315–322. MIT Press, Cambridge, MA, 1996.

Pierre Baldi and Kurt Hornik. Neural networks and principal component analysis: Learning from examples without local minima. *Neural Networks*, 2:53–58, 1989.

Anthony J. Bell and Terrence J. Sejnowski. An information maximisation approach to blind separation and blind deconvolution. *Neural Computation*, 7(6):1129–1159, 1995.

Yoshua Bengio. Learning deep architectures for AI. Technical Report 1312, Université de Montréal, dept. IRO, 2007.

Yoshua Bengio and Olivier Delalleau. Justifying and generalizing contrastive divergence. Technical Report 1311, Dept. IRO, Université de Montréal, 2007.

Yoshua Bengio and Yann Le Cun. Scaling learning algorithms towards AI. In L. Bottou, O. Chapelle, D. DeCoste, and J. Weston, editors, *Large Scale Kernel Machines*. MIT Press, 2007.

Yoshua Bengio, Olivier Delalleau, and Nicolas Le Roux. The curse of highly variable functions for local kernel machines. In Y. Weiss, B. Schölkopf, and J. Platt, editors, *Advances in Neural Information Processing Systems 18*, pages 107–114. MIT Press, Cambridge, MA, 2006.

Yoshua Bengio, Pascal Lamblin, Dan Popovici, and Hugo Larochelle. Greedy layer-wise training of deep networks. In B. Schölkopf, J. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems 19*, pages 153–160. MIT Press, 2007.

Guillaume Bouchard and Bill Triggs. The tradeoff between generative and discriminative classifiers. In *IASC International Symposium on Computational Statistics (COMPSTAT)*, pages 721–728, Prague, August 2004. URL `http://lear.inrialpes.fr/pubs/2004/BT04`.

Miguel A. Carreira-Perpiñan and Geoffrey E. Hinton. On contrastive divergence learning. In Robert G. Cowell and Zoubin Ghahramani, editors, *Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics, Jan 6-8, 2005, Savannah Hotel, Barbados*, pages 33–40. Society for Artificial Intelligence and Statistics, 2005.

Hsin Chen and Alan F. Murray. A continuous restricted Boltzmann machine with an implementable training algorithm. *IEE Proceedings of Vision, Image and Signal Processing*, 150(3):153–158, 2003.

Ronan Collobert and Jason Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the Twenty-fifth International Conference on Machine Learning (ICML 2008)*, pages 160–167, 2008. URL `http://www.kyb.tuebingen.mpg.de/bs/people/weston/papers/unified\-nlp.pdf`.

Pierre Comon. Independent component analysis - a new concept? *Signal Processing*, 36:287–314, 1994.

Garrison W. Cottrell, Paul Munro, and David Zipser. Learning internal representations from gray-scale images: An example of extensional programming. In *Ninth Annual Conference of the Cognitive Science Society*, pages 462–473, Seattle 1987, 1987. Lawrence Erlbaum, Hillsdale.

Peter Dayan, Geoffrey Hinton, Radford Neal, and Rich Zemel. The Helmholtz machine. *Neural Computation*, 7:889–904, 1995.

David DeMers and Garrison W. Cottrell. Non-linear dimensionality reduction. In C.L. Giles, S.J. Hanson, and J.D. Cowan, editors, *Advances in Neural Information Processing Systems 5*, pages 580–587, San Mateo CA, 1993. Morgan Kaufmann.

Scott E. Fahlman and Christian Lebiere. The cascade-correlation learning architecture. In D.S. Touretzky, editor, *Advances in Neural Information Processing Systems 2*, pages 524–532, Denver, CO, 1990. Morgan Kaufmann, San Mateo.

Ildiko E. Frank and Jerome H. Friedman. A statistical view of some chemometrics regression tools. *Technometrics*, 35(2):109–148, 1993.

Brendan J. Frey. *Graphical models for machine learning and digital communication*. MIT Press, 1998.

Kenji Fukumizu and Shun-ichi Amari. Local minima and plateaus in hierarchical structures of multilayer perceptrons. *Neural Networks*, 13(3):317–327, 2000.

Raia Hadsell, Ayse Erkan, Pierre Sermanet, Marco Scoffier, Urs Muller, and Yann LeCun. Deep belief net learning in a long-range vision system for autonomous off-road driving. In *Proc. Intelligent Robots and Systems (IROS'08)*, 2008. URL `http://www.cs.nyu.edu/~raia/docs/iros08-farod.pdf`.

Johan Håstad. Almost optimal lower bounds for small depth circuits. In *Proceedings of the 18th annual ACM Symposium on Theory of Computing*, pages 6–20, Berkeley, California, 1986. ACM Press.

Johan Hastad and M. Goldmann. On the power of small-depth threshold circuits. *Computational Complexity*, 1:113–129, 1991.

Geoffrey E. Hinton. Training products of experts by minimizing contrastive divergence. *Neural Computation*, 14:1771–1800, 2002.

Geoffrey E. Hinton. Connectionist learning procedures. *Artificial Intelligence*, 40:185–234, 1989.

Geoffrey E. Hinton. To recognize shapes, first learn to generate images. Technical Report UTML TR 2006-003, University of Toronto, 2006.

Geoffrey E. Hinton and Ruslan R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, July 2006.

Geoffrey E. Hinton, Peter Dayan, Brendan J. Frey, and Radford M. Neal. The wake-sleep algorithm for unsupervised neural networks. *Science*, 268:1558–1161, 1995.

Goeffrey E. Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neural Computation*, 18:1527–1554, 2006.

Alex Holub and Pietro Perona. A discriminative framework for modelling object classes. In *CVPR '05: Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 1*, pages 664–671, Washington, DC, USA, 2005. IEEE Computer Society. ISBN 0-7695-2372-2. doi: http://dx.doi.org/10.1109/CVPR.2005.25.

Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2:359–366, 1989.

Tommi S. Jaakkola and David Haussler. Exploiting generative models in discriminative classifiers. In M.S. Kearns, S.A. Solla, and D.A. Cohn, editors, *Advances in Neural Information Processing Systems 11*. MIT Press, Cambridge, MA, 1999.

Tony Jebara. *Machine Learning: Discriminative and Generative (The Kluwer International Series in Engineering and Computer Science)*. Springer, December 2003. ISBN 1402076479. URL `http://www.amazon.ca/exec/obidos/redirect?tag=citeulike09-20&path=ASIN/1402076479`.

Christian Jutten and Jeanny Herault. Blind separation of sources, part I: an adaptive algorithm based on neuromimetic architecture. *Signal Processing*, 24:1–10, 1991.

Hugo Larochelle and Yoshua Bengio. Classification using discriminative restricted boltzmann machines. In Andrew McCallum and Sam Roweis, editors, *Proceedings of the 25th Annual International Conference on Machine Learning (ICML 2008)*, pages 536–543. Omnipress, 2008.

Hugo Larochelle, Dumitru Erhan, Aaron Courville, James Bergstra, and Yoshua Bengio. An empirical evaluation of deep architectures on problems with many factors of variation. In Zoubin Ghahramani, editor, *Twenty-fourth International Conference on Machine Learning (ICML 2007)*, pages 473–480. Omnipress, 2007. URL `http://www.machinelearning.org/proceedings/icml2007/papers/331.pdf`.

Julia A. Lasserre, Christopher M. Bishop, and Thomas P. Minka. Principled hybrids of generative and discriminative models. In *CVPR '06: Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 87–94, Washington, DC, USA, 2006. IEEE Computer Society. ISBN 0-7695-2597-0. doi: http://dx.doi.org/10.1109/CVPR.2006.227.

Yann LeCun, Leon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, November 1998.

Régis Lengellé and Thierry Denoeux. Training MLPs layer by layer using an objective function for internal representations. *Neural Networks*, 9:83–97, 1996.

Javier R. Movellan, Paul Mineiro, and R. J. Williams. A monte-carlo EM approach for partially observable diffusion processes: theory and applications to neural networks. *Neural Computation*, 14:1501–1544, 2002.

Radford M. Neal. Connectionist learning of belief networks. *Artificial Intelligence*, 56:71–113, 1992.

Andrew Y. Ng and Michael I. Jordan. On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. In *NIPS*, pages 841–848, 2001.

Simon Osindero and Geoffrey E. Hinton. Modeling image patches with a directed hierarchy of markov random field. In *Neural Information Processing Systems Conference (NIPS) 20*, 2008.

Marc'Aurelio Ranzato, Fu-Jie Huang, Y-Lan Boureau, and Yann LeCun. Unsupervised learning of invariant feature hierarchies with applications to object recognition. In *Proc. Computer Vision and Pattern Recognition Conference (CVPR'07)*. IEEE Press, 2007a.

Marc'Aurelio Ranzato, Christopher Poultney, Sumit Chopra, and Yann LeCun. Efficient learning of sparse representations with an energy-based model. In B. Schölkopf, J. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems 19*. MIT Press, 2007b.

Marc'Aurelio Ranzato, Y-Lan Boureau, and Yann LeCun. Sparse feature learning for deep belief networks. In J.C. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems 20*. MIT Press, Cambridge, MA, 2008. URL `http://www.cs.nyu.edu/~ranzato/publications/ranzato-nips07.pdf`.

Ruslan Salakhutdinov and Geoffrey Hinton. Using deep belief nets to learn covariance kernels for gaussian processes. In J. C. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems 20*. MIT Press, Cambridge, MA, 2008. URL `http://www.csri.utoronto.ca/~hinton/absps/dbngp.pdf`.

Ruslan Salakhutdinov and Geoffrey Hinton. Semantic hashing. In *Proceedings of the 2007 Workshop on Information Retrieval and applications of Graphical Models (SIGIR 2007)*, Amsterdam, 2007a. Elsevier.

Ruslan Salakhutdinov and Geoffrey Hinton. Learning a nonlinear embedding by preserving class neighbourhood structure. In *Proceedings of AISTATS 2007*, San Juan, Porto Rico, 2007b. Omnipress.

Ruslan Salakhutdinov and Iain Murray. On the quantitative analysis of deep belief networks. In *Proceedings of the International Conference on Machine Learning*, volume 25, 2008.

Ruslan Salakhutdinov, Andriy Mnih, and Geoffrey Hinton. Restricted boltzmann machines for collaborative filtering. In *ICML '07: Proceedings of the 24th international conference on Machine learning*, pages 791–798, New York, NY, USA, 2007. ACM.

Lawrence K. Saul, Tommi Jaakkola, and Michael I. Jordan. Mean field theory for sigmoid belief networks. *Journal of Artificial Intelligence Research*, 4:61–76, 1996.

Eric Saund. Dimensionality-reduction using connectionist networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(3):304–314, 1989.

Paul Smolensky. Information processing in dynamical systems: Foundations of harmony theory. In D. E. Rumelhart and J. L. McClelland, editors, *Parallel Distributed Processing*, volume 1, chapter 6, pages 194–281. MIT Press, Cambridge, 1986.

Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In Andrew McCallum and Sam Roweis, editors, *Proceedings of the 25th Annual International Conference on Machine Learning (ICML 2008)*, pages 1096–1103. Omnipress, 2008. URL `http://icml2008.cs.helsinki.fi/papers/592.pdf`.

Ingo Wegener. *The Complexity of Boolean Functions*. John Wiley & Sons, 1987.

Max Welling and Geoffrey E. Hinton. A new learning algorithm for mean field boltzmann machines. In *ICANN '02: Proceedings of the International Conference on Artificial Neural Networks*, pages 351–357, London, UK, 2002. Springer-Verlag. ISBN 3-540-44074-7.

Max Welling, Michal Rosen-Zvi, and Geoffrey E. Hinton. Exponential family harmoniums with an application to information retrieval. In L.K. Saul, Y. Weiss, and L. Bottou, editors, *Advances in Neural Information Processing Systems 17*. MIT Press, 2005.

Jason Weston, Frédéric Ratle, and Ronan Collobert. Deep learning via semi-supervised embedding. In *Proceedings of the Twenty-fifth International Conference on Machine Learning (ICML 2008)*, 2008. URL `http://www.kyb.tuebingen.mpg.de/bs/people/weston/papers/deep-embed.pdf`.

Andrew Yao. Separating the polynomial-time hierarchy by oracles. In *Proceedings of the 26th Annual IEEE Symposium on Foundations of Computer Science*, pages 1–10, 1985.

Journal of Machine Learning Research 10 (2009) 41-70 Submitted 7/07; Revised 9/08; Published 1/09

Markov Properties for Linear Causal Models with Correlated Errors

Changsung Kang CSKANG@CS.IASTATE.EDU
Jin Tian JTIAN@CS.IASTATE.EDU
Department of Computer Science
Iowa State University
Ames, IA 50011, USA

Editor: Andre Elisseeff

Abstract

A linear causal model with correlated errors, represented by a DAG with bi-directed edges, can be tested by the set of conditional independence relations implied by the model. A global Markov property specifies, by the d-separation criterion, the set of all conditional independence relations holding in any model associated with a graph. A local Markov property specifies a much smaller set of conditional independence relations which will imply all other conditional independence relations which hold under the global Markov property. For DAGs with bi-directed edges associated with arbitrary probability distributions, a local Markov property is given in Richardson (2003) which may invoke an exponential number of conditional independencies. In this paper, we show that for a class of linear structural equation models with correlated errors, there is a local Markov property which will invoke only a linear number of conditional independence relations. For general linear models, we provide a local Markov property that often invokes far fewer conditional independencies than that in Richardson (2003). The results have applications in testing linear structural equation models with correlated errors.

Keywords: Markov properties, linear causal models, linear structural equation models, graphical models

1. Introduction

Linear causal models called structural equation models (SEMs) are widely used for causal reasoning in social sciences, economics, and artificial intelligence (Goldberger, 1972; Bollen, 1989; Spirtes et al., 2001; Pearl, 2000). One important problem in the applications of linear causal models is testing a hypothesized model against the given data. While the conventional method involves maximum likelihood estimation of the covariance matrix, an alternative approach has been proposed recently which involves testing for the conditional independence relationships implied by the model (Spirtes et al., 1998; Pearl, 1998; Pearl and Meshkat, 1999; Pearl, 2000; Shipley, 2000, 2003). The advantages of using this new test method instead of the traditional global fitting test have been discussed in Pearl (1998), Shipley (2000), McDonald (2002) and Shipley (2003). The method can be applied in small data samples and it can test "local" features of the model.

To apply this test method, one needs to be able to identify the conditional independence relationships implied by an SEM. This can be achieved by representing the SEM with a graph called a path diagram (Wright, 1934) and then reading independence relations from the path diagram. For a linear SEM without correlated errors, the corresponding path diagram is a directed acyclic graph (DAG). The set of all conditional independence relations holding in any model associated with a

DAG, often called a global Markov property for the DAG, can be read by the d-separation criterion (Pearl, 1988). However, it is not necessary to test for all the independencies implied by the model as a subset of those independencies may imply all others. A local Markov property specifies a much smaller set of conditional independence relations which will imply (using the laws of probability) all other conditional independence relations that hold under the global Markov property. A well-known local Markov property for DAGs is that each variable is conditionally independent of its non-descendants given its parents (Lauritzen et al., 1990; Lauritzen, 1996). Based on this local Markov property, Pearl and Meshkat (1999) and Shipley (2000) proposed testing methods for linear SEMs without correlated errors that involve at most one conditional independence test for each pair of variables.

On the other hand, the path diagrams for linear SEMs with correlated errors are DAGs with bi-directed edges ($\leftrightarrow$) where bi-directed edges are used to represent correlated errors. A DAG with bi-directed edges is called an *acyclic directed mixed graph (ADMG)* in Richardson (2003). The set of all conditional independence relations encoded in an ADMG can still be read by (a natural extension of) the d-separation criterion (called m-separation in Richardson, 2003) which provides the global Markov property for ADMGs (Spirtes et al., 1998; Koster, 1999; Richardson, 2003). A local Markov property for ADMGs is given in Richardson (2003), which, in the worst case, may invoke an exponential number of conditional independence relations, a sharp difference with the local Markov property for DAGs, where only one conditional independence relation is associated with each variable. Shipley (2003) suggested a method for testing linear SEMs with correlated errors but the method may or may not, depending on the actual models, be able to find a subset of conditional independence relations that imply all others.

In this paper, we seek to improve the local Markov property given in Richardson (2003) for linear SEMs with correlated errors. The local Markov property in Richardson (2003) is applicable for ADMGs associated with arbitrary probability distributions. Specifically, only semi-graphoid axioms which must hold in all probability distributions (Pearl, 1988) are used in showing that the set of conditional independence relations specified by the local Markov property will imply all those specified by the global Markov property. On the other hand, in linear SEMs, variables are assumed to have normal distributions, and it is known that normal distributions also satisfy the so-called composition axiom. Therefore, in this paper, we look for local Markov properties for ADMGs associated with probability distributions that satisfy the composition axiom. We will show that for a class of ADMGs, the local Markov property will invoke only one conditional independence relation for each variable, and therefore testing for the corresponding linear SEMs will involve at most one conditional independence test for each pair of variables. For general ADMGs, we provide a procedure that reduces the number of conditional independencies invoked by the local Markov property given in Richardson (2003), and therefore reduces the complexity of testing linear SEMs with correlated errors.

In the test of conditional independence relations, the efficiency of the test is influenced by the size of the conditioning set (that is, the number of conditioning variables) with a small conditioning set having advantage over a large one. The conditional independence relations invoked by the standard local Markov property for DAGs use a parent set as the conditioning set. Pearl and Meshkat (1999) have shown for linear SEMs without correlated errors how to find a set of conditional independence relations that may involve fewer conditioning variables. In this paper, we also generalize this result to linear SEMs with correlated errors.

The paper is organized as follows. In Section 2, we introduce linear SEMs, give basic notation and definitions, and present the local Markov property developed in Richardson (2003). In Section 3, we show that for a class of ADMGs, there is a local Markov property for probability distributions satisfying the composition axiom that invokes only a linear number of conditional independence relations. We also show a local Markov property that may involve fewer conditioning variables. In Section 4, we consider general ADMGs (for probability distributions satisfying the composition axiom) and show a local Markov property that invokes fewer conditional independencies than that in Richardson (2003). Section 5 concludes the paper.

2. Preliminaries and Motivation

In this section, we give basic definitions and introduce some relevant concepts.

2.1 Linear Causal Models

The SEM technique was developed by geneticists (Wright, 1934) and economists (Haavelmo, 1943) for assessing cause-effect relationships from a combination of statistical data and qualitative causal assumptions. It is an important causal analysis tool widely used in social sciences, economics, and artificial intelligence (Goldberger, 1972; Duncan, 1975; Bollen, 1989; Spirtes et al., 2001). For a review of SEMs and causality we refer to Pearl (1998).

In an SEM, the causal relationships among a set of variables are often assumed to be linear and expressed by linear equations. Each equation describes the dependence of one variable in terms of the others. For example, an equation

$$Y = aX + \varepsilon \tag{1}$$

represents that X may have a *direct* causal influence on Y and that no other variables have (direct) causal influences on Y except those factors (represented by the error term ε traditionally assumed to have normal distribution) that are omitted from the model. The parameter a quantifies the (direct) causal effect of X on Y. An equation like (1) with a causal interpretation represents an autonomous causal mechanism and is said to be *structural*.

As an example, consider the following model from Pearl (2000) that concerns the relations between smoking (X) and lung cancer (Y), mediated by the amount of tar (Z) deposited in a person's lungs:

$$\begin{aligned} X &= \varepsilon_1, \\ Z &= aX + \varepsilon_2, \\ Y &= bZ + \varepsilon_3. \end{aligned}$$

The model assumes that the amount of tar deposited in the lungs depends on the level of smoking (and external factors) and that the production of lung cancer depends on the amount of tar in the lungs but smoking has no effect on lung cancer except as mediated through tar deposits. To fully specify the model, we also need to decide whether those omitted factors (ε_1, ε_2, ε_3) are correlated or not. We may assume that no other factor that affects tar deposit is correlated with the omitted factors that affect smoking or lung cancer ($Cov(\varepsilon_1, \varepsilon_2) = Cov(\varepsilon_2, \varepsilon_3) = 0$). However, there might be unobserved factors (say some unknown carcinogenic genotype) that affect both smoking and lung

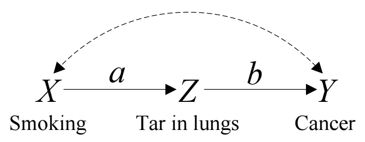


Figure 1: Causal diagram illustrating the effect of smoking on lung cancer

cancer ($Cov(\varepsilon_1, \varepsilon_3) \neq 0$), but the genotype nevertheless has no effect on the amount of tar in the lungs except indirectly (through smoking). Often, it is illustrative to express our qualitative causal assumptions in terms of a graphical representation, as shown in Figure 1.

We now formally define the model that we will consider in this paper. A *linear causal model* (or *linear SEM*) over a set of random variables $V = \{V_1, \ldots, V_n\}$ is given by a set of structural equations of the form

$$V_j = \sum_i c_{ji} V_i + \varepsilon_j, \; j = 1, \ldots, n, \tag{2}$$

where the summation is over the variables in V judged to be immediate causes of V_j. c_{ji}, called a *path coefficient*, quantifies the direct causal influence of V_i on V_j. ε_j's represent "error" terms due to omitted factors and are assumed to have normal distribution. We consider recursive models and assume that the summation in (2) is for $i < j$, that is, $c_{ji} = 0$ for $i \geq j$.

We denote the covariances between observed variables $\sigma_{ij} = Cov(V_i, V_j)$, and between error terms $\psi_{ij} = Cov(\varepsilon_i, \varepsilon_j)$. We denote the following matrices, $\Sigma = [\sigma_{ij}]$, $\Psi = [\psi_{ij}]$, and $C = [c_{ij}]$. The parameters of the model are the non-zero entries in the matrices C and Ψ. A parameterization of the model assigns a value to each parameter in the model, which then determines a unique covariance matrix Σ given by (see, for example, Bollen, 1989)

$$\Sigma = (I - C)^{-1} \Psi ((I - C)^t)^{-1}.$$

The structural assumptions encoded in the model are the zero path coefficients and zero error covariances. The model structure can be represented by a DAG G with (dashed) bi-directed edges (an ADMG), called a *causal diagram* (or *path diagram*), as follows: the nodes of G are the variables $V_1, \ldots, V_n$; there is a directed edge from V_i to V_j in G if V_i appears in the structural equation for V_j, that is, $c_{ji} \neq 0$; there is a bi-directed edge between V_i and V_j if the error terms ε_i and ε_j have non-zero correlation. For example, the smoking-and-lung-cancer SEM is represented by the causal diagram in Figure 1, in which each directed edge is annotated by the corresponding path coefficient.

We note that linear SEMs are often used without explicit causal interpretation. A linear SEM in which error terms are uncorrelated consists of a set of regression equations. Note that an equation as given by (2) is a regression equation if and only if ε_j is uncorrelated with each V_i ($Cov(V_i, \varepsilon_j) = 0$). Hence, an equation in an SEM with correlated errors may not be a regression equation. Linear SEMs provide a more powerful way to model data than the regression models taking into account correlated error terms.

2.2 Model Testing and Markov Properties

One important task in the applications of linear SEMs is to test a model against data. One approach for this task is to test for the conditional independence relationships implied by the model, which can be read from the causal diagram by the d-separation criterion as defined in the following.[1] A *path* between two vertices V_i and V_j in an ADMG consists of a sequence of consecutive edges of any type (directed or bi-directed). A vertex V_i is said to be an *ancestor* of a vertex V_j if there is a path $V_i \to \cdots \to V_j$. A non-endpoint vertex W on a path is called a *collider* if two arrowheads on the path meet at W, that is, $\to W \leftarrow$, $\leftrightarrow W \leftrightarrow$, $\leftrightarrow W \leftarrow$, $\to W \leftrightarrow$; all other non-endpoint vertices on a path are *non-colliders*, that is, $\leftarrow W \to$, $\leftarrow W \leftarrow$, $\to W \to$, $\leftrightarrow W \to$, $\leftarrow W \leftrightarrow$. A path between vertices V_i and V_j in an ADMG is said to be *d-connecting given a set* of vertices Z if

1. every non-collider on the path is not in Z, and
2. every collider on the path is an ancestor of a vertex in Z.

If there is no path d-connecting V_i and V_j given Z, then V_i and V_j are said to be *d-separated* given Z. Sets X and Y are said to be *d-separated* given Z, if for every pair V_i, V_j, with $V_i \in X$ and $V_j \in Y$, V_i and V_j are d-separated given Z. Let $I(X,Z,Y)$ denote that X is conditionally independent of Y given Z. The set of all the conditional independence relations encoded by a causal diagram G is specified by the following global Markov property.

Definition 1 (The Global Markov Property (GMP)) *A probability distribution P is said to satisfy the global Markov property for G if for arbitrary disjoint sets X,Y,Z with X and Y being nonempty,*

$$\text{(GMP)} \qquad X \text{ is d-separated from } Y \text{ given } Z \text{ in } G \Longrightarrow I(X,Z,Y).$$

The global Markov property typically involves a vast number of conditional independence relations and it is possible to test for a subset of those independencies that will imply all others. A local Markov property specifies a much smaller set of conditional independence relations which will imply by the laws of probability all other conditional independence relations that hold under the global Markov property. For example, a well-known local Markov property for DAGs is that each variable is conditionally independent of its non-descendants given its parents. The causal diagram for a linear SEM with correlated errors is an ADMG and a local Markov property for ADMGs is given in Richardson (2003).

Note that in linear SEMs, the conditional independence relations will correspond to zero partial correlations (Lauritzen, 1996):

$$\rho_{V_iV_j.Z} = 0 \Longleftrightarrow I(\{V_i\},Z,\{V_j\}).$$

As an example, for the linear SEM with the causal diagram in Figure 2, if we use the local Markov property in Richardson (2003), then we need to test for the vanishing of the following set of partial correlations (for ease of notation, we write $\rho_{ij.Z}$ to denote $\rho_{V_iV_j.Z}$):

$$\{\rho_{21}, \rho_{32.1}, \rho_{43.2}, \rho_{41.2}, \rho_{54.3}, \rho_{52.3}, \rho_{51.3}, \rho_{64.53}, \rho_{62.53}, \rho_{61.53}, \rho_{64.3}, \rho_{62.3}, \rho_{61.3}, \rho_{72.6543}, \rho_{71.6543}, \rho_{72.643}, \rho_{71.643}, \rho_{75.4}, \rho_{73.4}, \rho_{72.4}, \rho_{71.4}\}. \tag{3}$$

The local Markov property in Richardson (2003) is valid for any probability distributions. In fact, the equivalence of the global and local Markov properties is proved using the following so-called *semi-graphoid axioms* (Pearl, 1988) that probabilistic conditional independencies must satisfy:

1. The d-separation criterion was originally defined for DAGs (Pearl, 1988) but can be naturally extended for ADMGs and is called m-separation in Richardson (2003).

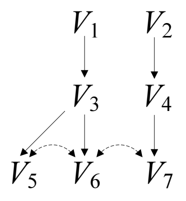


Figure 2: A causal diagram

- Symmetry
$$I(X,Z,Y) \Longleftrightarrow I(Y,Z,X).$$
- Decomposition
$$I(X,Z,Y\cup W) \Longrightarrow I(X,Z,Y)\ \&\ I(X,Z,W).$$
- Weak Union
$$I(X,Z,Y\cup W) \Longrightarrow I(X,Z\cup W,Y).$$
- Contraction
$$I(X,Z,Y)\ \&\ I(X,Z\cup Y,W) \Longrightarrow I(X,Z,Y\cup W).$$

where X, Y, Z, and W are disjoint sets of variables.

On the other hand, in linear SEMs the variables are assumed to have normal distributions, and normal distributions also satisfy the following *composition* axiom:

- Composition
$$I(X,Z,Y)\ \&\ I(X,Z,W) \Longrightarrow I(X,Z,Y\cup W).$$

Therefore, we expect a local Markov property for linear SEMs to invoke fewer conditional independence relations than that for arbitrary distributions. In this paper, we will derive reduced local Markov properties for linear SEMs by making use of the composition axiom. As an example, for the linear SEM in Figure 2, a local Markov property which we will present in this paper (see Section 3.3) says that we only need to test for the vanishing of the following set of partial correlations:

$$\{\rho_{21},\rho_{32},\rho_{43},\rho_{41},\rho_{54},\rho_{52},\rho_{51.3},\rho_{64},\rho_{62},\rho_{61.3},\rho_{75},\rho_{73},\rho_{71},\rho_{72.4}\}.$$

The number of tests needed and the size of the conditioning set Z are both substantially reduced compared with (3), thus leading to a more economical way of testing the given model.

2.3 A Local Markov Property for ADMGs

In this section, we describe the local Markov property for ADMGs associated with arbitrary probability distributions presented in Richardson (2003). In this paper, this Markov property will be used as an important tool to prove the equivalence of our local Markov properties and the global Markov property.

First, we define some graphical notations. For a vertex X in an ADMG G, $\text{pa}_G(X) \equiv \{Y|Y \to X$ in $G\}$ is the set of *parents* of X. $\text{sp}_G(X) \equiv \{Y|Y \leftrightarrow X$ in $G\}$ is the set of *spouses* of X. $\text{an}_G(X) \equiv$

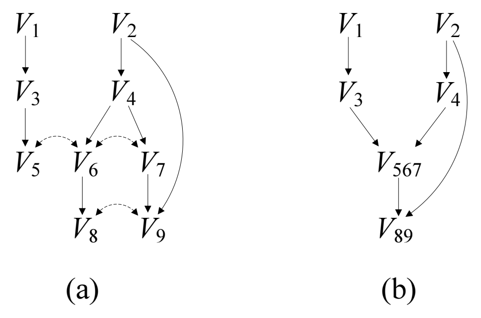


Figure 3: An ADMG and its compressed graph

$\{Y|Y \rightarrow \cdots \rightarrow X$ in G or $Y = X\}$ is the set of *ancestors* of X. And $\mathrm{de}_G(X) \equiv \{Y|Y \leftarrow \cdots \leftarrow X$ in G or $Y = X\}$ is the set of *descendants* of X. These definitions will be applied to sets of vertices, so that, for example, $\mathrm{pa}_G(A) \equiv \cup_{X\in A}\mathrm{pa}_G(X)$, $\mathrm{sp}_G(A) \equiv \cup_{X\in A}\mathrm{sp}_G(X)$, etc.

Definition 2 (C-component) *A c-component of G is a maximal set of vertices in G such that any two vertices in the set are connected by a path on which every edge is of the form $\leftrightarrow$; a vertex that is not connected to any bi-directed edge forms a c-component by itself.*

For example, the ADMG in Figure 3 (a) is composed of 6 c-components $\{V_1\}$, $\{V_2\}$, $\{V_3\}$, $\{V_4\}$, $\{V_5,V_6,V_7\}$ and $\{V_8,V_9\}$. The *district* of X in G is the c-component of G that includes X. Thus,

$$\mathrm{dis}_G(X) \equiv \{Y|Y \leftrightarrow \cdots \leftrightarrow X \text{ in } G \text{ or } Y = X\}.$$

For example, in Figure 3 (a), we have $\mathrm{dis}_G(V_5) = \{V_5,V_6,V_7\}$ and $\mathrm{dis}_G(V_8) = \{V_8,V_9\}$. A set A is said to be *ancestral* if it is closed under the ancestor relation, that is, if $\mathrm{an}_G(A) = A$. Let G_A denote the induced subgraph of G on the vertex set A, formed by removing from G all vertices that are not in A, and all edges that do not have both endpoints in A.

Definition 3 (Markov Blanket)[2] *If A is an ancestral set in an ADMG G, and X is a vertex in A that has no children in A then the* Markov blanket of vertex X with respect to the induced subgraph on A*, denoted* $\mathrm{mb}(X,A)$ *is defined to be*

$$\mathrm{mb}(X,A) \equiv \mathrm{pa}_{G_A}(\mathrm{dis}_{G_A}(X)) \cup (\mathrm{dis}_{G_A}(X) \setminus \{X\}).$$

For example, for an ancestral set $A = \mathrm{an}_G(\{V_5,V_6\}) = \{V_1,V_2,V_3,V_4,V_5,V_6\}$ in Figure 3 (a), we have

$$\mathrm{mb}(V_5,A) = \{V_3,V_4,V_6\}.$$

An ordering ($\prec$) on the vertices of G is said to be consistent with G if $X \prec Y \Rightarrow Y \notin \mathrm{an}_G(X)$. Given a consistent ordering $\prec$, let $\mathrm{pre}_{G,\prec}(X) \equiv \{Y|Y \prec X$ or $Y = X\}$.

Definition 4 (The Ordered Local Markov Property (LMP,$\prec$)) *A probability distribution P satisfies the ordered local Markov property for G with respect to a consistent ordering $\prec$, if, for any X and ancestral set A such that $X \in A \subseteq \mathrm{pre}_{G,\prec}(X)$,*

$$\text{(LMP,}\prec\text{)} \qquad I(\{X\},\mathrm{mb}(X,A),A \setminus (\mathrm{mb}(X,A) \cup \{X\})). \qquad (4)$$

2. The definition of Markov blanket here follows that in Richardson (2003) and is compatible with that in Pearl (1988).

Theorem 5 (Richardson, 2003) *If G is an ADMG and $\prec$ is a consistent ordering, then a probability distribution P satisfies the ordered local Markov property for G with respect to $\prec$ if and only if P satisfies the global Markov property for G.*

We will write (GMP) $\Longleftrightarrow$ (LMP,$\prec$) to denote the equivalence of the two Markov properties. Therefore the (smaller) set of conditional independencies specified in the ordered local Markov property will imply all other conditional independencies which hold under the global Markov property. It is possible to further reduce the number of conditional independence relations in the ordered local Markov property. An ancestral set A, with $X \in A \subseteq \mathrm{pre}_{G,\prec}(X)$ is said to be *maximal with respect to the Markov blanket* $\mathrm{mb}(X,A)$ if, whenever there is a set B such that $A \subseteq B \subseteq \mathrm{pre}_{G,\prec}(X)$ and $\mathrm{mb}(X,A) = \mathrm{mb}(X,B)$, then $A = B$. For example, suppose that we are given an ordering $\prec$: $V_1 \prec V_2 \prec V_3 \prec V_4 \prec V_5 \prec V_6 \prec V_7 \prec V_8 \prec V_9$ for the graph G in Figure 3 (a). While an ancestral set $A = \mathrm{an}_G(\{V_3,V_6,V_7\}) = \{V_1,V_2,V_3,V_4,V_6,V_7\}$ is maximal with respect to the Markov blanket $\mathrm{mb}(V_7,A) = \{V_4,V_6\}$, an ancestral set $A' = \mathrm{an}_G(\{V_6,V_7\}) = \{V_2,V_4,V_6,V_7\}$ is not. It was shown that we only need to consider ancestral sets A which are maximal with respect to $\mathrm{mb}(X,A)$ in the ordered local Markov property (Richardson, 2003). Thus, we will consider only maximal ancestral sets A when we discuss (LMP,$\prec$) for the rest of this paper. The following lemma characterizes maximal ancestral sets.

Lemma 6 (Richardson, 2003) *Let X be a vertex and A an ancestral set in G with consistent ordering $\prec$ such that $X \in A \subseteq \mathrm{pre}_{G,\prec}(X)$. The set A is maximal with respect to the Markov blanket* mb*(X,A) if and only if*

$$A = \mathrm{pre}_{G,\prec}(X) \setminus \mathrm{de}_G(\mathrm{h}(X,A))$$

where

$$\mathrm{h}(X,A) \equiv \mathrm{sp}_G\Big(\mathrm{dis}_{G_A}(X)\Big) \setminus \Big(\{X\} \cup \mathrm{mb}(X,A)\Big).$$

Even though we only consider maximal ancestral sets, the ordered local Markov property may still invoke an exponential number of conditional independence relations. For example, for a vertex X, if $\mathrm{dis}_G(X) \subseteq \mathrm{pre}_{G,\prec}(X)$ and $\mathrm{dis}_G(X)$ has a clique of n vertices joined by bi-directed edges, then there are at least $O(2^{n-1})$ different Markov blankets.

It should be noted that only the semi-graphoid axioms were used to prove Theorem 5 on the equivalence of the two Markov properties and no assumptions about probability distributions were made. Next we will show that the ordered local Markov property can be further reduced if we use the composition axiom in addition to the semi-graphoid axioms. The local Markov properties we obtained (in Sections 3 and 4) are not restricted to linear causal models in that they are actually valid for any probability distributions that satisfy the composition axiom.

3. Markov Properties for ADMGs without Directed Mixed Cycles

In this section, we introduce three local Markov properties for a class of ADMGs and show that they are equivalent to the global Markov property. Also, we discuss related work in maximal ancestral graphs and chain graphs. First, we give some definitions.

Definition 7 (Directed Mixed Cycle) *A path is said to be a directed mixed path from X to Y if it contains at least one directed edge and every edge on the path is either of the form $Z \leftrightarrow W$, or $Z \rightarrow W$ with W between Z and Y. A directed mixed path from X to Y together with an edge $Y \rightarrow X$ or $Y \leftrightarrow X$ is called a directed mixed cycle.*

$$\begin{tikzcd}
X \arrow[d] \arrow[r, leftrightarrow, dashed, bend left] & Y \\
Z \arrow[r, leftrightarrow, dashed, bend left] & W \arrow[u]
\end{tikzcd}$$

Figure 4: Directed mixed cycles

For example, the path $X \to Z \leftrightarrow W \to Y \leftrightarrow X$ in the graph in Figure 4 forms a directed mixed cycle. In this section, we will consider only ADMGs without directed mixed cycles.

Definition 8 (Compressed Graph) *Let G be an ADMG. The compressed graph of G is defined to be the graph $G' = (V', E')$, $V' = \{V_C \mid C$ is a c-component of $G\}$, $E' = \{V_{C_i} \to V_{C_j} \mid$ there is an edge $X \to Y$ in G such that $X \in C_i, Y \in C_j\}$.*

Figure 3 shows an ADMG and its compressed graph. If there exists a directed mixed cycle in an ADMG G, there will be a cycle or a self-loop in the compressed graph of G. For example, if for two vertices X and Y in a c-component C of G there exists an edge $X \to Y$, then the compressed graph of G contains a self-loop $\overset{\curvearrowright}{V_C}$. The following proposition holds.

Proposition 9 *Let G be an ADMG. The compressed graph of G is a DAG if and only if G has no directed mixed cycles.*

3.1 The Reduced Local Markov Property

In this section, we introduce a local Markov property for ADMGs without directed mixed cycles which only invokes a linear number of conditional independence relations and show that it is equivalent to the global local Markov property.

Definition 10 (The Reduced Local Markov Property (RLMP)) *Let G be an ADMG without directed mixed cycles. A probability distribution P is said to satisfy the reduced local Markov property for G if*

$$\text{(RLMP)} \qquad \forall X \in V, \quad I(\{X\}, \mathrm{pa}_G(X), V \setminus \mathrm{f}(X,G)) \tag{5}$$

where $\mathrm{f}(X,G) \equiv \mathrm{pa}_G(X) \cup \mathrm{de}_G(\{X\} \cup \mathrm{sp}_G(X))$.

The reduced local Markov property states that *a variable is independent of the variables that are neither its descendants nor its spouses' descendants given its parents.*

Theorem 11 *If a probability distribution P satisfies the composition axiom and an ADMG G has no directed mixed cycles, then*

$$\text{(GMP)} \Longleftrightarrow \text{(RLMP)}.$$

Proof: (GMP) $\Longrightarrow$ (RLMP)
We need to prove that any variable X is d-separated from $V \setminus \mathrm{f}(X,G)$ given $\mathrm{pa}_G(X)$ in G with no directed mixed cycle. Consider a vertex $\alpha \in V \setminus \mathrm{f}(X,G)$. We will show that there is no path d-connecting X and α given $\mathrm{pa}_G(X)$. There are four possible cases for any path between X and α.

1. $X \leftarrow \beta \cdots \alpha$
2. $X \rightarrow \cdots \rightarrow \delta \leftarrow\!\!* \cdots \alpha$
3. $X \leftrightarrow \gamma \leftarrow\!\!* \cdots \alpha$
4. $X \leftrightarrow \gamma \rightarrow \cdots \rightarrow \delta \leftarrow\!\!* \cdots \alpha$

A symbol $*$ serves as a wildcard for an end of an edge. For example, $\leftarrow\!\!*$ represents both $\leftarrow$ and $\leftrightarrow$. In case 1, $\beta \in \mathrm{pa}_G(X)$. In case 2, the collider δ is not an ancestor of a vertex in $\mathrm{pa}_G(X)$ (otherwise, there would be a cycle). In cases 3 and 4, neither γ nor δ is an ancestor of a vertex in $\mathrm{pa}_G(X)$ (otherwise, there would be directed mixed cycles). In any case, the path is not d-connecting given $\mathrm{pa}_G(X)$. ■

Proof: (RLMP) $\Longrightarrow$ (GMP)
We will show that for some consistent ordering $\prec$, (RLMP) $\Longrightarrow$ (LMP, $\prec$). Then, by Theorem 5, we have (RLMP) $\Longrightarrow$ (GMP).

We construct a consistent ordering with the desired property as follows.

1. Construct the compressed graph G' of G.
2. Let $\prec'$ be any consistent ordering on G'. Construct a consistent ordering $\prec$ from $\prec'$ by replacing each V_C (corresponding to each c-component C of G) in $\prec'$ with the vertices in C (the ordering of the vertices in C is arbitrary).

We now prove that (RLMP) $\Longrightarrow$ (LMP,$\prec$). Assume that a probability distribution P satisfies (RLMP). Consider the set of conditional independence relations invoked by (LMP,$\prec$) for each variable X given in (4). First, observe that for any vertex Y in $\mathrm{dis}_{G_A}(X)$, we have

$$A \setminus (\mathrm{pa}_G(Y) \cup \{Y\} \cup \mathrm{sp}_G(Y)) \subseteq V \setminus \mathrm{f}(Y,G),$$

since

$$\begin{aligned} & A \setminus (\mathrm{pa}_G(Y) \cup \{Y\} \cup \mathrm{sp}_G(Y)) \\ & = A \setminus \Big(\big(\mathrm{pa}_G(Y) \cup \{Y\} \cup \mathrm{sp}_G(Y)\big) \cup \big(\mathrm{de}_G(\{Y\} \cup \mathrm{sp}_G(Y)) \setminus (\{Y\} \cup \mathrm{sp}_G(Y))\big) \Big) \qquad (6) \\ & = A \setminus \mathrm{f}(Y,G). \end{aligned}$$

The equality (6) holds since the vertices in $\mathrm{de}_G(\{Y\} \cup \mathrm{sp}_G(Y)) \setminus (\{Y\} \cup \mathrm{sp}_G(Y))$ do not appear in A (because of the way $\prec$ is constructed, no descendant of $\mathrm{dis}_{G_A}(X)$ is in A). Thus, by (5), for all Y in $\mathrm{dis}_{G_A}(X)$, we have

$$I(\{Y\}, \mathrm{pa}_G(Y), A \setminus (\mathrm{pa}_G(Y) \cup \{Y\} \cup \mathrm{sp}_{G_A}(Y))).$$

Let $S_1 = \mathrm{pa}_G(\mathrm{dis}_{G_A}(X)) \setminus \mathrm{pa}_G(Y)$ and $S_2 = A \setminus (\mathrm{mb}(X,A) \cup \{X\})$. It follows that

$$\begin{aligned} & S_1 \subseteq A \setminus (\mathrm{pa}_G(Y) \cup \{Y\} \cup \mathrm{sp}_G(Y)) \text{ and} \\ & S_2 \subseteq A \setminus (\mathrm{pa}_G(Y) \cup \{Y\} \cup \mathrm{sp}_G(Y)). \end{aligned}$$

Also, we have

$$S_1 \cap S_2 = \emptyset,$$

since $S_1 \subseteq \mathrm{mb}(X,A)$. Therefore, for $Y \in \mathrm{dis}_{G_A}(X)$,

$$\begin{array}{ll}
I(\{Y\},\mathrm{pa}_G(Y),S_1 \cup S_2) & \text{by decomposition} \\
I(\{Y\},\mathrm{pa}_G(Y) \cup S_1,S_2) & \text{by weak union} \\
I(\mathrm{dis}_{G_A}(X),\mathrm{pa}_G(\mathrm{dis}_{G_A}(X)),A \setminus (\mathrm{mb}(X,A) \cup \{X\})) & \text{by composition} \\
I(\{X\},\mathrm{pa}_G(\mathrm{dis}_{G_A}(X)) \cup (\mathrm{dis}_{G_A}(X) \setminus \{X\}), & \\
\quad A \setminus (\mathrm{mb}(X,A) \cup \{X\})) & \text{by weak union.}
\end{array}$$

Thus, we have

$$I(\{X\},\mathrm{mb}(X,A),A \setminus (\mathrm{mb}(X,A) \cup \{X\}))$$

by the definition of the Markov blanket of X with respect to A. ■

As an example, consider the ADMG G in Figure 3 (a) which has no directed mixed cycles. The graph in Figure 3 (b) is the compressed graph G' of G described in the proof. From the ordering $\prec'$: $V_1 \prec V_2 \prec V_3 \prec V_4 \prec V_{567} \prec V_{89}$, we obtain the ordering $\prec$: $V_1 \prec V_2 \prec V_3 \prec V_4 \prec V_5 \prec V_6 \prec V_7 \prec V_8 \prec V_9$. The ordered local Markov property (LMP,$\prec$) involves the following conditional independence relations:

$$\begin{array}{ll}
I(\{V_2\},\emptyset,\{V_1\}), & I(\{V_3\},\{V_1\},\{V_2\}), \\
I(\{V_4\},\{V_2\},\{V_1,V_3\}), & I(\{V_5\},\{V_3\},\{V_1,V_2,V_4\}), \\
I(\{V_6\},\{V_3,V_4,V_5\},\{V_1,V_2\}), & I(\{V_6\},\{V_4\},\{V_1,V_2,V_3\}), \\
I(\{V_7\},\{V_3,V_4,V_5,V_6\},\{V_1,V_2\}), & I(\{V_7\},\{V_4,V_6\},\{V_1,V_2,V_3\}), \\
I(\{V_7\},\{V_4\},\{V_1,V_2,V_3,V_5\}), & I(\{V_8\},\{V_6\},\{V_1,V_2,V_3,V_4,V_5,V_7\}), \\
I(\{V_9\},\{V_2,V_6,V_7,V_8\},\{V_1,V_3,V_4,V_5\}), & I(\{V_9\},\{V_2,V_7\},\{V_1,V_3,V_4,V_5,V_6\}).
\end{array} \tag{7}$$

(RLMP) invokes the following conditional independence relations:

$$\begin{array}{ll}
I(\{V_1\},\emptyset,\{V_2,V_4,V_6,V_7,V_8,V_9\}), & I(\{V_2\},\emptyset,\{V_1,V_3,V_5\}), \\
I(\{V_3\},\{V_1\},\{V_2,V_4,V_6,V_7,V_8,V_9\}), & I(\{V_4\},\{V_2\},\{V_1,V_3,V_5\}), \\
I(\{V_5\},\{V_3\},\{V_1,V_2,V_4,V_7,V_9\}), & I(\{V_6\},\{V_4\},\{V_1,V_2,V_3\}), \\
I(\{V_7\},\{V_4\},\{V_1,V_2,V_3,V_5\}), & I(\{V_8\},\{V_6\},\{V_1,V_2,V_3,V_4,V_5,V_7\}), \\
I(\{V_9\},\{V_2,V_7\},\{V_1,V_3,V_4,V_5,V_6\}) &
\end{array} \tag{8}$$

which, by Theorem 11, imply all the conditional independence relations in (7).

For the special case of graphs containing only bi-directed edges,[3] Kauermann (1996) provides a local Markov property for probability distributions obeying the composition axiom as follows:

$$\forall X \in V, \quad I(\{X\},\emptyset,V \setminus (\{X\} \cup \mathrm{sp}_G(X))). \tag{9}$$

3. Kauermann (1996) actually used undirected graphs with dashed edges which are Markov equivalent to graphs with only bi-directed edges (see Richardson, 2003, for discussions).

Since a graph containing only bi-directed edges is a special case of ADMGs without directed mixed cycles, the reduced local Markov property (RLMP) is applicable, and it turns out that (RLMP) reduces to (9) for graphs containing only bi-directed edges. Therefore (RLMP) includes the local Markov property given in Kauermann (1996) as a special case.

3.2 The Ordered Reduced Local Markov Property

The set of zero partial correlations corresponding to a conditional independence relation $I(X,Z,Y)$ is

$$\{\rho_{V_iV_j.Z} = 0 \mid V_i \in X, V_j \in Y\}.$$

Although (RLMP) gives only a linear number of conditional independence relations, the number of zero partial correlations may be larger than that invoked by (LMP,$\prec$) in some cases. For example, 12 conditional independence relations in (7) involve 37 zero partial correlations while 9 conditional independence relations in (8) involve 41 zero partial correlations. In this section, we will show an ordered local Markov property such that at most one zero partial correlation is invoked for each pair of variables.

Definition 12 (C-ordering) *Let G be an ADMG. A consistent ordering $\prec$ on the vertices of G is said to be a c-ordering if all the vertices in each c-component of G are consecutively ordered in $\prec$.*

For example, the ordering $V_1 \prec V_2 \prec V_3 \prec V_4 \prec V_5 \prec V_6 \prec V_7 \prec V_8 \prec V_9$ is a c-ordering on the vertices of G in Figure 3 (a). The following holds.

Proposition 13 *There exists a c-ordering on the vertices of G if G does not have directed mixed cycles.*

We can easily construct a c-ordering from the compressed graph of G. We introduce the following Markov property.

Definition 14 (The Ordered Reduced Local Markov Property (RLMP,$\prec_c$)) *Let G be an ADMG without directed mixed cycles and $\prec_c$ be a c-ordering on the vertices of G. A probability distribution P is said to satisfy the ordered reduced local Markov property for G with respect to $\prec_c$ if*

$$\text{(RLMP,}\prec_c\text{)} \qquad \forall X \in V,\ I(\{X\}, \text{pa}_G(X), \text{pre}_{G,\prec_c}(X) \setminus (\{X\} \cup \text{pa}_G(X) \cup \text{sp}_G(X))). \qquad (10)$$

The ordered reduced local Markov property states that *a variable is independent of its predecessors, excluding its spouses, in a c-ordering given its parents*. We now establish the equivalence of (GMP) and (RLMP,$\prec_c$).

Theorem 15 *If a probability distribution P satisfies the composition axiom and an ADMG G has no directed mixed cycles, then for a c-ordering $\prec_c$ on the vertices of G,*

$$\text{(GMP)} \Longleftrightarrow \text{(RLMP,}\prec_c\text{)}.$$

Proof: (GMP) $\implies$ (RLMP,$\prec_c$)
The set $\mathrm{pre}_{G,\prec_c}(X)$ does not include any descendant of $\mathrm{dis}_G(X)$ since $\prec_c$ is a c-ordering. We have

$$\begin{aligned}
&\mathrm{pre}_{G,\prec_c}(X)\setminus(\{X\}\cup\mathrm{pa}_G(X)\cup\mathrm{sp}_G(X))\\
&=\mathrm{pre}_{G,\prec_c}(X)\setminus\Big(\Big(\{X\}\cup\mathrm{pa}_G(X)\cup\mathrm{sp}_G(X)\Big)\cup\Big(\mathrm{de}_G(\{X\}\cup\mathrm{sp}_G(X))\setminus(\{X\}\cup\mathrm{sp}_G(X))\Big)\Big)\\
&=\mathrm{pre}_{G,\prec_c}(X)\setminus\mathrm{f}(X,G)\\
&\subseteq V\setminus\mathrm{f}(X,G).
\end{aligned}$$

Hence, (RLMP,$\prec_c$) follows from (RLMP). ■

Proof: (RLMP,$\prec_c$) $\implies$ (GMP)
We will show that (RLMP,$\prec_c$) $\implies$ (LMP,$\prec_c$). Assume that a probability distribution P satisfies (RLMP,$\prec_c$). Let $\mathrm{g}(Y)=\mathrm{pre}_{G,\prec_c}(Y)\setminus(\{Y\}\cup\mathrm{pa}_G(Y)\cup\mathrm{sp}_G(Y))$. Consider the set of conditional independence relations invoked by (LMP,$\prec_c$) for each variable X given in (4) where A is maximal. By (10), for all Y in $\mathrm{dis}_{G_A}(X)$, we have

$$I(Y,\mathrm{pa}_G(Y),\mathrm{g}(Y)). \tag{11}$$

Let $S_1=\mathrm{pa}_G(\mathrm{dis}_{G_A}(X))\setminus\mathrm{pa}_G(Y)$ and $S_2=A\setminus(\mathrm{mb}(X,A)\cup\{X\})$. We have that

$$S_1\subseteq\mathrm{g}(Y).$$

Note that $S_2\setminus\mathrm{g}(Y)$ may be non-empty. Let $S_3=S_2\setminus\mathrm{g}(Y)$. It suffices to show that

$$I(Y,\mathrm{pa}_G(Y),S_3),$$

which implies $I(Y,\mathrm{pa}_G(Y),S_2)$ by composition. Then, the rest of the proof would be identical to that of Theorem 11.

We first characterize the vertices in S_3. We will show that

$$S_3=(\mathrm{pre}_{G,\prec_c}(X)\setminus\mathrm{pre}_{G,\prec_c}(Y))\setminus\mathrm{sp}_G(\mathrm{dis}_{G_A}(X)). \tag{12}$$

By Lemma 6, we have

$$S_2=\mathrm{pre}_{G,\prec_c}(X)\setminus\Big(\mathrm{de}_G(\mathrm{h}(X,A))\cup\mathrm{mb}(X,A)\cup\{X\}\Big).$$

Since $\prec_c$ is a c-ordering, no descendant of $\mathrm{dis}_G(X)$ will appear in A. Hence,

$$S_2=\mathrm{pre}_{G,\prec_c}(X)\setminus\Big(\mathrm{sp}_G(\mathrm{dis}_{G_A}(X))\cup\mathrm{pa}_G(\mathrm{dis}_{G_A}(X))\Big).$$

To identify some common elements of S_2 and $\mathrm{g}(Y)$, we will reformulate S_2 and $\mathrm{g}(Y)$ as follows.

$$\begin{aligned}
S_2&=\Big(B\setminus\mathrm{pa}_G(\mathrm{dis}_{G_A}(X))\Big)\cup\Big((\mathrm{dis}_G(X)\cap\mathrm{pre}_{G,\prec_c}(X))\setminus\mathrm{sp}_G(\mathrm{dis}_{G_A}(X))\Big),\\
\mathrm{g}(Y)&=\Big(B\setminus\mathrm{pa}_G(Y)\Big)\cup\Big((\mathrm{dis}_G(X)\cap\mathrm{pre}_{G,\prec_c}(Y))\setminus(\{Y\}\cup\mathrm{sp}_G(Y))\Big)
\end{aligned}$$

where $B = \mathrm{pre}_{G,\prec_c}(X) \setminus \mathrm{dis}_G(X)$. This can be verified by noting that $A_1 = A_2 \setminus (A_3 \cup A_4) = (A_{11} \setminus A_2) \cup (A_{12} \setminus A_3)$ if $A_1 = A_{11} \cup A_{12}, A_{11} \cap A_{12} = \emptyset, A_2 \subseteq A_{11}, A_3 \subseteq A_{12}$. From $\mathrm{pa}_G(Y) \subseteq \mathrm{pa}_G(\mathrm{dis}_{G_A}(X))$, it follows that $B \setminus \mathrm{pa}_G(\mathrm{dis}_{G_A}(X)) \subseteq B \setminus \mathrm{pa}_G(Y)$ and

$$\begin{aligned} S_3 =& S_2 \setminus \mathrm{g}(Y) \\ =& \Big((\mathrm{dis}_G(X) \cap \mathrm{pre}_{G,\prec_c}(X)) \setminus \mathrm{sp}_G(\mathrm{dis}_{G_A}(X)) \Big) \\ & \setminus \Big((\mathrm{dis}_G(X) \cap \mathrm{pre}_{G,\prec_c}(Y)) \setminus (\{Y\} \cup \mathrm{sp}_G(Y)) \Big). \end{aligned}$$

We can rewrite the first part of this expression as follows.

$$\begin{aligned} & (\mathrm{dis}_G(X) \cap \mathrm{pre}_{G,\prec_c}(X)) \setminus \mathrm{sp}_G(\mathrm{dis}_{G_A}(X)) \\ =& \Big((\mathrm{dis}_G(X) \cap \mathrm{pre}_{G,\prec_c}(Y)) \setminus \mathrm{sp}_G(\mathrm{dis}_{G_A}(X)) \Big) \\ & \cup \Big((\mathrm{pre}_{G,\prec_c}(X) \setminus \mathrm{pre}_{G,\prec_c}(Y)) \setminus \mathrm{sp}_G(\mathrm{dis}_{G_A}(X)) \Big). \end{aligned}$$

From $(\mathrm{dis}_G(X) \cap \mathrm{pre}_{G,\prec_c}(Y)) \setminus \mathrm{sp}_G(\mathrm{dis}_{G_A}(X)) \subseteq (\mathrm{dis}_G(X) \cap \mathrm{pre}_{G,\prec_c}(Y)) \setminus (\{Y\} \cup \mathrm{sp}_G(Y))$, (12) follows. Thus, the vertices in S_3 are those in the set $\mathrm{pre}_{G,\prec_c}(X) \setminus \mathrm{pre}_{G,\prec_c}(Y)$ and not in the set $\mathrm{sp}_G(\mathrm{dis}_{G_A}(X))$.

Now we are ready to prove $I(Y, \mathrm{pa}_G(Y), S_3)$. For any $Z \in S_3$, we have $Y \prec Z$ and $Z \notin \mathrm{sp}_G(Y)$. Hence,

$$\begin{array}{ll} I(\{Z\}, \mathrm{pa}_G(Z), \mathrm{g}(Z)), & \\ I(\{Z\}, \mathrm{pa}_G(Z), \{Y\} \cup (\mathrm{pa}_G(Y) \setminus \mathrm{pa}_G(Z))) & \text{by decomposition,} \\ I(\{Z\}, \mathrm{pa}_G(Z) \cup \mathrm{pa}_G(Y), \{Y\}) & \text{by weak union,} \\ I(\{Y\}, \mathrm{pa}_G(Y), \mathrm{pa}_G(Z) \setminus \mathrm{pa}_G(Y)) & \text{by } \mathrm{pa}_G(Z) \setminus \mathrm{pa}_G(Y)) \subseteq \mathrm{g}(Y), (11) \\ \quad \text{and decomposition,} & \\ I(\{Y\}, \mathrm{pa}_G(Y), \{Z\}) & \text{by contraction and decomposition.} \end{array}$$

Therefore, by composition, $I(Y, \mathrm{pa}_G(Y), S_3)$ holds. ■

(RLMP,$\prec_c$) invokes one zero partial correlation for each pair of nonadjacent variables. For example, for the ADMG G in Figure 3 (a) and a c-ordering $\prec_c$: $V_1 \prec V_2 \prec V_3 \prec V_4 \prec V_5 \prec V_6 \prec V_7 \prec V_8 \prec V_9$, (RLMP,$\prec_c$) invokes the following conditional independence relations:

$$\begin{array}{ll} I(\{V_2\}, \emptyset, \{V_1\}), & I(\{V_3\}, \{V_1\}, \{V_2\}), \\ I(\{V_4\}, \{V_2\}, \{V_1, V_3\}), & I(\{V_5\}, \{V_3\}, \{V_1, V_2, V_4\}), \\ I(\{V_6\}, \{V_4\}, \{V_1, V_2, V_3\}), & I(\{V_7\}, \{V_4\}, \{V_1, V_2, V_3, V_5\}), \\ I(\{V_8\}, \{V_6\}, \{V_1, V_2, V_3, V_4, V_5, V_7\}), & I(\{V_9\}, \{V_2, V_7\}, \{V_1, V_3, V_4, V_5, V_6\}) \end{array} \tag{13}$$

which involve 25 zero partial correlations while (7) involve 37 zero partial correlations.

3.3 The Pairwise Markov Property

In this section, we give a pairwise Markov property which specifies conditional independence relations between pairs of variables and show that it is equivalent to the global Markov property. In previous sections, we focused on minimizing the number of zero partial correlations. We now take into account the size of the conditioning set Z in each zero partial correlation $\rho_{XY.Z}$. When the size of $\mathrm{pa}_G(X)$ for a vertex X in (RLMP,$\prec_c$) is large, it might be advantageous to use a different conditioning set with smaller size (if the equivalence of the Markov properties still holds). Pearl and Meshkat (1999) introduced a pairwise Markov property for DAGs (without bi-directed edges) which may involve fewer conditioning variables and thus lead to more economical tests. The result can be easily generalized to ADMGs with no directed mixed cycles.

Let $d(X,Y)$ denote the shortest distance between two vertices X and Y, that is, the number of edges in the shortest path between X and Y. Two vertices X and Y are nonadjacent if X and Y are not connected by a directed nor a bi-directed edge.

Definition 16 (The Pairwise Markov Property (PMP,$\prec_c$)) *Let G be an ADMG without directed mixed cycles and $\prec_c$ be a c-ordering on the vertices of G. A probability distribution P is said to satisfy the pairwise Markov property for G with respect to $\prec_c$ if for any two nonadjacent vertices $V_i, V_j, V_j \prec_c V_i$*

$$\text{(PMP,}\prec_c\text{)} \qquad\qquad I(\{V_i\}, Z_{ij}, \{V_j\})$$

where Z_{ij} is any set of vertices such that Z_{ij} d-separates V_i from V_j and $\forall Z \in Z_{ij}, d(V_i,Z) < d(V_i,V_j)$.

Note that, in ADMGs with no directed mixed cycles, there always exists such a Z_{ij} for any two nonadjacent vertices. For example, the parent set of V_i always satisfies the condition for Z_{ij}. If the empty set d-separates V_i from V_j, then the empty set is defined to satisfy the condition for Z_{ij}. Therefore we can always choose a Z_{ij} with the smallest size, providing a more economical way to test zero partial correlations.

Theorem 17 *If a probability distribution P satisfies the composition axiom and an ADMG G has no directed mixed cycles, then*

$$\text{(GMP)} \Longleftrightarrow \text{(PMP,}\prec_c\text{)}.$$

Proof: Noting that two vertices X and Y are adjacent if $X \leftarrow Y$, $X \rightarrow Y$ or $X \leftrightarrow Y$, the proof of Theorem 1 by Pearl and Meshkat (1999) is directly applicable to ADMGs and it effectively proves that (RLMP,$\prec_c$) $\Longleftrightarrow$ (PMP,$\prec_c$). We do not reproduce the proof here. ■

As an example, for the ADMG G in Figure 3 (a) and a c-ordering $\prec_c$: $V_1 \prec V_2 \prec V_3 \prec V_4 \prec V_5 \prec V_6 \prec V_7 \prec V_8 \prec V_9$, the following conditional independence relations (for convenience, we combine the relations for each vertex that have the same conditioning set) can be given by (PMP,$\prec_c$):

$$\begin{array}{ll}
I(\{V_2\},\emptyset,\{V_1\}), & I(\{V_3\},\emptyset,\{V_2\}), \\
I(\{V_4\},\emptyset,\{V_3,V_1\}), & I(\{V_5\},\emptyset,\{V_4,V_2\}), \\
I(\{V_5\},\{V_3\},\{V_1\}), & I(\{V_6\},\emptyset,\{V_3,V_1\}), \\
I(\{V_6\},\{V_4\},\{V_2\}), & I(\{V_7\},\emptyset,\{V_5,V_3,V_1\}), \\
I(\{V_7\},\{V_4\},\{V_2\}), & I(\{V_8\},\{V_6\},\{V_7,V_5,V_4,V_2\}), \\
I(\{V_8\},\emptyset,\{V_3,V_1\}), & I(\{V_9\},\{V_2,V_7\},\{V_6,V_4\}), \\
I(\{V_9\},\emptyset,\{V_5,V_3,V_1\}) &
\end{array}$$

which involve the same number of zero partial correlations as (13) but involve smaller conditioning sets than those in (13).

3.4 Relation to Other Work

In this section, we contrast the class of ADMGs without directed mixed cycles to maximal ancestral graphs and chain graphs in terms of Markov properties.

3.4.1 MAXIMAL ANCESTRAL GRAPHS

It is easy to see that an ADMG without directed mixed cycles is a *maximal ancestral graph (MAG)* (Richardson and Spirtes, 2002). An ADMG is said to be *ancestral* if, for any edge $X \leftrightarrow Y$, X is not an ancestor of Y (and vice versa). Note that an edge $X \leftrightarrow Y$ and a directed path from X to Y (or Y to X) form a directed mixed cycle. Hence, an ADMG without directed mixed cycles is ancestral. An ancestral graph is said to be *maximal* if, for any pair of nonadjacent vertices X and Y, there exists a set $Z \subseteq V \setminus \{X,Y\}$ that d-separates X from Y. From Theorem 17, it follows that an ADMG without directed mixed cycles is maximal. On the other hand, there exist MAGs which have directed mixed cycles (see Figure 4). Thus, the class of ADMGs without directed mixed cycles is a strict subclass of MAGs.

Richardson and Spirtes (2002, p.979) showed the following pairwise Markov property for a MAG G:

$$I(\{V_i\}, \mathrm{an}_G(\{V_i,V_j\}) \setminus \{V_i,V_j\}, \{V_j\})$$

for any two nonadjacent vertices V_i and V_j. Richardson and Spirtes (2002) proved that this pairwise Markov property implies the global Markov property assuming a Gaussian parametrization. This does not trivially imply our results in Section 3.3 and our results cannot be considered as a special case of the results on MAGs. The two pairwise Markov properties involve two different forms of conditioning sets. The pairwise Markov property for MAGs involves considerably larger conditioning sets than our pairwise Markov property: the conditioning set includes all ancestors of V_i and V_j, which is undesirable for our purpose of using the zero partial correlations to test a model.

Also, it should be stressed that our results do not depend on a specific parameterization. We only require the composition axiom to be satisfied. In contrast, Richardson and Spirtes (2002) consider only Gaussian parameterizations. It requires further study whether the pairwise Markov property for MAGs can be generalized to the class of distributions satisfying the composition axiom.

In the next section, we consider general ADMGs and try to eliminate redundant conditional independence relations from (LMP,$\prec$). The class of MAGs is clearly a (strict) subclass of ADMGs. Hence, given a MAG, we have two options: either we use the result in the next section or the pairwise Markov property for MAGs. Although the pairwise Markov property for MAGs gives fewer zero partial correlations (one for each nonadjacent pair of vertices), it is possible that in some cases we are better off using the result in the next section (because of the cost incurred by the large conditioning sets in the pairwise Markov property for MAGs). An example of this situation will be given in the next section.

Richardson and Spirtes (2002) also proved that for a Gaussian distribution encoded by a MAG all the constraints on the distribution (that is, on the covariance matrix) are implied by the vanishing partial correlations given by the global Markov property. Hence, this also holds in a linear SEM represented by an ADMG without directed mixed cycles which is a special type of MAG.

3.4.2 CHAIN GRAPHS

The graph that results from replacing bi-directed edges with undirected edges in an ADMG without directed mixed cycles is a *chain graph*. The class of chain graphs has been studied extensively (see Lauritzen, 1996, for a review).

Some Markov properties have been proposed for chain graphs. The first Markov property for chain graphs has been proposed by Lauritzen and Wermuth (1989) and Frydenberg (1990). Andersson et al. (2001) have introduced another Markov property. These two Markov properties do not correspond to the Markov property for ADMGs. Let G be an ADMG without directed mixed cycles and G' be the chain graph obtained by replacing bi-directed edges with undirected edges. In general, the set of conditional independence relations given by the Markov property for G is not equivalent to that given by either of the two Markov properties for chain graphs. However, there are other Markov properties for chain graphs that correspond to the Markov property for ADMGs without directed mixed cycles (Cox and Wermuth, 1993; Wermuth and Cox, 2001, 2004).[4]

4. Markov Properties for General ADMGs

When an ADMG G has directed mixed cycles, (RLMP), (RLMP,$\prec_c$), and (PMP,$\prec_c$) are no longer equivalent to (GMP) while (LMP,$\prec$) still is. In this section, we show that the number of conditional independence relations given by (LMP,$\prec$) for an arbitrary ADMG that might have directed mixed cycles can still be reduced. We introduce a procedure to reduce (LMP,$\prec$). We then give an example to illustrate the procedure.

4.1 Reducing the Ordered Local Markov Property

First, we introduce a lemma that gives a condition by which a conditional independence relation renders another conditional independence relation redundant.

Lemma 18 *Given an ADMG G, a consistent ordering $\prec$ on the vertices of G and a vertex X, assume that a probability distribution P satisfies the global Markov property for $G_{\mathrm{pre}_{G,\prec}(X)\setminus\{X\}}$. Let $A = \mathrm{pre}_{G,\prec}(X)$ and A' be a maximal ancestral set with respect to $\mathrm{mb}(X,A')$ such that $X \in A' \subset A$, $A' \cap \mathrm{dis}_{G_A}(X) = \mathrm{dis}_{G_{A'}}(X)$ and $\mathrm{pa}_G(\mathrm{dis}_{G_A}(X) \setminus \mathrm{dis}_{G_{A'}}(X)) \subseteq \mathrm{mb}(X,A')$. Then,*

$$I(\{X\}, \mathrm{mb}(X,A), A \setminus (\mathrm{mb}(X,A) \cup \{X\})) \tag{14}$$

implies

$$I(\{X\}, \mathrm{mb}(X,A'), A' \setminus (\mathrm{mb}(X,A') \cup \{X\})).$$

We define $\mathrm{rd}_{G,\prec}(X)$ to be the set of all A' satisfying this condition.

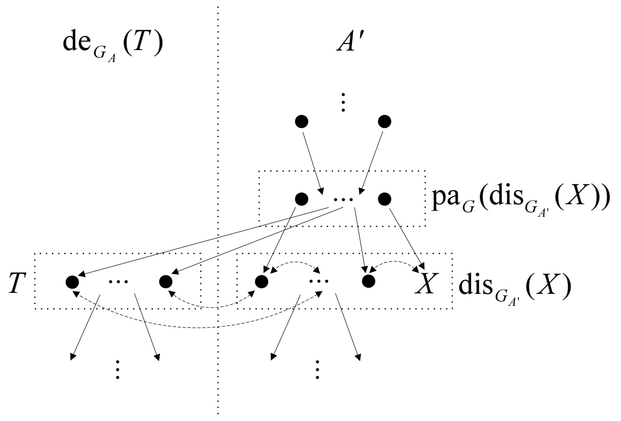
Proof: First, we show the relationships among $A, \mathrm{dis}_{G_A}(X)$, $\mathrm{mb}(X,A)$ and $A', \mathrm{dis}_{G_{A'}}(X)$, $\mathrm{mb}(X,A')$. By Lemma 6, we have

$$A' = A \setminus \mathrm{de}_{G_A}(\mathrm{h}(X,A')) \tag{15}$$

where

$$\mathrm{h}(X,A') \equiv \mathrm{sp}_{G_A}\Big(\mathrm{dis}_{G_{A'}}(X)\Big) \setminus \Big(\{X\} \cup \mathrm{mb}(X,A')\Big).$$

4. In their terminology, ADMGs without directed mixed cycles correspond to chain graphs with dashed arrows and dashed edges.



$\mathrm{de}_{G_A}(T)$
A'
$\mathrm{pa}_G(\mathrm{dis}_{G_{A'}}(X))$
T
X
$\mathrm{dis}_{G_{A'}}(X)$

Figure 5: The relationship between A and A' that satisfy the conditions in Lemma 18. The induced subgraph G_A is shown. The vertices of G_A are decomposed into two disjoint subsets $\mathrm{de}_{G_A}(T)$ and A'.

$\mathrm{dis}_{G_{A'}}(X)$ and $\mathrm{h}(X,A')$ are subsets of $\mathrm{dis}_{G_A}(X)$. Since $\mathrm{dis}_{G_{A'}}(X) \subseteq \{X\} \cup \mathrm{mb}(X,A')$ (by the definition of the Markov blanket), $\mathrm{dis}_{G_{A'}}(X) \cap \mathrm{h}(X,A') = \emptyset$. Thus, we can decompose the set $\mathrm{dis}_{G_A}(X)$ into 3 disjoint subsets as follows.

$$\mathrm{dis}_{G_A}(X) = \mathrm{dis}_{G_{A'}}(X) \cup \mathrm{h}(X,A') \cup B \tag{16}$$

where

$$B \equiv \mathrm{dis}_{G_A}(X) \setminus \Big(\mathrm{dis}_{G_{A'}}(X) \cup \mathrm{h}(X,A')\Big).$$

We have

$$\begin{aligned} A' \cap \mathrm{dis}_{G_A}(X) &= A' \cap \Big(\mathrm{dis}_{G_{A'}}(X) \cup \mathrm{h}(X,A') \cup B\Big) \\ &= \mathrm{dis}_{G_{A'}}(X) \cup B \end{aligned}$$

since $\mathrm{dis}_{G_{A'}}(X) \subseteq A', B \subseteq A'$ and $A' \cap \mathrm{h}(X,A') = \emptyset$. From the assumption in Lemma 18 that $A' \cap \mathrm{dis}_{G_A}(X) = \mathrm{dis}_{G_{A'}}(X)$, it follows that $B = \emptyset$. Thus, from (16), we have

$$\mathrm{dis}_{G_A}(X) \setminus \mathrm{dis}_{G_{A'}}(X) = \mathrm{h}(X,A'). \tag{17}$$

Let $T = \mathrm{dis}_{G_A}(X) \setminus \mathrm{dis}_{G_{A'}}(X) = \mathrm{h}(X,A')$. Then,

$$\begin{aligned} \mathrm{mb}(X,A) &= \mathrm{mb}(X,A') \cup T \cup \mathrm{pa}_G(T) \\ &= \mathrm{mb}(X,A') \cup T \end{aligned} \tag{18}$$

since $\mathrm{pa}_G(T) \subseteq \mathrm{mb}(X,A')$ by our assumption. Thus A decomposes into

$$A = A' \cup \mathrm{de}_{G_A}(T) \tag{19}$$

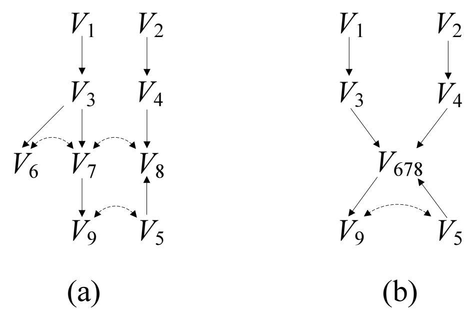


Figure 6: (a) An ADMG with directed mixed cycles (b) Illustration of the procedure **GetOrdering**. The modified graph after the first step is shown.

since $\mathrm{de}_{G_A}(T) \subseteq A$ and (15).

The key relationships among $A, \mathrm{dis}_{G_A}(X), \mathrm{mb}(X,A)$ and $A', \mathrm{dis}_{G_{A'}}(X), \mathrm{mb}(X,A')$ are given by (17)–(19). Figure 5 shows these relationships. We are now ready to prove that $I(\{X\}, \mathrm{mb}(X,A'), A' \setminus (\mathrm{mb}(X,A') \cup \{X\}))$ can be derived from $I(\{X\}, \mathrm{mb}(X,A), A \setminus (\mathrm{mb}(X,A) \cup \{X\}))$. From (18) and (19), it follows that

$$A \setminus (\mathrm{mb}(X,A) \cup \{X\}) = (A' \cup \mathrm{de}_{G_A}(T)) \setminus (\mathrm{mb}(X,A') \cup \{X\} \cup T).$$

Since $A' \cap \mathrm{de}_{G_A}(T) = \emptyset, (\mathrm{mb}(X,A') \cup \{X\}) \cap T = \emptyset, \mathrm{mb}(X,A') \cup \{X\} \subseteq A'$ and $T \subseteq \mathrm{de}_{G_A}(T)$, we have

$$A \setminus (\mathrm{mb}(X,A) \cup \{X\}) = \Big(A' \setminus (\mathrm{mb}(X,A') \cup \{X\})\Big) \cup \Big(\mathrm{de}_{G_A}(T) \setminus T\Big). \tag{20}$$

Plugging (18) and (20) into (14), we get

$$I\Big(\{X\}, \mathrm{mb}(X,A') \cup T, \Big(A' \setminus (\mathrm{mb}(X,A') \cup \{X\})\Big) \cup \Big(\mathrm{de}_{G_A}(T) \setminus T\Big)\Big).$$

From the decomposition axiom, it follows that

$$I(\{X\}, \mathrm{mb}(X,A') \cup T, A' \setminus (\mathrm{mb}(X,A') \cup \{X\})). \tag{21}$$

The last step is to remove T from the conditioning set to obtain $I(\{X\}, \mathrm{mb}(X,A'), A' \setminus (\mathrm{mb}(X,A') \cup \{X\}))$. We claim that

$$I(T, \mathrm{mb}(X,A'), A' \setminus (\mathrm{mb}(X,A') \cup \{X\})). \tag{22}$$

We first argue that T is d-separated from $A' \setminus (\mathrm{mb}(X,A') \cup \{X\})$ given $\mathrm{mb}(X,A')$. Consider a vertex $t \in T$ and a vertex $\alpha \in A' \setminus (\mathrm{mb}(X,A') \cup \{X\})$. Note that for any bi-directed edge $t \leftrightarrow \beta$ in G_A, β is either in T or $\mathrm{dis}_{G_{A'}}(X)$. There are only four possible cases for any path in G_A from t to α.

1. $t \leftarrow \gamma \cdots \alpha$

2. $t \rightarrow \cdots \rightarrow \gamma \leftarrow\!\ast \cdots \alpha$

3. $t \leftrightarrow \leftrightarrow \cdots \leftrightarrow \delta \leftarrow \gamma \cdots \alpha$

4. $t \leftrightarrow \leftrightarrow \cdots \leftrightarrow \delta \rightarrow \cdots \rightarrow \gamma \leftarrow\!\!* \cdots \alpha$

In case 1, $\gamma \in \text{mb}(X,A')$ since $\text{pa}_G(T) \subseteq \text{mb}(X,A')$. Thus, the path is not d-connecting. In case 2, γ is a descendant of t. Since $\text{mb}(X,A')$ does not contain any descendant of t, the path is not d-connecting. Case 3 is similar to case 1, but there are one or more bi-directed edges after t. δ is either in T or $\text{dis}_{G_{A'}}(X)$. It follows that $\gamma \in \text{mb}(X,A')$, so the path is not d-connecting. Case 4 is similar to case 2, but there are one or more bi-directed edges after t. If δ is in T, the argument for case 2 can be applied. If δ is in $\text{dis}_{G_{A'}}(X)$, then $\delta \in \text{mb}(X,A')$, which implies that the path is not d-connecting. This establishes that T is d-separated from $A' \setminus (\text{mb}(X,A') \cup \{X\})$ given $\text{mb}(X,A')$. By the assumption that P satisfies the global Markov property for $G_{\text{pre}_{G,\prec}(X)\setminus\{X\}}$, (22) holds. Finally, from (21),(22) and the contraction axiom, it follows that $I(\{X\}, \text{mb}(X,A'), A' \setminus (\text{mb}(X,A') \cup \{X\}))$. ■

For example, consider the ADMG G in Figure 2 and a consistent ordering $V_1 \prec V_2 \prec V_3 \prec V_4 \prec V_5 \prec V_6 \prec V_7$. Assume that the global Markov property for $G_{\text{pre}_{G,\prec}(V_6)}$ is satisfied. Let $A = \{V_1,V_2,V_3,V_4,V_5,V_6,V_7\}$ and $A' = \{V_1,V_2,V_3,V_4,V_6,V_7\}$. Then, $\text{dis}_{G_A}(V_7) = \{V_5,V_6,V_7\}$, $\text{dis}_{G_{A'}}(V_7) = \{V_6,V_7\}$, $A' \cap \text{dis}_{G_A}(V_7) = \{V_6,V_7\} = \text{dis}_{G_{A'}}(V_7)$ and $\text{pa}_G(\text{dis}_{G_A}(V_7) \setminus \text{dis}_{G_{A'}}(V_7)) = \{V_3\} \subseteq \{V_3,V_4,V_6\} = \text{mb}(V_7,A')$. Thus, $I(\{V_7\},\{V_3,V_4,V_6\},\{V_1,V_2\})$ follows from $I(\{V_7\},\{V_3,V_4,V_5,V_6\},\{V_1,V_2\})$. Note that in the proof of Lemma 18, the composition axiom is not used. Thus, Lemma 18 can be used to reduce the ordered local Markov property for ADMGs associated with an arbitrary probability distribution. Also, note that the condition that P satisfies the global Markov property for $G_{\text{pre}_{G,\prec}(X)\setminus\{X\}}$ is always satisfied in a recursive application of this lemma in Theorem 21.

We now introduce a key concept in eliminating redundant conditional independence relations from (LMP,$\prec$).

Definition 19 (C-ordered Vertex) *Given a consistent ordering $\prec$ on the vertices of an ADMG G, a vertex X is said to be c-ordered in $\prec$ if*

1. *all vertices in $\text{dis}_G(X) \cap \text{pre}_{G,\prec}(X)$ are consecutive in $\prec$ and*

2. *for any two vertices Y and Z in $\text{dis}_G(X) \cap \text{pre}_{G,\prec}(X)$, there is no directed edge between Y and Z.*

If no bi-directed edge is connected to X, then X is defined to be c-ordered. For example, consider the ADMG G in Figure 6 (a). $\prec$: $V_1 \prec V_2 \prec V_3 \prec V_4 \prec V_5 \prec V_6 \prec V_7 \prec V_8 \prec V_9$ is a consistent ordering on the vertices of G. $V_1,V_2,\ldots,V_8$ are c-ordered in $\prec$ but V_9 is not since V_5 and V_9 are not consecutive in $\prec$.

The key observation, which will be proved, is that c-ordered vertices contribute to eliminating many redundant conditional independence relations invoked by the ordered local Markov property (LMP,$\prec$). We provide two procedures. The first procedure **ReduceMarkov** in Figure 7 constructs a list of conditional independence relations in which some redundant conditional independence relations from (LMP,$\prec$) are not included (all the conditional independence relations identified by Lemma 18 are not included). **ReduceMarkov** takes as input a fixed ordering $\prec$. The second procedure **GetOrdering** in Figure 9 gives a good ordering that might have many c-ordered vertices.

We first describe the procedure **ReduceMarkov**. Given an ADMG G and a consistent ordering $\prec$, **ReduceMarkov** gives a set of conditional independence relations which will be shown to be

```
procedure ReduceMarkov
INPUT: An ADMG $G$ and a consistent ordering $\prec$ on the vertices of $G$
OUTPUT: A set of conditional independence relations $S$
$S \leftarrow \emptyset$
for $i = 1, \ldots, n$ do
  $I_i \leftarrow \emptyset$
  if $V_i$ is c-ordered in $\prec$ then
    for nonadjacent $V_j \prec V_i$ do
      $I_i \leftarrow I_i \cup I(\{V_i\}, Z_{ij}, \{V_j\})$ where $Z_{ij}$ is any set of vertices such that $Z_{ij}$ d-separates
      $V_i$ from $V_j$ and $\forall Z \in Z_{ij}, d(V_i, Z) < d(V_i, V_j)$
    end for
  else
    for all maximal ancestral sets $A$ with respect to $\text{mb}(V_i, A)$ such that
    $V_i \in A \subseteq \text{pre}_{G,\prec}(V_i), A \notin \text{rd}_{G,\prec}(V_i)$ do
      $I_i \leftarrow I_i \cup I(\{V_i\}, \text{mb}(V_i, A), A \setminus (\text{mb}(V_i, A) \cup \{V_i\}))$
    end for
  end if
  $S \leftarrow S \cup I_i$
end for
```

Figure 7: A procedure to generate a reduced set of conditional independence relations for an ADMG G and a consistent ordering $\prec$

equivalent to the global Markov property for G. For each vertex V_i, **ReduceMarkov** generates a set of conditional independence relations. If V_i is c-ordered, the relations that correspond to the pairwise Markov property are generated. Otherwise, the relations that correspond to the ordered local Markov property are generated, and Lemma 18 is used to remove some redundant relations. The output $S = \textbf{ReduceMarkov}(G, \prec)$ can be described as follows:

$$S = \bigcup_{X : X \text{ is c-ordered in } \prec} \Big(\bigcup_{Y : Y \prec X} I(\{X\}, Z_{XY}, \{Y\}) \Big) \bigcup \bigcup_{X : X \text{ is not c-ordered in } \prec} \Big(\bigcup_{\substack{\text{all maximal sets } A \\ \text{with respect to mb}(X,A): \\ X \in A \subseteq \text{pre}_{G,\prec}(X), \\ A \notin \text{rd}_{G,\prec}(X)}} I(\{X\}, \text{mb}(X,A), A \setminus (\text{mb}(X,A) \cup \{X\})) \Big) \tag{23}$$

where Z_{XY} is any set of vertices such that Z_{XY} d-separates X from Y and $\forall Z \in Z_{XY}$, $d(X,Z) < d(X,Y)$.

If a vertex X is c-ordered, $O(n)$ conditional independence relations (or zero partial correlations) are added to S. Otherwise, $O(2^n)$ conditional independence relations may be added to S and $O(n2^n)$ zero partial correlations may be invoked. Furthermore, a c-ordered vertex typically involves a smaller conditioning set. $I(\{X\}, Z_{XY}, \{Y\})$ has the conditioning set $|Z_{XY}| \leq |\text{pa}_G(X)|$ while $I(\{X\}, \text{mb}(X,A), A \setminus (\text{mb}(X,A) \cup \{X\}))$ has the conditioning set $|\text{mb}(X,A)| \geq |\text{pa}_G(X)|$.

We now prove that the conditional independence relations produced by **ReduceMarkov** can derive all the conditional independence relations invoked by the global Markov property.

Definition 20 **(S-Markov Property (S-MP,$\prec$))** *Let G be an ADMG and $\prec$ be a consistent ordering on the vertices of G. Let S be the set of conditional independence relations given by* ***ReduceMarkov****$(G,\prec)$. A probability distribution P is said to satisfy the S-Markov property for G with respect to $\prec$, if*

(S-MP,$\prec$) *P satisfies all the conditional independence relations in S.*

Theorem 21 *Let G be an ADMG and $\prec$ be a consistent ordering on the vertices of G. Let S be the set of conditional independence relations given by* ***ReduceMarkov****$(G,\prec)$. If a probability distribution P satisfies the composition axiom, then*

$$\text{(GMP)} \Longleftrightarrow (S\text{-MP},\prec).$$

Proof: (GMP) $\Longrightarrow$ (S-MP,$\prec$) since every conditional independence relation in (S-MP,$\prec$) corresponds to a valid d-separation. We show (S-MP,$\prec$) $\Longrightarrow$ (GMP). Without any loss of generality, let $\prec$: $V_1 \prec \ldots \prec V_n$. The proof is by induction on the sequence of ordered vertices. Suppose that (S-MP,$\prec$) $\Longrightarrow$ (GMP) holds for $V_1,\ldots V_{i-1}$. Let $S_{i-1} = I_1 \cup \ldots \cup I_{i-1}$. Then, by the induction hypothesis, S_{i-1} contains all the conditional independence relations invoked by (LMP,$\prec$) for $V_1,\ldots V_{i-1}$. If V_i is not c-ordered, $I_i = I(\{V_i\}, \mathrm{mb}(V_i,A), A \setminus (\mathrm{mb}(V_i,A) \cup \{V_i\}))$ for all maximal ancestral sets A such that $V_i \in A \subseteq \mathrm{pre}_{G,\prec}(V_i)$, $A \notin \mathrm{rd}_{G,\prec}(V_i)$. The conditional independence relations invoked by (LMP,$\prec$) with respect to V_i and any $A \in \mathrm{rd}_{G,\prec}(V_i)$ can be derived from other conditional independence relations by Lemma 18. Thus, $S_i = S_{i-1} \cup I_i$ contains all the conditional independence relations invoked by (LMP,$\prec$) for $V_1,\ldots V_i$, which implies (GMP). If V_i is c-ordered, applying the arguments in the proof of (GMP) $\Longleftrightarrow$ (PMP,$\prec_c$), we have

$$I(\{V_i\}, \mathrm{pa}_G(V_i), \mathrm{pre}_{G,\prec}(V_i) \setminus (\{V_i\} \cup \mathrm{pa}_G(V_i) \cup \mathrm{sp}_G(V_i))).$$

By the induction hypothesis and the definition of a c-ordered vertex, we have for all $V_j \in \mathrm{dis}_G(V_i) \cap \mathrm{pre}_{G,\prec}(V_i)$

$$I(\{V_j\}, \mathrm{pa}_G(V_j), \mathrm{pre}_{G,\prec}(V_j) \setminus (\{V_j\} \cup \mathrm{pa}_G(V_j) \cup \mathrm{sp}_G(V_j))).$$

By the arguments in the proof of (GMP) $\Longleftrightarrow$ (RLMP,$\prec_c$), we have for all maximal ancestral sets A such that $V_i \in A \subseteq \mathrm{pre}_{G,\prec}(V_i)$

$$I(\{V_i\}, \mathrm{mb}(V_i,A), A \setminus (\mathrm{mb}(V_i,A) \cup \{V_i\})).$$

Therefore, $S_i = S_{i-1} \cup I_i$ derives all the conditional independence relations invoked by (GMP). ■

As we have seen earlier, the number of zero partial correlations critically depends on the number of c-ordered vertices in a given ordering. This motivates us to find the ordering with the most c-ordered vertices. An obvious way of finding this ordering is to explore the space of all the consistent orderings. However, this exhaustive search may become infeasible as the number of vertices grows. We propose a greedy algorithm to get an ordering that has a large number of c-ordered vertices. The basic idea is to first find a large c-component in which many vertices can be c-ordered and place the vertices consecutively in the ordering, then repeating this until we cannot find a set of vertices that can be c-ordered. To describe the algorithm, we define the following notion, which identifies the largest subset of a c-component that can be c-ordered.

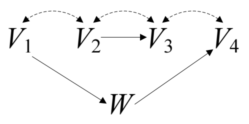


Figure 8: The c-component $\{V_1,V_2,V_3,V_4\}$ has the root set $\{V_1,V_2\}$

Definition 22 (Root Set) *The root set of a c-component C, denoted* $\mathrm{rt}(C)$ *is defined to be the set* $\{V_i \in C \mid \textit{there is no } V_j \in C \textit{ such that a directed path } V_j \to \ldots \to V_i \textit{ exists in } G\}$.

For example, the c-component $\{V_1,V_2,V_3,V_4\}$ in Figure 8 has the root set $\{V_1,V_2\}$. V_3 and V_4 are not in the root set since there are paths $V_2 \to V_3$ and $V_1 \to W \to V_4$. The root set has the following properties.

Proposition 23 *Let $\prec$ be a consistent ordering on the vertices of an ADMG G and C be a c-component of G. If the vertices in* $\mathrm{rt}(C)$ *are consecutive in $\prec$, then all the vertices in* $\mathrm{rt}(C)$ *are c-ordered in $\prec$.*

Proof: Assume that the vertices in $\mathrm{rt}(C)$ are consecutive in $\prec$. Then, for $X \in \mathrm{rt}(C)$, $\mathrm{dis}_G(X) \cap \mathrm{pre}_{G,\prec}(X) \subseteq \mathrm{rt}(C)$. Thus, there is no directed edge between any two vertices in $\mathrm{dis}_G(X) \cap \mathrm{pre}_{G,\prec}(X)$. ■

Proposition 24 *Let $\prec$ be a consistent ordering on the vertices of an ADMG G and C be a c-component of G. If a vertex X in C is c-ordered in $\prec$, then $X \in$* $\mathrm{rt}(C)$.

Proof: Assume that X is c-ordered in $\prec$. Suppose for a contradiction that $X \notin \mathrm{rt}(C)$. Then, there exists an ancestor Y of X in C. If there exists a vertex Z such that $Z \notin C$, $Y \to \cdots \to Z \to \cdots \to X$. Then, the first condition of a c-ordered vertex is violated. Otherwise, the second condition is violated. ■

Proposition 23 and 24 imply that the root set of a c-component is the largest subset of the c-component that can be c-ordered in a consistent ordering. If G does not have directed mixed cycles, $\mathrm{rt}(C) = C$ for every c-component C.

The procedure **GetOrdering** in Figure 9 is our proposed greedy algorithm that generates a good consistent ordering for G. In Step 1, it searches for the largest root set M and then merges all the vertices in M to one vertex V_M modifying edges accordingly. Then, it repeats the same operation for the modified graph until there is no root set that contains more than one vertex. Since the vertices in a root set are merged at each iteration, the modified graph is acyclic as otherwise there would be a directed path between two vertices in the root set, which contradicts the condition of a root set. After Step 1, we can easily obtain a consistent ordering for the original graph from the modified graph.

4.2 An Example

We show the application of the procedures **ReduceMarkov** and **GetOrdering** by considering the ADMG G in Figure 6 (a). First, we apply **GetOrdering** to get a consistent ordering on the vertices V of G. In Step 1, we first look for the largest root set. The c-component $\{V_6,V_7,V_8\}$ has the largest

```
procedure GetOrdering
INPUT: An ADMG $G$
OUTPUT: A consistent ordering $\prec$ on $V$
Step 1:
$G' \leftarrow G$ ($V'$ is the set of vertices of $G$)
while (there is a c-component $C$ of $G'$ such that $|\mathrm{rt}(C)| > 1$) do
  $M \leftarrow \emptyset$
  for each c-component $C$ of $G'$ do
    if $|\mathrm{rt}(C)| > |M|$ then
      $M \leftarrow \mathrm{rt}(C)$
    end if
  end for
  Add a vertex $V_M$ to $G'_{V' \setminus M}$
  Draw an edge $V_M \leftarrow X$ (respectively $V_M \rightarrow X, V_M \leftrightarrow X$) if there is
  $Y \leftarrow X$ (respectively $Y \rightarrow X, Y \leftrightarrow X$) in $G'$ such that $Y \in M, X \in V' \setminus M$
  Let $G'$ be the resulting graph
end while
Step 2:
Let $\prec'$ be any consistent ordering on $V'$. Construct a consistent ordering $\prec$ from $\prec'$ by replacing
each $V_S \in V' \setminus V$ with the vertices in $S$ (the ordering of the vertices in $S$ is arbitrary)
```

Figure 9: A greedy algorithm to generate a good consistent ordering on the vertices of an ADMG G

root set $\{V_6, V_7, V_8\}$. Then, the vertices in $\{V_6, V_7, V_8\}$ are merged into a vertex V_{678}. Figure 6 (b) shows the modified graph G' after the first iteration of the while loop. In the next iteration, we find that every c-component has the root set of size 1. Note that for $C = \{V_5, V_9\}$, $\mathrm{rt}(C) = \{V_5, V_9\}$ in G but $\mathrm{rt}(C) = \{V_5\}$ in G'. Thus, Step 1 ends. In Step 2, from G' in Figure 6 (b), we can obtain an ordering $\prec'$: $V_1 \prec V_2 \prec V_3 \prec V_4 \prec V_5 \prec V_{678} \prec V_9$. This is converted to a consistent ordering $\prec$: $V_1 \prec V_2 \prec V_3 \prec V_4 \prec V_5 \prec V_6 \prec V_7 \prec V_8 \prec V_9$ for G.

With the ordering $\prec$, we now apply **ReduceMarkov** to obtain a set of conditional independence relations that can derive those invoked by the global Markov property. It is easy to see that the vertices $V_1, \ldots, V_8$ are c-ordered in $\prec$. Thus, the following conditional independence relations corresponding to the pairwise Markov property are added to the set S (initially empty).

$$\begin{aligned}
&I(\{V_2\}, \emptyset, \{V_1\}), && I(\{V_3\}, \emptyset, \{V_2\}),\\
&I(\{V_4\}, \emptyset, \{V_3, V_1\}), && I(\{V_5\}, \emptyset, \{V_4, V_3, V_2, V_1\}),\\
&I(\{V_6\}, \emptyset, \{V_5, V_4, V_2\}), && I(\{V_6\}, \{V_3\}, \{V_1\}),\\
&I(\{V_7\}, \emptyset, \{V_5, V_4, V_2\}), && I(\{V_7\}, \{V_3\}, \{V_1\}),\\
&I(\{V_8\}, \emptyset, \{V_6, V_3, V_1\}), && I(\{V_8\}, \{V_4\}, \{V_2\}).
\end{aligned} \tag{24}$$

V_9 is not c-ordered in $\prec$ since V_5 is not adjacent in $\prec$. Thus, we use the ordered local Markov property (LMP,$\prec$) for V_9. The maximal ancestral sets that we need to consider are

$$A_1 = \text{an}_G(\{V_6,V_8,V_9\}) = \{V_1,V_2,V_3,V_4,V_5,V_6,V_7,V_8,V_9\} \text{ and}$$
$$A_2 = \text{an}_G(\{V_4,V_6,V_9\}) = \{V_1,V_2,V_3,V_4,V_6,V_7,V_9\}.$$

The corresponding conditional independence relations are

$$I(\{V_9\},\{V_7,V_5\},\{V_8,V_6,V_4,V_3,V_2,V_1\}), \tag{25}$$
$$I(\{V_9\},\{V_7\},\{V_6,V_4,V_3,V_2,V_1\}). \tag{26}$$

However, it turns out that $A_2 \in \text{rd}_{G,\prec}(V_9)$ and (26) is not added to S. We check the condition of Lemma 18. The global Markov property for $G_{\text{pre}_{G,\prec}(V_8)}$ is satisfied by (24). Also,

$$\text{dis}_{G_{A_1}}(V_9) = \{V_5,V_9\},$$
$$\text{dis}_{G_{A_2}}(V_9) = \{V_9\},$$
$$A_2 \cap \text{dis}_{G_{A_1}}(V_9) = \{V_9\} = \text{dis}_{G_{A_2}}(V_9),$$
$$\text{pa}_G(\text{dis}_{G_{A_1}}(V_9) \setminus \text{dis}_{G_{A_2}}(V_9)) = \emptyset \subseteq \{V_7\} = \text{mb}(V_9,A_2).$$

Therefore, the condition of Lemma 18 is satisfied and it follows that (26) is redundant. To see how much we reduced the testing requirements, the conditional independence relations invoked by (LMP,$\prec$) are shown below.

$$\begin{array}{ll}
I(\{V_2\},\emptyset,\{V_1\}), & I(\{V_3\},\{V_1\},\{V_2\}),\\
I(\{V_4\},\{V_2\},\{V_3,V_1\}), & I(\{V_5\},\emptyset,\{V_4,V_3,V_2,V_1\}),\\
I(\{V_6\},\{V_3\},\{V_5,V_4,V_2,V_1\}), & I(\{V_7\},\{V_3\},\{V_5,V_4,V_2,V_1\}),\\
I(\{V_7\},\{V_6,V_3\},\{V_5,V_4,V_2,V_1\}), & I(\{V_8\},\{V_5,V_4\},\{V_6,V_3,V_2,V_1\}),\\
I(\{V_8\},\{V_7,V_5,V_4,V_3\},\{V_2,V_1\}), & I(\{V_8\},\{V_7,V_6,V_5,V_4,V_3\},\{V_2,V_1\}),\\
I(\{V_9\},\{V_7\},\{V_6,V_4,V_3,V_2,V_1\}), & I(\{V_9\},\{V_7,V_5\},\{V_8,V_6,V_4,V_3,V_2,V_1\}).
\end{array} \tag{27}$$

S invokes 26 zero partial correlations while (LMP,$\prec$) invokes 39. Also, S involves much smaller conditioning sets. We have at most one vertex in each conditioning set in (24) and two vertices in (25) while 23 zero partial correlations in (27) involve more than 2 vertices in the conditioning set.

The ADMG G in this example turns out to be a MAG. As we discussed in Section 3.4.1, we have two options: either we use the constraints in (24) and (25) or the constraints given by the pairwise Markov property for MAGs. In this example, both sets of constraints involve the same number of zero partial correlations. However, the pairwise Markov property for MAGs involves much larger conditioning sets. For example, the pairwise Markov property for MAGs gives the following conditional independence relation for the pair V_6 and V_8: $I(\{V_8\},\{V_5,V_4,V_3,V_2,V_1\},\{V_6\})$. Our method uses an empty set as the conditioning set for the pair. Hence, in this example, we are better off using the constraints in (24) and (25).

4.3 Comparison of (LMP,$\prec$) and (S-MP,$\prec$)

From (23), it is clear that (S-MP,$\prec$) invokes fewer conditional independence relations than (LMP,$\prec$) if there are c-ordered vertices in $\prec$. But how much more economical is (S-MP,$\prec$) than (LMP,$\prec$) and for what type of graphs is the reduction large?

For simplicity, we will compare the number of conditional independence relations rather than zero partial correlations and ignore the reduction done by Lemma 18. For now assume

$$S = \bigcup_{X:X \text{ is c-ordered in } \prec} I(\{X\}, \mathrm{pa}_G(X), \mathrm{pre}_{G,\prec}(X) \setminus (\{X\} \cup \mathrm{pa}_G(X) \cup \mathrm{sp}_G(X))) \bigcup$$
$$\bigcup_{X:X \text{ is not c-ordered in } \prec} \Big(\bigcup_{\substack{\text{all maximal sets } A \\ \text{with respect to } \mathrm{mb}(X,A): \\ X \in A \subseteq \mathrm{pre}_{G,\prec}(X)}} I\big(\{X\}, \mathrm{mb}(X,A), A \setminus (\mathrm{mb}(X,A) \cup \{X\})\big)\Big).$$

Let $\mathrm{M}(X,\prec)$ be the number of different Markov blankets of a vertex X, that is, $\mathrm{M}(X,\prec) = \Big|\{\mathrm{dis}_{G_A}(X) \mid A \text{ is an ancestral set such that } X \in A \subseteq \mathrm{pre}_{G,\prec}(X)\}\Big|$, and $\mathrm{C}(\prec)$ be the set of vertices that are c-ordered in $\prec$. Then, (LMP,$\prec$) lists $\sum_{X \in V} \mathrm{M}(X,\prec)$ conditional independence relations and (S-MP,$\prec$) lists $|\mathrm{C}(\prec)| + \sum_{X \notin \mathrm{C}(\prec)} \mathrm{M}(X,\prec)$ conditional independence relations. Hence, the difference in the number of conditional independence relations between (LMP,$\prec$) and (S-MP,$\prec$) is

$$\sum_{X \in \mathrm{C}(\prec)} \Big(\mathrm{M}(X,\prec) - 1\Big).$$

This difference is large when $|\mathrm{C}(\prec)|$ or $\mathrm{M}(X,\prec)$ for each X is large.

The size of $\mathrm{C}(\prec)$ depends on the number of directed mixed cycles. From Definition 19, it follows that $\mathrm{C}(\prec)$ is large if there are a small number of directed mixed cycles. Note that a directed mixed cycle such as that in Figure 4 induces the violation of the first condition in Definition 19 and a directed mixed cycle of the form $\alpha \overset{\leftrightarrow}{\rightarrow} \beta$ induces the violation of the second condition in Definition 19.

$\mathrm{M}(X,\prec)$ depends on the structure of $\mathrm{dis}_G(X) \cap \mathrm{pre}_{G,\prec}(X)$. We will reformulate $\mathrm{M}(X,\prec)$ to show the properties that affect $\mathrm{M}(X,\prec)$. Let $G_{\leftrightarrow,\mathrm{dis}}(X,\prec) = (V', E')$ where $V' = \mathrm{dis}_G(X) \cap \mathrm{pre}_{G,\prec}(X)$ and $E' = \{V_i \leftrightarrow V_j \mid V_i \leftrightarrow V_j \text{ in } G_{V'}\}$. For example, for an ADMG G in Figure 8 and an ordering $V_1 \prec V_2 \prec V_3 \prec V_4$, $G_{\leftrightarrow,\mathrm{dis}}(V_3,\prec)$ is $V_1 \leftrightarrow V_2 \leftrightarrow V_3$. Let $G_{\leftrightarrow,\mathrm{dis}}(X,\prec)_S$ be the induced subgraph of $G_{\leftrightarrow,\mathrm{dis}}(X,\prec)$ on a set $S \subseteq \mathrm{dis}_G(X) \cap \mathrm{pre}_{G,\prec}(X)$. Then, $\mathrm{M}(X,\prec) = \Big|\{S \mid S \subseteq \mathrm{dis}_G(X) \cap \mathrm{pre}_{G,\prec}(X)$ such that $G_{\leftrightarrow,\mathrm{dis}}(X,\prec)_S$ is a *connected component* of $G_{\leftrightarrow,\mathrm{dis}}(X,\prec)_{S \cup (\mathrm{an}_G(S) \cap \mathrm{dis}_G(X) \cap \mathrm{pre}_{G,\prec}(X))}\}\Big|$ that is, $\mathrm{M}(X,\prec)$ corresponds to a set of subsets S of $\mathrm{dis}_G(X) \cap \mathrm{pre}_{G,\prec}(X)$ satisfying two conditions: (i) $G_{\leftrightarrow,\mathrm{dis}}(X,\prec)_S$ is connected; and (ii) for all $Y \in \Big(\mathrm{an}_G(S) \cap \mathrm{dis}_G(X) \cap \mathrm{pre}_{G,\prec}(X)\Big) \setminus S$, there is no path from Y to any vertices in S. The condition (i) implies that $\mathrm{M}(X,\prec)$ will be large if the vertices in $\mathrm{dis}_G(X) \cap \mathrm{pre}_{G,\prec}(X)$ are connected by many bi-directed edges. The condition (ii) implies that $\mathrm{M}(X,\prec)$ will be large if there are few directed mixed cycles. Note that for ADMGs without directed mixed cycles, (ii) trivially holds since $\Big(\mathrm{an}_G(S) \cap \mathrm{dis}_G(X) \cap \mathrm{pre}_{G,\prec}(X)\Big) \setminus S = \emptyset$. For example, consider a subset of vertices $\{V_1, \ldots, V_k\}$ in an ADMG with edges $V_i \leftrightarrow V_k, i = 1, \ldots, k-1$, which has no directed mixed cycles. Then, for an ordering $V_1 \prec \ldots \prec V_k$, $\mathrm{M}(V_k,\prec) = 2^{k-1}$. Also, consider a subset of vertices $\{V_1, \ldots, V_k\}$ in an ADMG with edges $V_1 \overset{\leftrightarrow}{\rightarrow} V_2 \overset{\leftrightarrow}{\rightarrow} \cdots \overset{\leftrightarrow}{\rightarrow} V_k$, which has $k-1$ directed mixed cycles. Then, $\mathrm{M}(V_k,\prec) = 1$. Hence, it is clear that $\mathrm{M}(X,\prec)$ is large if

1. the set $\mathrm{dis}_G(X) \cap \mathrm{pre}_{G,\prec}(X)$ is large,

2. there are many bi-directed edges connecting vertices in $\mathrm{dis}_G(X) \cap \mathrm{pre}_{G,\prec}(X)$, and

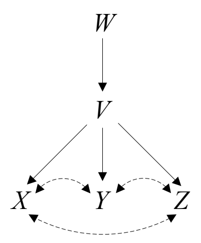


Figure 10: An example ADMG for which using (S-MP,$\prec$) is most beneficial. There is no directed mixed cycle and each c-component is a clique joined by bi-directed edges.

3. there are few directed mixed cycles.

Thus, (LMP,$\prec$) will invoke a large number of conditional independence relations for an ADMG with few directed mixed cycles and large c-components with many bi-directed edges. For such an ADMG, $\sum_{X\in \mathrm{C}(\prec)}\Big(\mathrm{M}(X,\prec)-1\Big)$, the reduction made by (S-MP,$\prec$), is also large. An extreme case is an ADMG that has no directed mixed cycles and each c-component of which is a clique joined by bi-directed edges. An example of such an ADMG is given in Figure 10. For this ADMG and an ordering $W \prec V \prec X \prec Y \prec Z$, (LMP,$\prec$) invokes $\mathrm{M}(W,\prec)+\mathrm{M}(V,\prec)+\mathrm{M}(X,\prec)+\mathrm{M}(Y,\prec)+\mathrm{M}(Z,\prec) = 1+1+1+2+4 = 9$ conditional independence relations while (S-MP,$\prec$) invokes $|\mathrm{C}(\prec)| = n = 5$ conditional independence relations. If we enlarge the clique joined by bi-directed edges such that it contains k vertices, then (LMP,$\prec$) invokes $2+\sum_{i=0}^{k-1}2^i = 1+2^k$ conditional independence relations while (S-MP,$\prec$) invokes $k+2$.

In general, although (S-MP,$\prec$) greatly reduces (LMP,$\prec$), it may still invoke an exponential number of conditional independence relations if there exist directed mixed cycles.

5. Conclusion and Discussion

We present local Markov properties for ADMGs representing linear SEMs with correlated errors. The results have applications in testing linear SEMs against the data by testing for zero partial correlations implied by the model. For general linear SEMs with correlated errors, we provide a procedure that lists a subset of zero partial correlations that will imply all other zero partial correlations implied by the model. In particular, for a class of models whose corresponding path diagrams contain no directed mixed cycles, this subset invokes one zero partial correlation for each pair of variables.

In general, our procedure may invoke an exponential number of zero partial correlations if the path diagram G satisfies all of the following properties: (i) G has large c-components; (ii) the vertices in each c-component are heavily connected by bi-directed edges; and (iii) G has directed mixed cycles. If one of these properties is not satisfied, then the number of zero partial correlations derived by our method is typically not exponential.

For the class of MAGs, which is a strict superclass of ADMGs without directed mixed cycles, one might use the pairwise Markov property for MAGs given in Richardson and Spirtes (2002) instead of our results in Section 4. However, when the two approaches give a similar number of constraints, it may be better to use our approach since it may use smaller conditioning sets as shown in the example in Section 4.2.

The potential advantages of testing linear SEMs based on vanishing partial correlations over the classical test method based on maximum likelihood estimation of the covariance matrix have been discussed in Pearl (1998), Shipley (2000), McDonald (2002) and Shipley (2003). The results presented in this paper provide a theoretical foundation for the practical applications of this test method in linear SEMs with correlated errors. How to implement this test method in practice still needs further study as it requires multiple testing of hypotheses about zero partial correlations (Shipley, 2000; Drton and Perlman, 2007). We also note that, in linear SEMs *without* correlated errors, all the constraints on the covariance matrix are implied by vanishing partial correlations. This also holds in linear SEMs *with* correlated errors that are represented by ADMGs *without* directed mixed cycles. However, it is possible that linear SEMs *with* correlated errors represented by ADMGs *with* directed mixed cycles may imply constraints on the covariance matrix that are not implied by zero partial correlations.

Although the intended application is in linear SEMs, the local Markov properties presented in the paper are valid for ADMGs associated with any probability distributions that satisfy the composition axiom. For example, any probability distribution that is faithful[5] to some DAG or undirected graph (and the marginals of the distribution) satisfies the composition axiom.

Model debugging for ADMGs using vanishing partial correlations is another area of current research. In this model debugging problem, the goal is to modify a graph based on the pattern of rejected hypotheses. The properties of ADMGs presented in this paper may facilitate the development of a new model debugging method.

Acknowledgments

We thank the anonymous reviewers for helpful comments. This research was partly supported by NSF grant IIS-0347846.

References

S.A. Andersson, D. Madigan, and M.D. Perlman. Alternative Markov properties for chain graphs. *Scandinavian Journal of Statistics*, 28:33–86, 2001.

K.A. Bollen. *Structural Equations with Latent Variables*. John Wiley, New York, 1989.

D.R. Cox and N. Wermuth. Linear dependencies represented by chain graphs. *Statistical Science*, 8(3):204–218, 1993.

M. Drton and M.D. Perlman. Multiple testing and error control in gaussian graphical model selection. *Statistical Science*, 22(3):430–449, 2007.

O.D. Duncan. *Introduction to Structural Equation Models*. Academic Press, New York, 1975.

M. Frydenberg. The chain graph markov property. *Scandinavian Journal of Statistics*, 17:333–353, 1990.

5. A probability distribution P is said to be faithful to a graph G if all the conditional independence relations embedded in P are encoded in G (via the global Markov property).

A.S. Goldberger. Structural equation models in the social sciences. *Econometrica: Journal of the Econometric Society*, 40:979–1001, 1972.

T. Haavelmo. The statistical implications of a system of simultaneous equations. *Econometrica*, 11: 1–12, 1943. Reprinted in D.F. Hendry and M.S. Morgan (Eds.), *The Foundations of Econometric Analysis*, Cambridge University Press, 477–490, 1995.

G. Kauermann. On a dualization of graphical Gaussian models. *Scandinavian Journal of Statistics*, 23:105–116, 1996.

J.T.A. Koster. On the validity of the Markov interpretation of path diagrams of gaussian structural equations systems with correlated errors. *Scandinavian Journal of Statistics*, 26:413–431, 1999.

S.L. Lauritzen. *Graphical Models*. Clarendon Press, Oxford, 1996.

S.L. Lauritzen and N. Wermuth. Graphical models for association between variables, some of which are qualitative and some quantitative. *Annals of Statistics*, 17:31–57, 1989.

S.L. Lauritzen, A.P. Dawid, B.N. Larsen, and H.G. Leimer. Independence properties of directed Markov fields. *Networks*, 20:491–505, 1990.

R.P. McDonald. What can we learn from the path equations?: Identifiability, constraints, equivalence. *Psychometrika*, 67(2):225–249, 2002.

J. Pearl. *Causality: Models, Reasoning, and Inference*. Cambridge University Press, NY, 2000.

J. Pearl. *Probabilistic Reasoning in Intelligent Systems*. Morgan Kaufmann, San Mateo, CA, 1988.

J. Pearl. Graphs, causality, and structural equation models. *Socioligical Methods and Research*, 27: 226–284, 1998.

J. Pearl and P. Meshkat. Testing regression models with fewer regressors. In *Proceedings of AI-STAT*, pages 255–259. 1999.

T. Richardson. Markov properties for acyclic directed mixed graphs. *Scandinavian Journal of Statistics*, 30(1):145–157, 2003.

T. Richardson and P. Spirtes. Ancestral graph Markov models. *Annals of Statistics*, 30(4):962–1030, 2002.

B. Shipley. A new inferential test for path models based on directed acyclic graphs. *Structural Equation Modeling*, 7:206–218, 2000.

B. Shipley. Testing recursive path models with correlated errors using d-separation. *Structural Equation Modeling*, 10:214–221, 2003.

P. Spirtes, T. Richardson, C. Meek, R. Scheines, and C. Glymour. Using path diagrams as a structural equation modeling tool. *Socioligical Methods and Research*, 27:182–225, 1998.

P. Spirtes, C. Glymour, and R. Scheines. *Causation, Prediction, and Search*. MIT Press, Cambridge, MA, 2001.

N. Wermuth and D.R. Cox. Graphical models: overview. *International Encyclopedia of the Social and Behavioral Sciences*, 9:6379–6386, 2001.

N. Wermuth and D.R. Cox. Joint response graphs and separation induced by triangular systems. *Journal of the Royal Statistical Society B*, 66:687–717, 2004.

S. Wright. The method of path coefficients. *Ann. Math. Statist.*, 5:161–215, 1934.

Journal of Machine Learning Research 10 (2008) 71-106 Submitted 4/08; Revised 10/08; Published 1/09

An Analysis of Convex Relaxations for MAP Estimation of Discrete MRFs

M. Pawan Kumar[*] PAWAN@ROBOTS.OX.AC.UK
Dept. of Engineering Science
University of Oxford
Parks Road, Oxford, OX1 3PJ
United Kingdom

Vladimir Kolmogorov VNK@ADASTRAL.UCL.AC.UK
Dept. of Computer Science
University College London
Martlesham Heath, IP5 3RE
United Kingdom

Philip H.S. Torr PHILIPTORR@BROOKES.AC.UK
Dept. of Computing
Oxford Brookes University
Wheatley, Oxford, OX33 1HX
United Kingdom

Editor: Martin Wainwright

Abstract

The problem of obtaining the maximum *a posteriori* estimate of a general discrete Markov random field (i.e., a Markov random field defined using a discrete set of labels) is known to be NP-hard. However, due to its central importance in many applications, several approximation algorithms have been proposed in the literature. In this paper, we present an analysis of three such algorithms based on convex relaxations: (i) LP-S: the linear programming (LP) relaxation proposed by Schlesinger (1976) for a special case and independently in Chekuri et al. (2001), Koster et al. (1998), and Wainwright et al. (2005) for the general case; (ii) QP-RL: the quadratic programming (QP) relaxation of Ravikumar and Lafferty (2006); and (iii) SOCP-MS: the second order cone programming (SOCP) relaxation first proposed by Muramatsu and Suzuki (2003) for two label problems and later extended by Kumar et al. (2006) for a general label set.

We show that the SOCP-MS and the QP-RL relaxations are equivalent. Furthermore, we prove that despite the flexibility in the form of the constraints/objective function offered by QP and SOCP, the LP-S relaxation *strictly dominates* (i.e., provides a better approximation than) QP-RL and SOCP-MS. We generalize these results by defining a large class of SOCP (and equivalent QP) relaxations which is dominated by the LP-S relaxation. Based on these results we propose some novel SOCP relaxations which define constraints using random variables that form cycles or cliques in the graphical model representation of the random field. Using some examples we show that the new SOCP relaxations strictly dominate the previous approaches.

Keywords: probabilistic models, MAP estimation, discrete MRF, convex relaxations, linear programming, second-order cone programming, quadratic programming, dominating relaxations

∗. Work done while at the Dept. of Computing, Oxford Brookes University.

1. Introduction

Discrete random fields are a powerful tool to obtain a probabilistic formulation for various applications in computer vision and related areas (Boykov et al., 2001; Cohen, 1986). Hence, developing accurate and efficient algorithms for performing inference on a given discrete random field is of fundamental importance. In this work, we will focus on the problem of maximum a posteriori (MAP) estimation. MAP estimation is a key step in obtaining the solutions to many applications such as stereo, image stitching and segmentation (Szeliski et al., 2006). Furthermore, it is closely related to many important Combinatorial Optimization problems such as MAXCUT (Goemans and Williamson, 1995), multi-way cut (Dalhaus et al., 1994), metric labelling (Boykov et al., 2001; Kleinberg and Tardos, 1999) and 0-extension (Boykov et al., 2001; Karzanov, 1998).

Given data $\mathbf{D}$, a discrete random field models the distribution (i.e., either the joint or the conditional probability) of a labelling for a set of random variables. Each of these variables $\mathbf{v} = \{v_0, v_1, \cdots, v_{n-1}\}$ can take a label from a discrete set $\mathbf{l} = \{l_0, l_1, \cdots, l_{h-1}\}$. A particular labelling of variables $\mathbf{v}$ is specified by a function f whose domain corresponds to the indices of the random variables and whose range is the index of the label set, that is,

$$f : \{0, 1, \cdots, n-1\} \rightarrow \{0, 1, \cdots, h-1\}.$$

In other words, random variable v_a takes label $l_{f(a)}$. For convenience, we assume the model to be a Markov random field (MRF) while noting that all the results of this paper also apply to conditional random fields (CRF).

An MRF specifies a neighbourhood relationship $\mathcal{E}$ between the random variables, that is, $(a,b) \in \mathcal{E}$ if, and only if, v_a and v_b are neighbouring random variables. Within this framework, the joint probability of a labelling f given data $\mathbf{D}$ is specified as

$$\Pr(f, \mathbf{D}|\theta) = \frac{1}{Z(\theta)} \exp(-Q(f, \mathbf{D}; \theta)).$$

Here θ represents the parameters of the MRF and $Z(\theta)$ is a normalization constant which ensures that the probability sums to one (known as the partition function). Assuming a pairwise MRF (i.e., an MRF with maximum clique size of 2 according to the neighbourhood relationship $\mathcal{E}$), the energy $Q(f, \mathbf{D}; \theta)$ is given by

$$Q(f, \mathbf{D}; \theta) = \sum_{v_a \in \mathbf{v}} \theta^1_{a;f(a)} + \sum_{(a,b) \in \mathcal{E}} \theta^2_{ab;f(a)f(b)}.$$

The term $\theta^1_{a;f(a)}$ is called a unary potential since its value depends on the labelling of one random variable at a time. Similarly, $\theta^2_{ab;f(a)f(b)}$ is called a pairwise potential as it depends on a pair of random variables. Note that the assumption of pairwise MRF is not a severe restriction as any MRF can be converted into an equivalent (i.e., representing the same probability distribution) pairwise MRF, for example, see Yedidia et al. (2001). For further simplicity of notation, we assume that $\theta^2_{ab;f(a)f(b)} = w(a,b)d(f(a), f(b))$ where $w(a,b)$ is the weight that indicates the strength of the pairwise relationship between variables v_a and v_b, with $w(a,b) = 0$ if $(a,b) \notin \mathcal{E}$, and $d(\cdot,\cdot)$ is a distance function on the labels.[1] As will be seen later, this formulation of the pairwise potentials would allow us to concisely describe our results.

1. The pairwise potentials for any MRF can be represented in the form $\theta^2_{ab;ij} = w(a,b)d(i,j)$. This can be achieved by using a larger set of labels $\hat{\mathbf{l}} = \{l_{0;0}, \cdots, l_{0;h_1}, \cdots, l_{n-1;h_1}\}$ such that the unary potential of v_a taking label $l_{b;i}$ is $\theta^1_{a;i}$ if

We note that a subclass of this problem where $w(a,b) \geq 0$ and the distance function $d(\cdot,\cdot)$ is a semi-metric or a metric has been well-studied in the literature (Boykov et al., 2001; Chekuri et al., 2001; Kleinberg and Tardos, 1999). However, we will focus on the general MAP estimation problem. In other words, unless explicitly stated, we do not place any restriction on the form of the unary and pairwise potentials.

The problem of MAP estimation for a discrete MRF is well known to be NP-hard in general. Since it plays a central role in several applications, many approximate algorithms have been proposed in the literature. In this work, we analyze three such algorithms which are based on convex relaxations. Specifically, we consider: (i) LP-S, the linear programming (LP) relaxation of Chekuri et al. (2001), Koster et al. (1998), Schlesinger (1976), and Wainwright et al. (2005); (ii) QP-RL, the quadratic programming (QP) relaxation of Ravikumar and Lafferty (2006); and (iii) SOCP-MS, the second order cone programming (SOCP) relaxation of Kumar et al. (2006) and Muramatsu and Suzuki (2003). In order to provide an outline of these relaxations, we formulate the problem of MAP estimation as an Integer Program (IP).

1.1 Integer Programming Formulation

We define a binary variable vector $\mathbf{x}$ of length nh. We denote the element of $\mathbf{x}$ at index $a \cdot h + i$ as $x_{a;i}$ where $v_a \in \mathbf{v}$ and $l_i \in \mathbf{l}$. These elements $x_{a;i}$ specify a labelling f such that

$$x_{a;i} = \begin{cases} 1 & \text{if} \quad f(a) = i, \\ -1 & \text{otherwise.} \end{cases}$$

We say that the variable $x_{a;i}$ *belongs to* variable v_a since it defines whether the variable v_a takes the label l_i. Let $\mathbf{X} = \mathbf{x}\mathbf{x}^\top$. We refer to the $(a \cdot h + i, b \cdot h + j)^{th}$ element of the matrix $\mathbf{X}$ as $X_{ab;ij}$ where $v_a, v_b \in \mathbf{v}$ and $l_i, l_j \in \mathbf{l}$. Clearly the sum of the unary potentials for a labelling specified by $\mathbf{x}$ is given by

$$\sum_{v_a, l_i} \theta^1_{a;i} \frac{(1 + x_{a;i})}{2}.$$

Similarly the sum of the pairwise potentials for a labelling $\mathbf{x}$ is given by

$$\sum_{(a,b) \in \mathcal{E}, l_i, l_j} \theta^2_{ab;ij} \frac{(1 + x_{a;i})}{2} \frac{(1 + x_{b;j})}{2} = \sum_{(a,b) \in \mathcal{E}, l_i, l_j} \theta^2_{ab;ij} \frac{(1 + x_{a;i} + x_{b;j} + X_{ab;ij})}{4}.$$

Hence, the following IP finds the labelling with the minimum energy, that is, it is equivalent to the MAP estimation problem:

$$\begin{aligned} \text{IP:} \quad & \mathbf{x}^* = \arg\min_{\mathbf{x}} \textstyle\sum_{v_a, l_i} \theta^1_{a;i} \frac{(1+x_{a;i})}{2} + \sum_{(a,b) \in \mathcal{E}, l_i, l_j} \theta^2_{ab;ij} \frac{(1+x_{a;i}+x_{b;j}+X_{ab;ij})}{4} \\ \text{s.t.} \quad & \mathbf{x} \in \{-1, 1\}^{nh}, & (1) \\ & \textstyle\sum_{l_i \in \mathbf{l}} x_{a;i} = 2 - h, & (2) \\ & \mathbf{X} = \mathbf{x}\mathbf{x}^\top. & (3) \end{aligned}$$

$a = b$ and ∞ otherwise. In other words, a variable v_a can only take labels from the set $\{l_{a;0}, \cdots, l_{a;h-1}\}$ since all other labels will result in an energy value of ∞. The pairwise potential for variables v_a and v_b taking labels $l_{a;i}$ and $l_{b;j}$ respectively can then be represented in the form $w(a,b)d(a;i,b;j)$ where $w(a,b) = 1$ and $d(a;i,b;j) = \theta^2_{ab;ij}$. Note that using a larger set of labels $\hat{\mathbf{l}}$ will increase the time complexity of MAP estimation algorithms, but does not affect the analysis presented in this paper.

Constraints (1) and (3) specify that the variables $\mathbf{x}$ and $\mathbf{X}$ are binary such that $X_{ab;ij} = x_{a;i}x_{b;j}$. We will refer to them as the *integer constraints*. Constraint (2), which specifies that each variable should be assigned only one label, is known as the *uniqueness constraint*. Note that one uniqueness constraint is specified for each variable v_a. Solving the above IP is in general NP-hard. It is therefore common practice to obtain an approximate solution using convex relaxations. We describe four such convex relaxations below.

1.2 Linear Programming Relaxation

The LP relaxation, proposed by Schlesinger (1976) for a special case (where the pairwise potentials specify a hard constraint, that is, they are either 0 or ∞) and independently in Chekuri et al. (2001), Koster et al. (1998), and Wainwright et al. (2005) for the general case, is given as follows:

$$\begin{aligned}
\text{LP-S:}\quad & \mathbf{x}^* = \arg\min_{\mathbf{x}} \textstyle\sum_{v_a,l_i} \theta^1_{a;i}\frac{(1+x_{a;i})}{2} + \sum_{(a,b)\in\mathcal{E},l_i,l_j}\theta^2_{ab;ij}\frac{(1+x_{a;i}+x_{b;j}+X_{ab;ij})}{4} \\
\text{s.t.}\quad & \mathbf{x}\in[-1,1]^{nh}, \mathbf{X}\in[-1,1]^{nh\times nh}, \\
& \textstyle\sum_{l_i\in\mathbf{l}} x_{a;i} = 2-h, \\
& \textstyle\sum_{l_j\in\mathbf{l}} X_{ab;ij} = (2-h)x_{a;i}, \qquad (4)\\
& X_{ab;ij} = X_{ba;ji}, \qquad (5)\\
& 1+x_{a;i}+x_{b;j}+X_{ab;ij} \geq 0. \qquad (6)
\end{aligned}$$

In the above relaxation, which we call LP-S, only those elements $X_{ab;ij}$ of $\mathbf{X}$ are used for which $(a,b)\in\mathcal{E}$ and $l_i,l_j\in\mathbf{l}$. Unlike the IP, the feasibility region of the above problem is relaxed such that the variables $x_{a;i}$ and $X_{ab;ij}$ lie in the interval $[-1,1]$. Further, the constraint (3) is replaced by Equation (4) which is called the *marginalization constraint* (Wainwright et al., 2005). One marginalization constraint is specified for each $(a,b)\in\mathcal{E}$ and $l_i\in\mathbf{l}$. Constraint (5) specifies that $\mathbf{X}$ is symmetric. Constraint (6) ensures that $\theta^2_{ab;ij}$ is multiplied by a number between 0 and 1 in the objective function. These constraints (5) and (6) are defined for all $(a,b)\in\mathcal{E}$ and $l_i,l_j\in\mathbf{l}$. The formulation of the LP-S relaxation presented here uses a slightly different notation to the ones described in Kolmogorov (2006) and Wainwright et al. (2005). However, it can easily be shown that the two formulations are equivalent by using the variables $\mathbf{y}$ and $\mathbf{Y}$ instead of $\mathbf{x}$ and $\mathbf{X}$ such that $y_{a;i} = \frac{1+x_{a;i}}{2}, Y_{ab;ij} = \frac{1+x_{a;i}+x_{b;j}+X_{ab;ij}}{4}$. Throughout this paper, we will make use of the variables $\mathbf{x}$ and $\mathbf{X}$ instead of $\mathbf{y}$ and $\mathbf{Y}$. As will be seen in the subsequent sections, this would allow us to concisely describe our results. Note that the above constraints are not exhaustive, that is, it is possible to specify other constraints for the problem of MAP estimation (e.g., see § 1.3 and 1.5).

1.2.1 PROPERTIES OF THE LP-S RELAXATION

- Since the LP-S relaxation specifies a linear program it can be solved in polynomial time. A labelling f can then be obtained by rounding the (possibly fractional) solution of the LP-S.

- Using the rounding scheme of Kleinberg and Tardos (1999), the LP-S provides a multiplicative bound[2] of 2 when the pairwise potentials form a Potts model (Chekuri et al., 2001).

2. Consider a set of optimization problems $\mathcal{A}$ and a relaxation scheme defined over this set $\mathcal{A}$. In other words, for every optimization problem A $\in\mathcal{A}$, the relaxation scheme provides a relaxation B $\in\mathcal{B}$ of A. Let e^A denote the optimal value of the optimization problem A. Further, let $\hat{e}^A$ denote the value of the objective function of A at the point obtained

- Using the rounding scheme of Chekuri et al. (2001), LP-S obtains a multiplicative bound of $2+\sqrt{2}$ for truncated linear pairwise potentials.
- LP-S provides a multiplicative bound of 1 when the energy function $Q(\cdot,\mathbf{D};\theta)$ of the MRF is submodular (Schlesinger and Flach, 2000) (also see Ishikawa 2003 and Schlesinger and Flach 2006 for the st-MINCUT graph construction for minimizing submodular energy functions).
- The LP-S relaxation provides the same optimal solution for all reparameterizations $\overline{\theta}$ of θ (Kolmogorov, 2006; Werner, 2007).[3]

We note here that, although the LP-S relaxation can be solved in polynomial time, the state of the art Interior Point algorithms can only handle up to a few thousand variables and constraints. In order to overcome this deficiency several efficient algorithms have been proposed in the literature for (approximately) solving the Lagrangian dual of LP-S (Kolmogorov, 2006; Komodakis et al., 2007; Schlesinger and Giginyak, 2007a,b; Wainwright et al., 2005; Werner, 2007). Recently, efficient methods have also been devised for solving the primal problem directly (Ravikumar et al., 2008).

1.3 Quadratic Programming Relaxation

We now describe the QP relaxation for the MAP estimation IP which was proposed by Ravikumar and Lafferty (2006). To this end, it would be convenient to reformulate the objective function of the IP using a vector of unary potentials of length nh (denoted by $\hat{\theta}^1$) and a matrix of pairwise potentials of size $nh \times nh$ (denoted by $\hat{\theta}^2$). The element of the unary potential vector at index $(a\cdot h+i)$ is defined as

$$\hat{\theta}^1_{a;i} = \theta^1_{a;i} - \sum_{v_c\in\mathbf{v}}\sum_{l_k\in\mathbf{l}} |\theta^2_{ac;ik}|,$$

where $v_a \in \mathbf{v}$ and $l_i \in \mathbf{l}$. The $(a\cdot h+i, b\cdot h+j)^{th}$ element of the pairwise potential matrix $\hat{\theta}^2$ is defined such that

$$\hat{\theta}^2_{ab;ij} = \begin{cases} \sum_{v_c\in\mathbf{v}}\sum_{l_k\in\mathbf{l}} |\theta^2_{ac;ik}|, & \text{if} \quad a=b, i=j, \\ \theta^2_{ab;ij} & \text{otherwise}, \end{cases}$$

where $v_a, v_b \in \mathbf{v}$ and $l_i, l_j \in \mathbf{l}$. In other words, the potentials are modified by defining a pairwise potential $\hat{\theta}^2_{aa;ii}$ and subtracting the value of that potential from the corresponding unary potential $\theta^1_{a;i}$. The advantage of this reformulation is that the matrix $\hat{\theta}^2$ is guaranteed to be positive semidefinite, that is, $\hat{\theta}^2 \succeq 0$. This can be seen by observing that for any vector $\mathbf{z}\in\mathbb{R}^{nh}$ the following holds true:

$$\begin{aligned} \mathbf{z}^\top\hat{\theta}^2\mathbf{z} &= \sum_{(a,b)\in\mathcal{E}}\sum_{l_i,l_j\in\mathbf{l}} \left(|\theta^2_{ab;ij}|z^2_{a;i} + |\theta^2_{ab;ij}|z^2_{b;j} + 2\theta^2_{ab;ij}z_{a;i}z_{b;j}\right), \\ &= \sum_{(a,b)\in\mathcal{E}}\sum_{l_i,l_j\in\mathbf{l}} \left(|\theta^2_{ab;ij}|\left(z_{a;i} + \frac{\theta^2_{ab;ij}}{|\theta^2_{ab;ij}|}z_{b;j}\right)^2\right), \\ &\geq 0. \end{aligned}$$

by rounding the optimal solution of its relaxation B. The relaxation scheme is said to provide a multiplicative bound of ρ for the set $\mathcal{A}$ if, and only if, the following condition is satisfied: $E(\hat{e}^A) \le \rho e^A, \forall A \in \mathcal{A}$, where $E(\cdot)$ denotes the expectation of its argument under the rounding scheme employed.

3. A parameter $\overline{\theta}$ is called a reparameterization of θ (denoted by $\overline{\theta}\equiv\theta$) if and only if $Q(f,\mathbf{D};\theta) = Q(f,\mathbf{D};\overline{\theta})$ for all labellings f.

Using the fact that for $x_{a;i} \in \{-1,1\}$,

$$\left(\frac{1+x_{a;i}}{2}\right)^2 = \frac{1+x_{a;i}}{2},$$

it can be shown that the following is equivalent to the MAP estimation problem (Ravikumar and Lafferty, 2006):

$$\begin{aligned}
\text{QP-RL:} \quad & \mathbf{x}^* = \arg\min_{\mathbf{x}} \left(\tfrac{\mathbf{1}+\mathbf{x}}{2}\right)^\top \hat{\theta}^1 + \left(\tfrac{\mathbf{1}+\mathbf{x}}{2}\right)^\top \hat{\theta}^2 \left(\tfrac{\mathbf{1}+\mathbf{x}}{2}\right), && (7)\\
\text{s.t.} \quad & \textstyle\sum_{l_i \in \mathbf{l}} x_{a;i} = 2-h, \forall v_a \in \mathbf{v}, && (8)\\
& \mathbf{x} \in \{-1,1\}^{nh}, && (9)
\end{aligned}$$

where $\mathbf{1}$ is a vector of appropriate dimensions whose elements are all equal to 1. It is worth noting that, although $\hat{\theta}^1$ and $\hat{\theta}^2$ define the same energy for a labelling f as the original parameter θ, they are not a valid reparameterization of θ since they contain non-zero pairwise potentials of the form $\hat{\theta}^2_{aa;ii}$. By relaxing the feasibility region of the above problem to $\mathbf{x} \in [-1,1]^{nh}$, the resulting QP can be solved in polynomial time since $\hat{\theta}^2 \succeq 0$ (i.e., the relaxation of the QP (7)-(9) is convex). We call the above relaxation QP-RL. Note that in Ravikumar and Lafferty (2006), the QP-RL relaxation was described using the variable $\mathbf{y} = \frac{\mathbf{1}+\mathbf{x}}{2}$. However, the above formulation can easily be shown to be equivalent to the one presented in Ravikumar and Lafferty (2006).

In Ravikumar and Lafferty (2006), the authors proposed a rounding scheme for QP-RL (different from the ones used in Chekuri et al. 2001 and Kleinberg and Tardos 1999) that provides an additive bound[4] of $\frac{S}{4}$ for the MAP estimation problem, where

$$S = \sum_{(a,b)\in\mathcal{E}} \sum_{l_i,l_j \in \mathbf{l}} |\theta^2_{ab;ij}|,$$

that is, S is the sum of the absolute values of all pairwise potentials (Ravikumar and Lafferty, 2006). Under their rounding scheme, this bound can be shown to be tight[5] using a random field defined over two random variables which specifies uniform unary potentials and Ising model pairwise potentials (see Fig. 2(a)). Further, they also proposed an efficient iterative procedure for solving the QP-RL relaxation approximately. However, unlike LP-S, no multiplicative bounds have been established for the QP-RL formulation for special cases of the MAP estimation problem.

1.4 Semidefinite Programming Relaxation

The SDP relaxation of the MAP estimation problem replaces the non-convex constraint $\mathbf{X} = \mathbf{x}\mathbf{x}^\top$ by the convex semidefinite constraint $\mathbf{X} - \mathbf{x}\mathbf{x}^\top \succeq 0$ (de Klerk et al., 2004; Goemans and Williamson, 1995; Lasserre, 2001) which can be expressed as

$$\begin{pmatrix} 1 & \mathbf{x}^\top \\ \mathbf{x} & \mathbf{X} \end{pmatrix} \succeq 0,$$

4. A relaxation scheme defined over the set of optimization problems $\mathcal{A}$ is said to provide an additive bound of σ for $\mathcal{A}$ if, and only if, the following holds true: $E(\hat{e}^A) \leq e^A + \sigma, \forall \text{A} \in \mathcal{A}$. Here e^A is the optimal value of A and $\hat{e}^A$ is the value obtained by rounding the solution of B.

5. The multiplicative bound specified by a relaxation scheme defined over the set of optimization problems $\mathcal{A}$ is said to be *tight* if, and only if, there exists an $\text{A} \in \mathcal{A}$ such that $E(\hat{e}^A) = \rho e^A$. Similarly, the additive bound specified by a relaxation scheme defined over $\mathcal{A}$ is said to be tight if, and only if, there exists an $\text{A} \in \mathcal{A}$ such that $E(\hat{e}^A) = e^A + \sigma$.

using Schur's complement (Boyd and Vandenberghe, 2004). Further, like LP-S, it relaxes the integer constraints by allowing the variables $x_{a;i}$ and $X_{ab;ij}$ to lie in the interval $[-1,1]$ with $X_{aa;ii} = 1$ for all $v_a \in \mathbf{v}, l_i \in \mathbf{l}$. Note that the value of $X_{aa;ii}$ is derived using the fact that $X_{aa;ii} = x_{a;i}^2$. Since $x_{a;i}$ can only take the values -1 or 1 in the MAP estimation IP, it follows that $X_{aa;ii} = 1$. The SDP relaxation is a well-studied approach which provides accurate solutions for the MAP estimation problem (e.g., see Wainwright and Jordan 2004). However, due to its computational inefficiency, it is not practically useful for large scale problems with $nh > 1000$. See however Olsson et al. (2007), Schellewald and Schnorr (2003) and Torr (2003).

1.5 Second Order Cone Programming Relaxation

We now describe the SOCP relaxation that was proposed by Muramatsu and Suzuki (2003) for the MAXCUT problem (i.e., MAP estimation with $h = 2$) and later extended for a general label set (Kumar et al., 2006). This relaxation, which we call SOCP-MS, is based on the technique of Kim and Kojima (2000). For completeness we first describe the general technique of Kim and Kojima (2000) and later show how SOCP-MS can be derived using it.

1.5.1 SOCP RELAXATIONS

In Kim and Kojima (2000), the authors observed that the SDP constraint $\mathbf{X} - \mathbf{x}\mathbf{x}^\top \succeq 0$ can be further relaxed to second order cone (SOC) constraints. Their technique uses the fact that the Frobenius inner product of two semidefinite matrices is non-negative. For example, consider the inner product of a fixed matrix $\mathbf{C} = \mathbf{U}\mathbf{U}^\top \succeq 0$ with $\mathbf{X} - \mathbf{x}\mathbf{x}^\top$ (which, by the SDP constraint, is also positive semidefinite). The non-negativity of this inner product can be expressed as an SOC constraint as follows:

$$\begin{aligned} \mathbf{C} \bullet (\mathbf{X} - \mathbf{x}\mathbf{x}^\top) &\geq 0, \\ \Rightarrow \|(\mathbf{U})^\top \mathbf{x}\|^2 &\leq \mathbf{C} \bullet \mathbf{X}. \end{aligned}$$

Hence, by using a set of matrices $\mathcal{S} = \{\mathbf{C}^k | \mathbf{C}^k = \mathbf{U}^k(\mathbf{U}^k)^\top \succeq 0, k = 1,2,\ldots,n_C\}$, the SDP constraint can be further relaxed to n_C SOC constraints,that is,

$$\Rightarrow \|(\mathbf{U}^k)^\top \mathbf{x}\|^2 \leq \mathbf{C}^k \bullet \mathbf{X}, k = 1, \cdots, n_C.$$

It can be shown that, for the above set of SOC constraints to be equivalent to the SDP constraint, $n_C = \infty$. However, in practice, we can only specify a finite set of SOC constraints. Each of these constraints may involve some or all variables $x_{a;i}$ and $X_{ab;ij}$. For example, if $C^k_{ab;ij} = 0$, then the k^{th} SOC constraint will not involve $X_{ab;ij}$ (since its coefficient will be 0).

1.5.2 THE SOCP-MS RELAXATION

Consider a pair of neighbouring variables v_a and v_b, that is, $(a,b) \in \mathcal{E}$, and a pair of labels l_i and l_j. These two pairs define the following variables: $x_{a;i}$, $x_{b;j}$, $X_{aa;ii} = X_{bb;jj} = 1$ and $X_{ab;ij} = X_{ba;ji}$ (since $\mathbf{X}$ is symmetric). For each such pair of variables and labels, the SOCP-MS relaxation specifies two SOC constraints which involve only the above variables (Kumar et al., 2006; Muramatsu and Suzuki, 2003). In order to specify the exact form of these SOC constraints, we need the following definitions.

Using the variables v_a and v_b (where $(a,b) \in \mathcal{E}$) and labels l_i and l_j, we define the submatrices $\mathbf{x}^{(a,b,i,j)}$ and $\mathbf{X}^{(a,b,i,j)}$ of $\mathbf{x}$ and $\mathbf{X}$ respectively as:

$$\mathbf{x}^{(a,b,i,j)} = \begin{pmatrix} x_{a;i} \\ x_{b;j} \end{pmatrix}, \mathbf{X}^{(a,b,i,j)} = \begin{pmatrix} X_{aa;ii} & X_{ab;ij} \\ X_{ba;ji} & X_{bb;jj} \end{pmatrix}.$$

The SOCP-MS relaxation specifies SOC constraints of the form:

$$||(\mathbf{U}^k_{MS})^\top \mathbf{x}^{(a,b,i,j)}||^2 \leq \mathbf{C}^k_{MS} \bullet \mathbf{X}^{(a,b,i,j)}, \tag{10}$$

for all pairs of neighbouring variables $(a,b) \in \mathcal{E}$ and labels $l_i, l_j \in \mathbf{l}$. To this end, it uses the following two matrices:

$$\mathbf{C}^1_{MS} = \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}, \mathbf{C}^2_{MS} = \begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix}.$$

In other words SOCP-MS specifies a total of $2|\mathcal{E}|h^2$ SOC constraints. Note that both the matrices $\mathbf{C}^1_{MS}$ and $\mathbf{C}^2_{MS}$ defined above are positive semidefinite, and hence can be written as $\mathbf{C}^1_{MS} = \mathbf{U}^1_{MS}(\mathbf{U}^1_{MS})^\top$ and $\mathbf{C}^2_{MS} = \mathbf{U}^2_{MS}(\mathbf{U}^2_{MS})^\top$ where

$$\mathbf{U}^1_{MS} = \begin{pmatrix} 0 & 1 \\ 0 & 1 \end{pmatrix} \text{ and } \mathbf{U}^2_{MS} = \begin{pmatrix} 0 & -1 \\ 0 & 1 \end{pmatrix},$$

Substituting these matrices in inequality (10) we see that the constraints defined by the SOCP-MS relaxation are given by

$$\begin{aligned}
& ||(\mathbf{U}^1_{MS})^\top \mathbf{x}^{(a,b,i,j)}||^2 \leq \mathbf{C}^1_{MS} \bullet \mathbf{X}^{(a,b,i,j)}, \\
& ||(\mathbf{U}^2_{MS})^\top \mathbf{x}^{(a,b,i,j)}||^2 \leq \mathbf{C}^2_{MS} \bullet \mathbf{X}^{(a,b,i,j)}, \\
\Rightarrow \quad & \left|\left|\begin{pmatrix} 0 & 0 \\ 1 & 1 \end{pmatrix}\begin{pmatrix} x_{a;i} \\ x_{b;j} \end{pmatrix}\right|\right|^2 = \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix} \bullet \begin{pmatrix} X_{aa;ii} & X_{ab;ij} \\ X_{ba;ji} & X_{bb;jj} \end{pmatrix}, \\
& \left|\left|\begin{pmatrix} 0 & 0 \\ -1 & 1 \end{pmatrix}\begin{pmatrix} x_{a;i} \\ x_{b;j} \end{pmatrix}\right|\right|^2 = \begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix} \bullet \begin{pmatrix} X_{aa;ii} & X_{ab;ij} \\ X_{ba;ji} & X_{bb;jj} \end{pmatrix}, \\
\Rightarrow \quad & (x_{a;i} + x_{b;j})^2 \leq X_{aa;ii} + X_{bb;jj} + X_{ab;ij} + X_{ba;ji}, \\
& (x_{a;i} - x_{b;j})^2 \leq X_{aa;ii} + X_{bb;jj} - X_{ab;ij} - X_{ba;ji}, \\
\Rightarrow \quad & (x_{a;i} + x_{b;j})^2 \leq 2 + 2X_{ab;ij}, \\
& (x_{a;i} - x_{b;j})^2 \leq 2 - 2X_{ab;ij}.
\end{aligned}$$

The last expression is obtained using the fact that $\mathbf{X}$ is symmetric and $X_{aa;ii} = 1$, for all $v_a \in \mathbf{v}$ and $l_i \in \mathbf{l}$. Hence, in the SOCP-MS formulation, the MAP estimation IP is relaxed to

$$\begin{aligned}
\text{SOCP-MS:} \quad & \mathbf{x}^* = \arg\min_{\mathbf{x}} \textstyle\sum_{v_a,l_i} \theta^1_{a;i} \frac{(1+x_{a;i})}{2} + \sum_{(a,b)\in\mathcal{E},l_i,l_j} \theta^2_{ab;ij} \frac{(1+x_{a;i}+x_{b;j}+X_{ab;ij})}{4} \\
\text{s.t.} \quad & \mathbf{x} \in [-1,1]^{nh}, \mathbf{X} \in [-1,1]^{nh \times nh}, \\
& \textstyle\sum_{l_i \in \mathbf{l}} x_{a;i} = 2 - h, \\
& (x_{a;i} - x_{b;j})^2 \leq 2 - 2X_{ab;ij}, && (11) \\
& (x_{a;i} + x_{b;j})^2 \leq 2 + 2X_{ab;ij}, && (12) \\
& X_{ab;ij} = X_{ba;ji}. && (13)
\end{aligned}$$

We refer the reader to Kumar et al. (2006) and Muramatsu and Suzuki (2003) for details. The SOCP-MS relaxation yields the supremum and infimum for the elements of the matrix $\mathbf{X}$ using constraints (11) and (12) respectively, that is,

$$\frac{(x_{a;i}+x_{b;j})^2}{2} - 1 \leq X_{ab;ij} \leq 1 - \frac{(x_{a;i}-x_{b;j})^2}{2}.$$

These constraints are specified for all $(a,b) \in \mathcal{E}$ and $l_i, l_j \in \mathbf{l}$. When the objective function of SOCP-MS is minimized, one of the two inequalities would be satisfied as an equality. This can be proved by assuming that the value for the vector $\mathbf{x}$ has been fixed. Hence, the elements of the matrix $\mathbf{X}$ should take values such that it minimizes the objective function subject to the constraints (11) and (12). Clearly, the objective function would be minimized when $X_{ab;ij}$ equals either its supremum or infimum value, depending on the sign of the corresponding pairwise potential $\theta^2_{ab;ij}$, that is,

$$X_{ab;ij} = \begin{cases} \frac{(x_{a;i}+x_{b;j})^2}{2} - 1 & \text{if} \quad \theta^2_{ab;ij} > 0, \\ 1 - \frac{(x_{a;i}-x_{b;j})^2}{2} & \text{otherwise.} \end{cases}$$

Similar to the LP-S and QP-RL relaxations defined above, the SOCP-MS relaxation can also be solved in polynomial time. To the best our knowledge, no bounds have been established for the SOCP-MS relaxation in earlier work. Our recent work (Kumar and Torr, 2008) provides an efficient approach for solving general SOCP relaxations of the MAP estimation problem for discrete MRF.

2. Comparing Relaxations

In order to compare the relaxations described above, we require the following definitions. We say that a relaxation A *dominates* (Chekuri et al., 2001) the relaxation B (alternatively, B is dominated by A) if and only if

$$\min_{(\mathbf{x},\mathbf{X})\in\mathcal{F}(\text{A})} e(\mathbf{x},\mathbf{X};\theta) \geq \min_{(\mathbf{x},\mathbf{X})\in\mathcal{F}(\text{B})} e(\mathbf{x},\mathbf{X};\theta), \forall\theta,$$

where $\mathcal{F}(\text{A})$ and $\mathcal{F}(\text{B})$ are the feasibility regions of the relaxations A and B respectively. The term $e(\mathbf{x},\mathbf{X};\theta)$ denotes the value of the objective function at $(\mathbf{x},\mathbf{X})$ (i.e., the energy of the possibly fractional labelling $(\mathbf{x},\mathbf{X})$) for the MAP estimation problem defined over the MRF with parameter θ. Thus the optimal value of the dominating relaxation A is always greater than or equal to the optimal value of relaxation B. We note here that the concept of domination has been used previously by Chekuri et al. (2001) (to compare LP-S with the linear programming relaxation of Kleinberg and Tardos 1999).

Relaxations A and B are said to be *equivalent* if A dominates B and B dominates A, that is, their optimal values are equal to each other for all MRFs. A relaxation A is said to *strictly dominate* relaxation B if A dominates B but B does not dominate A. In other words there exists at least one MRF with parameter θ such that

$$\min_{(\mathbf{x},\mathbf{X})\in\mathcal{F}(\text{A})} e(\mathbf{x},\mathbf{X};\theta) > \min_{(\mathbf{x},\mathbf{X})\in\mathcal{F}(\text{B})} e(\mathbf{x},\mathbf{X};\theta).$$

Note that, by definition, the optimal value of any relaxation would always be less than or equal to the energy of the optimal (i.e., the MAP) labelling. Hence, the optimal value of a strictly dominating relaxation A is closer to the optimal value of the MAP estimation IP compared to that of relaxation

B. In other words, A provides a tighter lower bound for MAP estimation than B and, in that sense, is a better relaxation than B. It is worth noting that the concept of domination (or strict domination) does not apply to the final solutions obtained by rounding the possibly fractional solutions of the relaxation. In other words, it is possible for a relaxation B which is strictly dominated by a relaxation A to provide a better final (integer) solution than A. However, the concept of domination provides an intuitive measure for comparing relaxations. One may also argue that if a dominating relaxation provides worse final solutions, then the deficiency of the method may be rooted in the rounding technique used.

We now describe two special cases of domination which are used extensively in the remainder of this paper.

Case I: Consider two relaxations A and B which share a common objective function. For example, the objective functions of the LP-S and the SOCP-MS relaxations described in the previous section have the same form. Further, let A and B differ in the constraints that they specify such that $\mathcal{F}(\text{A}) \subseteq \mathcal{F}(\text{B})$, that is, the feasibility region of A is a subset of the feasibility region of B.

Given two such relaxations, we claim that A dominates B. This can be proved by contradiction: assume that A does not dominate B then, by definition of domination, there exists at least one parameter θ for which B provides a greater value of the objective function than A. Let an optimal solution of A be $(\mathbf{x}_A, \mathbf{X}_A)$. Similarly, let $(\mathbf{x}_B, \mathbf{X}_B)$ be an optimal solution of B. By our assumption, the following holds true:

$$e(\mathbf{x}_A, \mathbf{X}_A; \theta) < e(\mathbf{x}_B, \mathbf{X}_B; \theta). \tag{14}$$

However, since $\mathcal{F}(\text{A}) \subseteq \mathcal{F}(\text{B})$ it follows that $(\mathbf{x}_A, \mathbf{X}_A) \in \mathcal{F}(\text{B})$. Hence, from Equation (14), we see that $(\mathbf{x}_B, \mathbf{X}_B)$ cannot be an optimal solution of B. This proves our claim.

We can also consider a case where $\mathcal{F}(\text{A}) \subset \mathcal{F}(\text{B})$, that is, the feasibility region of A is a strict subset of the feasibility region of B. Using the above argument we see that A dominates B. Further, assume that there exists a parameter θ such that the intersection of the set of all optimal solutions of A and the set of all optimal solutions of B is null. In other words if $(\mathbf{x}_B, \mathbf{X}_B)$ is an optimal solution of B then $(\mathbf{x}_B, \mathbf{X}_B) \notin \mathcal{F}(\text{A})$. Clearly, if such a parameter θ exists then A strictly dominates B.

Note that the converse is not true, that is, if A dominates (or strictly dominates) B it does not necessarily imply that $\mathcal{F}(\text{A}) \subseteq \mathcal{F}(\text{B})$ (or $\mathcal{F}(\text{A}) \subset \mathcal{F}(\text{B})$). In other words, the concept of domination is related to the value of the objective function and not to the feasibility region. In Section 5, we will consider some examples of relaxations which can be related through the concept of domination despite the fact that their feasibility regions are not subsets (or supersets) of each other.

Case II: Consider two relaxations A and B such that they share a common objective function. Further, let the constraints of B be a subset of the constraints of A. We claim that A dominates B. This follows from the fact that $\mathcal{F}(\text{A}) \subseteq \mathcal{F}(\text{B})$ and the argument used in Case I above.

2.1 Our Results

We prove that LP-S strictly dominates SOCP-MS (see Section 3). Further, in Section 4, we show that QP-RL is equivalent to SOCP-MS. This implies that LP-S strictly dominates the QP-RL relaxation. In Section 5 we generalize the above results by proving that a large class of SOCP (and equivalent QP) relaxations is dominated by LP-S. Based on these results, we propose a novel set of constraints which result in SOCP relaxations that dominate LP-S, QP-RL and SOCP-MS. These relaxations introduce SOC constraints on cycles and cliques formed by the neighbourhood relationship of the MRF.

A preliminary version of this article has appeared as Kumar et al. (2007).

3. LP-S vs. SOCP-MS

We now show that for the MAP estimation problem the linear constraints of LP-S, that is,

$$\mathbf{x} \in [-1,1]^{nh}, \mathbf{X} \in [-1,1]^{nh \times nh}, \tag{15}$$
$$\textstyle\sum_{l_i \in \mathbf{l}} x_{a;i} = 2-h, \tag{16}$$
$$\textstyle\sum_{l_j \in \mathbf{l}} X_{ab;ij} = (2-h)x_{a;i}, \tag{17}$$
$$X_{ab;ij} = X_{ba;ji}, \tag{18}$$
$$1 + x_{a;i} + x_{b;j} + X_{ab;ij} \geq 0. \tag{19}$$

are stronger than the SOCP-MS constraints, that is,

$$\mathbf{x} \in [-1,1]^{nh}, \mathbf{X} \in [-1,1]^{nh \times nh},$$
$$\textstyle\sum_{l_i \in \mathbf{l}} x_{a;i} = 2-h,$$
$$(x_{a;i} - x_{b;j})^2 \leq 2 - 2X_{ab;ij}, \tag{20}$$
$$(x_{a;i} + x_{b;j})^2 \leq 2 + 2X_{ab;ij}, \tag{21}$$
$$X_{ab;ij} = X_{ba;ji}.$$

In other words the feasibility region of LP-S is a strict subset of the feasibility region of SOCP-MS (i.e., $\mathcal{F}(\text{LP-S}) \subset \mathcal{F}(\text{SOCP-MS})$). This in turn would allow us to prove the following Theorem.

Theorem 1: The LP-S relaxation strictly dominates the SOCP-MS relaxation.

Proof: The LP-S and the SOCP-MS relaxations differ only in the way they relax the non-convex constraint $\mathbf{X} = \mathbf{x}\mathbf{x}^\top$. While LP-S relaxes $\mathbf{X} = \mathbf{x}\mathbf{x}^\top$ using the marginalization constraint (17), SOCP-MS relaxes it to constraints (20) and (21). The SOCP-MS constraints provide the supremum and infimum of $X_{ab;ij}$ as

$$\frac{(x_{a;i} + x_{b;j})^2}{2} - 1 \leq X_{ab;ij} \leq 1 - \frac{(x_{a;i} - x_{b;j})^2}{2}.$$

Consider a pair of neighbouring variables v_a and v_b and a pair of labels l_i and l_j. Recall that SOCP-MS specifies the constraints (20) and (21) for all such pairs of random variables and labels, that is, for all $x_{a;i}, x_{b;j}, X_{ab;ij}$ such that $(a,b) \in \mathcal{E}$ and $l_i, l_j \in \mathbf{l}$. In order to prove this Theorem we use the following two Lemmas.

Lemma 3.1: If $x_{a;i}$, $x_{b;j}$ and $X_{ab;ij}$ satisfy the LP-S constraints, that is, constraints (15)-(19), then

$$|x_{a;i} - x_{b;j}| \leq 1 - X_{ab;ij}.$$

The above result holds true for all $(a,b) \in \mathcal{E}$ and $l_i, l_j \in \mathbf{l}$.

Proof: From the LP-S constraints, we get

$$\frac{1 + x_{a;i}}{2} = \sum_{l_k \in \mathbf{l}} \frac{1 + x_{a;i} + x_{b;k} + X_{ab;ik}}{4},$$
$$\frac{1 + x_{b;j}}{2} = \sum_{l_k \in \mathbf{l}} \frac{1 + x_{a;k} + x_{b;j} + X_{ab;kj}}{4}. \tag{22}$$

Therefore,

$$\begin{aligned} |x_{a;i} - x_{b;j}| &= 2\left|\tfrac{1+x_{a;i}}{2} - \tfrac{1+x_{b;j}}{2}\right|, \\ &= 2\left|\left(\tfrac{1+x_{a;i}}{2} - \tfrac{1+x_{a;i}+x_{b;j}+X_{ab;ij}}{4}\right) - \left(\tfrac{1+x_{b;j}}{2} - \tfrac{1+x_{a;i}+x_{b;j}+X_{ab;ij}}{4}\right)\right|, \\ &\leq 2\left(\tfrac{1+x_{a;i}}{2} - \tfrac{1+x_{a;i}+x_{b;j}+X_{ab;ij}}{4}\right) + 2\left(\tfrac{1+x_{b;j}}{2} - \tfrac{1+x_{a;i}+x_{b;j}+X_{ab;ij}}{4}\right), \\ &= 1 - X_{ab;ij}. \end{aligned}$$

Note that the inequality holds since both the expressions in the parentheses, that is,

$$\left(\frac{1+x_{a;i}}{2} - \frac{1+x_{a;i}+x_{b;j}+X_{ab;ij}}{4}\right), \left(\frac{1+x_{b;j}}{2} - \frac{1+x_{a;i}+x_{b;j}+X_{ab;ij}}{4}\right),$$

are non-negative, as follows from equations (19) and (22). ∎

Using the above Lemma, we get

$$(x_{a;i} - x_{b;j})^2 \leq (1 - X_{ab;ij})(1 - X_{ab;ij}), \quad (23)$$
$$\Rightarrow \quad (x_{a;i} - x_{b;j})^2 \leq 2(1 - X_{ab;ij}), \quad (24)$$
$$\Rightarrow \quad (x_{a;i} - x_{b;j})^2 \leq 2 - 2X_{ab;ij}. \quad (25)$$

Inequality (24) is obtained using the fact that $-1 \leq X_{ab;ij} \leq 1$ and hence, $1 - X_{ab;ij} \leq 2$. Using inequality (23), we see that the necessary condition for the equality to hold true is $(1 - X_{ab;ij})(1 - X_{ab;ij}) = 2 - 2X_{ab;ij}$, that is, $X_{ab;ij} = -1$. Note that inequality (25) is equivalent to the SOCP-MS constraint (20). Thus LP-S provides a smaller supremum of $X_{ab;ij}$ when $X_{ab;ij} > -1$.

Lemma 3.2: If $x_{a;i}, x_{b;j}$ and $X_{ab;ij}$ satisfy the LP-S constraints then

$$|x_{a;i} + x_{b;j}| \leq 1 + X_{ab;ij}.$$

This holds true for all $(a,b) \in \mathcal{E}$ and $l_i, l_j \in \mathbf{l}$.

Proof: According to constraint (19),

$$-(x_{a;i} + x_{b;j}) \leq 1 + X_{ab;ij}. \quad (26)$$

Using Lemma 3.1, we get the following set of inequalities:

$$|x_{a;i} - x_{b;k}| \leq 1 - X_{ab;ik}, l_k \in \mathbf{l}, k \neq j.$$

Adding the above set of inequalities, we get

$$\begin{aligned} & \textstyle\sum_{l_k \in \mathbf{l}, k \neq j} |x_{a;i} - x_{b;j}| \leq \sum_{l_k \in \mathbf{l}, k \neq j} (1 - X_{ab;ik}), \\ \Rightarrow \quad & \textstyle\sum_{l_k \in \mathbf{l}, k \neq j} (x_{a;i} - x_{b;k}) \leq \sum_{l_k \in \mathbf{l}, k \neq j} (1 - X_{ab;ik}), \\ \Rightarrow \quad & (h-1)x_{a;i} - \textstyle\sum_{l_k \in \mathbf{l}, k \neq j} x_{b;k} \leq (h-1) - \sum_{l_k \in \mathbf{l}, k \neq j} X_{ab;ik}, \\ \Rightarrow \quad & (h-1)x_{a;i} + (h-2) + x_{b;j} \leq (h-1) + (h-2)x_{a;i} + X_{ab;ij}. \end{aligned}$$

The last step is obtained using constraints (16) and (17), that is,

$$\sum_{l_k \in \mathbf{l}} x_{b;k} = (2-h), \sum_{l_k \in \mathbf{l}} X_{ab;ik} = (2-h)x_{a;i}.$$

Rearranging the terms, we get

$$(x_{a;i} + x_{b;j}) \leq 1 + X_{ab;ij}. \tag{27}$$

Thus, according to inequalities (26) and (27)

$$|x_{a;i} + x_{b;j}| \leq 1 + X_{ab;ij}. \quad \blacksquare$$

Using the above Lemma, we obtain

$$\begin{aligned} & (x_{a;i} + x_{b;j})^2 \leq (1 + X_{ab;ij})(1 + X_{ab;ij}), \\ \Rightarrow \quad & (x_{a;i} + x_{b;j})^2 \leq 2 + 2X_{ab;ij}. \end{aligned}$$

where the necessary condition for the equality to hold true is $1 + X_{ab;ij} = 2$ (i.e., $X_{ab;ij} = 1$). Note that the above constraint is equivalent to the SOCP-MS constraint (21). Together with inequality (25), this proves that the LP-S relaxation provides smaller supremum and larger infimum of the elements of the matrix $\mathbf{X}$ than the SOCP-MS relaxation. Thus, $\mathcal{F}(\text{LP-S}) \subset \mathcal{F}(\text{SOCP-MS})$.

One can also construct a parameter θ for which the set of all optimal solutions of SOCP-MS do not lie in the feasibility region of LP-S. In other words the optimal solutions of SOCP-MS belong to the non-empty set $\mathcal{F}(\text{SOCP-MS}) - \mathcal{F}(\text{LP-S})$, for example, see Fig. 1. Using the argument of Case I in Section 2, this implies that LP-S strictly dominates SOCP-MS. $\blacksquare$

Note that the above Theorem does not apply to the variation of SOCP-MS described in Kumar et al. (2006) and Muramatsu and Suzuki (2003) which include *triangular inequalities* (Chopra and Rao, 1993). However, since triangular inequalities are linear constraints, LP-S can be extended to include them. The resulting LP relaxation would strictly dominate the SOCP-MS relaxation with triangular inequalities.

4. QP-RL vs. SOCP-MS

We now prove that QP-RL and SOCP-MS are equivalent (i.e., their optimal values are equal for MAP estimation problems defined over all MRFs). Specifically, we consider a vector $\mathbf{x}$ which lies in the feasibility regions of the QP-RL and SOCP-MS relaxations, that is, $\mathbf{x} \in [-1,1]^{nh}$. For this vector, we show that the values of the objective functions of the QP-RL and SOCP-MS relaxations are equal. This would imply that if $\mathbf{x}^*$ is an optimal solution of QP-RL for some MRF with parameter θ then there exists an optimal solution $(\mathbf{x}^*, \mathbf{X}^*)$ of the SOCP-MS relaxation. Further, if e^Q and e^S are the optimal values of the objective functions obtained using the QP-RL and SOCP-MS relaxation, then $e^Q = e^S$.

Theorem 2: The QP-RL relaxation and the SOCP-MS relaxation are equivalent.

Proof: Recall that the QP-RL relaxation is defined as follows:

$$\begin{aligned} \text{QP-RL:} \quad & \mathbf{x}^* = \arg\min_{\mathbf{x}} \left(\tfrac{\mathbf{1}+\mathbf{x}}{2}\right)^\top \hat{\theta}^1 + \left(\tfrac{\mathbf{1}+\mathbf{x}}{2}\right)^\top \hat{\theta}^2 \left(\tfrac{\mathbf{1}+\mathbf{x}}{2}\right), \\ \text{s.t.} \quad & \textstyle\sum_{l_i \in \mathbf{l}} x_{a;i} = 2 - h, \forall v_a \in \mathbf{v}, \\ & \mathbf{x} \in \{-1,1\}^{nh}, \end{aligned}$$

where the unary potential vector $\hat{\theta}^1$ and the pairwise potential matrix $\hat{\theta}^2 \succeq 0$ are defined as

$$\hat{\theta}^1_{a;i} = \theta^1_{a;i} - \sum_{v_c \in \mathbf{v}} \sum_{l_k \in \mathbf{l}} |\theta^2_{ac;ik}|, \tag{28}$$

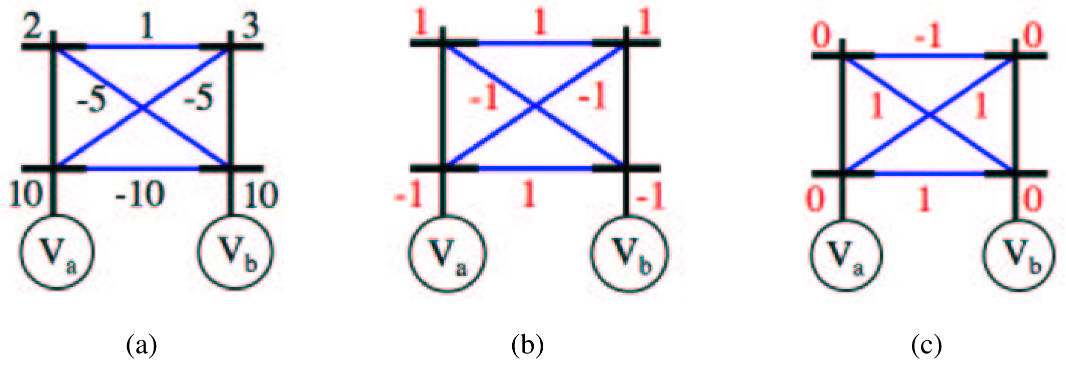


Figure 1: **(a)** An example MRF defined using two neighbouring random variables. Note that the observed nodes are not shown for the sake of clarity of the image. Each random variable can take one of two labels, represented by the branches (i.e., the horizontal lines) of the trellises (i.e., the vertical lines) on top of the variables. The value of the unary potential $\theta^1_{a;i}$ is shown next to the i^{th} branch of the trellis on top of v_a. For example, $\theta^1_{a;0} = 10$ (shown next to the lower branch of the trellis on top of v_a) and $\theta^1_{b;1} = 3$ (shown next to the upper branch of the trellis on top of v_b). The pairwise potential $\theta^2_{ab;ij}$ is shown next to the connection between the i^{th} and j^{th} branches of the trellises on top of v_a and v_b respectively. For example, $\theta^2_{ab;00} = -10$ (shown next to the bottom-most connection between the two trellises) and $\theta^2_{ab;01} = -5$ (shown next to the diagonal connection between the two trellises). **(b)** The optimal solution obtained using the LP-S relaxation. The value of $x_{a;i}$ is shown next to the i^{th} branch of the trellis on top of v_a. Similarly, the value of $X_{ab;ij}$ is shown next to the connection between the i^{th} and j^{th} branches of the trellises on top of v_a and v_b respectively. Note that the value of the objective function for the optimal solution is 6. **(c)** A feasible solution of the SOCP-MS relaxation which does not belong to the feasibility region of LP-S and has an objective function value of 2. It follows that the optimal solution of SOCP-MS would lie in $\mathcal{F}(\text{SOCP-MS}) - \mathcal{F}(\text{LP-S})$ and have a value of at most 2. Together with Lemmas 3.1 and 3.2, this proves that LP-S strictly dominates SOCP-MS.

$$\hat{\theta}^2_{ab;ij} = \begin{cases} \sum_{v_c \in \mathbf{v}} \sum_{l_k \in \mathbf{l}} |\theta^2_{ac;ik}|, & \text{if} \quad a = b, i = j, \\ \theta^2_{ab;ij} & \text{otherwise}. \end{cases} \tag{29}$$

Here, the terms $\theta^1_{a;i}$ and $\theta^2_{ac;ik}$ are the (original) unary potentials and pairwise potentials for the given MRF. Consider a feasible solution $\mathbf{x}$ of the QP-RL and the SOCP-MS relaxations. Further, let $\mathbf{X}$ be the solution obtained when minimizing the objective function of the SOCP-MS relaxation whilst keeping $\mathbf{x}$ fixed. We prove that the value of the objective functions for both relaxations at the above feasible solution is the same by equating the coefficient of $\theta^1_{a;i}$ and $\theta^2_{ab;ij}$ for all $v_a \in \mathbf{v}$, $(a,b) \in \mathcal{E}$ and $l_i, l_j \in \mathbf{l}$ in both formulations. Using Equation (28), we see that $\theta^1_{a;i}$ is multiplied by $\frac{1+x_{a;i}}{2}$ in the objective function of the QP-RL. Similarly, $\theta^1_{a;i}$ is multiplied by $\frac{1+x_{a;i}}{2}$ in the SOCP-MS relaxation. Therefore the coefficients of $\theta^1_{a;i}$ in both relaxations are equal for all $v_a \in \mathbf{v}$ and $l_i \in \mathbf{l}$.

We now consider the pairwise potentials, that is, $\theta^2_{ab;ij}$ and show that their coefficients are the same when obtaining the minimum of the objective function. We consider the following two cases.

Case I: Let $\theta^2_{ab;ij} = \theta^2_{ba;ji} \geq 0$. Using Equation (29) we see that, in the QP-RL relaxation, $\theta^2_{ab;ij} + \theta^2_{ba;ji}$ is multiplied by the following term:

$$\left(\frac{1+x_{a;i}}{2}\right)^2 + \left(\frac{1+x_{b;j}}{2}\right)^2 + 2\left(\frac{1+x_{a;i}}{2}\right)\left(\frac{1+x_{b;j}}{2}\right) - \frac{1+x_{a;i}}{2} - \frac{1+x_{b;j}}{2}. \tag{30}$$

In the case of SOCP-MS relaxation, since $\theta^2_{ab;ij} \geq 0$, the minimum of the objective function is obtained by using the minimum value that $X_{ab;ij}$ would take given the SOC constraints. Since $\mathbf{X}$ is symmetric, we see that $\theta^2_{ab;ij} + \theta^2_{ba;ji}$ is multiplied by the following term:

$$\begin{aligned} & \frac{1+x_{a;i}+x_{b;j}+\inf\{X_{ab;ij}\}}{2} \\ = & \frac{1+x_{a;i}+x_{b;j}+(x_{a;i}+x_{b;j})^2/2-1}{2}, \end{aligned} \tag{31}$$

where the infimum of $X_{ab;ij}$ is defined by constraint (21) in the SOCP-MS relaxation. It can easily be verified that the terms (30) and (31) are equal.

Case II: Now consider the case where $\theta^2_{ab;ij} = \theta^2_{ba;ji} < 0$. In the QP-RL relaxation, the term $\theta^2_{ab;ij} + \theta^2_{ba;ji}$ is multiplied by

$$\frac{1+x_{a;i}}{2} + \frac{1+x_{b;j}}{2} + 2\left(\frac{1+x_{a;i}}{2}\right)\left(\frac{1+x_{b;j}}{2}\right) - \left(\frac{1+x_{a;i}}{2}\right)^2 - \left(\frac{1+x_{b;j}}{2}\right)^2. \tag{32}$$

In order to obtain the minimum of the objective function, the SOCP-MS relaxation uses the maximum value that $X_{ab;ij}$ would take given the SOC constraints. Thus, $\theta^2_{ab;ij} + \theta^2_{ba;ji}$ is multiplied by

$$\begin{aligned} & \frac{1+x_{a;i}+x_{b;j}+\sup\{X_{ab;ij}\}}{2} \\ = & \frac{1+x_{a;i}+x_{b;j}+1-(x_{a;i}-x_{b;j})^2/2}{2}, \end{aligned} \tag{33}$$

where the supremum of $X_{ab;ij}$ is defined by constraint (20). Again, the terms (32) and (33) can be shown to be equivalent. ∎

Theorems 1 and 2 prove that the LP-S relaxation strictly dominates the QP-RL and SOCP-MS relaxations. A natural question that now arises is whether the additive bound of QP-RL (proved by Ravikumar and Lafferty 2006) is applicable to the LP-S and SOCP-MS relaxations. Our next Theorem answers this question in an affirmative. To this end, we use the rounding scheme proposed by Ravikumar and Lafferty (2006) to obtain the labelling f for all the random variables of the given MRF. This rounding scheme is summarized below:

- Pick a variable v_a which has not been assigned a label.
- Assign the label l_i to v_a (i.e., f(a) = i) with probability $\frac{1+x_{a;i}}{2}$.
- Continue till all variables have been assigned a label.

Recall that $\sum_{i=0}^{h-1} \frac{1+x_{a;i}}{2} = 1$ for all $v_a \in \mathbf{v}$. Hence, once v_a is picked it is guaranteed to be assigned a label. In other words the above rounding scheme terminates after $n = |\mathbf{v}|$ steps. To the best of our knowledge, this additive bound was previously known only for the QP-RL relaxation (Ravikumar and Lafferty, 2006).

Theorem 3: For the above rounding scheme, LP-S and SOCP-MS provide the same additive bound as the QP-RL relaxation of Ravikumar and Lafferty (2006), that is, $\frac{S}{4}$ where $S = \sum_{(a,b)\in\mathcal{E}} \sum_{l_i,l_j\in\mathbf{l}} |\theta^2_{ab;ij}|$ (i.e., the sum of the absolute values of all pairwise potentials). Furthermore, this bound is tight.

Proof: The QP-RL and SOCP-MS relaxations are equivalent. Thus the above Theorem holds true for SOCP-MS. We now consider the LP-S relaxation (Chekuri et al., 2001; Koster et al., 1998; Schlesinger, 1976; Wainwright et al., 2005). We denote the energy of the optimal labelling as e^*. Recall that e^L denotes the optimal value of the LP-S which is obtained using possibly fractional variables $(\mathbf{x}^*, \mathbf{X}^*)$. Clearly, $e^L \leq e^*$. The energy of the labelling $\hat{\mathbf{x}}$, obtained after rounding the solution of the LP-S relaxation, is represented by the term $\hat{e}^L$,

Using the above notation, we now show that the LP-S relaxation provides an additive bound of $\frac{S}{4}$ for the above rounding scheme. We first consider the unary potentials and observe that

$$E\left(\theta^1_{a;i}\left(\frac{1+\hat{x}_{a;i}}{2}\right)\right) = \theta^1_{a;i}\left(\frac{1+x^*_{a;i}}{2}\right),$$

where $E(\cdot)$ denotes the expectation of its argument under the above rounding scheme. Similarly, for the pairwise potentials,

$$E\left(\theta^2_{ab;ij}\left(\frac{1+\hat{x}_{a;i}}{2}\right)\left(\frac{1+\hat{x}_{b;j}}{2}\right)\right) = \theta^2_{ab;ij}\left(\frac{1+x^*_{a;i}+x^*_{b;ij}+x^*_{a;i}x^*_{b;j}}{4}\right).$$

We analyze the following two cases:

(i) $\theta^2_{ab;ij} \geq 0$: Using the fact that $X^*_{ab;ij} \geq |x^*_{a;i}+x^*_{b;j}| - 1$ (see Lemma 3.2), we get

$$\begin{array}{cc} & 1+x^*_{a;i}+x^*_{b;j}+x^*_{a;i}x^*_{b;j} - (1+x^*_{a;i}+x^*_{b;j}+X^*_{ab;ij}) \\ = & x^*_{a;i}x^*_{b;j} - X^*_{ab;ij} \\ \leq & x^*_{a;i}x^*_{b;j} + 1 - |x^*_{a;i}+x^*_{b;j}| \\ \leq & 1, \end{array}$$

where the equality holds when $x^*_{a;i} = x^*_{b;j} = 0$. Therefore,

$$E\left(\theta^2_{ab;ij}\left(\frac{1+\hat{x}_{a;i}}{2}\right)\left(\frac{1+\hat{x}_{b;j}}{2}\right)\right) \leq \theta^2_{ab;ij}\frac{(1+x^*_{a;i}+x^*_{b;ij}+X^*_{ab;ij})}{4} + \frac{|\theta^2_{ab;ij}|}{4}.$$

(ii) $\theta^2_{ab;ij} < 0$: Using the fact that $X^*_{ab;ij} \leq 1 - |x^*_{a;i} - x^*_{b;j}|$ (see Lemma 3.1), we get

$$\begin{array}{cc} & 1+x^*_{a;i}+x^*_{b;j}+x^*_{a;i}x^*_{b;j} - (1+x^*_{a;i}+x^*_{b;j}+X^*_{ab;ij}) \\ \geq & x^*_{a;i}x^*_{b;j} - 1 + |x^*_{a;i}-x^*_{b;j}| \\ \geq & -1, \end{array}$$

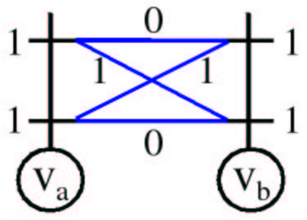
where the equality holds when $x^*_{a;i} = x^*_{b;j} = 0$. Therefore,

$$E\left(\theta^2_{ab;ij}\left(\frac{1+\hat{x}_{a;i}}{2}\right)\left(\frac{1+\hat{x}_{b;j}}{2}\right)\right) \leq \theta^2_{ab;ij}\frac{(1+x^*_{a;i}+x^*_{b;ij}+X^*_{ab;ij})}{4} + \frac{|\theta^2_{ab;ij}|}{4}.$$

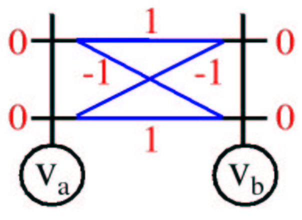
Summing the expectation of the unary and pairwise potentials for all $v_a \in \mathbf{v}$, $(a,b) \in \mathcal{E}$ and $l_i, l_j \in \mathbf{l}$, we get

$$e^* \leq E(\hat{e}^L) \leq e^L + \frac{S}{4} \leq e^* + \frac{S}{4},$$

which proves the additive bound for LP-S.



(a)



(b)

Figure 2: An example MRF for proving the tightness of the LP-S additive bound of $\frac{S}{4}$. **(a)** The MRF consisting of two random variables v_a and v_b. The potentials are shown similar to Fig. 1. Note that the unary potentials are uniform while the pairwise potentials form an Ising model. **(b)** An optimal solution of the LP-S relaxation for the MRF shown in (a). The values of the variables $x_{a;i}$ are shown next to the i^{th} branch of the trellis of v_a. Note that all variables $x_{a;i}$ have been assigned to 0. The values of the variables $X_{ab;ij}$ are shown next to the connection between the i^{th} and j^{th} branch of the trellises of v_a and v_b. Note that $X_{ab;ij} = -1$ if $\theta^2_{ab;ij} > 0$ and $X_{ab;ij} = 1$ otherwise.

This additive bound can indeed be shown to be tight by using the following simple example. Consider an instance of the MAP estimation problem for an MRF defined on two variables $\mathbf{v} = \{v_a, v_b\}$ each of which can take one of two labels from the set $\mathbf{l} = \{l_0, l_1\}$. Let the unary and pairwise potentials be as defined in Fig. 2(a), that is, the unary potentials are uniform and the pairwise potentials follow the Ising model. Note that this MRF is a chain of size 2 and can thus, be solved exactly using the LP-S relaxation (Chekuri et al., 2001; Wainwright and Jordan, 2003) (i.e., the optimal value of the objective function of the LP-S relaxation is exactly equal to the energy of the MAP estimate). However, our aim is to show that for a particular randomized rounding scheme the MRF in Fig. 2(a) offers an example of tightness for the additive bound.

An optimal solution of the LP-S relaxation is given in Fig. 2(b). Clearly, $e^* = 2$ (e.g., for the labelling $f = \{0,0\}$ or $f = \{1,1\}$) while $E(\hat{e}^L) = 2 + \frac{2}{4} = e^* + \frac{S}{4}$. Thus the additive bounds obtained for the LP-S, QP-RL and SOCP-MS algorithms are the same. In fact, one can construct arbitrarily large MRFs (i.e., MRF defined over a large number of variables) with uniform unary potentials and Ising model pairwise potentials for which the bound can be shown to be tight. ■

The above bound was proved for the case of binary variables (i.e., h = 2) by Hammer and Kalantari (1987) using a slightly different rounding scheme. Our result can be viewed as a generalization of this for any arbitrary number of labels. The above Theorem proves a tight additive bound using a simple rounding scheme. However, using this rounding scheme in practical applications may not be desirable as it generates an independent random number for rounding the fractional labelling of each random variable. We note here that better bounds can be obtained for some special cases of the MAP estimation problem using the LP-S relaxation together with more clever rounding schemes (such as those described in Chekuri et al. 2001 and Kleinberg and Tardos 1999).

5. QP and SOCP Relaxations over Trees and Cycles

We now generalize the results of Theorem 1 by defining a large class of SOCP relaxations which is dominated by LP-S. Specifically, we consider the SOCP relaxations which relax the non-convex constraint $\mathbf{X} = \mathbf{x}\mathbf{x}^\top$ using a set of second order cone (SOC) constraints of the form

$$||(\mathbf{U}^k)^\top \mathbf{x}|| \leq \mathbf{C}^k \bullet \mathbf{X}, k = 1, \cdots, n_C \tag{34}$$

where $\mathbf{C}^k = \mathbf{U}^k(\mathbf{U}^k)^\top \succeq 0$, for all $k = 1, \cdots, n_C$. In order to make the proofs of the subsequent Theorems easier, we make two assumptions. However, the Theorems would hold true even without these assumptions as discussed below.

Assumption 1: We assume that the integer constraints

$$\mathbf{x} \in \{-1, +1\}^{nh}, \mathbf{X} \in \{-1, +1\}^{nh \times nh}, \tag{35}$$

are relaxed to

$$\mathbf{x} \in [-1, +1]^{nh}, \mathbf{X} \in [-1, +1]^{nh \times nh}, \tag{36}$$

with $X_{aa;ii} = 1$, for all $v_a \in \mathbf{v}, l_i \in \mathbf{l}$. The constraints (36) provide the convex hull for all the points defined by the integer constraints (35). Recall that the convex hull of a set of points is the smallest convex set which contains all the points. We now discuss how the above assumption is not restrictive with respect to the results that follow. Let A be a relaxation which contains constraints (36). Using A it is possible to obtain another relaxation B by substituting constraints (36) by some other relaxation of the integer constraints. By the definition of convex hull, it would follow that $\mathcal{F}(\text{A}) \subset \mathcal{F}(\text{B})$. In other words A dominates B (see Case I in Section 2). Hence, if A is dominated by the LP-S relaxation, then LP-S would also dominate B.

Assumption 2: We assume that the set of constraints (34) contains all the constraints specified in the SOCP-MS relaxation. Recall that for a given pair of neighbouring random variables, that is, $(a,b) \in \mathcal{E}$, and a pair of labels $l_i, l_j \in \mathbf{l}$, SOCP-MS specifies SOC constraints using two matrices (say $\mathbf{C}^1$ and $\mathbf{C}^2$) which are 0 everywhere except for the following 2×2 submatrices:

$$\begin{pmatrix} C^1_{aa;ii} & C^1_{ab;ij} \\ C^1_{ba;ji} & C^1_{bb;jj} \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}, \begin{pmatrix} C^2_{aa;ii} & C^2_{ab;ij} \\ C^2_{ba;ji} & C^2_{bb;jj} \end{pmatrix} = \begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix}. \tag{37}$$

In the case where a given relaxation A does not contain the SOCP-MS constraints, we can define a new relaxation B. This new relaxation B is obtained by adding all the SOCP-MS constraints to A. By definition, B dominates A (although not necessarily strictly, see Case II in Section 2). Hence, if B is dominated by the LP-S relaxation then it follows that LP-S would also dominate A. Hence, our assumption about including the SOCP-MS constraints is not restrictive for the results presented in this section.

Note that each SOCP relaxation belonging to this class would not include the marginalization constraints. Hence, it would define an equivalent QP relaxation (similar to the equivalent QP-RL relaxation defined by the SOCP-MS relaxation). By definition of equivalent relaxations, all these QP relaxations will also be dominated by LP-S. Before we begin to describe our results in detail, we need to set up some notation as follows.

5.1 Notation

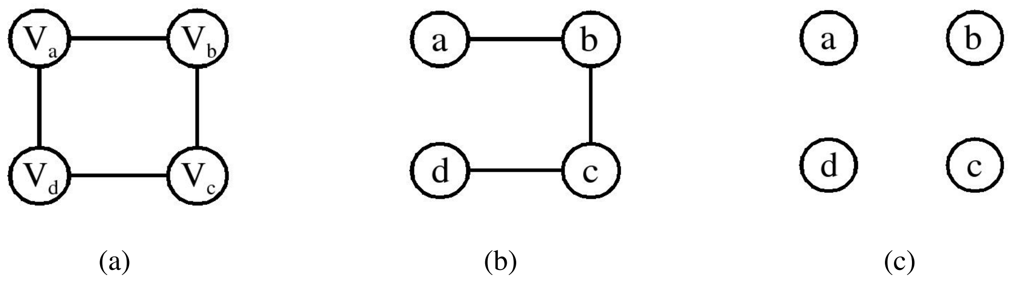


Figure 3: **(a)** An example MRF defined over four variables which form a cycle. **(b)** The set E^k specified by the matrix $\mathbf{C}^k$ shown in Equation (39), that is, $E^k = \{(a,b),(b,c),(c,d)\}$. **(c)** The set $V^k = \{a,b,c,d\}$. See text for definitions of these sets.

We consider an SOC constraint which is of the form described in Equation (34), that is,

$$||(\mathbf{U}^k)^\top \mathbf{x}|| \leq \mathbf{C}^k \bullet \mathbf{X}, \tag{38}$$

where $k \in \{1,\cdots,n_C\}$. In order to help the reader understand the notation better, we use an example MRF shown in Fig. 3(a). This MRF is defined over four variables $\mathbf{v} = \{v_a, v_b, v_c, v_d\}$ (connected to form a cycle of length 4), each of which take a label from the set $\mathbf{l} = \{l_0, l_1\}$. For this MRF we specify a constraint using a matrix $\mathbf{C}^k \succeq 0$ which is 0 everywhere, except for the following 4×4 submatrix:

$$\begin{pmatrix} C^k_{aa;00} & C^k_{ab;00} & C^k_{ac;00} & C^k_{ad;00} \\ C^k_{ba;00} & C^k_{bb;00} & C^k_{bc;00} & C^k_{bd;00} \\ C^k_{ca;00} & C^k_{cb;00} & C^k_{cc;00} & C^k_{cd;00} \\ C^k_{da;00} & C^k_{db;00} & C^k_{dc;00} & C^k_{dd;00} \end{pmatrix} = \begin{pmatrix} 2 & 1 & 1 & 0 \\ 1 & 2 & 1 & 1 \\ 1 & 1 & 2 & 1 \\ 0 & 1 & 1 & 2 \end{pmatrix}. \tag{39}$$

Using the SOC constraint shown in Equation (38) we define the following sets:

- The set E^k is defined such that $(a,b) \in E^k$ if, and only if, it satisfies the following conditions:

$$(a,b) \in \mathcal{E},$$
$$\exists l_i, l_j \in \mathbf{l} \text{ such that } C^k_{ab;ij} \neq 0.$$

  Recall that $\mathcal{E}$ specifies the neighbourhood relationship for the given MRF. In other words E^k is the subset of the edges in the graphical model of the MRF such that $\mathbf{C}^k$ specifies constraints for the random variables corresponding to those edges. For the example MRF (shown in Fig. 3(a)) and $\mathbf{C}^k$ matrix (in Equation 39), the set E^k obtained is shown in Fig. 3(b).

- The set V^k is defined as $a \in V^k$ if, and only if, there exists a $v_b \in \mathbf{v}$ such that $(a,b) \in E^k$. In other words V^k is the subset of hidden nodes in the graphical model of the MRF such that $\mathbf{C}^k$ specifies constraints for the random variables corresponding to those hidden nodes. Fig. 3(c) shows the set V^k for our example SOC constraint.

- The set $\mathcal{T}^k$ consists of elements $a;i \in \mathcal{T}^k$ which satisfy

$$a \in V^k, l_i \in \mathbf{l},$$
$$\exists b \in V^k, l_j \in \mathbf{l}, \text{ such that } C^k_{ab;ij} \neq 0.$$

In other words the set $\mathcal{T}^k$ consists of the set of indices for the vector $\mathbf{x}$ which are constrained by inequality (38), that is, the coefficient of $x_{a;i}$ where $a;i \in \mathcal{T}^k$ are non-zero in the LHS of inequality (38). Note that in Equation (39) the constraint is specified using only the label l_0 for all the random variables $\mathbf{v}$. Thus the set $\mathcal{T}^k$ is given by

$$\mathcal{T}^k = \{(a;0),(b;0),(c;0),(d;0)\}.$$

For each set $\mathcal{T}^k$ we define three disjoint subsets of $\mathcal{T}^k \times \mathcal{T}^k$ as follows.

- The set $\mathcal{T}^k_0$ is defined as

$$\mathcal{T}^k_0 = \{(a;i,b;j)|(a;i,b;j) \in \mathcal{T}^k \times \mathcal{T}^k, (a,b) \in \mathcal{E}, (a,b) \notin E^k\}.$$

Note that by definition $C^k_{ab;ij} = 0$ if $(a;i,b;j) \in \mathcal{T}^k_0$. Thus $\mathcal{T}^k_0$ indexes the elements of matrix $\mathbf{X}$ which are not constrained by inequality (38) but are present in the set $\mathcal{T}^k \times \mathcal{T}^k$. For the matrix $\mathbf{C}^k$ in Equation (39), the set $\mathcal{T}^k_0$ is given by

$$\mathcal{T}^k_0 = \{(a;0,d;0)\}.$$

- The set $\mathcal{T}^k_1$ is defined as

$$\mathcal{T}^k_1 = \{(a;i,b;j)|(a;i,b;j) \in \mathcal{T}^k \times \mathcal{T}^k, (a,b) \notin \mathcal{E}\}.$$

In other words the set $\mathcal{T}^k_1$ indexes the elements of matrix $\mathbf{X}$ which are constrained by inequality (38) but do not belong to any pair of neighbouring random variables. Note that the variables $X_{ab;ij}$ such that $(a;i,b;j) \in \mathcal{T}^k_1$ were not present in the LP-S relaxation. For the matrix $\mathbf{C}^k$ in Equation (39), the set $\mathcal{T}^k_1$ is given by

$$\mathcal{T}^k_1 = \{(a;0,c;0),(b;0,d;0)\}.$$

- The set $\mathcal{T}^k_2$ is defined as

$$\mathcal{T}^k_2 = \{(a;i,b;j)|(a;i,b;j) \in \mathcal{T}^k \times \mathcal{T}^k, (a,b) \in E^k\}.$$

In other words the set $\mathcal{T}^k_2$ indexes the elements of matrix $\mathbf{X}$ which are constrained by inequality (38) and belong to a pair of neighbouring random variables. For the matrix $\mathbf{C}^k$ in Equation (39), the set $\mathcal{T}^k_2$ is given by

$$\mathcal{T}^k_2 = \{(a;0,b;0),(b;0,c;0)(c;0,d;0)\}.$$

Note that $\mathcal{T}^k_0 \bigcup \mathcal{T}^k_1 \bigcup \mathcal{T}^k_2 = \mathcal{T}^k \times \mathcal{T}^k$. For a given set of pairwise potentials $\theta^2_{ab;ij}$ we define two disjoint sets of $\mathcal{T}^k_2$ as follows.

- The set $\mathcal{T}^k_{2+}$ corresponds to non-negative pairwise potentials, that is,

  $$\mathcal{T}^k_{2+} = \{(a;i,b;j) | (a;i,b;j) \in \mathcal{T}^k_2, \theta^2_{ab;ij} \geq 0\},$$

  Thus the set $\mathcal{T}^k_{2+}$ indexes the elements of matrix $\mathbf{X}$ which belong to $\mathcal{T}^k_2$ and are multiplied by a non-negative pairwise potential in the objective function of the relaxation.

- The set $\mathcal{T}^k_{2-}$ corresponds to negative pairwise potentials, that is,

  $$\mathcal{T}^k_{2-} = \{(a;i,b;j) | (a;i,b;j) \in \mathcal{T}^k_2, \theta^2_{ab;ij} < 0\},$$

  Thus the set $\mathcal{T}^k_{2-}$ indexes the elements of matrix $\mathbf{X}$ which belong to $\mathcal{T}^k_2$ and are multiplied by a negative pairwise potential in the objective function of the relaxation. Note that $\mathcal{T}^k_2 = \mathcal{T}^k_{2+} \bigcup \mathcal{T}^k_{2-}$. For the purpose of illustration let us assume that, for the example MRF in Fig. 3(a), $\theta^2_{ab;00} \geq 0$ while $\theta^2_{bc;00} < 0$ and $\theta^2_{cd;00} < 0$. Then,

  $$\mathcal{T}^k_{2+} = \{(a;0,b;0)\},$$
  $$\mathcal{T}^k_{2-} = \{(b;0,c;0),(c;0,d;0)\},$$

We also define a weighted graph $G^k = (V^k, E^k)$ whose vertices are specified by the set V^k and whose edges are specified by the set E^k. The weight of an edge $(a,b) \in E^k$ is given by $w(a,b)$. Recall that $w(a,b)$ specifies the strength of the pairwise relationship between two neighbouring variables v_a and v_b. Thus, for our example SOC constraint, the vertices of this graph are given in Fig. 3(c) while the edges are shown in Fig. 3(b). This graph can be viewed as a subgraph of the graphical model representation for the given MRF.

Further, we define the submatrices $\mathbf{x}^k_T$ and $\mathbf{X}^k_T$ of $\mathbf{x}$ and $\mathbf{X}$ respectively such that

$$\mathbf{x}^k_T = \{x_{a;i} | a;i \in \mathcal{T}^k\},$$
$$\mathbf{X}^k_T = \{X_{ab;ij} | (a;i,b;j) \in \mathcal{T}^k \times \mathcal{T}^k\}.$$

For our example, these submatrices will be given by

$$\mathbf{x}^k_T = \begin{pmatrix} x_{a;0} \\ x_{b;0} \\ x_{c;0} \\ x_{d;0} \end{pmatrix}, \mathbf{X}^k_T = \begin{pmatrix} X_{aa;00} & X_{ab;00} & X_{ac;00} & X_{ad;00} \\ X_{ba;00} & X_{bb;00} & X_{bc;00} & X_{bd;00} \\ X_{ca;00} & X_{cb;00} & X_{cc;00} & X_{cd;00} \\ X_{da;00} & X_{db;00} & X_{dc;00} & X_{dd;00} \end{pmatrix}.$$

Using the above notation, we are now ready to describe our results in detail.

5.2 QP and SOCP Relaxations over Trees

We begin by considering those relaxations where the SOC constraints are defined such that the graphs $G^k = (V^k, E^k)$ form trees. For example, the graph G^k defined by the SOC constraint in Equation (39) forms a tree as shown in Fig. 3(b). We denote such a relaxation, which specifies SOC constraints only over trees, by SOCP-T. Note that SOCP-MS (and hence, QP-RL) can be considered a special case of this class of relaxations where the number of vertices in each tree is equal to two (since the constraints are defined for all $(a,b) \in \mathcal{E}$).

We will remove this restriction by allowing the number of vertices in each tree to be arbitrarily large (i.e., between 1 and n). We consider one such tree $G = (V,E)$. Note that for a given relaxation SOCP-T, there may be several SOC constraints defined using this tree G (or its subtrees). Without loss of generality, we assume that the constraints

$$||(\mathbf{U}^k)^\top \mathbf{x}|| \leq \mathbf{C}^k \bullet \mathbf{X}, k = 1, \cdots, n'_C,$$

are defined on the tree G. In other words,

$$G^k \subseteq G, k = 1, \cdots, n'_C,$$

where $G^k \subseteq G$ implies that G^k is a subtree of G. In order to make the notation less cluttered, **we will drop the superscript *k* from the sets defined in the previous section** (since we will consider only one tree G in our analysis).

We will now show that SOCP-T is dominated by the LP-S relaxation. This result is independent of the choice of the tree G and matrices $\mathbf{C}^k$. To this end, we define the term $e_1(\mathbf{x}_T)$ for a given value of $\mathbf{x}_T$ as

$$e_1(\mathbf{x}_T) = \sum_{(a;i)\in\mathcal{T}} \left(\theta^1_{a;i} + \sum_{(b;j)\in\mathcal{T}} \frac{\theta^2_{ab;ij}}{2} \right) x_{a;i}.$$

Further, for a fixed $\mathbf{x}_T$ we also define the following two terms:

$$\begin{aligned} e^S_2(\mathbf{x}_T) &= \min_{(\mathbf{x}_T,\mathbf{X}_T)\in\mathcal{F}(\text{SOCP-T})} \sum_{(a;i,b;j)\in\mathcal{T}_2} \theta^2_{ab;ij} X_{ab;ij}, \\ e^L_2(\mathbf{x}_T) &= \min_{(\mathbf{x}_T,\mathbf{X}_T)\in\mathcal{F}(\text{LP-S})} \sum_{(a;i,b;j)\in\mathcal{T}_2} \theta^2_{ab;ij} X_{ab;ij}, \end{aligned}$$

where $\mathcal{F}$(SOCP-T) and $\mathcal{F}$(LP-S) are the feasibility regions of SOCP-T and LP-S respectively. We use the notation $(\mathbf{x}_T,\mathbf{X}_T) \in \mathcal{F}$(SOCP-T) loosely to mean that we can obtain a feasible solution $(\mathbf{x},\mathbf{X})$ of SOCP-T such that the values of the variables $x_{a;i}$ where $a;i \in \mathcal{T}$ and $X_{ab;ij}$ where $(a;i,b;j) \in \mathcal{T}\times\mathcal{T}$ are equal to the values specified by $\mathbf{x}_T$ and $\mathbf{X}_T$. The notation $(\mathbf{x}_T,\mathbf{X}_T) \in \mathcal{F}$(LP-S) is used similarly. Note that for a given $\mathbf{x}_T$ the possible values of $\mathbf{X}_T$ are constrained such that $(\mathbf{x}_T,\mathbf{X}_T) \in \mathcal{F}$(SOCP-T) and $(\mathbf{x}_T,\mathbf{X}_T) \in \mathcal{F}$(LP-S) (in the case of SOCP-T and LP-S respectively). Hence different values of $\mathbf{x}_T$ will provide different values of $e^S_2(\mathbf{x}_T)$ and $e^L_2(\mathbf{x}_T)$.

The contribution of the tree G to the objective function of SOCP-T and LP-S is given by

$$\begin{aligned} e^S &= \min_{\mathbf{x}_T} \frac{e_1(\mathbf{x}_T)}{2} + \frac{e^S_2(\mathbf{x}_T)}{4}, \\ e^L &= \min_{\mathbf{x}_T} \frac{e_1(\mathbf{x}_T)}{2} + \frac{e^L_2(\mathbf{x}_T)}{4}, \end{aligned}$$

respectively. Assuming that the trees G do not overlap, the total value of the objective function would simply be the sum of e^S (for SOCP-T) or e^L (for LP-S) over all trees G. However, since we use an arbitrary parameter θ in our analysis, it follows that the results do not depend on this assumption of non-overlapping trees. In other words if two trees G^1 and G^2 share an edge $(a,b) \in \mathcal{E}$ then we can simply consider two MAP estimation problems defined using arbitrary parameters θ_1 and θ_2 such that $\theta_1 + \theta_2 = \theta$. We can then add the contribution of G_1 for the MAP estimation problem with parameter θ_1 to the contribution of G_2 for the MAP estimation problem with parameter θ_2. This would then provide us with the total contribution of G_1 and G_2 for the original MAP estimation defined using parameter θ.

Using the above argument it follows that if, for all G and for all θ, the following holds true:

$$\begin{aligned} & \frac{e_1(\mathbf{x}_T)}{2} + \frac{e_2^S(\mathbf{x}_T)}{4} \leq \frac{e_1(\mathbf{x}_T)}{2} + \frac{e_2^L(\mathbf{x}_T)}{4}, \forall \mathbf{x}_T \in [-1,1]^{|\mathcal{T}|} \\ \Rightarrow \quad & e_2^S(\mathbf{x}_T) \leq e_2^L(\mathbf{x}_T), \forall \mathbf{x}_T \in [-1,1]^{|\mathcal{T}|}, \end{aligned} \tag{40}$$

then LP-S dominates SOCP-T (since this would imply that $e^S \leq e^L$, for all G and for all θ). This is the condition that we will use to prove that LP-S dominates all SOCP relaxations whose constraints are defined over trees. To this end, we define a vector $\omega = \{\omega_k, k = 1, \cdots, n'_C\}$ of non-negative real numbers such that

$$\sum_k \omega_k C^k_{ab;ij} = \theta^2_{ab;ij}, \forall (a;i,b;j) \in \mathcal{T}_2.$$

Due to the presence of the matrices $\mathbf{C}^k$ defined in Equation (37) (which result in the SOCP-MS constraints for all $(a,b) \in \mathcal{E}$ and $l_i, l_j \in \mathbf{l}$), such a vector ω would always exist for any MRF parameter θ. We denote the matrix $\sum_k \omega_k \mathbf{C}^k$ by $\mathbf{C}$. Clearly, $\mathbf{C} \succeq 0$, and hence can be written as $\mathbf{C} = \mathbf{U}\mathbf{U}^\top$.

Using the constraints $||(\mathbf{U}^k)^\top \mathbf{x}||^2 \leq \mathbf{C}^k \bullet \mathbf{X}_T$ together with the fact that $\omega_k \geq 0$, we get the following inequality:[6]

$$\begin{aligned} & \textstyle\sum_k \omega_k ||(\mathbf{U}^k)^\top \mathbf{x}||^2 \leq \sum_k \omega_k \mathbf{C}^k \bullet \mathbf{X}, \\ \Rightarrow \quad & ||\mathbf{U}^\top \mathbf{x}||^2 \leq \mathbf{C} \bullet \mathbf{X}, \\ \Rightarrow \quad & ||\mathbf{U}^\top \mathbf{x}||^2 \leq \textstyle\sum_{a;i \in \mathcal{T}} C_{aa;ii} X_{aa;ii} + \sum_{(a;i,b;j) \in \mathcal{T}_1} C_{ab;ij} X_{ab;ij} + \sum_{(a;i,b;j) \in \mathcal{T}_2} C_{ab;ij} X_{ab;ij}, \\ \Rightarrow \quad & ||\mathbf{U}^\top \mathbf{x}||^2 - \textstyle\sum_{a;i \in \mathcal{T}} C_{aa;ii} - \sum_{(a;i,b;j) \in \mathcal{T}_1} C_{ab;ij} X_{ab;ij} \leq \sum_{(a;i,b;j) \in \mathcal{T}_2} \theta^2_{ab;ij} X_{ab;ij}, \end{aligned} \tag{41}$$

where the last expression is obtained using the fact that $C_{ab;ij} = \theta^2_{ab;ij}$ for all $(a;i,b;j) \in \mathcal{T}_2$ and $X_{aa;ii} = 1$ for all $v_a \in \mathbf{v}$ and $l_i \in \mathbf{l}$. Note that the above inequality provides a lower bound of $e_2^S(\mathbf{x}_T)$. Another lower bound of $e_2^S(\mathbf{x}_T)$ is provided by the constraints that $-1 \leq X_{ab;ij} \leq 1$ for all $(a;i,b;j) \in \mathcal{T}_2$. Since the objective function being minimized contains $e_2^S(\mathbf{x}_T)$, it follows that in the absence of any other constraints (which is our assumption) $e_2^S(\mathbf{x}_T)$ would be equal to the maximum of the two lower bounds, that is,

$$\max \left\{ \sum_{(a;i,b;j) \in \mathcal{T}_2} -|\theta^2_{ab;ij} X_{ab;ij}|, ||\mathbf{U}^\top \mathbf{x}||^2 - \sum_{a;i \in \mathcal{T}} C_{aa;ii} - \sum_{(a;i,b;j) \in \mathcal{T}_1} C_{ab;ij} X_{ab;ij} \right\}.$$

The first expression is obtained by substituting the following values for $X_{ab;ij}$ where $(a;i,b;j) \in \mathcal{T}_2$:

$$X_{ab;ij} = \begin{cases} -1 & \text{if} \quad \theta^2_{ab;ij} \geq 0, \\ 1 & \text{otherwise}, \end{cases}$$

while the second expression is the LHS of inequality (41). Clearly, if $e_2^S(\mathbf{x}_T)$ is equal to the first expression then $e_2^L(\mathbf{x}_T) \geq e_2^S(\mathbf{x}_T)$, that is, LP-S dominates SOCP-T. In what follows, we will only

6. Note that there are no terms corresponding to $(a;i,b;j) \in \mathcal{T}_0$ in inequality (41) since $C_{ab;ij} = 0$ if $(a;i,b;j) \in \mathcal{T}_0$. In other words, $X_{ab;ij}$ vanishes from the above inequality if $(a;i,b;j) \in \mathcal{T}_0$.

consider the non-trivial case when

$$\begin{aligned} e_2^S(\mathbf{x}_T) &= \min \sum_{(a;i,b;j)\in\mathcal{T}_2} \theta^2_{ab;ij} X_{ab;ij}, \\ &= \|\mathbf{U}^\top \mathbf{x}\|^2 - \sum_{a;i\in\mathcal{T}} C_{aa;ii} - \sum_{(a;i,b;j)\in\mathcal{T}_1} C_{ab;ij} X_{ab;ij}. \end{aligned} \tag{42}$$

For the LP-S relaxation, from Lemmas 3.1 and 3.2, we obtain the following value of $e_2^L(\mathbf{x}_T)$:

$$\begin{aligned} & |x_{a;i}+x_{b;j}|-1 \leq X_{ab;ij} \leq 1-|x_{a;i}-x_{b;j}|, \\ \Rightarrow \quad & e_2^L(\mathbf{x}_T) = \min \textstyle\sum_{(a;i,b;j)\in\mathcal{T}_2} \theta^2_{ab;ij} X_{ab;ij}, \\ & = \textstyle\sum_{(a;i,b;j)\in\mathcal{T}_{2+}} \theta^2_{ab;ij}(|x_{a;i}+x_{b;j}|) - \sum_{(a;i,b;j)\in\mathcal{T}_{2-}} \theta^2_{ab;ij}(|x_{a;i}-x_{b;j}|) - \\ & \textstyle\sum_{(a;i,b;j)\in\mathcal{T}_2} |\theta^2_{ab;ij}|. \end{aligned} \tag{43}$$

We are now ready to prove one of our main results which generalizes Theorems 1 and 2. However, before proceeding with the proof, we note that our result can also be obtained using the *moment constraints* of Wainwright and Jordan (2003) (which imply that LP-S provides the exact solution for the MAP estimation problems defined over tree-structured random fields). However, as will be seen shortly, the proof presented here allows us to generalize our results to certain cycles.

Theorem 4: SOCP relaxations (and the equivalent QP relaxations) which define constraints only using graphs $G=(V,E)$ which form (arbitrarily large) trees are dominated by the LP-S relaxation.

Proof: We begin by assuming that $d(i,j)\geq 0$ for all $l_i, l_j \in \mathbf{l}$ and later drop this constraint on the distance function.[7] We will show that for any arbitrary tree G and any matrix $\mathbf{C}$, the value of $e_2^L(\mathbf{x}_T)$ is greater than the value of $e_2^S(\mathbf{x}_T)$ for all $\mathbf{x}_T$. This would prove inequality (40) which in turn would show that the LP-S relaxation dominates SOCP-T (and the equivalent QP relaxation which we call QP-T) whose constraints are defined over trees.

It is assumed that we do not specify any additional constraints for all the variables $X_{ab;ij}$ where $(a;i,b;j)\in\mathcal{T}_1$ (i.e., for $X_{ab;ij}$ not belonging to any of our trees). In other words these variables $X_{ab;ij}$ are bounded only by the relaxation of the integer constraint, that is, $-1\leq X_{ab;ij}\leq +1$. Thus in Equation (42) the minimum value of the RHS (which is equal to the value of $e_2^S(\mathbf{x}_T)$) is obtained by using the following value of $X_{ab;ij}$ where $(a;i,b;j)\in\mathcal{T}_1$:

$$X_{ab;ij} = \begin{cases} 1 & \text{if} \quad C_{ab;ij}\geq 0, \\ -1 & \text{otherwise.} \end{cases}$$

Substituting these values in Equation (42) we get

$$\begin{aligned} & e_2^S(\mathbf{x}_T) = ||\mathbf{U}^\top \mathbf{x}||^2 - \textstyle\sum_{a;i\in\mathcal{T}} C_{aa;ii} - \sum_{(a;i,b;j)\in\mathcal{T}_1} |C_{ab;ij}|, \\ \Rightarrow \quad & e_2^S(\mathbf{x}_T) = \textstyle\sum_{a;i\in\mathcal{T}} C_{aa;ii} x^2_{a;i} + \sum_{(a;i,b;j)\in\mathcal{T}_1} C_{ab;ij} x_{a;i} x_{b;j} + \sum_{(a;i,b;j)\in\mathcal{T}_2} \theta^2_{ab;ij} x_{a;i} x_{b;j} \\ & \qquad - \textstyle\sum_{a;i\in\mathcal{T}} C_{aa;ii} - \sum_{(a;i,b;j)\in\mathcal{T}_1} |C_{ab;ij}|, \end{aligned}$$

where the last expression is obtained using the fact that $\mathbf{C}=\mathbf{U}^\top\mathbf{U}$. Consider the term $\sum_{(a;i,b;j)\in\mathcal{T}_1} C_{ab;ij} x_{a;i} x_{b;j}$ which appears in the RHS of the above equation. For this term, clearly

7. Recall that $d(\cdot,\cdot)$ is a distance function on the labels. Together with the weights $w(\cdot,\cdot)$ defined over neighbouring random variables, it specifies the pairwise potentials as $\theta^2_{ab;ij} = w(a,b)d(i,j)$.

the following holds true

$$\sum_{(a;i,b;j)\in\mathcal{T}_1} C_{ab;ij}x_{a;i}x_{b;j} \le \sum_{(a;i,b;j)\in\mathcal{T}_1} \frac{|C_{ab;ij}|}{2}(x_{a;i}^2+x_{b;j}^2), \tag{44}$$

since for all $(a;i,b;j)\in\mathcal{T}_1$

$$\begin{aligned} C_{ab;ij} &\le |C_{ab;ij}|, \\ x_{a;i}x_{b;j} &\le \tfrac{(x_{a;i}^2+x_{b;j}^2)}{2}. \end{aligned}$$

Inequality (44) provides us with an upper bound on the value of $e_2^S(\mathbf{x}_T)$ as follows:

$$\begin{aligned} e_2^S(\mathbf{x}_T) \le \textstyle\sum_{a;i\in\mathcal{T}} C_{aa;ii}x_{a;i}^2 + \sum_{(a;i,b;j)\in\mathcal{T}_1} \frac{|C_{ab;ij}|}{2}(x_{a;i}^2+x_{b;j}^2) + \sum_{(a;i,b;j)\in\mathcal{T}_2} \theta^2_{ab;ij}x_{a;i}x_{b;j} \\ -\textstyle\sum_{a;i\in\mathcal{T}} C_{aa;ii} - \sum_{(a;i,b;j)\in\mathcal{T}_1}|C_{ab;ij}|. \end{aligned} \tag{45}$$

Note that in order to prove inequality (40), that is,

$$e_2^S(\mathbf{x}_T) \le e_2^L(\mathbf{x}_T), \forall \mathbf{x}_T \in [-1,1]^{|\mathcal{T}|},$$

it would be sufficient to show that $e_2^L(\mathbf{x}_T)$ specified in Equation (43) is greater than the RHS of inequality (45) (since the RHS of inequality (45) is greater than $e_2^S(\mathbf{x}_T)$). We now simplify the two infimums $e_2^L(\mathbf{x}_T)$ and $e_2^S(\mathbf{x}_T)$ as follows.

LP-S Infimum: Let $z_{a;i} = \sqrt{|x_{a;i}|(1-|x_{a;i}|)}$. From Equation (43), we see that the infimum provided by the LP-S relaxation is given by

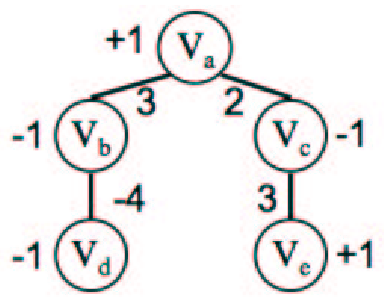
$$\begin{aligned} &\textstyle\sum_{(a;i,b;j)\in\mathcal{T}_{2+}} \theta^2_{ab;ij}(|x_{a;i}+x_{b;j}|) - \sum_{(a;i,b;j)\in\mathcal{T}_{2-}} \theta^2_{ab;ij}(|x_{a;i}-x_{b;j}|) - \sum_{(a;i,b;j)\in\mathcal{T}_2}|\theta^2_{ab;ij}| \\ &= -\textstyle\sum_{(a;i,b;j)\in\mathcal{T}_{2+}} |\theta^2_{ab;ij}|(1-|x_{a;i}+x_{b;j}|+x_{a;i}x_{b;j}) \\ &\quad -\textstyle\sum_{(a;i,b;j)\in\mathcal{T}_{2-}} |\theta^2_{ab;ij}|(1-|x_{a;i}-x_{b;j}|-x_{a;i}x_{b;j}) \\ &\quad +\textstyle\sum_{(a;i,b;j)\in\mathcal{T}_2} \theta^2_{ab;ij}x_{a;i}x_{b;j} \\ &\ge -\textstyle\sum_{(a;i,b;j)\in\mathcal{T}_2} |\theta^2_{ab;ij}|(1-|x_{a;i}|)(1-|x_{b;j}|) - 2\sum_{(a;i,b;j)\in\mathcal{T}_2}|\theta^2_{ab;ij}|z_{a;i}z_{b;j} + \\ &\quad +\textstyle\sum_{(a;i,b;j)\in\mathcal{T}_2} \theta^2_{ab;ij}x_{a;i}x_{b;j}. \end{aligned} \tag{46}$$

The last expression is obtained using the fact that

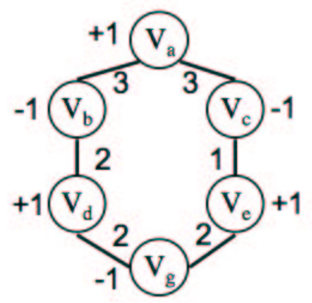
$$\begin{aligned} (1-|x_{a;i}+x_{b;j}|+x_{a;i}x_{b;j}) &\le (1-|x_{a;i}|)(1-|x_{b;j}|)+2z_{a;i}z_{b;j}, \\ (1-|x_{a;i}-x_{b;j}|-x_{a;i}x_{b;j}) &\le (1-|x_{a;i}|)(1-|x_{b;j}|)+2z_{a;i}z_{b;j}. \end{aligned}$$

SOCP Infimum: From inequality (45), we see that the infimum provided by the SOCP-T relaxation is given by

$$\begin{aligned} &\textstyle\sum_{a;i\in\mathcal{T}} C_{aa;ii}x_{a;i}^2 + \sum_{(a;i,b;j)\in\mathcal{T}_1} \frac{|C_{ab;ij}|}{2}(x_{a;i}^2+x_{b;j}^2) + \sum_{(a;i,b;j)\in\mathcal{T}_2}\theta^2_{ab;ij}x_{a;i}x_{b;j} \\ &\quad -\textstyle\sum_{a;i\in\mathcal{T}} C_{aa;ii} - \sum_{(a;i,b;j)\in\mathcal{T}_1}|C_{ab;ij}| \\ =&\quad -\textstyle\sum_{a;i\in\mathcal{T}} C_{aa;ii}(1-x_{a;i}^2) - \sum_{(a;i,b;j)\in\mathcal{T}_1}|C_{ab;ij}|(1-\frac{x_{a;i}^2}{2}-\frac{x_{b;j}^2}{2}) \\ &\quad +\textstyle\sum_{(a;i,b;j)\in\mathcal{T}_2}\theta^2_{ab;ij}x_{a;i}x_{b;j} \\ \le&\quad -\textstyle\sum_{a;i\in\mathcal{T}} C_{aa;ii}(1-|x_{a;i}|)^2 - \sum_{(a;i,b;j)\in\mathcal{T}_1}|C_{ab;ij}|(1-|x_{a;i}|)(1-|x_{b;j}|) \\ &\quad -2\textstyle\sum_{a;i\in\mathcal{T}} C_{aa;ii}z_{a;i}^2 - 2\sum_{(a;i,b;j)\in\mathcal{T}_1}|C_{ab;ij}|z_{a;i}z_{b;j} \\ &\quad +\textstyle\sum_{(a;i,b;j)\in\mathcal{T}_2}\theta^2_{ab;ij}x_{a;i}x_{b;j}. \end{aligned} \tag{47}$$



(a)



(b)

Figure 4: **(a)** An example subgraph G which forms a tree. The weights of the edges and corresponding elements of the vector $\mathbf{m}$ are also shown. **(b)** An example subgraph G which forms an even cycle where all weights are positive. The elements of $\mathbf{s}$ are defined using the $\{+1,-1\}$ assignments of the vertices.

The last expression is obtained using

$$1 - x_{a;i}^2 \geq (1-|x_{a;i}|)^2 + 2z_{a;i}^2,$$
$$1 - \frac{x_{a;i}^2}{2} - \frac{x_{b;j}^2}{2} \geq (1-|x_{a;i}|)(1-|x_{b;j}|) + 2z_{a;i}z_{b;j}.$$

In order to prove the Theorem, we use the following two Lemmas.

Lemma 5.1: The following inequality holds true for any matrix $\mathbf{C} \succeq 0$:

$$\sum_{a;i\in\mathcal{T}} C_{aa;ii}(1-|x_{a;i}|)^2 + \sum_{(a;i,b;j)\in\mathcal{T}_1} |C_{ab;ij}|(1-|x_{a;i}|)(1-|x_{b;j}|)$$
$$\geq \sum_{(a;i,b;j)\in\mathcal{T}_2} |\theta^2_{ab;ij}|(1-|x_{a;i}|)(1-|x_{b;j}|).$$

In other words, the first term in the RHS of inequality (46) exceeds the sum of the first two terms in the RHS of inequality (47).

Proof: The proof relies on the fact that $\mathbf{C}$ is positive semidefinite. We construct a vector $\mathbf{m} = \{m_a, a = 1,\cdots,n\}$ where n is the number of variables. Let $p(a)$ denote the parent of a non-root vertex a of tree G (where the root vertex can be chosen arbitrarily). The vector $\mathbf{m}$ is defined such that

$$m_a = \begin{cases} 0 & \text{if } \ a \text{ does not belong to tree } G, \\ 1 & \text{if } \ a \text{ is the root vertex of } G, \\ -m_{p(a)} & \text{if } \ w(a,p(a)) > 0, \\ m_{p(a)} & \text{if } \ w(a,p(a)) < 0. \end{cases}$$

Here $w(\cdot,\cdot)$ are the weights provided for a given MRF. Fig. 4(a) shows an example of a graph which forms a tree together with the corresponding elements of $\mathbf{m}$. Using the vector $\mathbf{m}$, we define a vector $\mathbf{s}$ of length nh (where $h = |\mathbf{l}|$) such that $s_{a;i} = 0$ if $a;i \notin \mathcal{T}$ and $s_{a;i} = m_a(1-|x_{a;i}|)$ otherwise. Since

$\mathbf{C}$ is positive semidefinite, we get

$$\begin{aligned}
& \mathbf{s}^\top \mathbf{C}\mathbf{s} \geq 0 \\
\Rightarrow\quad & \textstyle\sum_{a;i\in\mathcal{T}} C_{aa;ii}(1-|x_{a;i}|)^2 + \sum_{(a;i,b;j)\in\mathcal{T}_1} m_a m_b C_{ab;ij}(1-|x_{a;i}|)(1-|x_{b;j}) \\
& \quad + \textstyle\sum_{(a;i,b;j)\in\mathcal{T}_2} m_a m_b \theta^2_{ab;ij}(1-|x_{a;i}|)(1-|x_{b;j}) \geq 0, \\
\Rightarrow\quad & \textstyle\sum_{a;i\in\mathcal{T}} C_{aa;ii}(1-|x_{a;i}|)^2 + \sum_{(a;i,b;j)\in\mathcal{T}_1} m_a m_b C_{ab;ij}(1-|x_{a;i}|)(1-|x_{b;j}) \\
& \quad \geq \textstyle\sum_{(a;i,b;j)\in\mathcal{T}_2} |\theta^2_{ab;ij}|(1-|x_{a;i}|)(1-|x_{b;j}), \\
\Rightarrow\quad & \textstyle\sum_{a;i\in\mathcal{T}} C_{aa;ii}(1-|x_{a;i}|)^2 + \sum_{(a;i,b;j)\in\mathcal{T}_1} |C_{ab;ij}|(1-|x_{a;i}|)(1-|x_{b;j}|) \\
& \quad \geq \textstyle\sum_{(a;i,b;j)\in\mathcal{T}_2} |\theta^2_{ab;ij}|(1-|x_{a;i}|)(1-|x_{b;j}|). \quad \blacksquare
\end{aligned}$$

Lemma 5.2: The following inequality holds true for any matrix $\mathbf{C} \succeq 0$:

$$\sum_{a;i\in\mathcal{T}} C_{aa;ii} z_{a;i}^2 + \sum_{(a;i,b;j)\in\mathcal{T}_1} |C_{ab;ij}| z_{a;i} z_{b;j} \geq \sum_{(a;i,b;j)\in\mathcal{T}_2} |\theta^2_{ab;ij}| z_{a;i} z_{b;j}.$$

In other words the second term in the RHS of inequality (46) exceeds the sum of the third and fourth terms in inequality (47).

Proof: Similar to Lemma 5.1, we construct a vector $\mathbf{s}$ of length nh such that $s_{a;i} = 0$ if $a;i \notin \mathcal{T}$ and $s_{a;i} = m_a z_{a;i}$ otherwise. The proof follows by observing that $\mathbf{s}^\top\mathbf{C}\mathbf{s} \geq 0$. $\blacksquare$

Using the above two Lemmas, we see that the sum of the first two terms of inequality (46) exceed the sum of the first four terms of inequality (47). Further, the third and the fifth terms of inequalities (46) and (47) are the same. Since inequality (46) provides the lower limit of $e_2^L(\mathbf{x}_T)$ and inequality (47) provides the upper limit of $e_2^S(\mathbf{x}_T)$, it follows that $e_2^L(\mathbf{x}_T) \geq e_2^S(\mathbf{x}_T)$ for all $\mathbf{x}_T \in [-1,1]^{|\mathcal{T}|}$. Using condition (40), this proves the Theorem. $\blacksquare$

The proofs of Lemmas 5.1 and 5.2 make use of the fact that for any neighbouring random variables v_a and v_b (i.e., $(a,b) \in \mathcal{E}$), the pairwise potentials $\theta^2_{ab;ij}$ have the same sign for all $l_i, l_j \in \mathbf{l}$. This follows from the non-negativity property of the distance function. However, Theorem 4 can be extended to the case where the distance function does not obey the non-negativity property. To this end, we define a parameter $\overline{\theta}$ which satisfies the following condition:

$$Q(f,\mathbf{D};\overline{\theta}) = Q(f,\mathbf{D};\theta), \forall f.$$

Such a parameter $\overline{\theta}$ is called the reparameterization of θ (i.e., $\overline{\theta} \equiv \theta$). Note that there exist several reparameterizations of any parameter θ. We are interested in a parameter $\overline{\theta}$ which satisfies

$$\sum_{l_i,l_j\in\mathbf{l}} |\overline{\theta}^2_{ab;ij}| = |\sum_{l_i,l_j\in\mathbf{l}} \theta^2_{ab;ij}|, \forall (a,b) \in \mathcal{E}. \tag{48}$$

It can easily be shown that such a reparameterization always exists. Specifically, consider the general form of reparameterization described by Kolmogorov (2006), that is,

$$\begin{aligned}
\overline{\theta}^1_{a;i} &= \theta^1_{a;i} + M_{ba;i}, \\
\overline{\theta}^2_{ab;ij} &= \theta^2_{ab;ij} - M_{ba;i} - M_{ab;j}.
\end{aligned}$$

Clearly one can set the values of the terms $M_{ba;i}$ and $M_{ab;j}$ such that Equation (48) is satisfied. Further, the optimal value of LP-S for the parameter $\overline{\theta}$ is equal to its optimal value obtained using θ.

For details, we refer the reader to Kolmogorov (2006). Using this parameter $\overline{\theta}$, we obtain an LP-S infimum which is similar in form to the inequality (46) for any distance function (i.e., without the positivity constraint $d(i,j) \geq 0$ for all $l_i, l_j \in \mathbf{l}$). This LP-S infimum can then be easily compared to the SOCP-T infimum of inequality (47) (using slight extensions of Lemmas 5.1 and 5.2), thereby proving the results of Theorem 4 for a general distance function. We omit details.

As an upshot of the above Theorem, we see that the feasibility region of LP-S is always a subset of the feasibility region of SOCP-T (for any general set of trees and SOC constraints), that is, $\mathcal{F}(\text{LP-S}) \subset \mathcal{F}(\text{SOCP-T})$. This implies that $\mathcal{F}(\text{LP-S}) \subset \mathcal{F}(\text{QP-T})$, where QP-T is the equivalent QP relaxation defined by SOCP-T.

We now show how the above proof can be generalized to certain cycles. To the best of our knowledge, the alternative proof in Wainwright and Jordan (2003) cannot be trivially extended to these cases.

5.3 QP and SOCP Relaxations over Cycles

We now prove that the above result also holds true when the graph G forms an *even cycle*, that is, cycles with even number of vertices, whose weights are all non-negative or all non-positive provided $d(i,j) \geq 0$, for all $l_i, l_j \in \mathbf{l}$.

Theorem 5: When $d(i,j) \geq 0$ for all $l_i, l_j \in \mathbf{l}$, the SOCP relaxations which define constraints only using non-overlapping graphs G which form (arbitrarily large) even cycles with all positive or all negative weights are dominated by the LP-S relaxation.

Proof: It is sufficient to show that Lemmas 5.1 and 5.2 hold for a graph $G = (V,E)$ which forms an even cycle. We first consider the case where $\theta^2_{ab;ij} > 0$. Without loss of generality, we assume that $V = \{1,2,\ldots,t\}$ (where t is even) such that $(i,i+1) \in E$ for all $i = 1,\cdots,t-1$. Further, $(t,1) \in E$ thereby forming an even cycle. We construct a vector $\mathbf{m}$ of size n such that $m_a = -1^a$ if $a \in V$ and $m_a = 0$ otherwise. When $\theta^2_{ab;ij} < 0$, we define a vector $\mathbf{m}$ such that $m_a = 1$ if $a \in V$ and $m_a = 0$ otherwise. Fig. 4(b) shows an example of a graph G which forms an even cycle together with the corresponding elements of $\mathbf{m}$. Using $\mathbf{m}$, we construct a vector $\mathbf{s}$ of length nh (similar to the proofs of Lemmas 5.1 and 5.2). Lemmas 5.1 and 5.2 follow from the fact that $\mathbf{s}^\top \mathbf{C} \mathbf{s} \geq 0$. We omit details. ∎

It is worth noting that the feasibility region of the above SOCP relaxation (with SOC constraints defined using some special form of cycles) is not a subset of the LP-S relaxation. However, we are still able to show that LP-S dominates such an SOCP relaxation. The above Theorem helps illustrate the difference between comparing the feasibility regions of relaxations and the employing the concept of domination. Specifically, if the feasibility region of relaxation A is a subset of the feasibility region of relaxation B, then B dominates A. However, the converse of the above statement is not true.

Theorem 5 can be proved for cycles of any length whose weights are all negative by a similar construction. Further, it also holds true for *odd cycles* (i.e., cycles of odd number of variables) which have only one positive or only one negative weight. However, as will be seen in the next section, unlike trees it is not possible to extend these results for any general cycle.

6. Some Useful SOC Constraints

We now describe two SOCP relaxations which include all the marginalization constraints specified in LP-S. Note that the marginalization constraints can be incorporated within the SOCP framework but not in the QP framework.

6.1 The SOCP-C Relaxation

The SOCP-C relaxation (where C denotes cycles) defines second order cone (SOC) constraints using positive semidefinite matrices $\mathbf{C}$ such that the graph G (defined in § 5.1) form cycles. Let the variables corresponding to vertices of one such cycle G of length c be denoted as $\mathbf{v}_C = \{v_b | b \in \{a_1, a_2, \cdots, a_c\}\}$. Further, let $\mathbf{l}_C = \{l_j | j \in \{i_1, i_2, \cdots, i_c\}\} \in \mathbf{l}^c$ be a set of labels for the variables $\mathbf{v}_C$. The SOCP-C relaxation specifies the following constraints:

- The marginalization constraints, that is,

$$\sum_{l_j \in \mathbf{l}} X_{ab;ij} = (2-h)x_{a;i}, \forall (a,b) \in \mathcal{E}, l_i \in \mathbf{l}.$$

- A set of SOC constraints

$$||\mathbf{U}^\top \mathbf{x}|| \leq \mathbf{C} \bullet \mathbf{X}, \tag{49}$$

  such that the graph G defined by the above constraint forms a cycle. The matrix $\mathbf{C}$ is 0 everywhere except the following elements:

$$C_{a_k,a_l,i_k,i_l} = \begin{cases} \lambda_c & \text{if} \quad k = l, \\ D_c(k,l) & \text{otherwise}. \end{cases}$$

  Here $\mathbf{D}_c$ is a $c \times c$ matrix which is defined as follows:

$$D_c(k,l) = \begin{cases} 1 & \text{if} \quad |k-l| = 1, \\ (-1)^{c-1} & \text{if} \quad |k-l| = c-1, \\ 0 & \text{otherwise}, \end{cases}$$

  and λ_c is the absolute value of the smallest eigenvalue of $\mathbf{D}_c$.

In other words the submatrix of $\mathbf{C}$ defined by $\mathbf{v}_C$ and $\mathbf{l}_C$ has diagonal elements equal to λ_c and off-diagonal elements equal to the elements of $\mathbf{D}_c$. As an example we consider two cases when $c = 3$ and $c = 4$. In these cases the matrix $\mathbf{D}_c$ is given by

$$\mathbf{D}_3 = \begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix} \text{ and } \mathbf{D}_4 = \begin{pmatrix} 0 & 1 & 0 & -1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ -1 & 0 & 0 & 1 \end{pmatrix},$$

respectively, while $\lambda_3 = 1$ and $\lambda_4 = \sqrt{2}$. Clearly, $\mathbf{C} = \mathbf{U}^\top \mathbf{U} \succeq 0$ since its only non-zero submatrix $\lambda_c I + \mathbf{D}_c$ (where I is a $c \times c$ identity matrix) is positive semidefinite. This allows us to define a valid SOC constraint as shown in inequality (49). We choose to define the SOC constraint (49) for only those set of labels $\mathbf{l}_C$ which satisfy the following:

$$\sum_{(a_k,a_l) \in \mathcal{E}} D_c(k,l) \theta^2_{a_k a_l; i_k i_l} \geq \sum_{(a_k,a_l) \in \mathcal{E}} D_c(k,l) \theta^2_{a_k a_l; j_k j_l}, \forall \{j_1, j_2, \cdots, j_c\}.$$

Note that this choice is motivated by the fact that the variables $X_{a_k a_l;i_k i_l}$ corresponding to these sets $\mathbf{v}_C$ and $\mathbf{l}_C$ are assigned trivial values by the LP-S relaxation in the presence of non-submodular terms (see example below), that is,

$$X_{a_k a_l;i_k i_l} = \begin{cases} -1 & \text{if} \quad \theta^2_{a_k a_l;i_k i_l} \geq 0, \\ 1 & \text{otherwise.} \end{cases}$$

In order to avoid this trivial solution, we impose the SOC constraint (49) on them.

Since marginalization constraints are included in the SOCP-C relaxation, the value of the objective function obtained by solving this relaxation would at least be equal to the value obtained by the LP-S relaxation (i.e., SOCP-C dominates LP-S, see Case II in Section 2). We can further show that in the case where $|\mathbf{l}| = 2$ and the constraint (49) is defined over a frustrated cycle[8] SOCP-C strictly dominates LP-S. One such example is given below. Note that if the given MRF contains no frustrated cycle, then it can be solved exactly using the method described by Hammer et al. (1984).

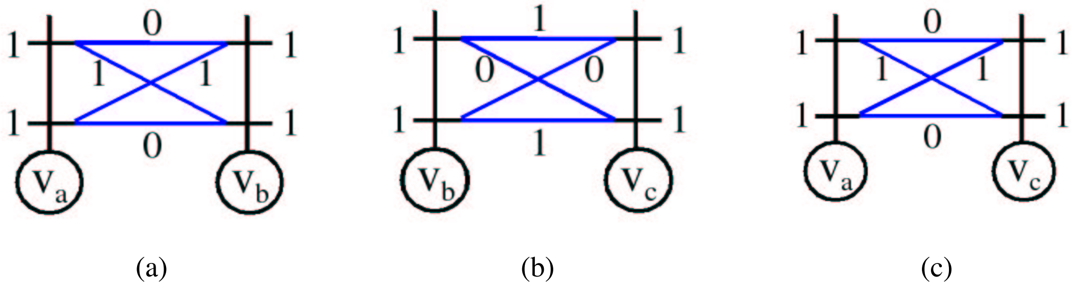


Figure 5: An example MRF defined over three random variables $\mathbf{v} = \{v_a, v_b, v_c\}$ shown as unfilled circles. Each of these variables can take one of two labels from the set $\mathbf{l} = \{l_0, l_1\}$ which are shown as branches (i.e., the horizontal lines) of trellises (i.e., the vertical lines) on top of the random variables. The unary potentials are shown next to the corresponding branches. The pairwise potentials are shown next to the edges connecting the branches of two neighbouring variables. Note that the pairwise potentials defined for (a,b) and (a,c) form a submodular Ising model (in **(a)** and **(b)** respectively). The pairwise potentials defined for (b,c) are non-submodular (in **(c)**). In other words, the above MRF defines a frustrated cycle.

Example: We consider a frustrated cycle and show that SOCP-C strictly dominates LP-S. Specifically, we consider an MRF with $\mathbf{v} = \{v_a, v_b, v_c\}$ and $\mathbf{l} = \{l_0, l_1\}$. The neighbourhood of this MRF is defined such that the variables form a cycle of length 3, that is, $\mathcal{E} = \{(a,b),(b,c),(c,a)\}$. We define a frustrated cycle which consists of all 3 variables of this MRF using the unary and pairwise potentials shown in Fig. 5, that is, the unary potentials are uniform and the pairwise potentials define only one non-submodular term (between the vertices b and c). Clearly, the energy of the optimal labelling for the above problem is 4. The value of the objective function obtained by solving the LP-S relaxation is 3 at an optimal solution shown in Fig. 6.

8. A cycle is called frustrated if it contains an odd number of non-submodular terms.

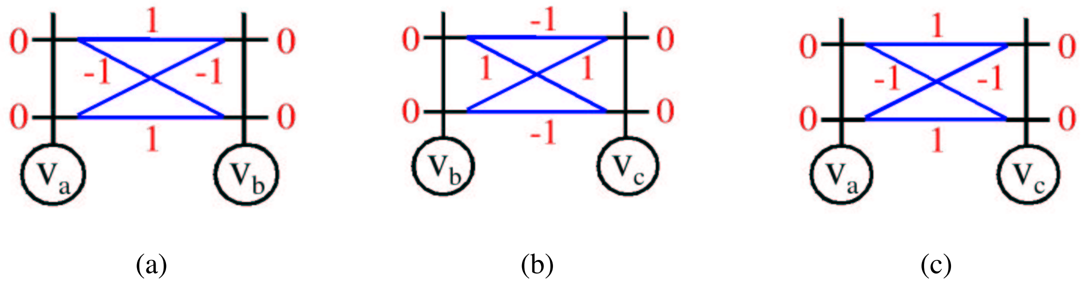


Figure 6: An optimal solution provided by the LP-S relaxation for the MRF shown in Fig. 5. The value of variable $x_{a;i}$ is shown next to the i^{th} branch of the trellis on top of v_a. In this optimal solution, all such variables $x_{a;i}$ are equal to 0. The value of the variable $X_{ab;ij}$ is shown next to the connection joining the i^{th} and the j^{th} branch of the trellises on top of v_a and v_b respectively. Note that $X_{ab;ij} = -1$ when $\theta^2_{ab;ij} > 0$ and $X_{ab;ij} = 1$ otherwise. This provides us with the minimum value of the objective function of LP-S, that is, 3.

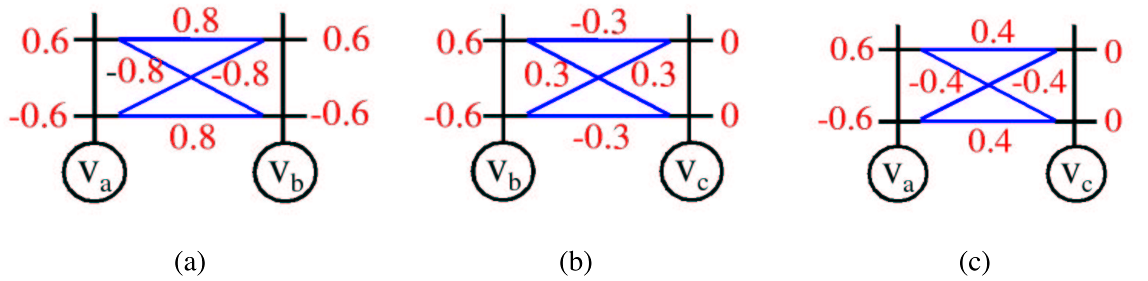


Figure 7: An optimal solution provided by the SOCP-C relaxation for the MRF shown in Fig. 5. This optimal solution provides us with the optimal value of 3.75 which greater than the LP-S optimal value for the solution shown in Fig. 6. Note that the optimal solution of LP-S does not belong to the feasibility region of SOCP-C as it violates constraint (50). This example proves that SOCP-C strictly dominates LP-S.

The LP-S optimal solution is no longer feasible when the SOCP-C relaxation is used. Specifically, the constraint

$$(x_{a;0} + x_{b;1} + x_{c;1})^2 \leq 3 + 2(X_{ab;01} + X_{ac;01} + X_{bc;11}), \tag{50}$$

is violated. In fact, the value of the objective function obtained using the SOCP-C relaxation is 3.75. Fig. 7 shows an optimal solution of the SOCP-C relaxation for the MRF in Fig. 5. The above example can be generalized to a frustrated cycle of any length. This proves that SOCP-C strictly dominates the LP-S relaxation (and hence, the QP-RL and SOCP-MS relaxations).

The constraint defined in Equation (49) is similar to the (linear) cycle inequality constraints (Chopra and Rao, 1993) which are given by

$$\sum_{k,l} D_c(k,l) X_{a_k a_l; i_k i_l} \geq 2 - c.$$

We believe that the feasibility region defined by cycle inequalities is a strict subset of the feasibility region defined by Equation (49). In other words a relaxation defined by adding cycle inequalities to LP-S would strictly dominate SOCP-C. We are not aware of a formal proof for this. We now describe the SOCP-Q relaxation.

6.2 The SOCP-Q Relaxation

In the previous section we saw that LP-S dominates SOCP relaxations whose constraints are defined on trees. However, the SOCP-C relaxation, which defines its constraints using cycles, strictly dominates LP-S. This raises the question whether matrices $\mathbf{C}$, which result in more complicated graphs G, would provide an even better relaxation for the MAP estimation problem. In this section, we answer this question in an affirmative. To this end, we define an SOCP relaxation which specifies constraints such that the resulting graph G form a clique. We denote this relaxation by SOCP-Q (where Q indicates cliques).

The SOCP-Q relaxation contains the marginalization constraint and the cycle inequalities (defined above). In addition, it also defines SOC constraints on graphs G which form a clique. We denote the variables corresponding to the vertices of clique G as $\mathbf{v}_Q = \{v_b | b \in \{a_1, a_2, \cdots, a_q\}\}$. Let $\mathbf{l}_Q = \{l_j | j \in \{i_1, i_2, \cdots, i_q\}\}$ be a set of labels for these variables $\mathbf{v}_Q$. Given this set of variables $\mathbf{v}_Q$ and labels $\mathbf{l}_Q$, we define an SOC constraint using a matrix $\mathbf{C}$ of size $nh \times nh$ which is zero everywhere except for the elements $C_{a_k a_l; i_k i_l} = 1$. Clearly, $\mathbf{C}$ is a rank 1 matrix with eigenvalue 1 and eigenvector $\mathbf{u}$ which is zero everywhere except $u_{a_k; i_k} = 1$ where $v_{a_k} \in \mathbf{v}_Q$ and $l_{i_k} \in \mathbf{l}_Q$. This implies that $\mathbf{C} \succeq 0$, which enables us to obtain the following SOC constraint:

$$\left(\sum_k x_{a_k; i_k}\right)^2 \leq q + \sum_{k,l} X_{a_k a_l; i_k i_l}. \tag{51}$$

We choose to specify the above constraint only for the set of labels $\mathbf{l}_Q$ which satisfy the following condition:

$$\sum_{(a_k, a_l) \in \mathcal{E}} \theta^2_{a_k a_l; i_k i_l} \geq \sum_{(a_k, a_l) \in \mathcal{E}} \theta^2_{a_k a_l; j_k j_l}, \forall \{j_1, j_2, \cdots, j_q\}.$$

Again, this choice is motivated by the fact that the variables $X_{a_k a_l; i_k i_l}$ corresponding to these sets $\mathbf{v}_Q$ and $\mathbf{l}_Q$ are assigned trivial values by the LP-S relaxation in the presence of non-submodular pairwise potentials.

When the clique contains a frustrated cycle, it can be shown that SOCP-Q dominates the LP-S relaxation (similar to SOCP-C). Further, using a counter-example, it can proved that the feasibility region given by cycle inequalities is not a subset of the feasibility region defined by constraint (51). One such example is given below.

Example: We present an example to prove that the feasibility region given by cycle inequalities is not a subset of the feasibility region defined by the SOC constraint

$$\left(\sum_k x_{a_k; i_k}\right)^2 \leq q + \sum_{k,l} X_{a_k a_l; i_k i_l}, \tag{52}$$

which is used in SOCP-Q. Note that it would be sufficient to provide a set of variables $(\mathbf{x}, \mathbf{X})$ which satisfy the cycle inequalities but not constraint (52).

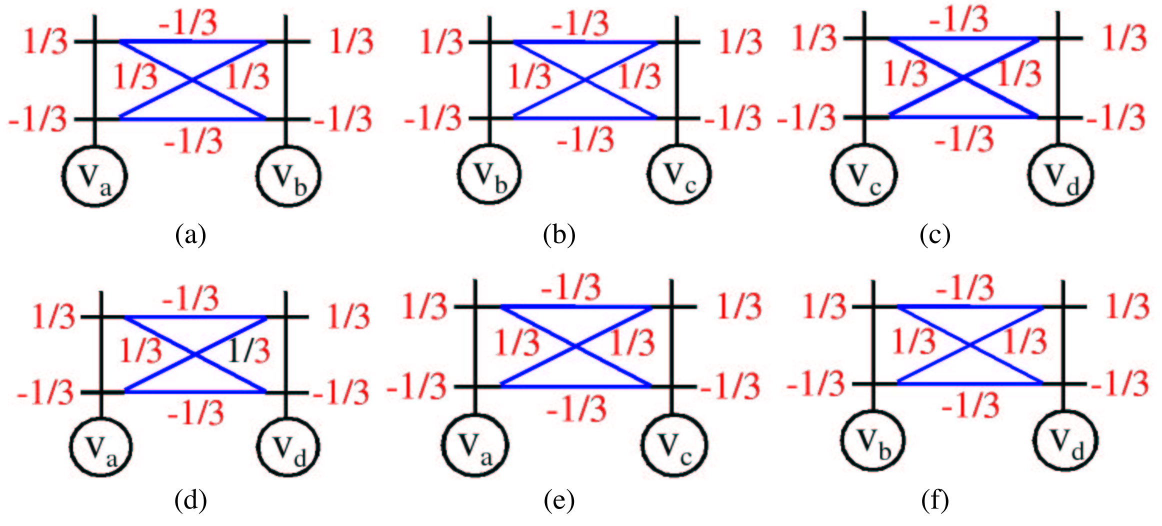


Figure 8: An infeasible solution for SOCP-Q. The value of the variable $x_{a;i}$ is shown next to the i^{th} branch of the trellis on top of v_a. The value of $X_{ab;ij}$ is shown next to the connection between the i^{th} and the j^{th} branches of the trellises on top of v_b and v_b respectively. It can easily be verified that these variables satisfy all cycle inequalities. However, they do not belong to the feasibility region of SOCP-Q since they violate constraint (53).

To this end, we consider an MRF defined over the random variables $\mathbf{v} = \{v_a, v_b, v_c, v_d\}$ which form a clique of size 4 with respect to the neighbourhood relationship $\mathcal{E}$, that is,

$$\mathcal{E} = \{(a,b), (b,c), (c,d), (a,d), (a,c), (b,d)\}.$$

Each of these variables takes a label from the set $\mathbf{l} = \{l_0, l_1\}$. Consider the set of variables $(\mathbf{x}, \mathbf{X})$ shown in Fig. 8 which do not belong to the feasibility region of SOCP-Q. It can be easily shown that these variables satisfy all the cycle inequalities (together with all the constraints of the LP-S relaxation). However, $(\mathbf{x}, \mathbf{X})$ defined in Fig. 8 does not belong to the feasibility region of the SOCP-Q relaxation since it does not satisfy the following SOC constraint:

$$\left(\sum_{v_a \in \mathbf{v}} x_{a;0} \right)^2 \leq 4 + 2 \left(\sum_{(a,b) \in \mathcal{E}} X_{ab;00} \right). \tag{53}$$

7. Discussion

We presented an analysis of approximate algorithms for MAP estimation which are based on convex relaxations. The surprising result of our work is that despite the flexibility in the form of the objective function/constraints offered by QP and SOCP, the LP-S relaxation dominates a large class of QP and SOCP relaxations. It appears that the authors who have previously used SOCP relaxations in the combinatorial optimization literature (Muramatsu and Suzuki, 2003) and those who have reported QP relaxation in the machine learning literature (Ravikumar and Lafferty, 2006) were unaware of this result. **However, our results do not discourage future research on SOCP based relaxations**.

On the contrary, we have proposed two new SOCP relaxations (SOCP-C and SOCP-Q) and presented some examples to prove that they provide a better approximation than LP-S. We also believe that SOC constraints are easier to obtain (using the method of Kim and Kojima 2000) than LP constraints. An interesting direction for future research would be to determine the best SOC constraints for a given MAP estimation problem (e.g., with truncated linear/quadratic pairwise potentials).

Acknowledgments

M. Pawan Kumar was funded by the EU CLASS project and by EPSRC grant EP/C006631/1(P). Vladimir Kolmogorov thanks ESPRC for support. Philip Torr is in receipt of a Royal Society Wolfson Research Merit Award, and would like to acknowledge support from the Royal Society and Wolfson foundation.

References

S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. *PAMI*, 23(11):1222–1239, 2001.

C. Chekuri, S. Khanna, J. Naor, and L. Zosin. Approximation algorithms for the metric labelling problem via a new linear programming formulation. In *SODA*, 2001.

S. Chopra and M. R. Rao. The parition problem. *Mathematical Programming*, 59:87–115, 1993.

F. Cohen. *Modeling and Applications of Stochastic Processes*. Kluwer, 1986.

E. Dalhaus, D. Johnson, C. Papadimitriou, P. Seymour, and M. Yannakakis. The complexity of multi-terminal cuts. *SICOMP*, 23(4):864–894, 1994.

E. de Klerk, D. Pasechnik, and J. Warners. Approximate graph colouring and max-k-cut algorithms based on the theta function. *Journal of Combinatorial Optimization*, 8(3):267–294, 2004.

M. Goemans and D. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of ACM*, 42:1115–1145, 1995.

P. Hammer and B. Kalantari. A bound on the roof duality gap. Technical Report RRR 46, Rutgers Center for Operations Research, Rutgers University, 1987.

P. Hammer, P. Hansen, and B. Simeone. Roof duality, complementation and persistency in quadratic 0-1 optimization. *Mathematical Programming*, 28:121–155, 1984.

H. Ishikawa. Exact optimization for Markov random fields with convex priors. *PAMI*, 25(10): 1333–1336, October 2003.

A. Karzanov. Minimum 0-extension of graph metrics. *European Journal of Combinatorics*, 19: 71–101, 1998.

S. Kim and M. Kojima. Second-order cone programming relaxation of nonconvex quadratic optimization problems. Technical report, Tokyo Institute of Technology, 2000.

J. Kleinberg and E. Tardos. Approximation algorithms for classification problems with pairwise relationships: Metric labeling and Markov random fields. In *STOC*, pages 14–23, 1999.

V. Kolmogorov. Convergent tree-reweighted message passing for energy minimization. *PAMI*, 28 (10):1568–1583, 2006.

N. Komodakis, N. Paragios, and G. Tziritas. MRF optimization via dual decomposition: Message-passing revisited. In *ICCV*, 2007.

A. Koster, C. van Hoesel, and A. Kolen. The partial constraint satisfaction problem: Facets and lifting theorems. *Operations Research Letters*, 23(3-5):89–97, 1998.

M. P. Kumar and P. H. S. Torr. Efficiently solving convex relaxations for MAP estimation. In *ICML*, 2008.

M. P. Kumar, P. H. S. Torr, and A. Zisserman. Solving Markov random fields using second order cone programming relaxations. In *CVPR*, volume I, pages 1045–1052, 2006.

M. P. Kumar, V. Kolmogorov, and P. H. S. Torr. An analysis of convex relaxations for MAP estimation. In *NIPS*, pages 1041–1048, 2007.

J. Lasserre. Global optimization with polynomials and the problem of moments. *SIAM Journal of Optimization*, 11:796–817, 2001.

M. Muramatsu and T. Suzuki. A new second-order cone programming relaxation for max-cut problems. *Journal of Operations Research of Japan*, 43:164–177, 2003.

C. Olsson, A.P. Eriksson, and F. Kahl. Solving large scale binary quadratic problems: Spectral methods vs. semidefinite programming. In *CVPR*, pages 1–8, 2007.

P. Ravikumar and J. Lafferty. Quadratic programming relaxations for metric labelling and Markov random field MAP estimation. In *ICML*, 2006.

P. Ravikumar, A. Agarwal, and M. Wainwright. Message-passing for graph-structured linear programs: Proximal projections, convergence and rounding schemes. In *ICML*, 2008.

C. Schellewald and C. Schnorr. Subgraph matching with semidefinite programming. In *IWCIA*, 2003.

D. Schlesinger and B. Flach. Transforming an arbitrary minsum problem into a binary one. Technical Report TUD-F106-01, Dresden University of Technology, 2006.

M. Schlesinger. Sintaksicheskiy analiz dvumernykh zritelnikh singnalov v usloviyakh pomekh (syntactic analysis of two-dimensional visual signals in noisy conditions). *Kibernetika*, 4:113–130, 1976.

M. Schlesinger and B. Flach. Some solvable subclass of structural recognition problems. In *Czech Pattern Recognition Workshop*, 2000.

M. Schlesinger and V. Giginyak. Solution to structural recognition (MAX,+)-problems by their equivalent transformations. Part 1. *Control Systems and Computers*, 1:3–15, 2007a.

M. Schlesinger and V. Giginyak. Solution to structural recognition (MAX,+)-problems by their equivalent transformations. Part 2. *Control Systems and Computers*, 2:3–18, 2007b.

R. Szeliski, R. Zabih, D. Scharstein, O. Veksler, V. Kolmogorov, A. Agarwala, M. Tappen, and C. Rother. A comparative study of energy minimization methods for markov random fields. In *ECCV*, pages II: 16–29, 2006.

P. H. S. Torr. Solving Markov random fields using semidefinite programming. In *AISTATS*, 2003.

M. Wainwright and M. Jordan. Graphical models, exponential families, and variational inference. Technical Report 649, University of California, Berkeley, 2003.

M. Wainwright and M. Jordan. Treewidth-based conditions for exactness of the Sherali-Adams and Lasserre relaxations. Technical Report 671, University of California, Berkeley, 2004.

M. Wainwright, T. Jaakola, and A. Willsky. MAP estimation via agreement on trees: Message passing and linear programming. *IEEE Trans. on Information Theory*, 51(11):3697–3717, 2005.

T. Werner. A linear programming approach to max-sum problem: A review. *PAMI*, 2007.

J. Yedidia, W. Freeman, and Y. Weiss. Understanding belief propagation and its generalizations. Technical Report TR2001-22, MERL, 2001.

Journal of Machine Learning Research 10 (2009) 107-140 Submitted 2/08; Revised 9/08; Published 1/09

Refinement of Reproducing Kernels

Yesheng Xu YXU06@SYR.EDU
Haizhang Zhang HZHANG12@SYR.EDU
Department of Mathematics
Syracuse University
Syracuse, NY 13244, USA

Editor: John Shawe-Taylor

Abstract

We continue our recent study on constructing a *refinement kernel* for a given kernel so that the reproducing kernel Hilbert space associated with the refinement kernel contains that with the original kernel as a subspace. To motivate this study, we first develop a refinement kernel method for learning, which gives an efficient algorithm for updating a learning predictor. Several characterizations of refinement kernels are then presented. It is shown that a nontrivial refinement kernel for a given kernel always exists if the input space has an infinite cardinal number. Refinement kernels for translation invariant kernels and Hilbert-Schmidt kernels are investigated. Various concrete examples are provided.

Keywords: reproducing kernels, reproducing kernel Hilbert spaces, learning with kernels, refinement kernels, translation invariant kernels, Hilbert-Schmidt kernels

1. Introduction

In our recent work (Xu and Zhang, 2007), we studied characterizations of a refinable kernel which offers a convenient way of enlarging its reproducing kernel Hilbert space (RKHS). Appropriately expanding a given RKHS is needed in learning theory when the given space is not adequate for a specific purpose. We will discuss this point in depth later. With a refinable kernel, a wavelet-like kernel or a *kernellet* was introduced in Xu and Zhang (2007). As pointed out there, the refinable kernel leaves out two important classes of kernels. Neither the Gaussian kernels nor kernels having finite dimensional feature spaces are refinable. Due to important applications of these classes of kernels, there is a need to develop a general method of enlarging a RKHS besides the particular one given by a refinable kernel. It is this need that leads to the study presented in this paper of *refinement* kernels for a given kernel.

We first review necessary notions related to kernels. Let X be a nonempty prescribed set called an input space. For $n \in \mathbb{N}$, we let $\mathbb{N}_n := \{1, 2, \dots, n\}$. A *kernel* K on X is a function from $X \times X$ to the field $\mathbb{C}$ of complex numbers such that for any finite set of inputs $\mathbf{x} := \{x_j : j \in \mathbb{N}_n\} \subseteq X$ the matrix

$$K[\mathbf{x}] := [K(x_j, x_k) : j, k \in \mathbb{N}_n] \tag{1}$$

is hermitian and positive semi-definite. Kernels are important in learning theory as they are used to measure the similarity between inputs in X (Evgeniou et al., 2000; Schölkopf and Smola, 2002; Shawe-Taylor and Cristianini, 2004; Vapnik, 1998). A *reproducing kernel Hilbert space* (RKHS) on

X is a Hilbert space of functions on X for which point evaluations are continuous linear functionals (Aronszajn, 1950).

There is a bijective correspondence between the set of kernels on X and that of reproducing kernel Hilbert spaces (RKHS) on X. In particular, for each kernel K on X there is a unique RKHS $\mathcal{H}_K$ such that

$$K(\cdot,x) \in \mathcal{H}_K, \ \text{ for all } x \in X \tag{2}$$

and for all $f \in \mathcal{H}_K$ there holds

$$f(x) = (f, K(\cdot,x))_{\mathcal{H}_K}, \ \ x \in X, \tag{3}$$

where $(\cdot,\cdot)_{\mathcal{H}_K}$ denotes the inner product on $\mathcal{H}_K$. Moreover, the linear span of $\{K(\cdot,x) : x \in X\}$ is dense in $\mathcal{H}_K$, namely,

$$\mathcal{H}_K = \overline{\operatorname{span}}\{K(\cdot,x) : x \in X\}, \tag{4}$$

and the inner product on $\mathcal{H}_K$ is determined by

$$(K(\cdot,y), K(\cdot,x))_{\mathcal{H}_K} = K(x,y), \ \ x,y \in X. \tag{5}$$

Conversely, for each RKHS $\mathcal{H}$ on X there exists exactly one kernel K on X such that (2) and (3) hold true with $\mathcal{H}_K$ replaced by $\mathcal{H}$. Equation (3) is interpreted as that a function in $\mathcal{H}_K$ can be *reproduced* through its inner product with the kernel K. For this reason, K is often called the *reproducing kernel* of $\mathcal{H}_K$.

The main purpose of this study is to investigate kernels K and G on X so that

$$\mathcal{H}_K \preceq \mathcal{H}_G \tag{6}$$

in the sense that $\mathcal{H}_K \subseteq \mathcal{H}_G$ and for all $f,g \in \mathcal{H}_K$, $(f,g)_{\mathcal{H}_K} = (f,g)_{\mathcal{H}_G}$. For an existing kernel K, we call a kernel G satisfying (6) a *refinement kernel* for K. If in addition, $\mathcal{H}_G$ contains $\mathcal{H}_K$ as a proper subspace, then we call G a *nontrivial* refinement kernel for K. The inclusion (6) was first considered by Aronszajn (1950). It was proved there that (6) holds true if and only if $L := G - K$ remains a kernel on X and $\mathcal{H}_K \cap \mathcal{H}_L = \{0\}$.

Our interest in refinement kernels is motivated by the widely used regularized learning algorithm, which and its variations have attracted 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). The algorithm aims at inferring from a finite set of training data $\mathbf{z} := \{(x_j, y_j) : j \in \mathbb{N}_m\} \subseteq X \times \mathbb{C}$ a function f_0 on X so that $f_0(x)$ would yield a meaningful output of an input $x \in X$. For a positive regularization parameter μ and the norm $\|\cdot\|_{\mathcal{H}_K}$ on $\mathcal{H}_K$, we set for each $f \in \mathcal{H}_K$

$$\mathcal{E}_{K,\mu}(f) := \sum_{j \in \mathbb{N}_m} |f(x_j) - y_j|^2 + \mu \|f\|_{\mathcal{H}_K}^2.$$

The learning algorithm then outputs a predictor f_0 as the minimizer of an error functional:

$$f_0 = \min_{f \in \mathcal{H}_K} \mathcal{E}_{K,\mu}(f). \tag{7}$$

The behavior of the predictor f_0 depends on the choice of the regularization parameter μ and as well as the RKHS. We will not consider the choice of μ in this paper. Rather, we will focus on the issue of expanding the RKHS associated with the original kernel K.

There are two possible situations where one may desire to find a nontrivial refinement kernel G for K in (7). The first happens when the predictor f_0 obtained from (7) does not work in a satisfactory way. One may hence be forced to replace K with a kernel G hoping that the corresponding learning algorithm would yield a better predictor. This is possible only if $\mathcal{H}_G$ is larger than $\mathcal{H}_K$. In other words, if the current RKHS underfits, then a refinement kernel may lead to a better predictor. The second situation occurs when the old training data $\mathbf{z}$ is expanded to be a new training data by adding to $\mathbf{z}$ more new samples from $X \times \mathbb{C}$. Since more information is available as the training data is increased, it is reasonable for people to expect a better predictor, which could be achieved by searching in a larger RKHS. This accounts for another reason one might want to find a refinement kernel for K.

In a recent paper (Xu and Zhang, 2007), we introduced a method of updating kernels via a composition of the kernel with a bijective mapping γ of the input space. Specifically, with a selected positive constant λ, we define for a kernel K on X a new kernel

$$G(x,y) := \lambda K(\gamma(x),\gamma(y)), \quad x,y \in X \tag{8}$$

and call K a γ-*refinable kernel* if (8) gives a nontrivial refinement kernel G for K. Various characterizations and many examples of refinable kernels were provided in Xu and Zhang (2007). The work was motivated by refinable functions in the context of wavelet analysis (Daubechies, 1992). As mentioned earlier, a purpose of the current study is to resolve two remaining questions in Xu and Zhang (2007). One is that a kernel is never refinable if it has a finite dimensional feature space. The other is that the commonly used Gaussian kernels are not refinable either. On the other hand, we know that kernels with a finite dimensional feature space, such as finite dot-product kernels (FitzGerald et al., 1995), and Gaussian kernels are important in learning (Micchelli and Pontil, 2005a; Schölkopf and Smola, 2002; Steinwart and Scovel, 2005; Walder et al., 2006). We would like to find nontrivial refinement kernels for them by considering general methods of updating kernels besides the particular one (8).

We organize this paper in six sections. Before delving into technical analysis of refinement kernels, we further motivate our study by proposing a refinement kernel method for learning in the next section. In Section 3, we present three basic characterizations of a refinement kernel. The first characterization is due to Aronszajn (1950), the second comes from a modification of a result in Xu and Zhang (2007) and the third result which is completely new serves as a base for further study in the remaining sections. We discuss in Section 4 the existence of a refinement kernel, and desired properties of kernels preserved by a refinement process. In particular, it will be shown that a nontrivial refinement kernel always exists if the input space contains infinite elements. In Sections 5 and 6, we study refinement kernels for translation invariant kernels and Hilbert-Schmidt kernels, respectively.

2. A Refinement Kernel Method for Learning

This section is devoted to development of learning algorithms based on refinement kernels. Suppose that a learning algorithm with kernel K has been given, that is, we have had a minimizer in the RKHS $\mathcal{H}_K$. But somehow we find that the minimizer is not good enough for a specific purpose. We then want to make a new search for a new minimizer in a larger RKHS $\mathcal{H}_G$, where G is a refinement kernel for K. We will demonstrate how a new search is done by making use of the previously computed results for the kernel K and the corresponding minimizer. We will refer to the methods described in this section as *refinement kernel methods* for learning.

For simplicity of presentation, we work only with real numbers in this section. Let K be a kernel on the input space X and $\mathbf{z} := \{(x_j, y_j) : j \in \mathbb{N}_m\} \subseteq X \times \mathbb{R}$ a finite set of sample data. We return to the learning algorithm described in the introduction which has the form

$$\min_{f \in \mathcal{H}_K} \left\{ \sum_{j \in \mathbb{N}_m} |f(x_j) - y_j|^2 + \mu \|f\|^2_{\mathcal{H}_K} \right\}. \tag{9}$$

The *representer theorem* (Kimeldorf and Wahba, 1971; Schölkopf et al., 2001; Schölkopf and Smola, 2002) in learning theory ensures that the minimizer $f_0 \in \mathcal{H}_K$ of (9) has the form

$$f_0 = \sum_{j \in \mathbb{N}_m} c_j K(\cdot, x_j).$$

In the above equation the vector $\mathbf{c} := [c_j : j \in \mathbb{N}_m]^T$ satisfies the linear system

$$(\mu I_m + K[\mathbf{x}])\mathbf{c} = \mathbf{y}, \tag{10}$$

where I_m denotes the $m \times m$ identity matrix, $\mathbf{x} := [x_j : j \in \mathbb{N}_m]^T$ and $\mathbf{y} := [y_j : j \in \mathbb{N}_m]^T$.

Suppose that the minimizer f_0 is not satisfactory and we need to have a new search in a larger RKHS. We assume that a refinement kernel G for K has been chosen and the training data $\mathbf{z}$ has been expanded to $\mathbf{z} \cup \mathbf{z}'$, where $\mathbf{z}' := \{(x'_k, y'_k) : k \in \mathbb{N}_q\} \subseteq X \times \mathbb{R}$. A new predictor can be obtained as the minimizer of

$$\min_{g \in \mathcal{H}_G} \left\{ \sum_{j \in \mathbb{N}_m} |g(x_j) - y_j|^2 + \sum_{k \in \mathbb{N}_q} |g(x'_k) - y'_k|^2 + \mu \|g\|^2_{\mathcal{H}_G} \right\}. \tag{11}$$

The purpose of this section is to develop an algorithm for efficiently solving (11) by using the existing information of the original minimization (9).

We proceed it in two steps.

2.1 Fixed Training Data

In this subsection, we assume that the training data set remains unchanged. Suppose that f_0 has been obtained, that is, linear system (10) has been solved, and one wishes to refine the kernel K in (9) to improve the predictor f_0. We consider a refinement kernel G for K for which the orthogonal complement of $\mathcal{H}_K$ in $\mathcal{H}_G$ is finite dimensional. We see from Aronszajn (1950) that there exist linearly independent functions ψ_j, $j \in \mathbb{N}_p$, on X, none of which lies in $\mathcal{H}_K$ such that the kernel $L := G - K$ has the form

$$L(x,y) := \sum_{j \in \mathbb{N}_p} \psi_j(x)\psi_j(y), \quad x, y \in X.$$

For instance, if K is the Gaussian kernel on $\mathbb{R}^d$ then L can be chosen as a finite dot-product kernel

$$\sum_{n \in \mathbb{Z}_+} a_n (x,y)^n, \quad x, y \in \mathbb{R}^d$$

or a *finite complex sinusoid kernel*

$$\sum_{n\in\mathbb{Z}^d} b_n e^{i(n,x-y)}, \ \ x,y\in\mathbb{R}^d,$$

where a and b are a nonnegative function on $\mathbb{Z}_+ := \mathbb{N}\cup\{0\}$ and $\mathbb{Z}^d$, respectively, with finite supports.

Using the refinement kernel G, we shall obtain a new predictor g_0 in a larger RKHS $\mathcal{H}_G$ that is the minimizer of

$$\min_{g\in\mathcal{H}_G}\left\{\sum_{j\in\mathbb{N}_m}|g(x_j)-y_j|^2+\mu\|g\|^2_{\mathcal{H}_G}\right\}.$$

By the representer theorem, the predictor g_0 is of the form

$$g_0=\sum_{j\in\mathbb{N}_m} d_j G(\cdot,x_j),$$

where $\mathbf{d} := [d_j : j\in\mathbb{N}_m]^T$ satisfies

$$(\mu I_m + K[\mathbf{x}] + L[\mathbf{x}])\mathbf{d} = \mathbf{y}. \tag{12}$$

Suppose that the computational results in solving (10) have been stored. Since in general $K[\mathbf{x}]$ is a dense positive semi-definite matrix, linear system (10) is usually solved by the Cholesky factorization method (Golub and van Loan, 1996). The method works at a cost of $O(m^3)$ multiplications of real numbers. We hence assume that we have obtained the Cholesky factorization of $\mu I_m + K[\mathbf{x}]$. The decomposition (12) enables us to solve a linear system whose coefficient matrix is $\mu I_m + K[\mathbf{x}]$ using only $O(m^2)$ multiplications.

Now we have to solve the new linear system (12). Instead of spending another $O(m^3)$ multiplications, we wish to make use of the stored computational results from solving (10) to reduce the computational complexity in solving (12). Specifically, for each $j\in\mathbb{N}_p$, we let $\psi_j(\mathbf{x}) := [\psi_j(x_k) : k\in\mathbb{N}_m]^T$ and let $\mathbf{e}_j$ denote the vector satisfying

$$(\mu I_m + K[\mathbf{x}])\mathbf{e}_j = \psi_j(\mathbf{x}). \tag{13}$$

We also need the vector

$$\beta := [-\psi_j(\mathbf{x})^T\mathbf{c} : j\in\mathbb{N}_p] \tag{14}$$

and the $p\times p$ matrix B defined by

$$B_{jk} := \psi_j(\mathbf{x})^T\mathbf{e}_k, \ \ j,k\in\mathbb{N}_p. \tag{15}$$

Since by (13) there holds for each $j,k\in\mathbb{N}_p$ that $B_{jk} = \mathbf{e}_j^T(\mu I_m + K[\mathbf{x}])\mathbf{e}_k$ and $\mu I_m + K[\mathbf{x}]$ is symmetric and strictly positive definite, B is symmetric and positive semi-definite. As a consequence, $I_p + B$ is invertible and we are allowed to introduce another vector $\alpha := [\alpha_j : j\in\mathbb{N}_p]^T\in\mathbb{R}^p$ as the unique solution of

$$(I_p + B)\alpha = \beta. \tag{16}$$

Proposition 1 *If vectors $\mathbf{e}_j\in\mathbb{R}^m$, $j\in\mathbb{N}_p$ are defined by (13), $\mathbf{c}\in\mathbb{R}^m$ by (10) and $\alpha\in\mathbb{R}^p$ by (16), then the solution $\mathbf{d}$ of linear system (12) is given by*

$$\mathbf{d} = \mathbf{c} + \sum_{j\in\mathbb{N}_p}\alpha_j\mathbf{e}_j. \tag{17}$$

If $\mathcal{N}(\mathbf{d})$ denotes the number of multiplications required for computing $\mathbf{d}$*, then*

$$\mathcal{N}(\mathbf{d}) = O(pm^2 + p^2m + p^3).$$

Proof Let $\mathbf{d}' := \mathbf{d} - \mathbf{c}$. It is clear by (10) and (12) that $\mathbf{d}$ satisfies (12) if and only if

$$(\mu I_m + K[\mathbf{x}])\mathbf{d}' + L[\mathbf{x}]\mathbf{d} = 0. \tag{18}$$

To prove the first statement of this proposition, it suffices to show that the vector $\mathbf{d}$ defined by (17) satisfies Equation (18). We denote by δ the left-hand side of (18). Substituting (17) into the left-hand side of (18) and noting that $L[\mathbf{x}] = \sum_{j\in\mathbb{N}_p}\psi_j(\mathbf{x})\psi_j(\mathbf{x})^T$, we obtain that

$$\delta = \sum_{j\in\mathbb{N}_p}\alpha_j\psi_j(\mathbf{x}) + \sum_{j\in\mathbb{N}_p}\psi_j(\mathbf{x})[\psi_j(\mathbf{x})^T\mathbf{c} + \sum_{k\in\mathbb{N}_p}\alpha_k\psi_j(\mathbf{x})^T\mathbf{e}_k].$$

By using (14) and (15), we have that

$$\delta = \sum_{j\in\mathbb{N}_p}\psi_j(\mathbf{x})\left[\alpha_j - \beta_j + \sum_{k\in\mathbb{N}_p}B_{jk}\alpha_k\right]. \tag{19}$$

Noting that α satisfies linear system (16) we observe for all $j\in\mathbb{N}_p$ that

$$\alpha_j - \beta_j + \sum_{k\in\mathbb{N}_p}B_{jk}\alpha_k = 0.$$

Combining this equation with (19) yields that $\delta = 0$. That is, the vector $\mathbf{d}$ defined by (17) satisfies Equation (18).

To prove the second statement, we make use of the assumption that the result of the Cholesky factorization of $\mu I_m + K[\mathbf{x}]$ has been computed and stored and we enumerate the additional number of multiplications required for computing the solution $\mathbf{d}$. Solving p linear systems (13) needs $O(pm^2)$ number of multiplications. Computing vector β and matrix B requires pm and $\frac{p(p+1)}{2}m$ multiplications respectively. Solving (16) costs $O(p^3)$ multiplications and finally, computing $\mathbf{d}$ by (17) requires pm multiplications. Summing these costs together yields the number of multiplications required to solve (12). ■

We remark that in applications, the number m of sample data is much larger than the number p of the dimension of the difference space $\mathcal{H}_L$. Therefore, under the condition $p \ll m$, we know from Proposition 1 that computing the solution $\mathbf{d}$ of the linear system (12) requires $O(m^2)$ additional number of multiplications. This is a big saving in comparison to $O(m^3)$ number of multiplications if the linear system (12) is solved directly by the Cholesky factorization without using the refinement kernel method. In other words, the use of the refinement kernel method reduces the number of multiplications from $O(m^3)$ to $O(m^2)$.

Moreover, we observe that most of the computational costs are used for solving (13), which is clearly independent of the output $\mathbf{y}$. Therefore, if $\mathbf{x}$ remains fixed for different applications, which is the case in many practical scenarios such as image analysis and processing of signals of the same size, then (13) can be calculated in advance and stored for repeated use. Taking this advantage, we only need $O(m)$ additional number of multiplications to solve (12) in order to obtain an updated predictor.

2.2 Expanded Training Data

We assume in this subsection that the training data $\mathbf{z}$ has been expanded to $\mathbf{z}\cup\mathbf{z}'$ while the kernel K remains the same. A new predictor $f_1 \in \mathcal{H}_K$ is obtained by solving the minimization problem

$$\min_{f\in\mathcal{H}_K} \sum_{j\in\mathbb{N}_m} |f(x_j)-y_j|^2 + \sum_{k\in\mathbb{N}_q} |f(x'_k)-y'_k|^2 + \mu\|f\|^2_{\mathcal{H}_K}. \tag{20}$$

The predictor f_1 can be written in terms of the kernel K. To this end, we introduce two vectors

$$\mathbf{x}' := [x'_k : k\in\mathbb{N}_q]^T,\ \ \mathbf{y}' := [y'_k : k\in\mathbb{N}_q]^T,$$

and matrices

$$K[\mathbf{x},\mathbf{x}'] := [K(x_j,x'_k) : j\in\mathbb{N}_m, k\in\mathbb{N}_q],\ \ K[\mathbf{x}',\mathbf{x}] := K[\mathbf{x},\mathbf{x}']^T.$$

For notational simplicity, we also set $A := \mu I_m + K[\mathbf{x}]$, $B := \mu I_q + K[\mathbf{x}']$ and $C := K[\mathbf{x},\mathbf{x}']$. By the representer theorem, the minimizer f_1 of (20) is given by

$$f_1 = \sum_{j\in\mathbb{N}_m} d_j K(\cdot,x_j) + \sum_{k\in\mathbb{N}_q} d'_k K(\cdot,x'_k),$$

where $\mathbf{d} := [d_j : j\in\mathbb{N}_m]^T$ and $\mathbf{d}' := [d'_k : k\in\mathbb{N}_q]^T$ satisfy

$$\begin{bmatrix} A & C \\ C^T & B \end{bmatrix}\begin{bmatrix} \mathbf{d} \\ \mathbf{d}' \end{bmatrix} = \begin{bmatrix} \mathbf{y} \\ \mathbf{y}' \end{bmatrix}. \tag{21}$$

The above linear system generally cost $O((m+q)^3)$ number of multiplications to solve by using the Cholesky factorization. With the known Cholesky factorization of $\mu I_m + K[\mathbf{x}]$, we propose a method to solve system (21) with reduction in the computational costs.

To solve system (21), we first find the $m\times q$ matrix M that satisfies

$$AM = C. \tag{22}$$

Note that

$$\begin{bmatrix} A & 0 \\ 0 & B-C^TM \end{bmatrix} = \begin{bmatrix} I_m & 0 \\ -C^TA^{-1} & I_q \end{bmatrix}\begin{bmatrix} A & C \\ C^T & B \end{bmatrix}\begin{bmatrix} I_m & -A^{-1}C \\ 0 & I_q \end{bmatrix}.$$

Thus $B-C^TM$ is symmetric and strictly positive definite. Consequently, we can solve the following system for a unique vector $\eta\in\mathbb{R}^q$

$$(B-C^TM)\eta = C^T\mathbf{c}-\mathbf{y}' \tag{23}$$

by using again the Cholesky factorization.

Proposition 2 *The solution* $\mathbf{d}$, $\mathbf{d}'$ *for linear system (21) is given as*

$$\mathbf{d} := \mathbf{c}+M\eta,\ \mathbf{d}' := -\eta. \tag{24}$$

Moreover, if $\mathcal{N}(\mathbf{d},\mathbf{d}')$ *denotes the number of multiplications required for computing both* $\mathbf{d}$ *and* $\mathbf{d}'$*, then*

$$\mathcal{N}(\mathbf{d},\mathbf{d}') = O(qm^2+q^2m+q^3).$$

Proof The first statement of this theorem follows by a direct computation using (10), (21), (22) and (23).

We now count the number of multiplications used for computing both $\mathbf{d}$ and $\mathbf{d}'$. Computing matrix M by solving the matrix Equation (22) needs $O(qm^2)$ number of multiplications. Finding η from the system (23) takes up $O(q^2m+q^3)$ number of multiplications. Computing $\mathbf{d}$ and $\mathbf{d}'$ by using (24) costs $O(qm)$ number of multiplications. Summing all these costs up proves the second part of this proposition. ■

We remark that under the assumption that $q \ll m$, we only need additional $O(m^2)$ number of multiplications to solve system (21) based on the known Cholesky factorization of matrix A from solving (10). Since (22) is independent of outputs, making use of this feature, if the inputs $\mathbf{x}$, $\mathbf{x}'$ remain unchanged in different applications, the computational costs can be further reduced as we have mentioned at the end of the last subsection.

2.3 The General Case

We now return to the general case where we have both a refinement kernel G for K such that $\mathcal{H}_{G-K}$ is p-dimensional, and an expanded training data $\mathbf{z}\cup\mathbf{z}'$. To make use of the computational result in solving (10) to compute the minimizer of (11), we divide the updating process into two steps. In step one, we fix the training data to be $\mathbf{z}$ and refine the kernel K to G. We then solve the system using the method described in Section 2.1. In step two, we fix the kernel to be G and expand $\mathbf{z}$ to $\mathbf{z}\cup\mathbf{z}'$ and solve the problem described in Section 2.2 with K replaced by G. Under reasonable hypotheses, the number of multiplications is $O(m^2)$. We state this result in the next proposition.

Proposition 3 *If $p \ll m$ and $q \ll m$ then the number of multiplications required for computing the minimizer of (11) using the algorithm described above is given by $O(m^2)$.*

The refinement kernel learning method allows us to reduce the number of multiplications from $O(m^3)$ to $O(m^2)$ by using the known Cholesky factorization of the matrix corresponding to the old kernel K.

Here we focus on the computational complexity of the refinement kernel method for the regularized learning algorithm in order to motivate the study of the refinement kernel. Although convergence and consistency of the refinement kernel method, and extensions of the method to other learning algorithms such as support vector machines are important, they are not the focus of this paper. They will be addressed in different occasions. The rest of this paper will be devoted to theoretical analysis of refinement kernels such as characterizations, existence and constructions.

3. Characterizations of Refinement Kernels

We present in this section several characterizations of refinement kernels. We begin with a review of a well-known characterization from Aronszajn (1950).

Lemma 4 *Let K,G be kernels on X. Then G is a refinement kernel for K if and only if $L := G-K$ is a kernel on X and $\mathcal{H}_K \cap \mathcal{H}_L = \{0\}$. If G is a refinement kernel for K then $\mathcal{H}_L$ is the orthogonal complement of $\mathcal{H}_K$ in $\mathcal{H}_G$, and G is a nontrivial refinement kernel for K if and only if L is not the zero kernel.*

Proof This is a direct consequence of Property 7 on page 345 and the theorem on page 353 of Aronszajn (1950). ■

In general, the conditions in the characterization presented in Lemma 4 are not easy to verify. Aiming at a characterization convenient for use, we next characterize refinement kernels in terms of feature maps for kernels, since most kernels are identified with their feature maps. A *feature map* for a kernel K on X is a mapping Φ from X to a Hilbert space $\mathcal{W}$ over $\mathbb{C}$ such that

$$K(x,y)=(\Phi(x),\Phi(y))_{\mathcal{W}},\ \ x,y\in X. \tag{25}$$

The Hilbert space $\mathcal{W}$ is called a *feature space* for K. It is well-known that K is a kernel on X if and only if it can be represented as (25) for some mapping $\Phi:X\to\mathcal{W}$ (Schölkopf and Smola, 2002). We denote by $\Phi(X)$ the image of X under the mapping Φ, by $\overline{\operatorname{span}}\Phi(X)$ the closure of the linear span of $\Phi(X)$ in $\mathcal{W}$, and by P_Φ the orthogonal projection from $\mathcal{W}$ to $\overline{\operatorname{span}}\Phi(X)$. There is a well-known characterization (Micchelli and Pontil, 2005a; Opfer, 2006; Schölkopf and Smola, 2002; Shawe-Taylor and Cristianini, 2004; Xu and Zhang, 2007) of the RKHS $\mathcal{H}_K$ of K in terms of its feature maps.

Lemma 5 *If K is a kernel on X represented as (25) by a feature map Φ from X to $\mathcal{W}$, then $\mathcal{H}_K=\{(\Phi(\cdot),u)_{\mathcal{W}}:u\in\mathcal{W}\}$ with the inner product*

$$((\Phi(\cdot),u)_{\mathcal{W}},(\Phi(\cdot),v)_{\mathcal{W}})_{\mathcal{H}_K}=(P_\Phi v,P_\Phi u)_{\mathcal{W}},\ \ u,v\in\mathcal{W}. \tag{26}$$

We shall always assume in the application of Lemma 5 that

$$\overline{\operatorname{span}}\Phi(X)=\mathcal{W} \tag{27}$$

since (25) remains valid if we replace $\mathcal{W}$ there with $\overline{\operatorname{span}}\Phi(X)$. Convenience which results from this assumption is that P_Φ in (26) would become the identity operator on $\mathcal{W}$.

We next state a characterization of refinement kernels in terms of their feature maps. Recall that we call a linear operator T from Hilbert space $\mathcal{W}_1$ to Hilbert space $\mathcal{W}_2$ *isometric* if for each $u\in\mathcal{W}_1$, $\|Tu\|_{\mathcal{W}_2}=\|u\|_{\mathcal{W}_1}$. A linear operator T from Hilbert space $\mathcal{W}_1$ to Hilbert space $\mathcal{W}_2$ is called an *isomorphism* if it is a bijective isometric linear mapping from $\mathcal{W}_1$ to $\mathcal{W}_2$. If there is an isomorphism from $\mathcal{W}_1$ to $\mathcal{W}_2$, we say that $\mathcal{W}_1$ is *isomorphic* to $\mathcal{W}_2$.

Theorem 6 *Suppose that K is a kernel on X with a feature map $\Phi:X\to\mathcal{W}$ satisfying (27) and G is a kernel on X with a feature map $\Phi':X\to\mathcal{W}'$ that satisfies*

$$\overline{\operatorname{span}}\Phi'(X)=\mathcal{W}'.$$

Then G is a refinement kernel for K if and only if there exists a bounded linear operator $T:\mathcal{W}'\to\mathcal{W}$ such that

$$T\Phi'(x)=\Phi(x),\ \ x\in X \tag{28}$$

and the adjoint operator $T^*:\mathcal{W}\to\mathcal{W}'$ of T is isometric. Moreover, G is a nontrivial refinement kernel for K if and only if T in (28) is not injective.

Proof The proof for the case when $\mathcal{W}' = \mathcal{W}$ and G is given by (8) can be found in Xu and Zhang (2007) (see Theorems 6, 7 therein). Based on Lemma 5, the arguments used there can be extended in a direct way to prove the general result described here. ■

We now explain how the general context of Theorem 6 allows us to refine kernels with a finite dimensional feature space while the refinement approach (8) does not. Suppose that K has the feature map representation (25) and G is given by (8). Then it can be seen that G has the form

$$G(x,y) = (\lambda^{1/2}\Phi(\gamma(x)), \lambda^{1/2}\Phi(\gamma(y)))_{\mathcal{W}},\ \ x,y \in X$$

with the feature space $\mathcal{W}$. By Theorem 6, G is a nontrivial refinement kernel for K if and only if there exists a bounded linear operator $T : \mathcal{W} \to \mathcal{W}$ such that

$$\lambda^{1/2} T\Phi(\gamma(x)) = \Phi(x),\ \ x \in X,$$

its adjoint T^* is isometric from $\mathcal{W}$ to $\mathcal{W}$, and T is not injective. By the latter two conditions, $\mathcal{W}$ must be isomorphic to a proper subspace of itself. This is impossible if $\mathcal{W}$ is finite dimensional. Therefore, we can not get a nontrivial refinement kernel by (8) if $\mathcal{W}$ is of finite dimension. On the other hand, by considering a general refinement process

$$G(x,y) = (\Phi'(x), \Phi'(y))_{\mathcal{W}'},\ \ x,y \in X,$$

we have the freedom to choose $\mathcal{W}'$ different from $\mathcal{W}$. The fact that $\mathcal{W}$ is finite dimensional actually makes it easier to find $\mathcal{W}'$ such that $\mathcal{W}$ is isomorphic to a proper subspace of $\mathcal{W}'$. Examples of nontrivial refinement kernels for kernels with a finite dimensional feature space will be provided in Section 6.

Our next task is to present a characterization of refinement kernels in terms of their feature spaces defined by finite positive Borel measures. This result is crucial for our discussion later on translation invariant kernels and Hilbert-Schmidt kernels. Suppose that Y is a topological space and denote by $\mathcal{B}(Y)$ the set of finite positive Borel measures on Y. For each $\rho \in \mathcal{B}(Y)$ and $p \in [1, +\infty)$ we let $L^p(Y, \rho)$ denote the space of Borel measurable functions f on Y such that

$$\int_Y |f(\xi)|^p d\rho(\xi) < +\infty.$$

In particular, $L^2(Y, \rho)$ is a Hilbert space with the inner product

$$(f,g)_{L^2(Y,\rho)} := \int_Y f(\xi)\overline{g(\xi)} d\rho(\xi),\ \ f,g \in L^2(Y,\rho).$$

For two measures $\rho_1, \rho_2 \in \mathcal{B}(Y)$, ρ_1 is said to be *absolutely continuous* with respect to ρ_2, denoted as $\rho_1 \ll \rho_2$, if for each Borel subset $V \subseteq Y$ with $\rho_2(V) = 0$, we have $\rho_1(V) = 0$. By the Radon-Nikodym theorem (see, for example, Rudin, 1987, page 121), if $\rho_1 \ll \rho_2$ then there exists a nonnegative $h \in L^1(Y, \rho_2)$ such that there holds for each Borel subset $V \subseteq Y$

$$\rho_1(V) = \int_Y h(\xi)\chi_V(\xi) d\rho_2(\xi),$$

where χ_V denotes the characteristic function of V. We sometimes write the function h satisfying the above equation as $d\rho_1/d\rho_2$.

For $\mu,\nu \in \mathcal{B}(Y)$, we let

$$\omega := \frac{\mu+\nu}{2} + \frac{|\mu-\nu|}{2},$$

where $|\mu-\nu|$ denotes the *total variation measure* of $\mu-\nu$. For the definition and properties of total variations of signed measures, see Rudin (1987, page 116). We remark that $|\mu-\nu| \in \mathcal{B}(Y)$ and $\mu(V),\nu(V) \le \omega(V)$ for all Borel subsets V of Y. It follows that μ,ν are absolutely continuous with respect to ω. We assume that a function $\phi : X \times Y \to \mathbb{C}$ has the property that for each $x \in X$, $\phi(x,\cdot) \in L^2(Y,\omega)$ and

$$\overline{\operatorname{span}}\{\phi(x,\cdot) : x \in X\} = L^2(Y,\omega). \tag{29}$$

Lemma 7 *If $\phi : X \times Y \to \mathbb{C}$ satisfies $\phi(x,\cdot) \in L^2(Y,\omega)$ for each $x \in X$ and condition (29), then $\phi(x,\cdot)$ lies in $L^2(Y,\mu)$ and $L^2(Y,\nu)$ for all $x \in X$, and* span $\{\phi(x,\cdot) : x \in X\}$ *is dense in $L^2(Y,\mu)$ and $L^2(Y,\nu)$.*

Proof We present only the case for μ since the case for ν can be similarly handled. Since $\mu \ll \omega$, we may introduce a function $h_\mu := d\mu/d\omega$. By the fact that $\mu(V) \le \omega(V)$ for each Borel $V \subseteq Y$, h_μ is less than or equal to 1 almost everywhere on Y with respect to ω. For $x \in X$, the assumption that $\phi(x,\cdot) \in L^2(Y,\omega)$ implies that it is Borel measurable. We also verify that

$$\int_Y |\phi(x,\xi)|^2 d\mu(\xi) = \int_Y |\phi(x,\xi)|^2 h_u(\xi) d\omega(\xi) \le \int_Y |\phi(x,\xi)|^2 d\omega(\xi) < +\infty.$$

This yields that $\phi(x,\cdot) \in L^2(Y,\mu)$ for all $x \in X$.

Now we assume that $f \in L^2(Y,\mu)$ is orthogonal to $\phi(x,\cdot)$ for each $x \in X$, that is,

$$\int_Y \phi(x,\xi)\overline{f(\xi)} d\mu(\xi) = 0, \ \ x \in X.$$

In the above equation, we substitute $d\mu(\xi) = h_\mu(\xi) d\omega(\xi)$ to obtain that

$$\int_Y \phi(x,\xi)\overline{f(\xi)} h_\mu(\xi) d\omega(\xi) = 0, \ \ x \in X. \tag{30}$$

The function fh_μ belongs to $L^2(Y,\omega)$ since h_μ is less than or equal to 1 almost everywhere on Y with respect to ω. Thus, by condition (29), Equation (30) implies that $fh_\mu = 0$ with respect to ω. We then observe for each Borel subset $V \subseteq Y$ that

$$\int_V |f(\xi)| d\mu(\xi) = \int_V |f(\xi)| h_\mu(\xi) d\omega(\xi) = 0.$$

This ensures that f vanishes almost everywhere on Y with respect to μ. We conclude that span $\{\phi(x,\cdot) : x \in X\}$ is dense in $L^2(Y,\mu)$. ■

By virtue of the above lemma, we introduce two kernels K_μ, K_ν by setting for all $x,y \in X$

$$K_\mu(x,y) := (\phi(x,\cdot),\phi(y,\cdot))_{L^2(Y,\mu)}, \ \ K_\nu(x,y) := (\phi(x,\cdot),\phi(y,\cdot))_{L^2(Y,\nu)}. \tag{31}$$

By Lemmas 5 and 7, functions in $\mathcal{H}_\mu := \mathcal{H}_{K_\mu}$ have the form

$$f_{\phi,\mu}(x) := (\phi(x,\cdot), f)_{L^2(Y,\mu)}, \ \ x \in X, \ \ f \in L^2(Y,\mu)$$

and the inner product on $\mathcal{H}_\mu$ is given by

$$(f_{\phi,\mu}, g_{\phi,\mu})_{\mathcal{H}_\mu} = (g, f)_{L^2(Y,\mu)}, \ \ f, g \in L^2(Y,\mu).$$

Similar results hold for $\mathcal{H}_\nu := \mathcal{H}_{K_\nu}$.

We shall characterize the relation $\mathcal{H}_\mu \preceq \mathcal{H}_\nu$ in terms of a relation of the measures μ, ν. To this end, we write $\mu \preceq \nu$ to indicate that $\mu \ll \nu$, and $d\mu/d\nu$ equals 0 or 1 almost everywhere with respect to ν, that is,

$$\nu\left(Y \setminus \left\{x \in Y : \frac{d\mu}{d\nu}(x) = 0 \text{ or } 1\right\}\right) = 0.$$

Note that $\mu \preceq \nu$ if and only if there exists a Borel subset $E \subseteq Y$ such that $\mu(Y \setminus E) = 0$ and for each Borel subset $V \subseteq E, \mu(V) = \nu(V)$.

Theorem 8 *Suppose that $\phi : X \times Y \to \mathbb{C}$ satisfies (29) and K_μ, K_ν are defined by (31). Then $\mathcal{H}_\mu \preceq \mathcal{H}_\nu$ if and only if $\mu \preceq \nu$. If $\mu \preceq \nu$ then K_ν is a nontrivial refinement kernel for K_μ if and only if*

$$\nu(Y) - \mu(Y) > 0.$$

Proof Suppose that $\mathcal{H}_\mu \preceq \mathcal{H}_\nu$. Hence, by Lemma 5, for each $f \in L^2(Y,\mu)$ there exists some $g \in L^2(Y,\nu)$ such that

$$\int_Y \phi(x,\xi)\overline{f(\xi)}d\mu(\xi) = \int_Y \phi(x,\xi)\overline{g(\xi)}d\nu(\xi) \tag{32}$$

and

$$\int_Y |f(\xi)|^2 d\mu(\xi) = \int_Y |g(\xi)|^2 d\nu(\xi). \tag{33}$$

With the derivatives $h_u := d\mu/d\omega$ and $h_\nu := d\nu/d\omega$, Equation (32) is rewritten as

$$\int_Y \phi(x,\xi)\overline{f(\xi)}h_\mu(\xi)d\omega(\xi) = \int_Y \phi(x,\xi)\overline{g(\xi)}h_\nu(\xi)d\omega(\xi).$$

This together with the density condition (29) implies that $fh_\mu = gh_\nu$ almost everywhere on Y with respect to ω. Thus for each $f \in L^2(Y,\mu)$ there exists some $g \in L^2(Y,\nu)$ satisfying for all Borel $V \subseteq Y$ that

$$\int_V f(\xi)d\mu(\xi) = \int_V f(\xi)h_\mu(\xi)d\omega(\xi) = \int_V g(\xi)h_\nu(\xi)d\omega(\xi) = \int_V g(\xi)d\nu(\xi). \tag{34}$$

We claim that $\mu \ll \nu$. We assume to the contrary that there exists a Borel set $V \subseteq Y$ for which $\nu(V) = 0$ and $\mu(V) > 0$. Letting $f = \chi_V$ in (34) yields $\mu(V) = 0$, a contradiction. Set $h := d\mu/d\nu$. By (34), the function g satisfying (32) and (33) can be taken as $g := fh$. With this choice, we obtain from (33) for each $f \in L^2(Y,\mu)$ that

$$\int_Y |f(\xi)|^2 h(\xi)d\nu(\xi) = \int_Y |f(\xi)|^2 h^2(\xi)d\nu(\xi).$$

The above equation implies that h equals 1 or 0 almost everywhere on Y with respect to ν. Consequently, $\mu \preceq \nu$.

Conversely, we suppose that $\mu \preceq \nu$ and proceed the proof by using Theorem 6. To this end, we set $E := \{x \in Y : \frac{d\mu}{d\nu}(x) = 1\}$ and introduce a linear operator $T : L^2(Y,\nu) \to L^2(Y,\mu)$ by

$$Tf := f\chi_E, \ \ f \in L^2(Y,\nu).$$

By the hypothesis $\mu \preceq \nu$, we have that $\mu(Y \setminus E) = 0$. This guarantees that for each $x \in X$, $T\phi(x,\cdot) = \phi(x,\cdot)$ in $L^2(Y,\mu)$. Note that for $g \in L^2(Y,\mu)$ and $f \in L^2(Y,\nu)$,

$$\begin{aligned}\int_Y g(\xi)\overline{(Tf)(\xi)}d\mu(\xi) &= \int_Y g(\xi)\overline{f(\xi)\chi_E(\xi)}d\mu(\xi) = \int_E g(\xi)\overline{f(\xi)}d\mu(\xi)\\ &= \int_E g(\xi)\overline{f(\xi)}d\nu(\xi) = \int_Y g(\xi)\chi_E(\xi)\overline{f(\xi)}d\nu(\xi).\end{aligned}$$

This ensures that the adjoint $T^* : L^2(Y,\mu) \to L^2(Y,\nu)$ of T is given by $T^*g = g\chi_E$, for $g \in L^2(Y,\mu)$. Moreover, it can be verified that

$$\int_Y |(T^*g)(\xi)|^2 d\nu(\xi) = \int_Y |g(\xi)\chi_E(\xi)|^2 d\nu(\xi) = \int_E |g(\xi)|^2 d\nu(\xi).$$

Since $d\mu/d\nu = 1$ on E and $\mu(Y \setminus E) = 0$, we get that

$$\int_E |g(\xi)|^2 d\nu(\xi) = \int_E |g(\xi)|^2 \frac{d\mu}{d\nu}(\xi) d\nu(\xi) = \int_E |g(\xi)|^2 d\mu(\xi) = \int_Y |g(\xi)|^2 d\mu(\xi).$$

Combining the above two equations yields that T^* is isometric. By Theorem 6, K_ν is a refinement kernel for K_μ.

If $\mu \preceq \nu$ then K_ν is a nontrivial refinement kernel for K_μ if and only if the operator T is not injective, that is, there exists $f \in L^2(Y,\nu)$ such that

$$\|f\|_{L^2(Y,\nu)} > 0 \ \text{ but } \ \|Tf\|_{L^2(Y,\mu)} = 0.$$

This is equivalent to that

$$\int_Y |f(\xi)|^2 d\nu(\xi) > 0 \ \text{ but } \ \int_E |f(\xi)|^2 d\nu(\xi) = 0.$$

Clearly, such an f exists if and only if $\nu(Y \setminus E) > 0$. Because

$$\nu(Y) - \mu(Y) = \nu(Y) - \mu(E) = \nu(Y) - \nu(E) = \nu(Y \setminus E),$$

we conclude the second statement of the theorem. ■

4. Existence of Refinement Kernels

With characterizations in Lemma 4 and Theorem 6, we shall consider existence of nontrivial refinement kernels and properties of kernels preserved by the refinement process. We let $\mathbb{C}^X$ denote the space of all the complex-valued functions on X.

Lemma 9 *A kernel K on X does not have a nontrivial refinement kernel if and only if $\mathcal{H}_K = \mathbb{C}^X$.*

Proof If $\mathcal{H}_K = \mathbb{C}^X$ then since for all kernels G on X, $\mathcal{H}_G \subseteq \mathbb{C}^X$, K does not have a nontrivial refinement kernel. Conversely, if $\mathcal{H}_K \neq \mathbb{C}^X$ then we choose an arbitrary function $\varphi \in \mathbb{C}^X \setminus \mathcal{H}_K$ and define the kernel

$$G(x,y) := K(x,y) + \varphi(x)\overline{\varphi(y)}, \ \ x,y \in X.$$

It is clear that $L := G - K$ is a kernel on X since it has a feature map $\varphi : X \to \mathbb{C}$. Moreover, $\mathcal{H}_L = \operatorname{span}\{\varphi\}$, which does not have a nontrivial intersection with $\mathcal{H}_K$. By Lemma 4, G is a nontrivial refinement kernel for K. ■

According to Lemma 9, one may expect that every kernel has a nontrivial refinement kernel since in general it should be impossible to impose an inner product on $\mathbb{C}^X$ so that it becomes a RKHS. Our next two results confirm this expectation.

Proposition 10 *If the input space X has a finite cardinality, then a kernel K on X has a nontrivial refinement kernel if and only if $K[X]$ is singular.*

Proof By Lemma 9, it suffices to show that $\mathcal{H}_K = \mathbb{C}^X$ if and only if the matrix $K[X]$ is invertible. Suppose that the cardinality of X is n and $X = \{x_j : j \in \mathbb{N}_n\}$. Assume that $K[X]$ is invertible. Then for each function $\varphi \in \mathbb{C}^X$ there exists a unique vector $[c_j : j \in \mathbb{N}_n] \in \mathbb{C}^n$ such that

$$\sum_{k\in\mathbb{N}_n} c_k K(x_j, x_k) = \varphi(x_j), \quad j \in \mathbb{N}_n,$$

which implies that $\varphi = \sum_{k\in\mathbb{N}_n} c_k K(\cdot, x_k)$. By (4), we have $\varphi \in \mathcal{H}_K$, thereby proving that $\mathcal{H}_K = \mathbb{C}^X$. Conversely, suppose that $\mathcal{H}_K = \mathbb{C}^X$. For each $j \in \mathbb{N}_n$, we introduce a function $\varphi_j \in \mathbb{C}^X$ by setting for each $l \in \mathbb{N}_n$, $\varphi_j(x_l) := \delta_{j,l}$, where δ denotes the Kronecker delta function. By (4) and the assumption that $\mathcal{H}_K = \mathbb{C}^X$, there exists a vector $[c_{j,k} : k \in \mathbb{N}_n] \in \mathbb{C}^n$ such that

$$\sum_{k\in\mathbb{N}_n} c_{j,k} K(x_l, x_k) = \varphi_j(x_l) = \delta_{j,l}, \quad j, l \in \mathbb{N}_n.$$

That is, $[c_{j,k} : j, k \in \mathbb{N}_n]$ is the inverse of the transpose of $K[X]$. Therefore, the matrix $K[X]$ is invertible. The proof is complete. ■

Theorem 11 *If the input space X has an infinite cardinality, then every kernel on it has a nontrivial refinement kernel.*

Proof According to Lemma 9, it suffices to show that there does not exist a kernel K on X such that $\mathcal{H}_K$ contains every function on X. We prove this by contradiction. Assume that there were a kernel K on X such that $\mathcal{H}_K = \mathbb{C}^X$. Since X has a countable subset of distinct points $x_n, n \in \mathbb{N}$, we may define a fixed function $f \in \mathbb{C}^X$ by setting $f(x_j) := j$ for each $j \in \mathbb{N}$ and $f(x) := 0$ for $x \in X \setminus \{x_j : j \in \mathbb{N}\}$. By (3), we would have for each $n \in \mathbb{N}$ that

$$n = |f(x_n)| = |(f, K(\cdot, x_n))_{\mathcal{H}_K}| \le \|f\|_{\mathcal{H}_K} \|K(\cdot, x_n)\|_{\mathcal{H}_K}. \tag{35}$$

Note that

$$\|K(\cdot, x_n)\|_{\mathcal{H}_K} = \sqrt{K(x_n, x_n)}, \quad n \in \mathbb{N}. \tag{36}$$

Combining (35) and (36) yields that

$$\lim_{n\to\infty} K(x_n, x_n) = +\infty. \tag{37}$$

However, again by (3) for the function $g \in \mathbb{C}^X$ defined by $g(x) := K(x,x)$, $x \in X$, we observe for each $n \in \mathbb{N}$ that

$$\begin{aligned} K(x_n,x_n) &= |g(x_n)| = |(g, K(\cdot,x_n))_{\mathcal{H}_K}| \\ &\le \|g\|_{\mathcal{H}_K}\|K(\cdot,x_n)\|_{\mathcal{H}_K} = \|g\|_{\mathcal{H}_K}\sqrt{K(x_n,x_n)}. \end{aligned}$$

This implies that

$$K(x_n,x_n) \le \|g\|^2_{\mathcal{H}_K}, \ \ n \in \mathbb{N},$$

which contradicts (37). The contradiction proves the desired result. ■

In the rest of this section, we show that the refinement process preserves the strictly positive definiteness, the continuity and the universality of the original kernel. A kernel K on X is said to be *strictly positive definite* if for any finite inputs $\mathbf{x} := \{x_j : j \in \mathbb{N}_n\} \subseteq X$ the matrix $K[\mathbf{x}]$ defined by (1) is strictly positive definite. Strictly positive definite kernels are important to the minimum norm interpolation in RKHS.

Proposition 12 *If K is a strictly positive definite kernel on X and G is a refinement kernel for K, then G is also strictly positive definite.*

Proof By Lemma 4, if G is a refinement kernel for K then $G-K$ remains a kernel on X. As a consequence, we have for all $\mathbf{x} := \{x_j : j \in \mathbb{N}_n\} \subseteq X$ that

$$G[\mathbf{x}] = K[\mathbf{x}] + (G-L)[\mathbf{x}].$$

Since $K[\mathbf{x}]$ is strictly positive definite and $(G-L)[\mathbf{x}]$ is positive semi-definite, $G[\mathbf{x}]$ is strictly positive definite. ■

The next kernel property that we consider is continuity. Suppose that the input space X is a topological space. We call a kernel on X a *continuous kernel* if it is at the same time a continuous function on $X \times X$. Given a continuous kernel K on X, can we find a nontrivial refinement kernel G for K that is also continuous? Assuming that X is a metric space with an infinite cardinality, the answer to this question is positive. Recall the Tietze extension theorem in topology (see, for example, Munkres, 2000, page 219), which states that a continuous function defined on a closed subspace of X can be extended to a continuous function on X.

Theorem 13 *If X is a metric space with an infinite cardinality, then every continuous kernel on X has a nontrivial continuous refinement kernel.*

Proof If the topology on X is discrete then any function on X is continuous. In this case, the result holds true by Theorem 11.

Now we suppose that X has an accumulation point x_0. In other words, there exists a sequence of distinct points $x_n \in X$, $n \in \mathbb{N}$ that converges to x_0 and none of the points is the same as x_0. By the arguments used in the proof of Lemma 9, it suffices to prove that there is not a continuous kernel K on X for which $\mathcal{H}_K$ contains all the continuous functions on X. Suppose to the contrary that there is such a kernel K. Then we introduce a sequence of nonnegative numbers by setting

$$c_n := \|K(\cdot,x_n) - K(\cdot,x_{n+1})\|_{\mathcal{H}_K}, \ \ n \in \mathbb{N}.$$

For each $n \in \mathbb{N}$, we define a function φ_n on the closed set $\{x_n, x_{n+1}\}$ as $\varphi_n(x_n) := 1, \varphi_n(x_{n+1}) := 0$. The function φ_n is continuous on $\{x_n, x_{n+1}\}$. Therefore, by the Tietze extension theorem, it can be extended to a continuous function on X. As a consequence, for each $n \in \mathbb{N}$, c_n is positive since otherwise we would get by (3) for each $f \in C(X) \subseteq \mathcal{H}_K$ that $f(x_n) = f(x_{n+1})$. By (5), we have that

$$c_n = \sqrt{K(x_n, x_n) + K(x_{n+1}, x_{n+1}) - K(x_n, x_{n+1}) - K(x_{n+1}, x_n)}, \ \ n \in \mathbb{N}.$$

Since K is continuous and x_n converges to x_0, c_n converges to zero as n tends to infinity.

Set $\mathcal{Z} := \{x_n : n \in \mathbb{N}\} \cup \{x_0\}$. Then $\mathcal{Z}$ is a closed subspace of X. We define a continuous function f on $\mathcal{Z}$ by

$$f(x) := \begin{cases} \sqrt{c_{2n-1}}, & x = x_{2n-1},\ n \in \mathbb{N}, \\ 0, & \text{otherwise}. \end{cases}$$

By the Tietze extension theorem, f can be extended to a continuous function on X, which we still denote by f. By the assumption that $C(X) \subseteq \mathcal{H}_K$, $f \in \mathcal{H}_K$. We now obtain by the reproducing property (3) for each $n \in \mathbb{N}$ that

$$\begin{aligned} \sqrt{c_{2n-1}} &= |f(x_{2n-1}) - f(x_{2n})| = |(f, K(\cdot, x_{2n-1}) - K(\cdot, x_{2n}))_{\mathcal{H}_K}| \\ &\leq \|f\|_{\mathcal{H}_K} \|K(\cdot, x_{2n-1}) - K(\cdot, x_{2n})\|_{\mathcal{H}_K} = \|f\|_{\mathcal{H}_K} c_{2n-1}. \end{aligned}$$

Thus we get that

$$\|f\|_{\mathcal{H}_K} \geq \frac{1}{\sqrt{c_{2n-1}}}, \ \ n \in \mathbb{N},$$

which contradicts the fact that c_n converges to zero. This contradiction implies that $\mathcal{H}_K$ can not contain the space $C(X)$. ■

The result in Theorem 13 remains valid if we only assume that X is a *normal* topological space (see, Munkres, 2000, page 195).

The last kernel property with which we are concerned is universality (Micchelli et al., 2003, 2006; Steinwart, 2001). Suppose that X is a locally compact Hausdorff space. We say that a function $K : X \times X \to \mathbb{C}$ is a *universal kernel* if it is a continuous kernel on X and for all compact subsets $\mathcal{Z} \subseteq X$, span$\{K(\cdot, x) : x \in \mathcal{Z}\}$ is dense in the Banach space $C(\mathcal{Z})$ of the continuous functions on $\mathcal{Z}$. Universal kernels were extensively studied in Micchelli et al. (2006). They are those kernels that can be used to approximate any continuous target function uniformly on a compact input space.

Proposition 14 *If K is a universal kernel on X, then any continuous refinement kernel for K is universal.*

Proof Suppose that K is a universal kernel on X and G is a continuous refinement kernel for K. Assume that K and G have feature maps $\Phi : X \to \mathcal{W}$ and $\Phi' : X \to \mathcal{W}'$, respectively. By Theorem 4 in Micchelli et al. (2006), G is universal if and only if for all compact $\mathcal{Z} \subseteq X$, span$\{(\Phi'(\cdot), u')_{\mathcal{W}'} : u' \in \mathcal{W}'\}$ is dense in $C(\mathcal{Z})$. Let $\mathcal{Z}$ be a compact subset of X. Since K is universal, Theorem 4 in Micchelli et al. (2006) ensures that

$$\overline{\text{span}}\{(\Phi(\cdot), u)_{\mathcal{W}} : u \in \mathcal{W}\} = C(\mathcal{Z}). \tag{38}$$

By Theorem 6, there exists a bounded linear operator $T : \mathcal{W}' \to \mathcal{W}$ that satisfies (28). We hence get for each $u \in \mathcal{W}$ that

$$(\Phi(\cdot), u)_{\mathcal{W}} = (T\Phi'(\cdot), u)_{\mathcal{W}} = (\Phi'(\cdot), T^* u)_{\mathcal{W}'}.$$

Therefore, there holds

$$\{(\Phi(\cdot), u)_{\mathcal{W}} : u \in \mathcal{W}\} \subseteq \{(\Phi'(\cdot), u')_{\mathcal{W}'} : u' \in \mathcal{W}'\}.$$

The above inclusion together with (38) proves that G is also universal. ■

By Theorems 11 and 13, it is reasonable to conjecture that there exist nontrivial refinement kernels for most kernels used in machine learning. To verify this conjecture and present concrete examples, we shall discuss refinement kernels for translation invariant kernels and Hilbert-Schmidt kernels in the next two sections.

5. Refinement of Translation Invariant Kernels

In this section, we specify our input space as $\mathbb{R}^d$, $d \in \mathbb{N}$ and investigate refinement kernels for translation invariant kernels on $\mathbb{R}^d$. The presentation of this section is organized into six subsections. We discuss in the first subsection the notion of translation invariant kernels. In Section 5.2 we establish various characterizations of refinement for general translation invariant kernels. We then consider several types of specific translation invariant kernels. Specifically, refinement of B-spline kernels, radial kernels and periodic kernels is studied in Sections 5.3, 5.4 and 5.5, respectively. Finally in Section 5.6, we deal with refinement through an expanding matrix.

5.1 Translation Invariant Kernels

A kernel K on $\mathbb{R}^d$ is said to be *translation invariant* if for all $a \in \mathbb{R}^d$,

$$K(x-a, y-a) = K(x,y), \ \ x, y \in \mathbb{R}^d. \tag{39}$$

For each $a \in \mathbb{R}^d$ we introduce the translation operator τ_a by setting for all functions f on $\mathbb{R}^d$

$$\tau_a f := f(\cdot - a).$$

It can be seen by (3) and (39) that a translation invariant kernel K on $\mathbb{R}^d$ satisfies for all $a, b, x, y \in \mathbb{R}^d$ that

$$\tau_a K(\cdot, x) = K(\cdot - a, x) = K(\cdot, x + a) \tag{40}$$

and

$$(\tau_a K(\cdot, y), \tau_b K(\cdot, x))_{\mathcal{H}_K} = (K(\cdot, y+a), K(\cdot, x+b))_{\mathcal{H}_K} = K(x+b, y+a). \tag{41}$$

Recall that $\mathcal{B}(\mathbb{R}^d)$ denotes the set of all the finite positive Borel measures on $\mathbb{R}^d$. It was established by Bochner in Bochner (1959) that K is a continuous translation invariant kernel on $\mathbb{R}^d$ if and only if there exists a $\mu \in \mathcal{B}(\mathbb{R}^d)$ such that

$$K(x,y) = \int_{\mathbb{R}^d} e^{i(x-y,\xi)} d\mu(\xi), \ \ x, y \in \mathbb{R}^d,$$

where $(\cdot,\cdot)$ denotes the standard inner product on $\mathbb{R}^d$. This result is referred to as the Bochner theorem. We present below a characterization of translation invariant kernels in terms of their RKHS. To this end, we call a linear operator from a Hilbert space $\mathcal{W}$ to itself an *isomorphism* on $\mathcal{W}$ if it is isomorphic from $\mathcal{W}$ to $\mathcal{W}$.

Proposition 15 *A kernel K on $\mathbb{R}^d$ is translation invariant if and only if for each $a \in \mathbb{R}^d$, τ_a is an isomorphism on $\mathcal{H}_K$.*

Proof Suppose that K is a translation invariant kernel on $\mathbb{R}^d$. Set $a \in \mathbb{R}^d$, $f \in \mathcal{H}_K$ and $\mathcal{H} := \operatorname{span}\{K(\cdot,x) : x \in \mathbb{R}^d\}$. By (4), $\mathcal{H}$ is a dense subspace of $\mathcal{H}_K$. Thus there exists a sequence of functions $f_n \in \mathcal{H}$, $n \in \mathbb{N}$ that converges to f in $\mathcal{H}_K$. By (40) and (41), $\tau_a f_n \in \mathcal{H}$ and $\|\tau_a f_n\|_{\mathcal{H}_K} = \|f_n\|_{\mathcal{H}_K}$ for each $n \in \mathbb{N}$. The latter implies that $\tau_a f_n$ form a Cauchy sequence in $\mathcal{H}_K$. Let g be their limit in $\mathcal{H}_K$. We have that $\|g\|_{\mathcal{H}_K} = \|f\|_{\mathcal{H}_K}$. To prove that τ_a is isometric on $\mathcal{H}_K$, it remains to prove that $g = \tau_a f$. To this end, we verify for each $x \in \mathbb{R}^d$ by (3), (40) and (41) that

$$\begin{aligned} g(x) &= (g, K(\cdot,x))_{\mathcal{H}_K} = \lim_{n\to\infty} (\tau_a f_n, K(\cdot,x))_{\mathcal{H}_K} \\ &= \lim_{n\to\infty} (f_n, K(\cdot,x-a))_{\mathcal{H}_K} = (f, K(\cdot,x-a))_{\mathcal{H}_K} = f(x-a). \end{aligned}$$

Similarly, it can be proved that $\tau_{-a} f_n$ converges to $\tau_{-a} f$, implying that $\tau_{-a} f \in \mathcal{H}_K$. Since $\tau_a \tau_{-a} f = f$, τ_a is surjective from $\mathcal{H}_K$ to $\mathcal{H}_K$. This together with τ_a being isometric shows that it is an isomorphism on $\mathcal{H}_K$.

Conversely, suppose that for each $a \in \mathbb{R}^d$, τ_a is an isomorphism on $\mathcal{H}_K$. This implies that the adjoint operator τ_a^* of τ_a is identified with τ_{-a} (see, Conway, 1990, page 32). It follows for each $x,y \in \mathbb{R}^d$ that $\tau_a K(\cdot,y), \tau_{-a} K(\cdot,x) \in \mathcal{H}_K$ and

$$\begin{aligned} (\tau_a K(\cdot,y), K(\cdot,x))_{\mathcal{H}_K} &= (K(\cdot,y), \tau_{-a} K(\cdot,x))_{\mathcal{H}_K} \\ &= (K(\cdot,y), K(\cdot + a, x))_{\mathcal{H}_K} = K(x, y+a). \end{aligned}$$

On the other hand, we have by (5) that

$$(\tau_a K(\cdot,y), K(\cdot,x))_{\mathcal{H}_K} = (K(\cdot - a, y), K(\cdot,x))_{\mathcal{H}_K} = K(x-a, y).$$

Combining the above two equations, we obtain that $K(x-a,y) = K(x,y+a)$ for all $a,x,y \in \mathbb{R}^d$. Replacing y with $y-a$ yields (39). ■

5.2 Characterizations

Suppose that K is a continuous translation invariant kernel on $\mathbb{R}^d$. We are interested in constructing refinement kernels for K that are continuous and translation invariant as well. Specifically, with a different measure $\mu' \in \mathcal{B}(\mathbb{R}^d)$ we introduce a new kernel

$$G(x,y) := \int_{\mathbb{R}^d} e^{i(x-y,\xi)} d\mu'(\xi), \ \ x,y \in \mathbb{R}^d$$

and characterize G being a refinement kernel for K in terms of a relation between μ and μ'.

Theorem 16 *There holds $\mathcal{H}_K \preceq \mathcal{H}_G$ if and only if $\mu \preceq \mu'$. Moreover, if $\mu \preceq \mu'$ then G is a nontrivial refinement kernel for K if and only if*

$$\mu'(\mathbb{R}^d) - \mu(\mathbb{R}^d) > 0.$$

Proof We prove this result by employing Theorem 8 with $Y := \mathbb{R}^d$. To this end, we introduce a mapping $\phi : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{C}$ by setting $\phi(x,\xi) := e^{i(x,\xi)}$, for $x,\xi \in \mathbb{R}^d$. One can see that K can be represented by

$$K(x,y) = \int_{\mathbb{R}^d} \phi(x,\xi)\overline{\phi(y,\xi)}d\mu(\xi), \;\; x,y \in \mathbb{R}^d$$

and likewise

$$G(x,y) = \int_{\mathbb{R}^d} \phi(x,\xi)\overline{\phi(y,\xi)}d\mu'(\xi), \;\; x,y \in \mathbb{R}^d.$$

Note also that for any $\omega \in \mathcal{B}(\mathbb{R}^d)$, span $\{\phi(x,\cdot) : x \in \mathbb{R}^d\}$ is dense in $L^2(\mathbb{R}^d,\omega)$. Thus, the result of this theorem is an immediate consequence of Theorem 8. ■

We next characterize the inclusion $\mathcal{H}_K \preceq \mathcal{H}_G$ by using the structure of $\mathcal{H}_K$ and $\mathcal{H}_G$. Let us prepare for this analysis by recalling some basic facts about Borel measures on $\mathbb{R}^d$. Suppose $\nu,\omega \in \mathcal{B}(\mathbb{R}^d)$. If there is a Borel subset $V \subseteq \mathbb{R}^d$ such that for each Borel $U \subseteq \mathbb{R}^d$, $\nu(U) = \nu(U \cap V)$, we say that ν is *concentrated* on V. We call ν a *singular* measure with respect to ω if there exist disjoint Borel subsets U,V of $\mathbb{R}^d$ such that ω is concentrated on U and ν is concentrated on V. The Lebesgue decomposition theorem (see, for example, Rudin, 1987, page 121) asserts that for two measures $\nu,\omega \in \mathcal{B}(\mathbb{R}^d)$, there exist two unique measures $\nu_c,\nu_s \in \mathcal{B}(\mathbb{R}^d)$ with ν_c being absolutely continuous with respect to ω and ν_s being singular with respect to ω such that ν has the Lebesgue decomposition with respect to ω

$$\nu = \nu_c + \nu_s.$$

The Lebesgue decomposition of measures with respect to the Lebesgue measure leads to a decomposition of the corresponding continuous translation invariant kernel. Specifically, for a continuous translation invariant kernel K on $\mathbb{R}^d$, we have the *Lebesgue decomposition*

$$K = K_c + K_s, \tag{42}$$

where

$$K_c(x,y) = \int_{\mathbb{R}^d} e^{i(x-y,\xi)}d\mu_c(\xi), \;\; K_s(x,y) = \int_{\mathbb{R}^d} e^{i(x-y,\xi)}d\mu_s(\xi), \;\; x,y \in \mathbb{R}^d.$$

Likewise, for a continuous translation invariant kernel G on $\mathbb{R}^d$, we also have its Lebesgue decomposition

$$G = G_c + G_s, \tag{43}$$

where

$$G_c(x,y) = \int_{\mathbb{R}^d} e^{i(x-y,\xi)}d\mu'_c(\xi), \;\; G_s(x,y) = \int_{\mathbb{R}^d} e^{i(x-y,\xi)}d\mu'_s(\xi), \;\; x,y \in \mathbb{R}^d.$$

In the above equations, the measures μ_c,μ'_c are absolutely continuous with respect to the Lebesgue measure and μ_s,μ'_s are singular with respect to the Lebesgue measure. By the Radon-Nikodym theorem, there exist nonnegative functions $k,g \in L^1(\mathbb{R}^d)$ such that

$$K_c(x,y) = \int_{\mathbb{R}^d} e^{i(x-y,\xi)}k(\xi)d\xi, \; G_c(x,y) = \int_{\mathbb{R}^d} e^{i(x-y,\xi)}g(\xi)d\xi, \;\; x,y \in \mathbb{R}^d. \tag{44}$$

The next task is to characterize $\mathcal{H}_K \preceq \mathcal{H}_G$ in terms of k, g, μ_s, μ'_s. We start with a simple observation. We associate with each $\psi \in L^1(\mathbb{R}^d)$ a finite Borel measure μ_ψ on $\mathbb{R}^d$ defined on each Borel subset $V \subseteq \mathbb{R}^d$ by

$$\mu_\psi(V) := \int_V \psi(\xi) d\xi.$$

Lemma 17 *If L is a continuous translation invariant kernel on $\mathbb{R}^d$ with a Lebesgue decomposition $L = L_c + L_s$, then $\mathcal{H}_L$ is equal to the orthogonal direct sum of $\mathcal{H}_{L_c}$ and $\mathcal{H}_{L_s}$, namely, $\mathcal{H}_L = \mathcal{H}_{L_c} \bigoplus \mathcal{H}_{L_s}$.*

Proof By Lemma 4, it suffices to show that $\mathcal{H}_{L_c} \cap \mathcal{H}_{L_s} = \{0\}$. We assume that

$$L_c(x,y) = \int_{\mathbb{R}^d} e^{i(x-y,\xi)} l(\xi) d\xi,\ L_s(x,y) = \int_{\mathbb{R}^d} e^{i(x-y,\xi)} d\rho(\xi),\ \ x,y \in \mathbb{R}^d,$$

where $l \in L^1(\mathbb{R}^d)$ is nonnegative and $\rho \in \mathcal{B}(\mathbb{R}^d)$ is singular with respect to the Lebesgue measure. Suppose that $f \in \mathcal{H}_{L_c} \cap \mathcal{H}_{L_s}$. By Lemma 5, there exists $g \in L^2(\mathbb{R}^d, \mu_l)$ and $h \in L^2(\mathbb{R}^d, \rho)$ such that

$$f(x) = \int_{\mathbb{R}^d} e^{i(x,\xi)} g(\xi) l(\xi) d\xi = \int_{\mathbb{R}^d} e^{i(x,\xi)} h(\xi) d\rho(\xi),\ \ x \in \mathbb{R}^d.$$

Define the Borel measure $\mu_{h,\rho}$ on each Borel set $V \subseteq \mathbb{R}^d$ by

$$\mu_{h,\rho}(V) := \int_V h(\xi) d\rho(\xi).$$

By the uniqueness of Fourier transforms (Grafakos, 2004), the two Borel measures μ_{gl}, $\mu_{h,\rho}$ are identical. However, μ_{gl} is absolutely continuous with respect to the Lebesgue measure while $\mu_{h,\rho}$ is singular with respect to the Lebesgue measure. Therefore, we must have $\mu_{gl} = \mu_{h,\rho} = 0$. Consequently, $f = 0$. The proof is complete. ■

The next result allows us to identify the inclusion $\mathcal{H}_K \preceq \mathcal{H}_G$ with two independent inclusions $\mathcal{H}_{K_c} \preceq \mathcal{H}_{G_c}$ and $\mathcal{H}_{K_s} \preceq \mathcal{H}_{G_s}$.

Proposition 18 *Suppose that K and G are continuous translation invariant kernels on $\mathbb{R}^d$ defined by (42) and (43). Then $\mathcal{H}_K \preceq \mathcal{H}_G$ if and only if $\mathcal{H}_{K_c} \preceq \mathcal{H}_{G_c}$ and $\mathcal{H}_{K_s} \preceq \mathcal{H}_{G_s}$.*

Proof Suppose that $\mathcal{H}_{K_c} \preceq \mathcal{H}_{G_c}$ and $\mathcal{H}_{K_s} \preceq \mathcal{H}_{G_s}$. Let f be an arbitrary function in $\mathcal{H}_K$. By Lemma 17, there exists $f_c \in \mathcal{H}_{K_c}$ and $f_s \in \mathcal{H}_{K_s}$ such that $f = f_c + f_s$ and

$$\|f\|^2_{\mathcal{H}_K} = \|f_c\|^2_{\mathcal{H}_{K_c}} + \|f_s\|^2_{\mathcal{H}_{K_s}}.$$

By the assumption, $f_c \in \mathcal{H}_{G_c}$, $f_s \in \mathcal{H}_{G_s}$ and

$$\|f_c\|_{\mathcal{H}_{G_c}} = \|f_c\|_{\mathcal{H}_{K_c}},\ \|f_s\|_{\mathcal{H}_{G_s}} = \|f_s\|_{\mathcal{H}_{K_s}}.$$

Lemma 17 asserts that $\mathcal{H}_G = \mathcal{H}_{G_c} \bigoplus \mathcal{H}_{G_s}$. As a result, we get that $f \in \mathcal{H}_G$ and

$$\|f\|^2_{\mathcal{H}_G} = \|f_c\|^2_{\mathcal{H}_{G_c}} + \|f_s\|^2_{\mathcal{H}_{G_s}} = \|f_c\|^2_{\mathcal{H}_{K_c}} + \|f_s\|^2_{\mathcal{H}_{K_s}} = \|f\|^2_{\mathcal{H}_K}.$$

Therefore, we have proved that $\mathcal{H}_K \preceq \mathcal{H}_G$.

Conversely, we assume that G is a refinement kernel for K. Suppose that $f \in \mathcal{H}_{K_c}$. By Lemma 17, $f \in \mathcal{H}_K$ and $\|f\|_{\mathcal{H}_K} = \|f\|_{\mathcal{H}_{K_c}}$. By the assumption and Lemma 17, there exist two functions $g_c \in \mathcal{H}_{G_c}$ and $g_s \in \mathcal{H}_{G_s}$ such that $f = g_c + g_s$ and

$$\|f\|^2_{\mathcal{H}_K} = \|f\|^2_{\mathcal{H}_G} = \|g_c\|^2_{\mathcal{H}_{G_c}} + \|g_s\|^2_{\mathcal{H}_{G_s}}.$$

To obtain that $f \in \mathcal{H}_{G_c}$ and $\|f\|_{\mathcal{H}_{K_c}} = \|f\|_{\mathcal{H}_{G_c}}$, it suffices to show that $g_s = 0$. Arguments similar to those used in the proof of Lemma 17 serve this purpose. Since f is an arbitrary function in $\mathcal{H}_{K_c}$, we obtain that $\mathcal{H}_{K_c} \preceq \mathcal{H}_{G_c}$. Likewise, one can show that $\mathcal{H}_{K_s} \preceq \mathcal{H}_{G_s}$. The proof is thus complete. ∎

We next consider $\mathcal{H}_{K_c} \preceq \mathcal{H}_{G_c}$. For a nonnegative function f on $\mathbb{R}^d$, we let $\Omega_f := \{x \in \mathbb{R}^d : f(x) > 0\}$. We write $k \preceq g$ to mean that $g = k$ almost everywhere on Ω_k with respect to the Lebesgue measure.

Theorem 19 *There holds $\mathcal{H}_{K_c} \preceq \mathcal{H}_{G_c}$ if and only if $k \preceq g$. Moreover, if $k \preceq g$ then G_c is a nontrivial refinement kernel for K_c if and only if*

$$\int_{\mathbb{R}^d} (g(\xi) - k(\xi))d\xi > 0. \tag{45}$$

Proof The proof for this result when G_c has the form $\lambda K_c(2\cdot, 2\cdot)$ for some positive constant λ was provided in Xu and Zhang (2007), Theorem 23. Arguments similar to those in Xu and Zhang (2007) can prove the general result described here. We give a different proof below based on Theorem 16.

By Theorem 16, $\mathcal{H}_{K_c} \preceq \mathcal{H}_{G_c}$ if and only if $\mu_k \ll \mu_g$ and $d\mu_k/d\mu_g$ equals 1 almost everywhere on Ω_k and equals 0 almost everywhere elsewhere. Therefore, $\mathcal{H}_{K_c} \preceq \mathcal{H}_{G_c}$ if and only if for each Borel $V \subseteq \Omega_k$

$$\int_V k(\xi)d\xi = \int_V g(\xi)d\xi.$$

Clearly, the above equation holds for all Borel $V \subseteq \Omega_k$ if and only if $k \preceq g$. The second statement of the result follows from the observation that the right hand side of (45) is equal to $\mu_g(\mathbb{R}^d) - \mu_k(\mathbb{R}^d)$. ∎

We are ready to present a characterization of $\mathcal{H}_K \preceq \mathcal{H}_G$ in terms of conditions on k, g, μ_s, μ'_s.

Theorem 20 *Let K and G be continuous translation invariant kernels on $\mathbb{R}^d$ defined by (42) and (43). Then $\mathcal{H}_K \preceq \mathcal{H}_G$ if and only if $k \preceq g$ and $\mu_s \preceq \mu'_s$. The refinement kernel G is nontrivial for K if and only if there holds (45) or $\mu'_s(\mathbb{R}^d) - \mu_s(\mathbb{R}^d) > 0$.*

Proof The result of this theorem follows directly from Proposition 18 and Theorems 19, 16. ∎

The next result is a direct consequence of Theorem 19. Suppose that $k \in L^1(\mathbb{R}^d)$ is positive almost everywhere on $\mathbb{R}^d$ and define

$$K(x,y) := \int_{\mathbb{R}^d} e^{i(x-y,\xi)} k(\xi)d\xi, \ \ x, y \in \mathbb{R}^d. \tag{46}$$

Corollary 21 *For $k \in L^1(\mathbb{R}^d)$ positive almost everywhere on $\mathbb{R}^d$, define K as in (46). Suppose that G is a continuous translation invariant kernel on $\mathbb{R}^d$ with a Lebesgue decomposition (43). Then $\mathcal{H}_K \preceq \mathcal{H}_G$ if and only if $G_c = K$. The kernel G is a nontrivial refinement kernel for K if and only if $G_c = K$ and $G_s \neq 0$.*

5.3 B-spline Kernels

Our next example is concerned with the B-spline kernel. For a nontrivial compactly supported function $f_0 \in L^2(\mathbb{R}^d)$ such that

$$\overline{f_0(-x)} = f_0(x), \ \ x \in \mathbb{R}^d, \tag{47}$$

we define recursively

$$f_n(x) := \int_{\mathbb{R}^d} f_{n-1}(x-y) f_0(y) dy, \ \ x \in \mathbb{R}^d, \ \ n \in \mathbb{N}.$$

For each odd integer $p \in \mathbb{N}$, we let

$$K(x,y) := f_p(x-y), \ \ x,y \in \mathbb{R}^d. \tag{48}$$

In the next proposition, we show that K is a kernel on $\mathbb{R}^d$ and characterize refinement kernels for K. To this end, we need the Fourier transform $\hat{f}$ of a function $f \in L^1(\mathbb{R}^d)$ defined as

$$\hat{f}(\xi) := \int_{\mathbb{R}^d} f(x) e^{-i(x,\xi)} dx, \ \ \xi \in \mathbb{R}^d.$$

By a standard approximation process (Grafakos, 2004), the Fourier transform can be extended to be a bounded operator on $L^2(\mathbb{R}^d)$.

Proposition 22 *For each odd integer $p \in \mathbb{N}$, K defined by (48) is a kernel on $\mathbb{R}^d$. Moreover, suppose that G is a continuous translation invariant kernel on $\mathbb{R}^d$ with a Lebesgue decomposition (43). Then G is a refinement kernel for K if and only if $G_c = K$.*

Proof Since f_0 is compactly supported, $f_0 \in L^1(\mathbb{R}^d)$. By the Schwartz inequality and by induction, we have that $f_p \in L^1(\mathbb{R}^d)$. By the Fourier transform of convolutions, we know that

$$(f_p)\hat{} = ((f_0)\hat{})^{p+1}.$$

Condition (47) ensures that $(f_0)\hat{}$ is real on $\mathbb{R}^d$. Since p is odd, $(f_p)\hat{}$ is nonnegative. By the Bochner theorem, to prove that K is a kernel on $\mathbb{R}^d$, it suffices to show that $((f_0)\hat{})^{p+1} \in L^1(\mathbb{R}^d)$. This is clear since $((f_0)\hat{})^2 \in L^1(\mathbb{R}^d)$ and $(f_0)\hat{}$ is bounded.

Since f_0 is compactly supported, by the Paley-Wiener theorem (see, for example, Gasquet and Witomski, 1999, page 293), $(f_0)\hat{}$ is real-analytic on $\mathbb{R}^d$. Also, it is nontrivial since f_0 is assumed to be nontrivial. By Corollary 21, to conclude our second statement it suffices to point out the well-known fact that the zeros of a nontrivial real-analytic function on $\mathbb{R}^d$ form a set of Lebesgue measure zero in $\mathbb{R}^d$. ■

A particular example of (48) are the *B-spline kernels* (see, for example, Schölkopf and Smola, 2002, page 98, and the references therein), which are defined as (48) by f_0 that is the characteristic function of a ball in $\mathbb{R}^d$ centered at the origin.

5.4 Radial Kernels

We next turn to the *radial* kernels on $\mathbb{R}^d$. They are kernels of the form

$$K(x,y) := r(\|x-y\|), \ \ x,y \in \mathbb{R}^d, \tag{49}$$

where r is a function on $\mathbb{R}_+ := [0,+\infty)$ and $\|\cdot\|$ denotes the standard Euclidean norm on $\mathbb{R}^d$. It was proved in Schoenberg (1938) that the function K in the form (49) with a continuous function r on $\mathbb{R}_+$ defines a kernel on $\mathbb{R}^d$ for all $d \in \mathbb{N}$ if and only if there exists some $\mu \in \mathcal{B}(\mathbb{R}_+)$ such that

$$r(t) := \int_{\mathbb{R}_+} e^{-\sigma t^2} d\mu(\sigma), \ \ t \in \mathbb{R}_+. \tag{50}$$

Theorem 23 *Suppose that K is a nontrivial radial kernel defined by (49), (50) with $\mu(\{0\}) = 0$. Then a continuous translation invariant kernel G on $\mathbb{R}^d$ with a Lebesgue decomposition (43) is a refinement kernel for K if and only if $G_c = K$.*

Proof We prove this theorem by applying Corollary 21 with identifying the function $k \in L^1(\mathbb{R}^d)$ that is positive almost everywhere.

Recalling that for all $\sigma > 0$ there holds

$$\exp\left(-\sigma\|x-y\|^2\right) = \frac{1}{(2\pi)^d}\int_{\mathbb{R}^d}\left(\frac{\pi}{\sigma}\right)^{d/2} e^{i(x-y,\xi)} e^{-\frac{\|\xi\|^2}{4\sigma}} d\xi, \ \ x,y \in \mathbb{R}^d,$$

by the hypothesis that $\mu(\{0\}) = 0$, we have for all $x,y \in \mathbb{R}^d$ that

$$K(x,y) = \int_{(0,+\infty)} \frac{1}{(2\pi)^d}\int_{\mathbb{R}^d}\left(\frac{\pi}{\sigma}\right)^{d/2} e^{i(x-y,\xi)} e^{-\frac{\|\xi\|^2}{4\sigma}} d\xi d\mu(\sigma).$$

Define

$$k(\xi) := \int_{(0,+\infty)} \frac{1}{(2\pi)^d}\left(\frac{\pi}{\sigma}\right)^{d/2} e^{-\frac{\|\xi\|^2}{4\sigma}} d\mu(\sigma), \ \ \xi \in \mathbb{R}^d.$$

It can be verified by the Fubini theorem (see, Rudin, 1987, page 164) that

$$\int_{\mathbb{R}^d} k(\xi)d\xi = \int_{(0,+\infty)} d\mu(\sigma)\int_{\mathbb{R}^d}\frac{1}{(2\pi)^d}\left(\frac{\pi}{\sigma}\right)^{d/2} e^{-\frac{\|\xi\|^2}{4\sigma}} d\xi = \int_{(0,+\infty)} d\mu(\sigma) = \mu(\mathbb{R}_+).$$

Therefore, k is a nontrivial function in $L^1(\mathbb{R}^d)$. Again, by the Fubini theorem we get that

$$K(x,y) = \int_{\mathbb{R}^d} e^{i(x-y,\xi)} k(\xi)d\xi, \ \ x,y \in \mathbb{R}^d.$$

Thus, to conclude the result of this theorem by Corollary 21, it remains to show that k is positive almost everywhere on $\mathbb{R}^d$. To this end, we observe that the function

$$\varphi(t) := \int_{(0,+\infty)} \frac{1}{(2\pi)^d}\left(\frac{\pi}{\sigma}\right)^{d/2} e^{-\frac{t}{4\sigma}} d\mu(\sigma), \ \ t > 0$$

belongs to $C^\infty(0,+\infty)$ and satisfies for each nonnegative integer j that $(-1)^j\varphi^{(j)}(t) > 0$, for $t > 0$. In other words, φ is *completely monotonic* in $(0,+\infty)$ and is hence real-analytic on the interval (see, Widder, 1941, pages 145–146). Consequently, $k = \varphi(\|\cdot\|^2)$ is real-analytic on $\mathbb{R}^d \setminus \{0\}$. This together with $\|k\|_{L^1(\mathbb{R}^d)} > 0$ proves that k is almost everywhere positive on $\mathbb{R}^d$. The proof is complete. ■

We remark that if $\mu(\{0\}) > 0$ then K is the sum of a constant kernel and a radial kernel satisfying the hypothesis of the last theorem. Note that a constant kernel is defined by a singular Borel measure. Therefore, by Theorems 20 and 23, a continuous kernel G with a Lebesgue decomposition (43) is a refinement kernel for K if and only if $G_c = K - \mu(\{0\})$ and $\mu_s'(\{0\}) = \mu(\{0\})$ since μ_s' and μ must agree at the origin.

As a direct consequence of Theorem 23, we have the following result about the Gaussian kernels

$$\mathcal{G}_\sigma(x,y) := \exp\left(-\sigma\|x-y\|^2\right), \ \ x,y \in \mathbb{R}^d, \ \ \sigma > 0.$$

Corollary 24 *A continuous translation invariant kernel G on $\mathbb{R}^d$ with a Lebesgue decomposition (43) is a refinement kernel for the Gaussian kernel $\mathcal{G}_\sigma$ if and only if $G_c = \mathcal{G}_\sigma$. It is a nontrivial refinement kernel for $\mathcal{G}_\sigma$ if and only if $G_c = \mathcal{G}_\sigma$ and $G_s \neq 0$.*

The corollary above suggests that we may refine the Gaussian kernel $\mathcal{G}_\sigma$ by adding to it a kernel G_s defined by a singular measure on $\mathbb{R}^d$. The RKHS for a Gaussian kernel has been well understood (see, for example, Walder et al., 2006). In particular, we can see by Lemma 5 that

$$\mathcal{H}_{\mathcal{G}_\sigma} := \left\{ f \in L^2(\mathbb{R}^d) : \int_{\mathbb{R}^d} |\hat{f}(\xi)|^2 e^{\frac{\|\xi\|^2}{4\sigma}} d\xi < +\infty \right\}, \tag{51}$$

and the inner product on $\mathcal{H}_{\mathcal{G}_\sigma}$ is given as

$$(f,g)_{\mathcal{H}_{\mathcal{G}_\sigma}} = \frac{1}{(2\pi)^d}\left(\frac{\sigma}{\pi}\right)^{d/2} \int_{\mathbb{R}^d} \hat{f}(\xi)\overline{\hat{g}(\xi)} e^{\frac{\|\xi\|^2}{4\sigma}} d\xi.$$

By (51), $\mathcal{H}_{\mathcal{G}_\sigma} \subseteq \mathcal{H}_{\mathcal{G}_{\sigma'}}$ for $\sigma < \sigma'$. However, since functions with a continuous compactly supported Fourier transform are contained in $\mathcal{H}_{\mathcal{G}_\sigma}$ and are dense in $\mathcal{H}_{\mathcal{G}_{\sigma'}}$, $\mathcal{H}_{\mathcal{G}_\sigma}$ is dense in $\mathcal{H}_{\mathcal{G}_{\sigma'}}$. But, $\mathcal{H}_{\mathcal{G}_\sigma}$ is not closed under the norm of $\mathcal{H}_{\mathcal{G}_{\sigma'}}$. Therefore, there does not exist σ' with $\sigma < \sigma'$ such that $\mathcal{G}_{\sigma'}$ is a refinement kernel for $\mathcal{G}_\sigma$.

5.5 Periodic Kernels

We now investigate kernels defined by continuous periodic functions and their refinement. For this purpose, we recall the Fourier coefficients of a function $f \in L^2([0,2\pi]^d)$ which are defined by setting for each $n \in \mathbb{Z}^d$

$$c_n(f) := \frac{1}{(2\pi)^d} \int_{[0,2\pi]^d} f(x) e^{-i(n,x)} dx.$$

A function f on $\mathbb{R}^d$ is called 2π-*periodic* if for all $n \in \mathbb{Z}^d$, $f = f(\cdot + 2\pi n)$.

Proposition 25 *Let f be a continuous 2π-periodic function on $\mathbb{R}^d$. Then $K(x,y) := f(x-y)$ defines a kernel on $\mathbb{R}^d$ if and only if $\mathbf{f} := [c_n(f) : n \in \mathbb{Z}^d] \in \ell^1(\mathbb{Z}^d)$ and $\mathbf{f} \geq 0$.*

Proof If $\mathbf{f} \in \ell^1(\mathbb{Z}^d)$ and $\mathbf{f} \geq 0$, then we define the Borel measure μ that is supported on $\mathbb{Z}^d$ with $\mu(n) := c_n(f)$, $n \in \mathbb{Z}^d$. By the Fourier series expansion of f, we get that

$$K(x,y) = f(x-y) = \sum_{n \in \mathbb{Z}^d} c_n(f) e^{i(n,x-y)} = \int_{\mathbb{R}^d} e^{i(x-y,\xi)} d\mu(\xi), \;\; x,y \in \mathbb{R}^d.$$

By the Bochner theorem, K is a kernel on $\mathbb{R}^d$.

Conversely, if K is a kernel on $\mathbb{R}^d$, then again by the Bochner theorem, there is a $\mu \in \mathcal{B}(\mathbb{R}^d)$ such that

$$f(x) = \int_{\mathbb{R}^d} e^{i(x,\xi)} d\mu(\xi), \;\; x \in \mathbb{R}^d. \tag{52}$$

Since f is 2π-periodic, we have for each $n \in \mathbb{Z}^d$ that

$$\int_{\mathbb{R}^d} e^{i(x,\xi)} d\mu(\xi) = \int_{\mathbb{R}^d} e^{i(x,\xi)} e^{i2\pi(n,\xi)} d\mu(\xi), \;\; x \in \mathbb{R}^d.$$

By the uniqueness of Fourier transforms, there holds for almost every $\xi \in \mathbb{R}^d$ with respect to μ that $e^{i2\pi(n,\xi)} = 1$, for $n \in \mathbb{Z}^d$. Note that this equation holds for all $n \in \mathbb{Z}^d$ if and only if $\xi \in \mathbb{Z}^d$. Therefore, $\mu(\mathbb{R}^d \setminus \mathbb{Z}^d) = 0$. Consequently, by (52) we obtain that

$$f(x) = \sum_{n \in \mathbb{Z}^d} \mu(\{n\}) e^{i(n,x)}, \;\; x \in \mathbb{R}^d.$$

It is implied that $c_n(f) = \mu(\{n\})$, $n \in \mathbb{Z}^d$. Since μ is finite and positive, $\mathbf{f} \in \ell^1(\mathbb{Z}^d)$ and $\mathbf{f} \geq 0$. ■

By the last proposition, for $c := [c_n : n \in \mathbb{Z}^d] \in \ell^1(\mathbb{Z}^d)$ with $c_n \geq 0$ for each $n \in \mathbb{Z}^d$, we introduce kernel

$$f_c(x-y) := \sum_{n \in \mathbb{Z}^d} c_n e^{i(n,x-y)}, \;\; x,y \in \mathbb{R}^d, \tag{53}$$

and set $\Omega_c := \{n \in \mathbb{Z}^d : c_n > 0\}$. The function $c \in \ell^1(\mathbb{Z}^d)$ will be viewed at the same time as a Borel measure on $\mathbb{Z}^d$ whose measure on $n \in \mathbb{Z}^d$ is defined to be c_n.

Proposition 26 *Suppose that $a, b \in \ell^1(\mathbb{Z}^d)$ with $a_n, b_n \geq 0$ for each $n \in \mathbb{Z}^d$ and define f_a, f_b as in (53). Then $\mathcal{H}_{f_a} \preceq \mathcal{H}_{f_b}$ if and only if*

$$\Omega_a \subseteq \Omega_b \;\; \textit{and for all} \;\; n \in \Omega_a, \;\; a_n = b_n. \tag{54}$$

If $\mathcal{H}_{f_a} \preceq \mathcal{H}_{f_b}$ then f_b is a nontrivial refinement kernel for f_a if and only if Ω_a is a proper subset of Ω_b.

Proof By Theorem 20, $\mathcal{H}_{f_a} \preceq \mathcal{H}_{f_b}$ if and only if $a \preceq b$. We now show that $a \preceq b$ is equivalent to (54).

Suppose that condition (54) holds. By $\Omega_a \subseteq \Omega_b$ we have that $a \ll b$. The derivative $c := da/db$ is given by

$$c_n := \begin{cases} \dfrac{a_n}{b_n}, & n \in \Omega_a, \\ 0, & \text{otherwise.} \end{cases} \tag{55}$$

Therefore, $a \preceq b$.

Conversely, suppose that $a \preceq b$. Since $a \ll b$, we have $\Omega_a \subseteq \Omega_b$. Also, since the derivative $c := da/db$ given by (55) is equal to 1 or 0 almost everywhere with respect to b, we have $a_n = b_n$ for each $n \in \Omega_a$. Hence, (54) holds. ■

5.6 Refinement via an Expanding Matrix

To close this section, we consider a special refinement process in the sense of refinable kernels introduced in Xu and Zhang (2007). The translation invariant kernels K_c defined by (44) via a non-negative function $k \in L^1(\mathbb{R}^d)$ are of special interest. We call them translation invariant kernels of continuous type. Next, we consider updating kernels of this type through an expanding matrix. Let D be a $d \times d$ real matrix with determinant $\det D$ bigger than 1. Such a matrix is called expanding. Refinable functions with respect to an expanding matrix and the corresponding wavelets were studied by many authors (see, for example, Chen et al., 2003, 2007; Daubechies, 1992; Goodman and Lee, 1994; Jia, 1999; Jia et al., 1999; Micchelli and Sauer, 1997; Micchelli and Xu, 1994; Wang, 2002, and the references cited therein). In particular, the interesting relation between wavelets and tiling was investigated in Wang (2002). For a continuous translation invariant kernel K_c defined by (44), we consider a refinement kernel G_c having the form

$$G_c(x,y) = \lambda K_c(Dx, Dy), \;\; x, y \in \mathbb{R}^d. \tag{56}$$

We are interested in characterizing $\mathcal{H}_{K_c} \preceq \mathcal{H}_{G_c}$ in terms of k, λ and D. A special case of this problem when D is the dilation matrix $2I$ was studied in Xu and Zhang (2007).

Theorem 27 *Suppose that K_c is defined by (44) via a nonnegative $k \in L^1(\mathbb{R}^d)$ and G_c is given by (56). Then, $\mathcal{H}_{K_c} \preceq \mathcal{H}_{G_c}$ if and only if for almost every $\xi \in \Omega_k$,*

$$k(\xi) = \frac{\lambda}{\det D} k((D^T)^{-1}\xi).$$

If $\mathcal{H}_{K_c} \preceq \mathcal{H}_{G_c}$ then G_c is a nontrivial refinement kernel for K_c if and only if

$$\int_{\Omega_k \setminus (D^T)^{-1}\Omega_k} k(\xi) d\xi > 0. \tag{57}$$

Proof Through a change of variables, we obtain from (56) that

$$G_c(x,y) = \frac{\lambda}{\det D} \int_{\mathbb{R}^d} e^{i(x-y,\xi)} k((D^T)^{-1}\xi) d\xi, \;\; x, y \in \mathbb{R}^d.$$

We then identify the nonnegative function $g \in L^1(\mathbb{R}^d)$ in the definition (44) of the translation invariant kernel G_c of continuous type as follows

$$g(\xi) = \frac{\lambda}{\det D} k((D^T)^{-1}\xi), \;\; \xi \in \mathbb{R}^d.$$

The first statement of this theorem now follows directly from the first statement of Theorem 19.

If $\mathcal{H}_{K_c} \preceq \mathcal{H}_{G_c}$, by the first statement of this theorem, we have that

$$\int_{\mathbb{R}^d} g(\xi)d\xi = \int_{\mathbb{R}^d} k(\xi)d\xi + \int_{\mathbb{R}^d\setminus\Omega_k} \frac{\lambda}{\det D} k((D^T)^{-1}\xi)d\xi.$$

Using this identity, we observe that inequality (45) is equivalent to inequality

$$\int_{\mathbb{R}^d\setminus\Omega_k} k((D^T)^{-1}\xi)d\xi > 0.$$

By a change of variables, we find that the inequality above is equivalent to inequality (57). ■

Along this direction, we present a corollary to Proposition 26.

Corollary 28 *Let D be an invertible matrix in $\mathbb{Z}^{d\times d}$, f_a a kernel defined as in (53) and G defined as $G(x,y) = \lambda f_a(Dx - Dy)$ for some positive constant λ. Then G is a refinement kernel for f_a if and only if $\Omega_a \subseteq D^T\Omega_a$ and for each $n \in \Omega_a$, $a_n = \lambda a_{(D^T)^{-1}n}$.*

We shall see in the next section that Proposition 26 and Corollary 28 are special instances of refinement of Hilbert-Schmidt kernels.

6. Refinement of Hilbert-Schmidt Kernels

We characterize in this section refinement kernels for two types of Hilbert-Schmidt kernels. As special examples, we study refinement of the Bergman kernels, the Szegö kernels, the Schoenberg kernels, and kernels having finite dimensional feature spaces.

Let a be a nonnegative function on $\mathbb{N}$ and set $a_n := a(n)$, $n \in \mathbb{N}$. We denote by $\ell_a^2(\mathbb{N})$ the set of functions f on $\mathbb{N}$ such that $\sum_{n\in\mathbb{N}} a_n|f_n|^2 < +\infty$. It is a Hilbert space with the inner product

$$(f,g)_{\ell_a^2(\mathbb{N})} := \sum_{n\in\mathbb{N}} a_n f_n \overline{g_n}, \ \ f,g \in \ell_a^2(\mathbb{N}).$$

Suppose that we have a sequence of functions ϕ_n on the input space X, $n \in \mathbb{N}$ such that for each $x \in X$ the function $\Phi(x)$ on $\mathbb{N}$ defined as

$$\Phi(x)(n) := \phi_n(x), \ \ n \in \mathbb{N} \tag{58}$$

belongs to $\ell_a^2(\mathbb{N})$. The *Hilbert-Schmidt kernel* K_a associated with a is given as

$$K_a(x,y) := (\Phi(x),\Phi(y))_{\ell_a^2(\mathbb{N})} = \sum_{n\in\mathbb{N}} a_n\phi_n(x)\overline{\phi_n(y)}, \ \ x,y \in X. \tag{59}$$

The Mercer theorem (see, for example, Cucker and Smale, 2002; Hochstadt, 1973; Mercer, 1909; Sun, 2005) in the theory of reproducing kernels indicates that (59) represents a large class of kernels. Hilbert-Schmidt kernels are a key element of recent studies Opfer (2006) and Rakotomamonjy and Canu (2005).

The support of a function a on $\mathbb{N}$, denoted as $\operatorname{supp} a$, is the set of $n \in \mathbb{N}$ for which $a_n \neq 0$. Let a,b be two nonnegative functions on $\mathbb{N}$. Set $c := \max\{a,b\}$ and assume that the sequence ϕ_n satisfies for each $x \in X$ that $\Phi(x) \in \ell_c^2(\mathbb{N})$ and

$$\overline{\operatorname{span}}\{\Phi(x) : x \in X\} = \ell_c^2(\mathbb{N}).$$

We shall consider the inclusion $\mathcal{H}_{K_a} \preceq \mathcal{H}_{K_b}$. For this purpose, we make the convention that whenever we write $a \preceq b$ for two functions a, b on $\mathbb{N}$, it means that $\operatorname{supp} a \subseteq \operatorname{supp} b$ and $a_n = b_n$ for each $n \in \operatorname{supp} a$.

Theorem 29 *There holds $\mathcal{H}_{K_a} \preceq \mathcal{H}_{K_b}$ if and only if $a \preceq b$. Moreover, if $a \preceq b$ then K_b is a nontrivial refinement kernel for K_a if and only if* $\operatorname{supp} a$ *is a proper subset of* $\operatorname{supp} b$.

Proof This theorem is proved by using Theorem 8 with an identification of measures μ and ν. We introduce three nonnegative functions $\tilde{a}, \tilde{b}, \tilde{c}$ in $\ell^1(\mathbb{N})$ by setting for $n \in \mathbb{N}$

$$\tilde{a}_n := \begin{cases} \frac{a_n}{n^2 c_n}, & n \in \operatorname{supp} a, \\ 0, & \text{otherwise}, \end{cases} \quad \tilde{b}_n := \begin{cases} \frac{b_n}{n^2 c_n}, & n \in \operatorname{supp} b, \\ 0, & \text{otherwise}, \end{cases}$$

and

$$\tilde{c}_n := \begin{cases} \frac{1}{n^2}, & n \in \operatorname{supp} c, \\ 0, & \text{otherwise}. \end{cases}$$

We also define a function $\phi : X \times \mathbb{N} \to \mathbb{C}$ by

$$\phi(x,n) := n\sqrt{c_n}\phi_n(x), \ \ x \in X, \ n \in \mathbb{N}.$$

It is observed that $\tilde{c} = (\tilde{a}+\tilde{b})/2 + |\tilde{a}-\tilde{b}|/2$, $\phi(x,\cdot) \in \ell^2_{\tilde{c}}(\mathbb{N})$ for all $x \in X$ and $\operatorname{span}\{\phi(x,\cdot) : x \in X\}$ is dense in $\ell^2_{\tilde{c}}(\mathbb{N})$. Moreover,

$$K_a(x,y) = (\phi(x,\cdot), \phi(y,\cdot))_{\ell^2_{\tilde{a}}(\mathbb{N})}, \ K_b(x,y) = (\phi(x,\cdot), \phi(y,\cdot))_{\ell^2_{\tilde{b}}(\mathbb{N})}, \ \ x, y \in X.$$

Let $Y := \mathbb{N}$. We identify measures $\mu, \nu \in \mathcal{B}(Y)$ with $\tilde{a}$ and $\tilde{b}$, respectively, such that

$$K_a(x,y) = \int_Y \phi(x,\xi)\overline{\phi(y,\xi)}d\mu(\xi), \ \ K_b(x,y) = \int_Y \phi(x,\xi)\overline{\phi(y,\xi)}d\nu(\xi), \ \ x, y \in X.$$

Note that $\mu \preceq \nu$ is equivalent to $a \preceq b$. The result of this theorem now follows immediately from Theorem 8. ■

One can see that Proposition 26 may be viewed as a corollary of Theorem 29. We next present two more concrete applications of Theorem 29.

For the first example, we assume $R \in (0, +\infty]$ and specify our input space X to be $\{z \in \mathbb{C} : |z| < R^{1/2}\}$. Here we make the convention that if $R = +\infty$ then $X := \mathbb{C}$. For two nonnegative functions a, b defined on $\mathbb{N}$ satisfying

$$\max\left\{\limsup_{n\to\infty} \sqrt[n]{a_n}, \ \limsup_{n\to\infty} \sqrt[n]{b_n}\right\} \leq \frac{1}{R}, \tag{60}$$

we define the kernels

$$\mathcal{K}_a(\xi,\eta) := \sum_{n\in\mathbb{N}} a_n \xi^{n-1}\overline{\eta}^{n-1}, \ \mathcal{K}_b(\xi,\eta) := \sum_{n\in\mathbb{N}} b_n \xi^{n-1}\overline{\eta}^{n-1}, \ \ \xi, \eta \in X. \tag{61}$$

Classical kernels such as the Bergman kernels and the Szegö kernels (see, for example, Saitoh, 1988) have the above form.

Proposition 30 *Suppose that a, b are nonnegative functions defined on $\mathbb{N}$ satisfying (60) and kernels $\mathcal{K}_a, \mathcal{K}_b$ are defined in (61). Then $\mathcal{H}_{\mathcal{K}_a} \preceq \mathcal{H}_{\mathcal{K}_b}$ if and only if $a \preceq b$, and the refinement is nontrivial if and only if* $\operatorname{supp} a$ *is a proper subset of* $\operatorname{supp} b$.

Proof We define a sequence of functions $\phi_n, n \in \mathbb{N}$ on the input space X by setting

$$\phi_n(\xi) := \xi^{n-1}, \ \ \xi \in X.$$

Let $c := \max\{a, b\}$. Condition (60) ensures that Φ defined by (58) with the ϕ_n defined above satisfies the condition that $\Phi(\xi) \in \ell^2_c(\mathbb{N})$ for each $\xi \in X$. It is clear that $\operatorname{span}\{\Phi(\xi) : \xi \in X\}$ is dense in $\ell^2_c(\mathbb{N})$. The result of this proposition follows directly from Theorem 29. ■

Our second example concerns the *Schoenberg kernels* (Schoenberg, 1942) on the unit sphere $\mathbb{S}^d$ in $\mathbb{R}^{d+1}$. We shall need the ultraspherical polynomials $P^d_n, n \in \mathbb{Z}_+$. When $d = 1, P^1_n$ is the Chebyshev polynomial of degree n (Rivlin, 1990) and for $d > 1, P^d_n$ is determined by

$$\frac{1}{(1-2zt+z^2)^{(d-1)/2}} = \sum_{n \in \mathbb{Z}_+} P^d_n(t) z^n, \ \ |z| < 1, \ t \in [-1, 1].$$

For a nonnegative function h defined on $\mathbb{N}$ satisfying the conditions

$$\sum_{n \in \mathbb{N}} h_n P^d_{n-1}(1) < +\infty, \tag{62}$$

we introduce a Schoenberg kernel on $\mathbb{S}^d$ by setting

$$\mathcal{S}_h(x, y) := \sum_{n \in \mathbb{N}} h_n P^d_{n-1}((x, y)), \ \ x, y \in \mathbb{S}^d, \tag{63}$$

where $(\cdot, \cdot)$ denotes the inner product on $\mathbb{R}^{d+1}$.

Theorem 31 *Suppose that a and b are two nonnegative functions defined on $\mathbb{N}$ satisfying the condition (62) and $\mathcal{S}_a$ and $\mathcal{S}_b$ are the corresponding Schoenberg kernels on $\mathbb{S}^d$ defined as in (63). Then, $\mathcal{H}_{\mathcal{S}_a} \preceq \mathcal{H}_{\mathcal{S}_b}$ if and only if $a \preceq b$. If $a \preceq b$ then $\mathcal{S}_b$ is a nontrivial refinement kernel for $\mathcal{S}_a$ if and only if* $\operatorname{supp} a$ *is a proper subset of* $\operatorname{supp} b$.

Proof We shall write the kernels $\mathcal{S}_a, \mathcal{S}_b$ in the form of Hilbert-Schmidt kernels and then apply Theorem 29. To this end, we recall some basic facts of *spherical harmonics* (Stein and Weiss, 1971). For each $n \in \mathbb{Z}_+$ we let $\mathcal{H}_n$ be the set of all homogeneous harmonic polynomials of total degree n on $\mathbb{R}^{d+1}$ restricted to $\mathbb{S}^d$. We consider $\mathcal{H}_n$ as a subspace of $L^2(\mathbb{S}^d, \omega)$ where ω is the Lebesgue measure on $\mathbb{S}^d$. Let d_n denote the dimension of $\mathcal{H}_n$ and $\{Y^n_j : j \in \mathbb{N}_{d_n}\}$ an orthonormal basis for $\mathcal{H}_n$. If $n \neq n'$ then $\mathcal{H}_n$ is orthogonal to $\mathcal{H}_{n'}$ (Stein and Weiss, 1971). For each $n \in \mathbb{Z}_+$, there exists a positive constant c_n such that

$$P^d_n((x, y)) = c_n \sum_{j \in \mathbb{N}_{d_n}} Y^n_j(x) Y^n_j(y), \ \ x, y \in \mathbb{S}^d. \tag{64}$$

By Equations (63) and (64), we have that

$$\mathcal{S}_a(x, y) = \sum_{n \in \mathbb{N}} a_n c_{n-1} \sum_{j \in \mathbb{N}_{d_{n-1}}} Y^{n-1}_j(x) Y^{n-1}_j(y), \ \ x, y \in \mathbb{S}^d$$

and

$$\mathcal{S}_b(x,y) = \sum_{n\in\mathbb{N}} b_n c_{n-1} \sum_{j\in\mathbb{N}_{d_{n-1}}} Y_j^{n-1}(x) Y_j^{n-1}(y), \ \ x,y\in\mathbb{S}^d.$$

The result of this theorem hence follows immediately from Theorem 29 and the orthogonality of Y_j^n. ■

We now return to general Hilbert-Schmidt kernels defined by a sequence of functions ϕ_n on X and investigate the case when ϕ_n's are coupled. We shall work in the Hilbert space $\ell^2(\mathbb{N})$ under the assumption that $\Phi(x)\in\ell^2(\mathbb{N})$ for each $x\in X$ and

$$\overline{\operatorname{span}}\{\Phi(x): x\in X\} = \ell^2(\mathbb{N}). \tag{65}$$

For each bounded, positive, and self-adjoint linear operator C on $\ell^2(\mathbb{N})$, we denote by $\ell^2_C(\mathbb{N})$ the Hilbert space completed upon the linear space $\ell^2(\mathbb{N})$ under the inner product

$$(u,v)_{\ell^2_C(\mathbb{N})} := (Cu,v)_{\ell^2(\mathbb{N})}, \ \ u,v\in\ell^2(\mathbb{N}).$$

Note that $\ell^2_C(\mathbb{N})$ is a Hilbert space of equivalent classes. In other words, two elements $u,v\in\ell^2(\mathbb{N})$ are identical in $\ell^2_C(\mathbb{N})$ if and only if $u-v\in\ker C := \{x\in\ell^2(\mathbb{N}): Cx=0\}$. For two bounded, positive and self-adjoint linear operators A,B from $\ell^2(\mathbb{N})$ to itself, we introduce two kernels by setting

$$K_A(x,y) := (\Phi(x),\Phi(y))_{\ell^2_A(\mathbb{N})}, \ K_B(x,y) := (\Phi(x),\Phi(y))_{\ell^2_B(\mathbb{N})}, \ \ x,y\in X.$$

Before delving into conditions equivalent to $\mathcal{H}_{K_A}\preceq\mathcal{H}_{K_B}$, we review necessary results from functional analysis. For each bounded, positive and self-adjoint linear operator C on $\ell^2(\mathbb{N})$, we let $P_{C,\perp}$ denote the orthogonal projection from $\ell^2(\mathbb{N})$ to the orthogonal complement $(\ker C)^\perp$ of $\ker C$. Note that $u\in\ell^2(\mathbb{N})$ satisfies $(Cu,u)_{\ell^2(\mathbb{N})}=0$ if and only if $u\in\ker C$. Moreover, for each $u\in\ell^2(\mathbb{N})$ there holds that

$$(Cu,u)_{\ell^2(\mathbb{N})} = (CP_{C,\perp}u, P_{C,\perp}u)_{\ell^2(\mathbb{N})}.$$

Thus C is a bijective mapping from $(\ker C)^\perp$ to $\operatorname{ran} C$. We denote by $\tilde{C}^{-1}$ its inverse from $\operatorname{ran} C$ to $(\ker C)^\perp$.

Our characterization of $\mathcal{H}_{K_A}\preceq\mathcal{H}_{K_B}$ is as follows.

Theorem 32 *Suppose that A,B are bounded, positive and self-adjoint linear operators from $\ell^2(\mathbb{N})$ to itself. Then, $\mathcal{H}_{K_A}\preceq\mathcal{H}_{K_B}$ if and only if $\operatorname{ran} A\subseteq\operatorname{ran} B$ and for each $v\in\operatorname{ran} A$,*

$$P_{A,\perp}\tilde{B}^{-1}v = \tilde{A}^{-1}v. \tag{66}$$

Proof By Lemma 5, $\mathcal{H}_{K_A}\preceq\mathcal{H}_{K_B}$ if and only if for each $u\in\ell^2(\mathbb{N})$ there exists $v\in\ell^2(\mathbb{N})$ such that

$$(\Phi(x),Au)_{\ell^2(\mathbb{N})} = (\Phi(x),Bv)_{\ell^2(\mathbb{N})}, \ \ x\in X \tag{67}$$

and

$$(Au,u)_{\ell^2(\mathbb{N})} = (Bv,v)_{\ell^2(\mathbb{N})}. \tag{68}$$

By the density assumption (65), (67) is equivalent to that

$$Au = Bv. \tag{69}$$

Suppose that $\mathcal{H}_{K_A} \preceq \mathcal{H}_{K_B}$. Then for each $u \in \ell^2(\mathbb{N})$ there exists $v \in \ell^2(\mathbb{N})$ satisfying (68) and (69). By (69), we have that $\operatorname{ran} A \subseteq \operatorname{ran} B$. Let $u \in (\ker A)^\perp$. Then v in (69) can be taken as $\tilde{B}^{-1}Au$. Similarly, if we choose $u' \in (\ker A)^\perp$ and let $v' := \tilde{B}^{-1}Au'$ then (67) and (69) hold true with u, v replaced by u', v'. Since $\mathcal{H}_{K_A}$ is a subspace of $\mathcal{H}_{K_B}$, we have

$$\begin{aligned}(Au', u)_{\ell^2(\mathbb{N})} &= ((\Phi(x), Au)_{\ell^2(\mathbb{N})}, (\Phi(x), Au')_{\ell^2(\mathbb{N})})_{\mathcal{H}_{K_A}} \\ &= ((\Phi(x), Bv)_{\ell^2(\mathbb{N})}, (\Phi(x), Bv')_{\ell^2(\mathbb{N})})_{\mathcal{H}_{K_B}} \\ &= (Bv', v)_{\ell^2(\mathbb{N})}.\end{aligned}$$

By the above equation and (69), we get that

$$(Au', P_{A,\perp}\tilde{B}^{-1}Au)_{\ell^2(\mathbb{N})} = (Au', \tilde{B}^{-1}Au)_{\ell^2(\mathbb{N})} = (Bv', v)_{\ell^2(\mathbb{N})} = (Au', u)_{\ell^2(\mathbb{N})}.$$

Note that the above equation is true for all $u, u' \in (\ker A)^\perp$. Thus there must hold

$$P_{A,\perp}\tilde{B}^{-1}Au = u = \tilde{A}^{-1}Au.$$

Since A is surjective from $(\ker A)^\perp$ onto $\operatorname{ran} A$, we obtain (66) for each $v \in \operatorname{ran} A$.

Conversely, suppose that $\operatorname{ran} A \subseteq \operatorname{ran} B$ and there holds for each $v \in \operatorname{ran} A$ Equation (66). Therefore, for each $u \in (\ker A)^\perp$ there exists $v \in \ell^2(\mathbb{N})$ satisfying (69). Moreover, we calculate by (66) that

$$\begin{aligned}(Bv, v)_{\ell^2(\mathbb{N})} &= (Bv, P_{B,\perp}v)_{\ell^2(\mathbb{N})} = (Bv, \tilde{B}^{-1}Au)_{\ell^2(\mathbb{N})} = (Au, \tilde{B}^{-1}Au)_{\ell^2(\mathbb{N})} \\ &= (Au, P_{A,\perp}\tilde{B}^{-1}Au)_{\ell^2(\mathbb{N})} = (Au, \tilde{A}^{-1}Au)_{\ell^2(\mathbb{N})} = (Au, u)_{\ell^2(\mathbb{N})}.\end{aligned}$$

Thus (68) holds true. We hence prove that $\mathcal{H}_{K_A} \preceq \mathcal{H}_{K_B}$. ■

To close this section, as an application of Theorem 32, we consider kernels of finite dimensional feature spaces. Let $n \le m$ be two positive integers and A, B hermitian and strictly positive definite matrices of sizes $n \times n$ and $m \times m$, respectively. Suppose that ϕ_j, $j \in \mathbb{N}_m$ form a sequence of linearly independent functions on X. The kernels we consider are

$$G_A(x,y) := \sum_{j,k \in \mathbb{N}_n} A_{jk}\phi_k(x)\overline{\phi_j(y)},\ G_B(x,y) := \sum_{j,k \in \mathbb{N}_m} B_{jk}\phi_k(x)\overline{\phi_j(y)},\ \ x, y \in X. \tag{70}$$

We remark that a kernel has a finite dimensional feature space if and only if its RKHS has finite dimension. It was proven in Aronszajn (1950) that a RKHS is finite dimensional if and only if its reproducing kernel has the form of (70). The following corollary is a direct consequence of Theorem 32.

Corollary 33 *Let kernels G_A, G_B be defined by (70). Then $\mathcal{H}_{G_A} \preceq \mathcal{H}_{G_B}$ if and only if B^{-1} is an augmentation of A^{-1}, namely, $B^{-1}_{jk} = A^{-1}_{jk}$, $j, k \in \mathbb{N}_n$. In particular, if G_A, G_B have the form*

$$G_A(x,y) := \sum_{j \in \mathbb{N}_n} a_j\phi_j(x)\overline{\phi_j(y)},\ G_B(x,y) := \sum_{k \in \mathbb{N}_m} b_k\phi_k(x)\overline{\phi_k(y)},\ \ x, y \in X$$

for some positive constants a_j, b_k then $\mathcal{H}_{G_A} \preceq \mathcal{H}_{G_B}$ if and only if $a_j = b_j$ for each $j \in \mathbb{N}_n$. In both cases, if $\mathcal{H}_{G_A} \preceq \mathcal{H}_{G_B}$ then G_B is a nontrivial refinement kernel for G_A if and only if $m > n$.

Acknowledgments

Yuesheng Xu was supported in part by the US National Science Foundation under grants CCR-0407476 and DMS-0712827, by the Natural Science Foundation of China under grants 10371122 and 10631080, by the Education Ministry of the People's Republic of China under the Changjiang Scholar Chair Professorship Program through Sun Yat-sen University, and by Institute of Mathematics, Academy of Mathematics and System Science, Chinese Academy of Sciences.

References

N. Aronszajn. Theory of reproducing kernels. *Trans. Amer. Math. Soc.*, 68:337–404, 1950.

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.

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.

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.

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.

C. Gasquet and P. Witomski. *Fourier Analysis and Applications*. Springer-Verlag, New York, 1999.

G. H. Golub and C. F. van Loan. *Matrix Computations*. 3rd Edition, Johns Hopkins University Press, Baltimore, MD, 1996.

T. N. T. Goodman and S. L. Lee. Wavelets of multiplicity r. *Trans. Amer. Math. Soc.*, 342:307–324, 1994.

L. Grafakos. *Classical and Modern Fourier Analysis*. Prentice Hall, New Jersey, 2004.

H. Hochstadt. *Integral Equations*. Wiley, New York, 1973.

R. Q. Jia. Characterization of smoothness of multivariate refinable functions in Sobolev spaces. *Trans. Amer. Math. Soc.*, 351:4089–4112, 1999.

R. Q. Jia, S. D. Riemenschneider and D. X. Zhou. Smoothness of multiple refinable functions and multiple wavelets. *SIAM J. Matrix Anal. Appl.*, 21:1–28, 1999.

G. Kimeldorf and G. Wahba. Some results on Tchebycheffian spline functions. *J. Math. Anal. Appl.*, 33:82–95, 1971.

J. Mercer. Functions of positive and negative type and their connection with the theory of integral equations. *Philos. Trans. R. Soc. Lond. Ser. A Math. Phys. Eng. Sci.*, 209: 415–446, 1909.

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 and T. Sauer. Regularity of multiwavelets. *Adv. Comput. Math.*, 7:455–545, 1997.

C. A. Micchelli and Y. Xu. Using the matrix refinement equation for the construction of wavelets on invariant sets. *Appl. Comput. Harmon. Anal.*, 1:391–401, 1994.

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.

J. R. Munkres. *Topology*. 2nd Edition, Prentice Hall, Upper Saddle River, New Jersey, 2000.

R. Opfer. Multiscale kernels. *Adv. Comput. Math.*, 25: 357–380, 2006.

A. Rakotomamonjy and S. Canu. Frames, reproducing kernels, regularization and learning. *Journal of Machine Learning Research*, 6:1485–1515, 2005.

T. J. Rivlin. *Chebyshev Polynomials*. 2nd Edition, Wiley, New York, 1990.

W. Rudin. *Real and Complex Analysis*. 3rd Edition, McGraw-Hill, New York, 1987.

S. Saitoh. *Theory of Reproducing Kernels and its Applications*. Pitman Research Note in Mathematics Series 189, Longman, UK, 1988.

I. J. Schoenberg. Metric spaces and completely monotone functions. *Ann. of Math. (2)*, 39:811–841, 1938.

I. J. Schoenberg. Positive definite functions on spheres. *Duke. Math. J.*, 9: 96–108, 1942.

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.

E. M. Stein and G. Weiss. *Introduction to Fourier Analysis on Euclidean Spaces*. Princeton University Press, New Jersey, 1971.

I. Steinwart. On the influence of the kernel on the consistency of support vector machines. *Journal of Machine Learning Research*, 2:67–93, 2001.

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.

H. Sun. Mercer theorem for RKHS on noncompact sets. *J. Complexity*, 21:337–349, 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, B. Schölkopf and O. Chapelle. Implicit surface modelling with a globally regularised basis of compact support. *Computer Graphics Forum*, 25:635–644, 2006.

Y. Wang. Wavelets, tiling, and spectral sets. *Duke Math. J.*, 114:43–57, 2002.

D. V. Widder. *The Laplace Transform*. Princeton University Press, Princeton, 1941.

Y. Xu and H. Zhang. Refinable kernels. *Journal of Machine Learning Research*, 8:2083–2120, 2007.

Y. Ying and D. X. Zhou. Learnability of Gaussians with flexible variances. *Journal of Machine Learning Research*, 8:249–276, 2007.

T. Zhang. Statistical behavior and consistency of classification methods based on convex risk minimization. *Ann. Statis.*, 32:56–85, 2004.

Journal of Machine Learning Research 10 (2009) 141-158 Submitted 7/07; Revised 7/08; Published 2/09

Subgroup Analysis via Recursive Partitioning

Xiaogang Su XSU@PEGASUS.CC.UCF.EDU
Department of Statistics and Actuarial Science
University of Central Florida
Orlando, FL 32816, USA

Chih-Ling Tsai CLTSAI@UCDAVIS.EDU
Graduate School of Management
University of California, Davis
Davis, CA 95616, USA

Hansheng Wang HANSHENG@GSM.PKU.EDU.CN
Guanghua School of Management
Peking University
Beijing 100871, P. R. China

David M. Nickerson NICKERSN@MAIL.UCF.EDU
Department of Statistics and Actuarial Science,
University of Central Florida
Orlando, FL 32816, USA

Bogong Li LI.TIMOTHY@BLS.GOV
Office of Survey Methods Research
U.S. Bureau of Labor Statistics
Washington, DC 20212, USA

Editor: Saharon Rosset

Abstract

Subgroup analysis is an integral part of comparative analysis where assessing the treatment effect on a response is of central interest. Its goal is to determine the heterogeneity of the treatment effect across subpopulations. In this paper, we adapt the idea of recursive partitioning and introduce an interaction tree (IT) procedure to conduct subgroup analysis. The IT procedure automatically facilitates a number of objectively defined subgroups, in some of which the treatment effect is found prominent while in others the treatment has a negligible or even negative effect. The standard CART (Breiman et al., 1984) methodology is inherited to construct the tree structure. Also, in order to extract factors that contribute to the heterogeneity of the treatment effect, variable importance measure is made available via random forests of the interaction trees. Both simulated experiments and analysis of census wage data are presented for illustration.

Keywords: CART, interaction, subgroup analysis, random forests

1. Introduction

In comparative studies where two or more treatments are compared, subgroup analysis emerges after the overall assessment of the treatment effect and plays an important role in determining whether and how the treatment effect (i.e., comparison among treatments) varies across subgroups induced by covariates. In designed clinical trials, for example, practitioners and regulatory agencies are keen to know if there are subgroups of trial participants who are more (or less) likely to be helped (or harmed) by the intervention under investigation. Subgroup analysis helps explore the heterogeneity of the treatment effect and extract the maximum amount of information from the available data. According to a survey conducted by Assmann et al. (2000), 70% of trials published over a three-month period in four leading medical journals involved subgroup analyses.

The research questions in subgroup analysis can be either pre-planned or raised in a post-hoc manner. If there is a prior hypothesis of the treatment effect being different in particular subgroups, then this hypothesis and its assessment should be part of the planned study design (CPMP, 1995). At the same time, subgroup analysis is also often used for post-hoc exploratory identification of unusual or unexpected results (Chow and Liu, 2004). Suppose, for example, that it is of interest to investigate whether the treatment effect is consistent among three age groups: young, middle-aged, and older individuals. The evaluation is formally approached by means of an interaction test between treatment and age. If the resulting test is significant, multiple comparisons are then used to find out further details about the magnitude and direction of the treatment effect within each age group. To ensure a valid experimentwise false positive rate, adjustment methods such as Bonferroni typed correction are often applied.

Limitations of traditional subgroup analysis have been extensively noted (see, e.g., Assmann et al., 2000, Sleight, 2000, and Lagakos, 2006). First of all, the subgroups themselves, as well as the number of subgroups to be examined, are specified by the investigator beforehand in the current practice of subgroup analysis, which renders subgroup analysis a highly subjective process. Even for the field expert, it is a daunting task to determine which specific subgroups should be used in subgroup analysis. The subjectivity may lead directly to dubious results and willful manipulations. For example, one may fail to identify a subgroup of great prospective interest or intentionally avoid reporting subgroups where the investigational treatment is found unsuccessful or even potentially harmful. Reliance on such analyses is likely to be erroneous and harmful. Moreover, significance testing is the main approach in subgroup analysis. Because there is no general guideline for selecting the number of subgroups, one has to examine numerous plausible possibilities to have a thorough assessment of the treatment effect. However, a large number of subgroups inevitably causes concerns related to multiplicity and lack of power.

In this paper, we propose a data-driven tree procedure, labelled as "interaction trees" (IT), to explore the heterogeneity structure of the treatment effect across a number of subgroups that are objectively defined in a post hoc manner. The tree method, also called recursive partitioning, was first proposed by Morgan and Sonquist (1963). By recursively bisecting the predictor space, the hierarchical tree structure partitions the data into meaningful groups and makes available a piecewise approximation of the underlying relationship between the response and its associated predictors. The applications of tree models have been greatly advanced in various fields especially since the development of CART (classification and regression trees) by Breiman et al. (1984). Their pruning idea for tree size selection has become and remains the current standard in constructing tree models.

The remainder of the paper is organized as follows. In Section 2, the IT procedure is presented in detail. Section 3 contains simulation studies designed for assessing the proposed method. In Section 4, we apply the IT procedure to analyze a wage data set, in which the goal is to determine whether or not women are underpaid or overpaid as compared to their male counterparts and, if so, by what amount. Section 5 concludes the paper with a brief discussion.

2. Tree-Structured Subgroup Analysis

We consider a study designed to assess a binary treatment effect on a continuous response or output while adjusting or controlling for a number of covariates. Suppose that the data available contain n i.i.d. observations $\{(y_i, \text{trt}_i, \mathbf{x}_i) : i = 1, \ldots, n\}$, where y_i is the continuous response; trt_i is the binary treatment indicator taking values 1 or 0; and $\mathbf{x}_i = (x_{i1}, \ldots, x_{ip})'$ is the associated p-dimensional covariate vector where its components can be of mixed types, that is, categorical or continuous. Our goal in subgroup analysis is to find out whether there exist subgroups of individuals in which the treatment shows heterogeneous effects, and if so, how the treatment effect varies across them.

Applying a tree procedure to guide the subgroup analysis is rather intuitive. First, subgroup analysis essentially involves the interaction between the treatment and the covariates. The tree algorithm is known as an excellent tool for exploring interactions. As a matter of fact, the first implementation of decision trees is referred to as Automatic Interaction Detection (AID; Morgan and Sonquist, 1963). However, tree methods handle interactions *implicitly*. It is often hard to determine whether or not interaction really exists among variables for a given tree structure. We shall seeks ways that enable us to *explicitly* assess the interaction between the treatment and the covariates. Secondly, the hierarchical binary tree structure naturally groups data in an optimal way. By recursively partitioning the data into two subgroups that show the greatest heterogeneity in treatment effect, we are able to optimize the subgroup analysis and make it more efficient in representing the heterogeneity structure of the treatment effect. Thirdly, the tree procedure is objective, data-driven, and automated. The grouping strategy and the number of subgroups are automatically determined by the procedure. The proposed method would result in a set of objectively recognized and mutually exclusive subgroups, ranking from the most effective to the least effective in terms of treatment effect. To the best of our knowledge, Negassa et al. (2005) and Su et al. (2008), who studied tree-structured subgroup analysis in the context of censored survival data, are the only previous works along similar lines to our proposal.

To construct the IT model, we follow the CART (Breiman et al., 1984) convention, which consists of three major steps: (1) growing a large initial tree; (2) a pruning algorithm; and (3) a validation method for determining the best tree size. Once a final tree structure is obtained, the subgroups are naturally induced by its terminal nodes. To achieve better efficiency, an amalgamation algorithm is used to merge terminal nodes that show homogenous treatment effects. In addition, we adopt the variable importance technique in the context of random forest to extract covariates that exhibit important interactions with the treatment.

2.1 Growing a Large Initial Tree

We start with a single split, say s, of the data. This split is induced by a threshold on a predictor X. If X is continuous, then the binary question whether $X \leq c$ is considered. Observations answering "yes" go to the left child node t_L and observations answering "no" go to the right child node t_R. If X is nominal with r distinct categories $C = \{c_1, \ldots, c_r\}$, then the binary question becomes "Is $X \in A$?" for any subset $A \subset C$. When X has many distinct categories, one may place them into order according to the treatment effect estimate within each category and then proceed as if X is ordinal. This strategy helps reduce the computational burden. A theoretical justification is provided by Theorem 1 in the appendix.

For a given node t, a split s yields the following 2×2 table:

	child node							
treatment	t_L				t_R			
1	μ_1^L	$\bar{y}_1^L$	s_1^2	n_1	μ_1^R	$\bar{y}_1^R$	s_2^2	n_2
0	μ_0^L	$\bar{y}_0^L$	s_3^2	n_3	μ_0^R	$\bar{y}_0^R$	s_4^2	n_4

Here, $\{\mu_1^L, \bar{y}_1^L, s_1^2, n_1\}$ are the population mean, the sample mean, the sample variance, and the sample size for the treatment 1 group in the left child node t_L, respectively. Similar notation applies to the other quantities. To evaluate the heterogeneity of the treatment effect between t_L and t_R, we compare $(\mu_1^L - \mu_0^L)$ with $(\mu_1^R - \mu_0^R)$ so that the interaction between split s and treatment is under investigation.

A natural measure for assessing the interaction is given by

$$t(s) = \frac{(\bar{y}_1^L - \bar{y}_0^L) - (\bar{y}_1^R - \bar{y}_0^R)}{\hat{\sigma} \cdot \sqrt{1/n_1 + 1/n_2 + 1/n_3 + 1/n_4}}, \tag{1}$$

where $\hat{\sigma}^2 = \sum_{i=1}^4 w_i s_i^2$ is a pooled estimator of the constant variance, and $w_i = (n_i - 1)/\sum_{j=1}^4 (n_j - 1)$. For a given split s, $G(s) = t^2(s)$ converges to a $\chi^2(1)$ distribution. We will use $G(s)$ as the splitting statistic, while remaining aware that the sign of $t(s)$ supplies useful information regarding the direction of the comparison.

The best split $s^\star$ is the one that yields the maximum G statistic among all permissible splits. That is,

$$G(s^\star) = \max_s G(s).$$

The data in node t are then split according to the best split $s^\star$. The same procedure is applied to split both child nodes: t_L and t_R. Recursively doing so results in a large initial tree, denoted by T_0.

Remark 1: It can be easily seen that $t(s)$ given in (1) is equivalent to the t test for testing $H_0 : \gamma = 0$ in the following threshold model

$$y_i = \beta_0 + \beta_1 \cdot \text{trt}_i + \delta \cdot z_i^{(s)} + \gamma \cdot \text{trt}_i \cdot z_i^{(s)} + \varepsilon_i, \tag{2}$$

where $z_i^{(s)} = 1_{\{X_i \le c\}}$ is the binary variable associated with split s. This observation sheds light on extensions of the proposed method to situations involving different types of responses and multi-level treatments. Note that inclusion of both trt_i and $z_i^{(s)}$ in model (2) is important to assure the correspondence between the regression coefficient γ and the interaction, that is, $\gamma = (\mu_1^L - \mu_0^L) - (\mu_1^R - \mu_0^R)$.

Remark 2: In constructing the large initial tree T_0, a terminal node is declared when any one of the following conditions is met: the node gets pure in the sense that all the covariates have the same values; the total number of observations in the node is less than some preset minimum node size; the depth of the node is greater than some preset maximum tree depth; for all permissible splits, the minimum of (n_1, n_2, n_3, n_4) is below some preset threshold. The last condition is particularly useful in preventing the "end-cut preference" problem as discussed in CART (Breiman et al., 1984, pp. 313–317). For this purpose, the threshold can be set adaptively with respect to the depth of the node, following the suggestion of Torgo (2001).

2.2 Pruning

The final tree could be any subtree of T_0. To narrow down the choices, Breiman et al. (1984) proposed a pruning algorithm which results in a sequence of optimally pruned subtrees by iteratively truncating the "weakest link" of T_0. In the following, we briefly describe an interaction-complexity pruning procedure, which is analogous to the split-complexity pruning algorithm in LeBlanc and Crowley (1993). We refer the reader to their paper for a detailed description.

To evaluate the performance of an interaction tree T, we define an interaction-complexity measure:

$$G_\lambda(T) = G(T) - \lambda \cdot \mid T - \widetilde{T} \mid, \tag{3}$$

where $G(T) = \sum_{h \in T - \widetilde{T}} G(h)$ measures the overall amount of interaction in T; the total number of internal nodes $|T - \widetilde{T}|$ corresponds to the complexity of the tree; and the complexity parameter $\lambda \geq 0$ acts as a penalty for each added split. Given a fixed λ, a tree with larger $G_\lambda(T)$ is preferable.

Start with T_0. For any internal node h of T_0, calculate $g(h) = G(T_h)/ \mid T_h - \widetilde{T_h} \mid$, where T_h is the branch with h as root and $|T_h - \widetilde{T_h}|$ denotes the number of internal nodes of T_h. Then, the node $h^\star$ with smallest $g(h^\star)$ is the "weakest link," in the sense that $h^\star$ becomes ineffective first as λ increases. Next, let T_1 be the subtree after pruning off the branch $T_{h^\star}$ from T_0, and subsequently apply the same procedure to prune T_1. Repeating this procedure yields a nested sequence of subtrees $T_M \prec \cdots \prec T_m \prec T_{m-1} \prec \cdots \prec T_1 \prec T_0$, where T_M is the null tree with the root node only and $\prec$ means "is a subtree of".

2.3 Selecting the Best-Sized Subtree

The final tree will be selected from the nested subtree sequence $\{T_m : m = 0, 1, \ldots, M\}$, again, based on the maximum interaction-complexity measure $G_\lambda(T_m)$ given in (3). For tree size determination purposes, λ is suggested to be fixed within the range $2 \leq \lambda \leq 4$, where $\lambda = 2$ is in the spirit of the Akaike information criterion (AIC; Akaike, 1973) and $\lambda = 4$ corresponds roughly to the 0.05 significance level on the $\chi^2(1)$ curve. Another choice is $\lambda = \ln(n)$, citing the Bayesian information criterion (BIC; Schwarz, 1978).

To overcome the over-optimism due to greedy search, an "honest" estimate of the goodness-of-split measure $G(T_m)$ is needed. This can be achieved by a validation method. If the sample size is large, $G(T_m)$ can be recalculated using an independent subset of the data (termed the validation sample). If the sample size is small or moderate, one has to resort to techniques such as v-fold cross-validation or bootstrapping methods in order to validate $G(T_m)$. The bootstrap method used in LeBlanc and Crowley (1993), for example, can be readily adapted to interaction trees.

Remark 3: As another alternative, one referee suggested the use of in-sample BIC to determine the optimally-sized subtree, which can be promising, although further research effort is needed to determine the effective degrees of freedom (see, e.g., Ye, 1998, and Tibshirani and Knight, 1999) associated with interaction trees.

2.4 Summarizing the Terminal Nodes

The total number of subgroups, which could be further reduced, corresponds to the automatically selected best tree size. Existence of the overall interaction between the treatment and the covariates can be roughly assessed by inspecting whether a null final tree structure is obtained. Unlike conventional subgroup analysis, the subgroups obtained from the IT procedure are mutually exclusive. Due to the subjectivity in determining the subgroups and multiplicity emerging from significance testings across subgroups, it is generally agreed that subgroup analysis should be regarded as exploratory and hypothesis-generating. As recommended by Lagakos (2006), it is best not to present p-values for within-subgroup comparisons, but rather to give an estimate of the magnitude of the treatment difference. To summarize the terminal nodes, one can compute the average responses for both treatments within each terminal node, as well as their relative differences or ratios and the associated standard errors. If, however, one would like to perform some formal statistical testings, then it is important to conduct these tests based on yet another independent sample. To do so, one partitions the data into three sets: the learning sample $\mathcal{L}_1$, the validation sample $\mathcal{L}_2$, and the test sample $\mathcal{L}_3$. The best IT structure will be developed using $\mathcal{L}_1$ and $\mathcal{L}_2$, and then reconfirmed using the test sample $\mathcal{L}_3$.

It happens often that the treatment shows homogeneous effects for entirely different causal reasons in terminal nodes stemming from different branches. In this case, a merging scheme analogous to the approach of Ciampi et al. (1986) can be useful to further bring down the number of final subgroups. A smaller number of subgroups are easier to summarize and comprehend and more efficient to represent the heterogeneity structure for the treatment effect. It also ameliorates the multiplicity issue in within-subgroup comparisons. In this scheme, one computes the t statistic in equation (1) between every pair of the terminal nodes. The pair showing the least heterogeneity in treatment effect are then merged together. The same procedure is executed iteratively until all the remaining subgroups display outstanding heterogeneity. In the end, one can sort the final subgroups based on the strength of the treatment effect, from the most effective to the least effective.

Remark 4: The hierarchical tree structure is appealing mainly because of its easy interpretability. Apparently the merging scheme would invalidate the tree structure, yet lead to better interpretations. It is noteworthy that this merging scheme contributes additional optimism to the results. Thus, we suggest that amalgamation be executed with data $(\mathcal{L}_1 + \mathcal{L}_2)$ that pool the learning sample $\mathcal{L}_1$ and the validation sample $\mathcal{L}_2$ together, so that the resultant subgroups can be validated using the test sample $\mathcal{L}_3$. We shall illustrate this scheme with an example presented in Section 4.

2.5 Variable Importance Measure via Random Forests

Variable importance measure is another attractive feature offered by recursive partitioning. In the context of subgroup analysis, a covariate is called an *effect-modifier* of the treatment if it strongly interacts with the treatment. Variable importance measure helps answer questions such as which features or predictors are important in modifying the treatment effect. This issue cannot be fully addressed by simply examining the splitting variables shown in a single final IT structure, as an important variable can be completely masked by other correlated ones. While there are many methods available for extracting variable importance information, we propose an algorithm analogous to the procedure used in random forests (Breiman, 2001), which is among the newest and most promising developments in this regards.

Algorithm 1: Computing Variable Importance Measure via Random Forests.

```
Initialize all $V_j$'s to 0.
For $b = 1, 2, \ldots, B$, do
    • Generate bootstrap sample $\mathcal{L}_b$ and obtain the out-of-bag sample $\mathcal{L} - \mathcal{L}_b$.
    • Based on $\mathcal{L}_b$, grow a large IT tree $T_b$ by searching over $m_0$ randomly selected
    covariates at each split.
    • Send $\mathcal{L} - \mathcal{L}_b$ down $T_b$ to compute $G(T_b)$.
    • For all covariates $X_j$, $j = 1, \ldots, p$, do
        ◦ Permute the values of $X_j$ in $\mathcal{L} - \mathcal{L}_b$.
        ◦ Send the permuted $\mathcal{L} - \mathcal{L}_b$ down to $T_b$ to compute $G_j(T_b)$.
        ◦ Update $V_j \leftarrow V_j + \frac{G(T_b) - G_j(T_b)}{G(T_b)}$.
    • End do.
End do.
Average $V_j \leftarrow V_j / B$.
```

Let V_j denote the importance measure of the j-th covariate or feature X_j for $j = 1, \ldots, p$. We construct random forests of IT trees by taking B bootstrap samples $\mathcal{L}_b$, $b = 1, \ldots, B$. This is done by searching over only a subset of randomly selected m_0 covariates at each split. For each IT tree T_b, the b-th out-of-bag sample (denoted as $\mathcal{L} - \mathcal{L}_b$), which contains all observations that are not in $\mathcal{L}_b$, is sent down T_b to compute the interaction measure $G(T_b)$. Next, the values of the j-th covariate in $\mathcal{L} - \mathcal{L}_b$ are randomly permuted. The permuted out-of-bag sample is then sent down T_b to recompute $G_j(T_b)$. The relative difference between $G(T_b)$ and $G_j(T_b)$ is recorded. The procedure is repeated for B bootstrap samples. As a result, the importance measure V_j is the average of the relative differences over all B bootstrap samples. The whole procedure is summarized in Algorithm 1.

The variable importance technique in random forests has been increasingly studied in its own right and applied as a tool for variable selection in various fields. This method generally belongs to the "cost-of-exclusion" (Kononenko and Hong, 1997) feature selection category, in which the importance or relevance of a feature is determined by the difference in some model performance measure with and without the given feature included in the modeling process. In Algorithm 1, random forests make available a flexible modeling process while exclusion of a given feature is carried out by permutating its values.

3. Simulated Studies

This section contains simulated experiments designed to investigate the performance of the IT procedure. We generate data from six models outlined in Table 1. Each data set consists of a continuous response Y, a binary treatment, and four covariates X_1–X_4 simulated from a discrete uniform distribution over $(0.02, 0.04, \ldots, 1.00)$. However, only a subset of the covariates interact with the treatment.

Model	Form	Error Distribution
A	$Y = 2 + 2 \cdot \mathrm{trt} + 2Z_1 + 2Z_2 + \varepsilon$	$\mathcal{N}(0,1)$
B	$Y = 2 + 2 \cdot \mathrm{trt} + 2Z_1 + 2Z_2 + 2 \cdot \mathrm{trt} \cdot Z_1 Z_2 + \varepsilon$	$\mathcal{N}(0,1)$
C	$Y = 2 + 2 \cdot \mathrm{trt} + 2Z_1 + 2Z_2 + 2 \cdot \mathrm{trt} \cdot Z_1 + 2 \cdot \mathrm{trt} \cdot Z_2 + \varepsilon$	$\mathcal{N}(0,1)$
D	$Y = 10 + 10 \cdot \mathrm{trt} \cdot \exp\{(X_1 - 0.5)^2 + (X_2 - 0.5)^2\} + \varepsilon$	$\mathcal{N}(0,1)$
E	$Y = 2 + 2 \cdot \mathrm{trt} + 2Z_1 + 2Z_2 + 2 \cdot \mathrm{trt} \cdot Z_1 + 2 \cdot \mathrm{trt} \cdot Z_2 + \varepsilon$	$\mathrm{Unif}(-\sqrt{3}, \sqrt{3})$
F	$Y = 2 + 2 \cdot \mathrm{trt} + 2Z_1 + 2Z_2 + 2 \cdot \mathrm{trt} \cdot Z_1 + 2 \cdot \mathrm{trt} \cdot Z_2 + \varepsilon$	$\exp(1)$

Table 1: Models used for assessing the performance of the IT procedure. Note that $Z_1 = 1_{\{X_1 \leq 0.5\}}$ and $Z_2 = 1_{\{X_2 \leq 0.5\}}$.

Specifically, Model A is a plain additive model with no interaction. It helps assess the type I error or false positive rate when using the IT procedure. Model B involves a second-order interaction between the treatment and two terms of thresholds, both at 0.5, on X_1 and X_2. If the IT procedure works, a tree structure with three terminal nodes should be selected. In Model C, the two thresholds, each interacting with the treatment, are present in an additive manner. Model D involves interaction of complex forms other than cross-products. A large tree is expected in order to represent the interaction structure in this case. Models E and F are similar to Model C, but with different error distributions. They are useful in evaluating the robustness of the IT procedure to deviations from normality.

Only one set of sample sizes is reported: 800 observations in the learning sample $\mathcal{L}_1$ and 400 observations in the validation sample $\mathcal{L}_2$. Each model is examined for 200 simulation runs and four choices of λ, $\{2, 3, 4, \ln(400)\}$, are considered for determining the best tree structure. The relative frequencies of the final tree sizes selected by the IT procedure are presented in Table 2. The expected final tree size for each model is highlighted in boldface. Note that both X_1 and X_2, but neither X_3 nor X_4, are actually involved in models B-F. To address the variable selection issue, we count the frequency of "hits" (i.e., the final tree selected by the IT procedure is split by X_1 and X_2 and only by them). The results are presented in the last column of Table 2.

We first examine the results for Model A, which involves no interaction. The IT procedure correctly selects the null tree structure at least 83.5% of the time. When $\lambda = \ln(n)$, the percentage of correct selections becomes 98.5%. The 98.5% of correct selections yields an empirical size of 100%-98.5% = 1.5%, which is well within the acceptable level. This implies that the chance for the IT procedure to extract an unsolicited interactions is really small. For models B-F, the IT procedure also successfully identifies the true final tree structure and selects the desired splitting variables a majority of the time. Moreover, from results for models E and F, it seems rather robust against deviations from normality. When comparing different complexity parameters, we see that $\lambda = \ln(n)$ provides the best selection. This is mainly because of the large sample size and relatively strong signals considered in our simulation configuration (see, e.g., McQuarrie and Tsai, 1998).

To evaluate the proposed variable importance technique, we generate a data set containing 1,200 observations from each model. Then a total number of 500 random interaction trees with $m_0 = 1$ are used to compute the variable importance. In Figure 1, graphs A-I, B-I, ..., F-I display the resultant importance scores for models A-F, respectively. For comparison, we also apply a simple feature

	Complexity	Final Tree Size							
Model	λ	1	2	3	4	5	6	≥ 7	Hits
A	2	**83.5**	5.0	2.5	2.5	2.0	0.5	4.0	83.5
	3	**94.0**	3.5	0.5	0.5	1.0	0.5	0.0	94.0
	4	**97.5**	2.5	0.0	0.0	0.0	0.0	0.0	97.5
	$\ln(n)$	**98.5**	1.5	0.0	0.0	0.0	0.0	0.0	98.5
B	2	0.0	0.0	**67.0**	9.0	9.0	2.5	12.5	77.5
	3	0.0	0.0	**83.0**	6.5	5.5	1.0	4.0	90.0
	4	0.0	0.0	**89.0**	6.5	3.0	1.0	0.5	95.0
	$\ln(n)$	1.0	0.0	**91.5**	6.5	2.0	0.0	0.0	97.5
C	2	0.0	0.0	0.0	**66.5**	10.5	9.5	13.5	77.5
	3	0.0	0.0	0.0	**82.5**	7.5	6.0	4.0	90.5
	4	0.0	0.0	0.0	**88.0**	6.5	4.5	1.0	95.0
	$\ln(n)$	0.0	0.0	0.0	**94.0**	4.0	1.5	0.5	98.5
D	2	0.0	0.0	0.0	0.0	13.0	10.5	76.5	66.5
	3	0.0	0.0	0.0	0.5	24.5	16.0	59.0	83.5
	4	0.0	0.0	0.0	0.5	35.5	18.5	45.5	91.5
	$\ln(n)$	0.0	0.0	2.0	3.5	54.0	16.0	24.5	96.0
E	2	0.0	0.0	0.0	**74.0**	12.0	6.0	8.0	82.0
	3	0.0	0.0	0.0	**88.5**	7.5	3.5	0.5	93.0
	4	0.0	0.0	0.0	**93.5**	5.0	1.0	0.5	97.5
	$\ln(n)$	0.0	0.0	0.0	**97.0**	2.5	0.0	0.5	98.5
F	2	0.0	0.0	0.0	**67.5**	13.0	8.0	11.5	76.5
	3	0.0	0.0	0.0	**84.5**	7.0	4.5	4.0	91.0
	4	0.0	0.0	0.0	**90.0**	7.5	1.5	1.0	96.5
	$\ln(n)$	0.0	0.0	0.0	**95.0**	4.5	0.5	0.0	99.0

Table 2: Relative frequencies (in percentages) of the final tree size identified by the interaction tree (IT) procedure in 200 runs.

selection approach, in which the importance of a covariate, say, X, is determined by the p-value for testing $H_0 : \beta_3 = 0$ in the interaction model $y = \beta_0 + \beta_1 x + \beta_2 \,\mathrm{trt} + \beta_3 \,\mathrm{trt} \cdot x + \varepsilon$. In Figure 1, graphs A-II, B-II, ..., F-II depict the resulting logworths for models A-F, where the logworth is defined as minus base 10 logarithm of the p-value. Both methods are able to pick up important variables in many cases. Nevertheless, the latter approach is focused on cross-product and first-order interactions only, which accounts for its failure in models B and D. In Figure B-II, it fails to identify X_1

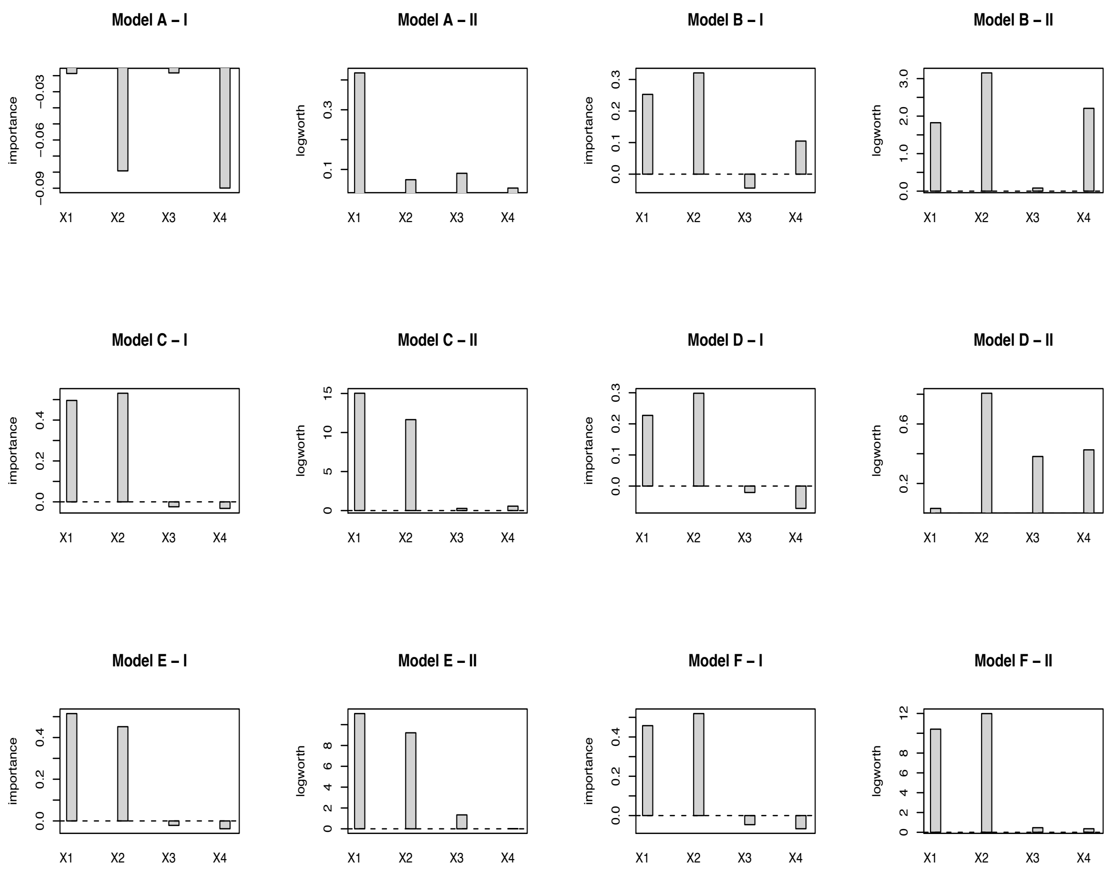


Figure 1: Variable Importance via Random Forests and Feature Selection for the Six Simulation Models. With a data set consisting of 1200 observations generated from Model A, Graph A-I plots the variable importance score computed from 500 random interaction trees while Graph A-II plots the logworth of the p-value from the simple feature selection method. Note that logworth $= -\log_{10}$(p-value). Similar methods were used to make other graphs.

and mistakenly selects X_4. In Figure D-II, none of the covariates are found important.

Remark 5: Comparatively, the random forest approach facilitates better comprehensive evaluation by automatically taking into consideration interactions of higher orders and complex forms. As pointed out by an anonymous reviewer, some other feature selection methods such as the wrapper and/or embedded algorithms can be adopted to detect interactions. It would be interesting in future research to have a more comprehensive study in comparing various algorithms with the variable importance measure obtained from random forests.

	Name	Type	Levels	Description
X_1	`age`	continuous	70	age
X_2	`workclass`	categorical	4	class of worker
X_3	`education`	ordinal	16	education levels
X_4	`marital`	categorical	7	marital status
X_5	`industry`[1]	categorical	22	major industry
X_6	`occupation`[2]	categorical	13	major occupation
X_7	`race`	categorical	5	race
X_8	`union`	categorical	2	member of a labor union
X_9	`fulltime`	categorical	6	full or part time employment
X_{10}	`tax.status`	categorical	6	tax filer status
X_{11}	`household.sum`[3]	categorical	7	detailed household summary
X_{12}	`n.emplyee`	ordinal	7	number of persons worked for employer
X_{13}	`country.birth`	categorical	39	country of birth
X_{14}	`citizenship`	categorical	7	US citizen or not
X_{15}	`self.employed`	categorical	3	own business or self employed
X_{16}	`weeks.worked`	continuous	53	weeks worked in year

[1]The specific levels for `industry` (X_5): 1 - Agriculture; 2 - Business and repair services; 3 - Communications; 4 - Construction; 5 - Education; 6 - Entertainment; 7 - Finance insurance and real estate; 8 - Forestry and fisheries; 9 - Hospital services; 10 - Manufacturing-durable goods; 11 - Manufacturing-nondurable goods; 12 - Medical except hospital; 13 - Mining; 14 - Other professional services; 15 - Personal services except private HH; 16 - Private household services; 17 - Public administration; 19 - Retail trade; Social services; 20 - Transportation; 21 - Utilities and sanitary services; 22 - Wholesale trade.

[2]The specific levels for `occupation` (X_6): 1- Adm support including clerical; 2 - Executive admin and managerial; 3 - Farming forestry and fishing; 4 - Handlers equip cleaners etc; 5 - Machine operators assmblrs & inspctrs; 6 - Other service; 7 - Precision production craft & repair; 8 - Private household services; 9 - Professional specialty; 10 - Protective services; 11 - Sales; 12 - Technicians and related support; 13 - Transportation and material moving.

[3]The specific levels for `household.sum` (X_{11}): 1 - Child 18 or older; 2 - Child under 18 never married; 3 - Group Quarters, Secondary individual; 4 - Householder; 5 - Nonrelative of householder; 6 - Other relative of householder; 7 - Spouse of householder.

Table 3: Variable Description for the Census Wage Data.

4. An Example - The CPS Data

Society has long been arguing for pay equality between women and men. Although the pay gap has narrowed, according to current statistics, gaps between the two sexes still exist. For example, the Bureau of Labor Statistics of the U.S. Department of Labor, in 2004, the most recent year for which statistics are available, reported that women's median weekly earnings were only 80 percent that of men. This represents an improvement over 1979, when women brought home only 62 percent of earnings compared to their male counterparts. A public policy advocate would be very interested in specific subgroups of the working population where the pay gap between sexes is still dominant. Traditional statistical methods consider simple cross tabulations according to a number of variables.

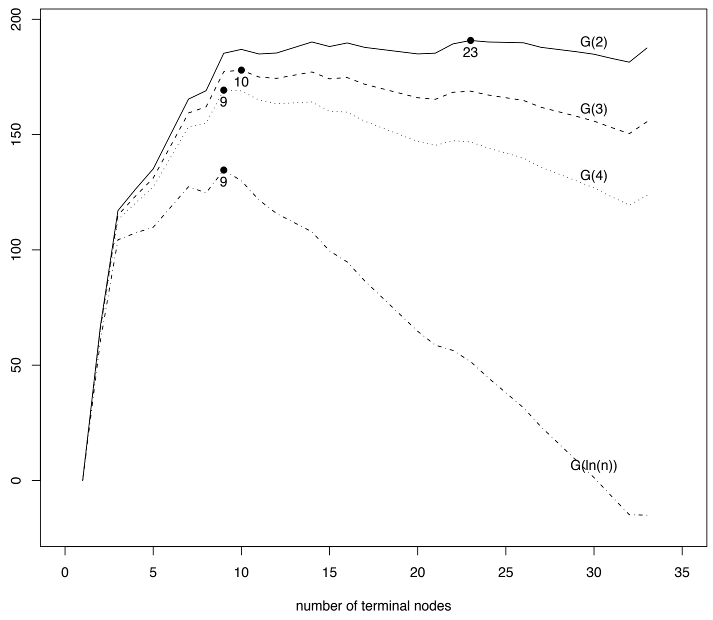


Figure 2: Tree size selection for the CPS data (the plot of $G_\lambda(T_m)$ vs. tree size).

The quest becomes complicated when there are a large number of classification variables to choose from, which motivates the IT procedure.

We extract data from the Current Population Survey (CPS) database, which can be accessed at (`http://www.census.gov/cps/`). The CPS is a national monthly survey of approximately 60,000 households conducted by the U.S. Census Bureau for the Bureau of Labor Statistics. Information collected in the survey includes employment status, hours worked and income from work as well as a number of demographic characteristics of the household members. From the CPS 1995 March Supplement, we compile a data set, which contains 16,602 individuals with no missing value involved. We use hourly wages in U.S. dollars as a measure of pay. Besides gender, there are a total of 16 demographical covariates included. A brief description for these covariates is given in Table 3. Several of them are nominal with many levels. We only list detailed codings for those variables appearing in the final tree structure.

To apply IT, we take a logarithmic transformation on wage. Then, we randomly divide the entire data (denoted as $\mathcal{L}$) into three sets with a ratio of approximately 2: 1: 1. A large initial tree with 33 terminal nodes is constructed and pruned using data in $\mathcal{L}_1$. Sending the validation sample $\mathcal{L}_2$ down each subtree, Figure 2 depicts the resultant $G_\lambda(T_m)$ score versus the subtree size. It can be seen that the four choices of complexity parameter $\lambda = 2, 3, 4$, and $\ln(n)$ yield the best tree sizes of 23, 10, 9, and 9, respectively. For the sake of illustration, the best tree structure with 9 terminal nodes as well as some related summary statistics are given in Figure 3. To merge those terminal nodes among which the wage discrepancies due to gender are not significantly different, we run the amalgamation algorithm with the pooled data $(\mathcal{L}_1 + \mathcal{L}_2)$. It results in four final subgroups, which are then ranked as I-IV according to the ratio of women versus men in terms of average wage. In Group

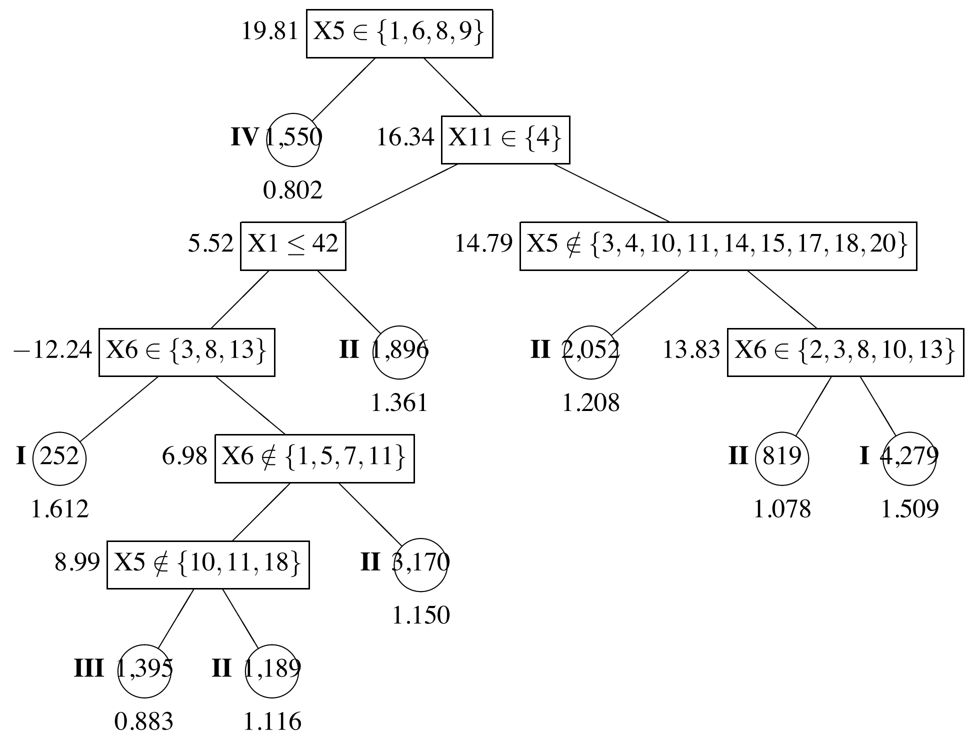


Figure 3: The best-sized interaction tree for the CPS data selected by BIC. For each internal node denoted by a box, the splitting rule is given inside the box. Observations satisfying the condition go to the left node while observations not satisfying the condition go the right node. To the left of each internal node is the $t(s)$ test statistic recomputed using $\mathcal{L}_3$. Terminal nodes are denoted by circles and ranked with Roman numerals on the left. The node size (i.e., number of individuals) is given inside the circle and the ratio of average wages for males versus females is given underneath, both based on the entire data set $\mathcal{L}$.

I, women are most underpaid compared to men. In Group II, women are somewhat underpaid, etc. This specification was also made available to each terminal node in Figure 3.

Table 4 summarizes the final four subgroups. For each subgroup, the number of men and women, as well as their average wages, are computed based on the whole data $\mathcal{L}$. The two-sample t test for comparing the average wages between men and women is also presented. This test was computed by using the test sample $\mathcal{L}_3$. Since there are four tests performed, one may compare the resultant p-values with $0.05/4 = 0.0125$ by applying the Bonferroni-typed adjustment to the joint significant level $\alpha = 0.05$. As a result, Groups I, II, and IV, each marked with an asterisk, show significant differences in wage between men and women.

Figure 3 indicates that the wage disparity between men and women varies with their occupation (X_5), industry of the job (X_6), household compositional situation (X_{11}), and age (X_1). For both Groups I and II, which constitute the majority of the population, women are paid significantly less than men. It is particularly pronounced in Group I, where the average wage of men is \$12.29 per

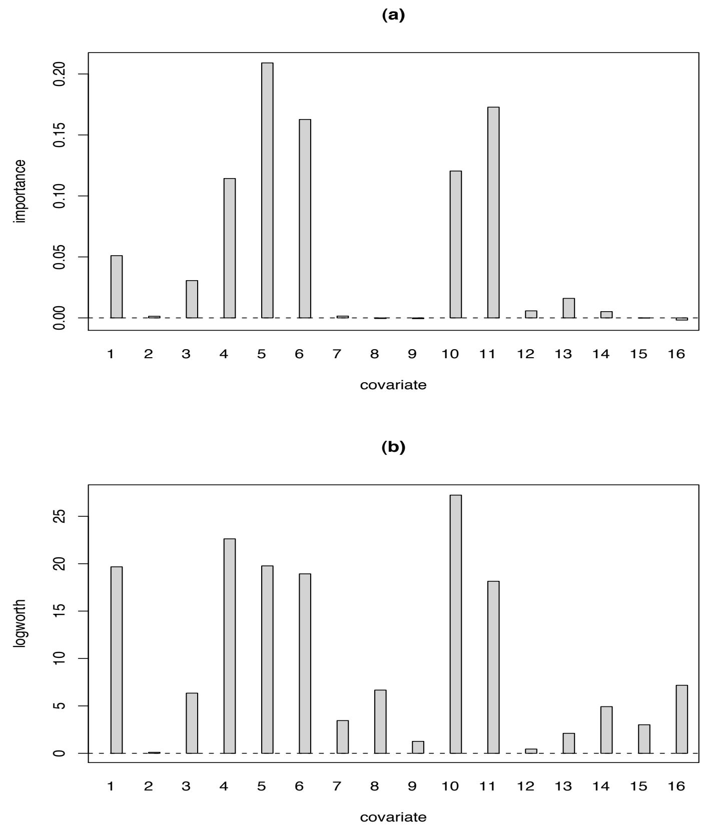


Figure 4: Variable importance measures for the CPS Data: (a), via random forests of interaction trees; (b), based on the p-value associated with the interaction terms in an interaction model that includes the treatment and the covariate only. Note that the logworth is $-\log_{10}$(p-value). The sixteen covariates are `age` (X_1), `workclass` (X_2), `education` (X_3), `marital` (X_4), `industry` (X_5), `occupation` (X_6), `race` (X_7), `union` (X_8), `fulltime` (X_9), `tax.status` (X_{10}), `household.sum` (X_{11}), `n.employee` (X_{12}), `country.birth` (X_{13}), `citizenship` (X_{14}), `self.employed` (X_{15}), and `weeks.worked` (X_{16}).

hour while the average wage of women is only \$8.03 per hour. Interestingly the IT tree has also identified a subgroup, Group IV, in which men are underpaid compared to women. This occurs in the following industries (X_5): 1 - Agriculture; 6 - Entertainment; 8 - Forestry and fisheries; 9 - Hospital services.

Finally, Figure 4(a) displays the computed variance importance scores for each of the 16 covariates. In this calculation, $B = 2,000$ bootstrap samples are used and the number of randomly selected variables, m_0, is set at 4 when splitting data. It can be seen that `industry` (X_5) and `household.sum` (X_{11}) stand out as the most important factors that contribute to the pay gap between males and females, followed by `occupation` (X_6), `tax.status` (X_{10}), `marital` (X_4), and then `age` X_1. This finding is consistent with the tree structure in Figure 3, except for that `tax.status` (X_{10}) and `marital`

	Male		Female		Two-Sample t Test (computed from the test sample)	
	size	average	size	average		
Node	(n_1)	wage $(\bar{y}_1)$	(n_0)	wage $(\bar{y}_0)$	t	p-value
I	3220	12.29	1311	8.03	13.449	$< 0.0001^\star$
II	3800	10.11	5326	8.25	7.678	$< 0.0001^\star$
III	449	8.08	946	9.15	−0.306	0.7598
IV	501	9.43	1049	11.76	−5.560	$< 0.0001^\star$

Table 4: Summary statistics for the four final subgroups of the CPS data. Note that $(n_1, \bar{y}_1, n_0, \bar{y}_0)$ are computed using the whole data. The two-sample t test statistics are calculated from the test sample $\mathcal{L}_3$. The associated p-values are obtained for two-sided tests.

(X_4) have been masked out. The simple feature selection technique is also used to determine the importance of each covariate. Figure 4(b) depicts the resulting logworths. While both plots show some similarities, one should keep in mind that evaluation based on the latter approach is rather restrictive.

5. Discussion

In data analysis, it is important to distinguish two types of interactions. If there is no directional change in terms of the comparison, that is, $(\mu_1^L - \mu_0^L) \cdot (\mu_1^R - \mu_0^R) > 0$, the interaction is said to be *quantitative*; otherwise, it is termed as *qualitative*. The presence of qualitative interactions causes more concerns than quantitative ones (see, e.g., Gail and Simon, 1985). The results from the IT procedure can help to address this issue. For instance, the qualitative interaction in the the CPS example is obviously present among the final four subgroups.

The proposed IT procedure is applicable to a number of different areas. For example, the IT structure can help identify the most and least effective subgroups for the investigational medicine. If the new medicine shows an overall effect that is significant, and, if even in the least effective subgroup under examination, it does not present any harmful side effects, then its release may be endorsed without reservation. In trials where the proposed compound is not found to be significantly effective, the tree-structured subgroup analysis may identify sub-populations that contribute to the failure of the compound. Information gained in this manner could provide the basis for establishing inclusion/exclusion criteria in future trials, and as such, could be of considerable value to the existing efforts in synthesizing compounds to fight deadly diseases such as cancer and HIV/AIDS. We believe that efforts along these lines make subgroup analysis an efficient and valuable tool for research in many application fields.

Acknowledgments

The authors would like to thank the editor and three anonymous referees, whose insightful comments and constructive suggestions have greatly improved a preliminary version of this manuscript. Hansheng Wang's research is supported in part by a NSFC grant (#10771006) and a grant from Microsoft Research Asia.

Appendix A. Categorical Splits

The following theorem provides motivation and justification for the computationally efficient strategy of "ordinalizing" categorical covariates in interaction trees as discussed in Section 2.1. It is analogous to Theorem 9.6 of CART (Breiman et al., 1984, pp. 274–278) yet makes a stronger statement.

Consider splits based on a categorical covariate X, whose possible values range over a finite set $C = \{c_1, \ldots, c_r\}$. Then any subset $A \subset C$, together with its complement $A' = C - A$, induces a partition of the data into $t_L = \{(y_i, \text{trt}_i, \mathbf{x}_i) : x_{il} \in A\}$ and $t_R = \{(y_i, \text{trt}_i, \mathbf{x}_i) : x_{il} \in A'\}$. In the setting of interaction trees, the splitting rule seeks an optimal partition $(A + A')$ to maximize the difference in treatment effect between two child nodes $\left\{(\mu_1^L - \mu_0^L) - (\mu_1^R - \mu_0^R)\right\}^2$, where $\mu_1^L = \mu_{A1} = E\left(y \,|\, x_{il} \in A, \text{trt}_i = 1\right)$ denotes the treatment mean in the left child node, and a similar definition applies for the other μ's. For the sake of convenience, we assume $(\mu_1^L - \mu_0^L) > (\mu_1^R - \mu_0^R)$ so that an optimal split maximizes $(\mu_1^L - \mu_0^L) - (\mu_1^R - \mu_0^R)$.

Theorem 1. *Let $\mu_{ck} = E(y \,|\, X = c, trt = k)$ for any element $c \in C$ and $k = 0, 1$. If $(A + A')$ forms an optimal partition of C, then we have*

$$\mu_{c_1 1} - \mu_{c_1 0} \geq \mu_{c_2 1} - \mu_{c_2 0},$$

for any element $c_1 \in A$ and $c_2 \in A'$. ■

Proof. Define

$$d_1 = \min_{c \in A} (\mu_{c1} - \mu_{c0}) \quad \text{and} \quad d_2 = \max_{c \in A'} (\mu_{c1} - \mu_{c0}).$$

Accordingly, it suffices to show that $d_1 \geq d_2$.

To proceed, let

$$\begin{aligned} A_1 &= \{c \in A : \mu_{c1} - \mu_{c0} = d_1\} \\ A_2 &= \left\{c \in A' : \mu_{c1} - \mu_{c0} = d_2\right\} \\ A_3 &= \{c \in A : \mu_{c1} - \mu_{c0} > d_1\} \\ A_4 &= \left\{c \in A' : \mu_{c1} - \mu_{c0} < d_2\right\}. \end{aligned}$$

Thus $A = A_1 + A_3$ and $A' = A_2 + A_4$. We reserve a generic notation $d = \mu_1 - \mu_0$ for the treatment effect. Define

$$\begin{aligned} d_i &= \sum_{c \in A_i} (\mu_{c1} - \mu_{c0}) \Pr\{X_j = c\}, \\ d_{ij} &= \sum_{c \in A_i \cup A_j} (\mu_{c1} - \mu_{c0}) \Pr\{X_j = c\}, \\ d_{ijk} &= \sum_{c \in A_i \cup A_j \cup A_k} (\mu_{c1} - \mu_{c0}) \Pr\{X_j = c\}. \end{aligned}$$

for $i \neq j \neq k = 1, 2, 3, 4$. We also introduce notation $Q_i = \Pr\{X_j \in A_i\}$ for $i = 1, 2, 3, 4$. It can be verified that

$$d_{ij} = \frac{d_i Q_i + d_j Q_j}{Q_{ij}} \text{ with } Q_{ij} = Q_i + Q_j$$

and

$$d_{ijk} = \frac{d_i Q_i + d_j Q_j + d_k Q_k}{Q_{ijk}} \text{ with } Q_{ijk} = Q_i + Q_j + Q_k.$$

Since partition $(A + A')$ provides an optimal partition, $d_{13} - d_{24} > 0$ reaches the maximum among all possible partitions. In particular, for partition $(A_3 + A_3')$, we have $d_{13} - d_{24} \geq d_3 - d_{124}$, which can be shown equivalent to

$$d_1 \geq (1 - Q_3)\, d_{13} + Q_3\, d_{24}, \tag{4}$$

by first plugging in the following two relationships

$$\begin{aligned} d_3 &= \frac{Q_{13} d_{13} - Q_1 d_1}{Q_3} \\ d_{124} &= \frac{Q_{24} d_{24} + d_1 Q_1}{Q_{124}} \end{aligned}$$

and then simplifying. Similarly, for partition $(A_4' + A_4)$, we can establish

$$Q_4\, d_{13} + (1 - Q_4)\, d_{24} \geq d_2 \tag{5}$$

starting with $d_{13} - d_{24} \geq d_{123} - d_4$.

Now suppose that $d_2 \geq d_1$. It follows from (5) and (4) that

$$\begin{aligned} & Q_4\, d_{13} + (1 - Q_4)\, d_{24} \geq d_2 \geq d_1 \geq (1 - Q_3)\, d_{13} + Q_3\, d_{24} \\ \Longrightarrow\ & \{Q_4\, d_{13} + (1 - Q_4)\, d_{24}\} - \{(1 - Q_3)\, d_{13} + Q_3\, d_{24}\} \ \geq\ d_2 - d_1 \\ \Longrightarrow\ & Q_{12}(d_{24} - d_{13}) \ \geq\ d_2 - d_1, \text{ since } 1 - Q_3 - Q_4 = Q_1 + Q_2 = Q_{12}. \end{aligned}$$

However, $0 > d_{24} - d_{13}$ leads to $0 > d_2 - d_1$, which contradicts the condition $d_1 \leq d_2$. Thus we must have $d_1 > d_2$.

The theorem also holds in extreme cases such as $d_1 = d_3$ or $d_2 = d_4$, etc. The proofs are omitted.

References

H. Akaike. Information theory and an extension of the maximum likelihood principle. In B. N. Petrov and F. Czaki, editors, *2nd Int. Symp. Inf. Theory*, pages 267–281. Budapest: Akad Kiado, 1973.

S. F. Assmann, S. J. Pocock, L. E. Enos, and L. E. Kasten. Subgroup analysis and other (mis)uses of baseline data in clinical trials. *Lancet*, 255:1064–1069, 2000.

L. Breiman. Random forests. *Machine Learning*, 45:5–32, 2001.

L. Breiman, J. Friedman, R. Olshen, and C. Stone. *Classification And Regression Trees*. Wadsworth International Group, Belmont, CA, 1984.

S.-C. Chow and J.-P. Liu. *Design and Analysis of Clinical Trials: Concepts and Methodologies*. Hoboken, NJ: Wiley-Interscience, 2004.

A. Ciampi, J. Thiffault, J.-P. Nakache, and B. Asselain. Stratification by stepwise regression, correspondence analysis and recursive partition. *Computational Statistics and Data Analysis*, 4:185–204, 1986.

CPMP Working Party on Efficacy on Medicinal Products. Biostatistical methodology in clinical trials in applications for marketing authorisations for medicinal products: Note for guidance. *Statistics in Medicine*, 14:1659–1682, 1995.

M. Gail and R. Simon. Testing for qualitative interactions between treatment effects and patient subsets. *Biometrics*, 41:362–372, 1985.

I. Kononenko and S. J. Hong. Attribute Selection for Modeling. In: *Future Generation Computer Systems*, 13:181–195, 1997.

S. W. Lagakos. The challenge of subgroup analyses - reporting without distorting. *The New England Journal of Medicine*, 354:1667–1669, 2006.

M. Leblanc and J. Crowley. Survival trees by goodness of split. *Journal of the American Statistical Association*, 88:457–467, 1993

A. D. R. McQuarrie and C.-L. Tsai. *Regression and Time Series Model Selection*. Singapore: World Scientific, 1998.

J. Morgan and J. Sonquist. Problems in the analysis of survey data and a proposal. *Journal of the American Statistical Association*, 58:415–434, 1963.

A. Negassa, A. Ciampi, M. Abrahamowicz, S. Shapiro, and J.-F. Boivin. Tree-structured subgroup analysis for censored survival data: validation of computationally inexpensive model selection criteria. *Statistics and Computing*, 15:231–239, 2005.

G. Schwarz. Estimating the dimension of a model. *Annals of Statistics*, 6:461–464, 1978.

P. Sleight. Debate: Subgroup analyses in clinical trials: Fun to look at, but don't believe them! *Current Controlled Trials on Cardiovascular Medicine*, 1:25–27, 2000.

X. G. Su, T. Zhou, X. Yan, J. Fan, and S. Yang. Interaction trees with censored survival data. *The International Journal of Biostatistics*, vol. 4 : Iss. 1, Article 2, 2008. Available at: `http://www.bepress.com/ijb/vol4/iss1/2`.

R. Tibshirani and K. Knight. The covariance inflation criterion for adaptive model selection. *Journal of the Royal Statistical Society, Series B*, 61:529–546, 1999.

L. Torgo. A study on end-cut preference in least squares regression trees. Proceedings of the 10th Portuguese Conference on Artificial Intelligence. *Lecture Notes In Computer Science*, 2258:104–115. Springer-Verlag: London, UK, 2001.

J. Ye. On measuring and correcting the effects of data mining and model selection. *Journal of the American Statistical Association*, 93:120–131, 1998.

Journal of Machine Learning Research 10 (2009) 159-162 Submitted 6/08; Revised 10/08; Published 2/09

Python Environment for Bayesian Learning: Inferring the Structure of Bayesian Networks from Knowledge and Data

Abhik Shah SHAHAD@UMICH.EDU
Peter Woolf PWOOLF@UMICH.EDU
Department of Chemical Engineering
3320 G.G. Brown
Ann Arbor, MI 48103, USA

Editor: Cheng Soon Ong

Abstract

In this paper, we introduce PEBL, a Python library and application for learning Bayesian network structure from data and prior knowledge that provides features unmatched by alternative software packages: the ability to use interventional data, flexible specification of structural priors, modeling with hidden variables and exploitation of parallel processing.
PEBL is released under the MIT open-source license, can be installed from the Python Package Index and is available at `http://pebl-project.googlecode.com`.

Keywords: Bayesian networks, python, open source software

1. Introduction

Bayesian networks (BN) have become a popular methodology in many fields because they can model nonlinear, multimodal relationships using noisy, inconsistent data. Although learning the structure of BNs from data is now common, there is still a great need for high-quality open-source software that can meet the needs of various users. End users require software that is easy to use; supports learning with different data types; can accommodate missing values and hidden variables; and can take advantage of various computational clusters and grids. Researchers require a framework for developing and testing new algorithms and translating them into usable software. We have developed the Python Environment for Bayesian Learning (PEBL) to meet these needs.

2. PEBL Features

PEBL provides many features for working with data and BNs; some of the more notable ones are listed below.

2.1 Structure Learning

PEBL can load data from tab-delimited text files with continuous, discrete and class variables and can perform maximum entropy discretization. Data collected following an intervention is important for determining causality but requires an altered scoring procedure (Pe’er et al., 2001; Sachs et al., 2002). PEBL uses the BDe metric for scoring networks and handles interventional data using the method described by Yoo et al. (2002).

```
from pebl import data, result
from pebl.learner.greedy import GreedyLearner
from pebl.taskcontroller.xgrid import XgridController

dataset = data.fromfile('mydata.txt')
runner = XgridController('grid.com', 'pass')
learners = [GreedyLearner(dataset) for i in range(10)]
results = runner.run(learners)
result.merge(results).tohtml('./myoutput')
```

(a) Python script

```
[data]
filename = mydata.txt
[learner]
type = greedy.GreedyLearner
numtasks = 10
[taskcontroller]
type=xgrid.XgridController
[xgrid]
controller = grid.com
password = pass
[result]
output = ./myoutput
```

(b) PEBL configuration file

Figure 1: Two ways of using PEBL: with a Python script and a configuration file. Both methods create 10 greedy learners with default parameters and run them on an Apple Xgid. The Python script can be typed in an interactive shell, run as a script or included as part of a larger application.

PEBL can handle missing values and hidden variables using exact marginalization and Gibbs sampling (Heckerman, 1998). The Gibbs sampler can be resumed from a previously suspended state, allowing for interactive inspection of preliminary results or a manual strategy for determining satisfactory convergence.

A key strength of Bayesian analysis is the ability to use prior knowledge. PEBL supports structural priors over edges specified as 'hard' constraints or 'soft' energy matrices (Imoto et al., 2003) and arbitrary constraints specified as Python functions or lambda expressions.

PEBL includes greedy hill-climbing and simulated annealing learners and makes writing custom learners easy. Efficient implementaion of learners requires careful programming to eliminate redundant computation. PEBL provides components to alter, score and rollback changes to BNs in a simple, transactional manner and with these, efficient learners look remarkably similar to pseudocode.

2.2 Convenience and Scalability

PEBL includes both a library and a command line application. It aims for a balance between ease of use, extensibility and performance. The majority of PEBL is written in Python, a dynamically-typed programming language that runs on all major operating systems. Critical sections use the numpy library (Ascher et al., 2001) for high-performance matrix operations and custom extensions written in ANSI C for portability and speed.

PEBL's use of Python makes it suitable for both programmers and domain experts. Python provides interactive shells and notebook interfaces and includes an extensive standard library and many third-party packages. It has a strong presence in the scientific computing community (Oliphant, 2007). Figure 1 shows a script and configuration file example that showcase the ease of using PEBL.

	BANJO	**BNT**	**Causal Explorer**	**Deal**	**LibB**	**PEBL**
Latest Version	2.0.1	1.04	1.4	1.2-25	2.1	0.9.10
License	Academic [1]	GPL	Academic [1]	GPL	Academic [1]	MIT
Scripting Language	Matlab [2]	Matlab	Matlab	R	N/A	Python
Application	Yes	No	No	No	Yes	Yes
Interventional Data	No	Yes	No	No	No	Yes
DBN	Yes	Yes	No	No	No	No
Structural Priors	Yes [3]	No	No	No	No	Yes
Missing Data	No	Yes	No	No	Yes	Yes
Parallel Execution	No	No	No	No	No	Yes

[1] Custom academic, non-commercial license; not OSI approved.
[2] Via a Matlab-Java bridge.
[3] Only constraints/hard-priors supported.

Table 1: Comparing the features of popular Bayesian network structure learning software.

While many tasks related to Bayesian learning are embarrassingly parallel in theory, few software packages take advantage of it. PEBL can execute learning tasks in parallel over multiple processors or CPU cores, an Apple Xgrid,[1] an IPython cluster[2] or the Amazon EC2 platform.[3] The EC2 platform is especially attractive for scientists because it allows one to rent processing power on an on-demand basis and execute PEBL tasks on them.

With appropriate configuration settings and the use of parallel execution, PEBL can be used for large learning tasks. Although PEBL has been tested successfully with data sets with 10000 variables and samples, BN structure learning is a known NP-Hard problem (Chickering et al., 1994) and analysis using data sets with more than a few hundred variables is likely to result in poor results due to poor coverage of the search space.

3. PEBL Development

The benefits of open source software derive not just from the freedoms afforded by the software license but also from the open and collaborative development model. PEBL's source code repository and issue tracker are hosted at Google Code and freely available to all. Additionally, PEBL includes over 200 automated unit tests and mandates that every source code submission and resolved error be accompanied with tests.

4. Related Software

While there are many software tools for working with BNs, most focus on parameter learning and inference rather than structure learning. Of the few tools for structure learning, few are open-source and none provide the set of features included in PEBL. As shown in Table 1, the ability to handle interventional data, model with missing values and hidden variables, use soft and arbitrary priors and exploit parallel platforms are unique to PEBL. PEBL, however, does not currently provide any

1. Grid computing solution by Apple, Inc. `http://www.apple.com/server/macosx/technology/xgrid.html`.
2. Cluster of Python interpreters (`http://ipython.scipy.org`).
3. A pay-per-use, on-demand computing platform by Amazon, Inc. (`http://aws.amazon.com`).

features for inference or learning Dynamic Bayesian Networks (DBN). Despite its use of optimized matrix libraries and custom C extension modules, PEBL can be an order of magnitude or more slower than software written in Java or C/C++; the ability to use a wider range of data and priors, the parallel processing features and the ease-of-use, however, should make it an attractive option for many users.

5. Conclusion and Future Work

We have developed a library and application for learning BNs from data and prior knowledge. The set of features found in PEBL is unmatched by alternative packages and we hope that our open development model will convince others to use PEBL as a platform for BN algorithms research.

Acknowledgments

We would like to acknowledge support for this project from the NIH grant #U54 DA021519.

References

D. Ascher, P. F. Dubois, K. Hinsen, J. Hugunin, and T. Oliphant. An open source project: Numerical python. Technical Report UCRL-MA-128569, Lawrence Livermore National Laboratory, September 2001.

D. M. Chickering, D. Geiger, and D. Heckerman. Learning bayesian networks is np-hard. Technical Report MSR-TR-94-17, Microsoft Research, November 1994.

D. Heckerman. A tutorial on learning with bayesian networks. In *Learning in Graphical Models*, pages 301–354. The MIT Press, 1998.

S. Imoto, T. Higuchi, T. Goto, K. Tashiro, S. Kuhara, and S. Miyano. Combining microarrays and biological knowledge for estimating gene networks via bayesian networks. *Bioinformatics Conference, 2003. CSB 2003. Proceedings of the 2003 IEEE*, pages 104–113, 2003.

T. E. Oliphant. Python for scientific computing. *Computing in Science & Engineering*, pages 10–20, 2007.

D. Pe’er, A. Regev, G. Elidan, and N. Friedman. Inferring subnetworks from perturbed expression profiles. *Bioinformatics*, 1(1):1–9, 2001.

K. Sachs, D. Gifford, T. Jaakkola, P. Sorger, and D. Lauffenburger. Bayesian network approach to cell signaling pathway modeling. *Science’s STKE*, 2002.

C. Yoo, V. Thorsson, and G. F. Cooper. Discovery of causal relationships in a gene-regulation pathway from a mixture of experimental and observational DNA microarray data. *Pac Symp Biocomput*, 7:498–509, 2002.

Journal of Machine Learning Research 10 (2009) 163-182 Submitted 9/08; Published 2/09

On The Power of Membership Queries in Agnostic Learning[*]

Vitaly Feldman[†] VITALY@POST.HARVARD.EDU
IBM Almaden Research Center
650 Harry Rd.
San Jose, CA 95120

Editor: Rocco Servedio

Abstract

We study the properties of the agnostic learning framework of Haussler (1992) and Kearns, Schapire, and Sellie (1994). In particular, we address the question: is there any situation in which membership queries are useful in agnostic learning?

Our results show that the answer is negative for distribution-independent agnostic learning and positive for agnostic learning with respect to a specific marginal distribution. Namely, we give a simple proof that any concept class learnable agnostically by a distribution-independent algorithm with access to membership queries is also learnable agnostically without membership queries. This resolves an open problem posed by Kearns et al. (1994). For agnostic learning with respect to the uniform distribution over $\{0,1\}^n$ we show a concept class that is learnable with membership queries but computationally hard to learn from random examples alone (assuming that one-way functions exist).

Keywords: agnostic learning, membership query, separation, PAC learning

1. Introduction

The agnostic framework (Haussler, 1992; Kearns et al., 1994) is a natural generalization of Valiant's PAC learning model (Valiant, 1984). In this model no assumptions are made on the labels of the examples given to the learning algorithm, in other words, the learning algorithm has no prior beliefs about the target concept (and hence the name of the model). The goal of the agnostic learning algorithm for a concept class $\mathcal{C}$ is to produce a hypothesis h whose error on the target concept is close to the best possible by a concept from $\mathcal{C}$. This model reflects a common empirical approach to learning, where few or no assumptions are made on the process that generates the examples and a limited space of candidate hypothesis functions is searched in an attempt to find the best approximation to the given data.

Designing algorithms that learn efficiently in this model is notoriously hard and very few positive results are known (Kearns et al., 1994; Lee et al., 1995; Goldman et al., 2001; Gopalan et al., 2008; Kalai et al., 2008a,b). Furthermore, strong computational hardness results are known for agnostic learning of even the simplest classes of functions such as parities, monomials and halfspaces (Håstad, 2001; Feldman, 2006; Feldman et al., 2006; Guruswami and Raghavendra, 2006) (albeit only for *proper* learning). Reductions from long-standing open problems for PAC learning to ag-

∗. Parts of this work are published in the Proceedings of 18th Annual Conference on Learning Theory, 2008.

†. Part of the work done while the author was at Harvard University supported by grants from the National Science Foundation NSF-CCF-04-32037 and NSF-CCF-04-27129.

nostic learning of simple classes of functions provide another indication of the hardness of agnostic learning (Kearns et al., 1994; Kalai et al., 2008a; Feldman et al., 2006).

A membership oracle allows a learning algorithm to obtain the value of the unknown target function f on any point in the domain. It can be thought of as modeling the access to an expert or ability to conduct experiments. Learning with membership queries in both PAC and Angluin's exact models (Angluin, 1988) was studied in numerous works. For example monotone DNF formulas, finite automata and decision trees are only known to be learnable with membership queries (Valiant, 1984; Angluin, 1988; Bshouty, 1995). It is well-known and easy to prove that the PAC model with membership queries is strictly stronger than the PAC model without membership queries (if one-way functions exist).

Membership queries are also used in several agnostic learning algorithms. The first one is the famous algorithm of Goldreich and Levin (1989) introduced in a cryptographic context (even before the definition of the agnostic learning model). Their algorithm learns parities agnostically with respect to the uniform distribution using membership queries. Kushilevitz and Mansour (1993) used this algorithm to PAC learn decision trees and it has since found numerous other significant applications. More efficient versions of this algorithm were also given by Levin (1993), Bshouty, Jackson, and Tamon (2004) and Feldman (2007). Recently, Gopalan, Kalai, and Klivans (2008) gave an elegant algorithm that learns decision trees agnostically over the uniform distribution and uses membership queries.

1.1 Our Contribution

In this work we study the power of membership queries in the agnostic learning model. The question of whether or not membership queries can aid in agnostic learning was first asked by Kearns et al. (1994) who conjectured that the answer is no. To the best of our knowledge, the question has not been addressed prior to our work. We present two results on this question. In the first result we prove that every concept class learnable agnostically with membership queries is also learnable agnostically without membership queries (see Th. 6 for a formal statement). This proves the conjecture of Kearns et al. (1994). The reduction we give modifies the distribution of examples and therefore is only valid for distribution-independent learning, that is, when a single learning algorithm is used for every distribution over the examples. The simple proof of this result explains why the known distribution-independent agnostic learning algorithm do not use membership queries (Kearns et al., 1994; Kalai et al., 2008a,b).

The proof of this result also shows equivalence of two standard agnostic models: the one in which examples are labeled by an unrestricted function and the one in which examples come from a joint distribution over the domain and the labels.

Our second result is a proof that there exists a concept class that is agnostically learnable with membership queries over the uniform distribution on $\{0,1\}^n$ but hard to learn in the same setting without membership queries (see Th. 8 for a formal statement). This result is based on the most basic cryptographic assumption, namely the existence of one-way functions. Note that an unconditional separation of these two models would imply $\mathsf{NP} \neq \mathsf{P}$. Cryptographic assumptions are essential for numerous other hardness results in learning theory (cf., Kearns and Valiant, 1994; Kharitonov, 1995). Our construction is based on the use of pseudorandom function families, list-decodable codes and a variant of an idea from the work of Elbaz, Lee, Servedio, and Wan (2007). Sections 4.1 and 4.2 describe the technique and its relation to prior work in more detail.

This results is, perhaps, unsurprising since agnostic learning of parities with respect to the uniform distribution from random examples only is commonly considered hard and is known to be equivalent to learning of parities with random noise (Feldman et al., 2006), a long standing open problem which itself is equivalent to decoding of random linear codes, a long-standing open problem in coding theory. The best known algorithm for this problem runs in time $O(2^{n/\log n})$ (Blum et al., 2003; Feldman et al., 2006). If one assumes that learning of parities with noise is intractable then it immediately follows that membership queries are provably helpful in agnostic learning over the uniform distribution on $\{0,1\}^n$. The goal of our result is to replace this assumption by a possibly weaker and more general cryptographic assumption. It is known that if learning of parities with noise is hard then one-way functions exist (Blum et al., 1993) but, for all we know, it is possible that the converse is not true. The proof of our result however is substantially less straightforward than one might expect (and than the analogous separation for PAC learning). Here the main obstacle is the same as in proving positive results for agnostic learning: the requirements of the model impose severe limits on concept classes for which the agnostic guarantees can be provably satisfied.

1.2 Organization

Following the preliminaries, our first result is described in Section 3. The second result appears in Section 4.

2. Preliminaries

Let X denote the domain or the *input space* of a learning problem. The domain of the problems that we study is $\{0,1\}^n$, or the n-dimensional *Boolean hypercube*. A *concept* over X is a $\{-1,1\}$ function over the domain and a *concept class* $\mathcal{C}$ is a set of concepts over X. The unknown function $f \in \mathcal{C}$ that a learning algorithm is trying to learn is referred to as the *target concept*.

A parity function is a function equal to the *XOR* of some subset of variables. For a Boolean vector $a \in \{0,1\}^n$ we define the parity function $\chi_a(x) = (-1)^{a\cdot x} = (-1)^{\oplus_{i\le n} a_i x_i}$. We denote the concept class of parity functions $\{\chi_a \mid a \in \{0,1\}^n\}$ by PAR. A *k-junta* is a function that depends only on k variables.

A *representation class* is a concept class defined by providing a specific way to represent each function in the concept class. In fact all the classes of functions that we discuss are representation classes. We often refer to a representation class simply as concept class when the representation is implicit in the description of the class. For a representation class $\mathcal{F}$, we say that an algorithm outputs $f \in \mathcal{F}$ if the algorithm outputs f in the representation associated with $\mathcal{F}$.

2.1 PAC Learning Model

The learning models discussed in this work are based on Valiant's well-known PAC model (Valiant, 1984). In this model, for a concept f and distribution D over X, an *example oracle* $\mathrm{EX}(D,f)$ is the oracle that, upon request, returns an example $\langle x, f(x)\rangle$ where x is chosen randomly with respect to D. For $\varepsilon \ge 0$ we say that a function g ε-approximates a function f with respect to distribution D if $\mathbf{Pr}_D[f(x) = g(x)] \ge 1-\varepsilon$. In the PAC learning model the learner is given access to $\mathrm{EX}(D,f)$ where f is assumed to belong to a fixed concept class $\mathcal{C}$.

Definition 1 *For a representation class $\mathcal{C}$, we say that an algorithm* `Alg` PAC learns $\mathcal{C}$*, if for every $\varepsilon > 0$, $\delta > 0$, $f \in \mathcal{C}$, and distribution D over X,* `Alg`*, given access to $EX(D, f)$, outputs, with probability at least $1 - \delta$, a hypothesis h that ε-approximates f.*

The learning algorithm is *efficient* if its running time and the time to evaluate h are polynomial in $1/\varepsilon, 1/\delta$ and the *size* σ of the learning problem. Here by the size we refer to the maximum description length of an element in X (e.g., n when $X = \{0,1\}^n$) plus the maximum description length of an element in $\mathcal{C}$ in the representation associated with $\mathcal{C}$.

An algorithm is said to *weakly* learn $\mathcal{C}$ if it produces a hypothesis h that $(\frac{1}{2} - \frac{1}{p(\sigma)})$-approximates f for some polynomial $p(\cdot)$.

2.2 Agnostic Learning Model

The *agnostic* learning model was introduced by Haussler (1992) and Kearns et al. (1994) in order to model situations in which the assumption that examples are labeled by some $f \in \mathcal{C}$ does not hold. In its least restricted version the examples are generated from some unknown distribution A over $X \times \{-1,1\}$. The goal of an agnostic learning algorithm for a concept class $\mathcal{C}$ is to produce a hypothesis whose error on examples generated from A is close to the best possible by a concept from $\mathcal{C}$. Class $\mathcal{C}$ is referred to as the *touchstone* class in this setting. More generally, the model allows specification of the assumptions made by a learning algorithm by describing a set $\mathcal{A}$ of distributions over $X \times \{-1,1\}$ that restricts the distributions over $X \times \{-1,1\}$ seen by a learning algorithm. Such $\mathcal{A}$ is referred to as the *assumption class*. Any distribution A over $X \times \{-1,1\}$ can be described uniquely by its marginal distribution D over X and the expectation of b given x. That is, we refer to a distribution A over $X \times \{-1,1\}$ by a pair (D_A, ϕ_A) where $D_A(z) = \mathbf{Pr}_{\langle x,b\rangle \sim A}[x = z]$ and

$$\phi_A(z) = \mathbf{E}_{\langle x,b\rangle \sim A}[b \mid z = x].$$

Formally, for a Boolean function h and a distribution $A = (D, \phi)$ over $X \times \{-1,1\}$, we define

$$\Delta(A,h) = \mathbf{Pr}_{\langle x,b\rangle \sim A}[h(x) \neq b] = \mathbf{E}_D[|\phi(x) - h(x)|/2] \ .$$

Similarly, for a concept class $\mathcal{C}$, define

$$\Delta(A,\mathcal{C}) = \inf_{h \in \mathcal{C}} \{\Delta(A,h)\} \ .$$

Kearns et al. (1994) define agnostic learning as follows.

Definition 2 *An algorithm* `Alg` agnostically *learns a representation class $\mathcal{C}$ by a representation class $\mathcal{H}$ assuming $\mathcal{A}$ if for every $\varepsilon > 0, \delta > 0, A \in \mathcal{A}$,* `Alg` *given access to examples drawn randomly from A, outputs, with probability at least $1 - \delta$, a hypothesis $h \in \mathcal{H}$ such that $\Delta(A,h) \leq \Delta(A,\mathcal{C}) + \varepsilon$.*

The learning algorithm is *efficient* if it runs in time polynomial $1/\varepsilon, \log(1/\delta)$ and σ (the size of the learning problem). If $\mathcal{H} = \mathcal{C}$ then, by analogy with the PAC model, the learning is referred to as *proper*. We drop the reference to $\mathcal{H}$ to indicate that $\mathcal{C}$ is learnable by some representation class.

A number of special cases of the above definition are commonly considered (and often referred to as *the* agnostic learning model). In fully agnostic learning $\mathcal{A}$ is the set of all distributions over $X \times \{-1,1\}$. Another version assumes that examples are labeled by an unrestricted function. That is, the set $\mathcal{A}$ contains distribution $A = (D, f)$ for every Boolean function f and distribution D. Note that access to random examples from $A = (D, f)$ is equivalent to access to $\mathrm{EX}(D, f)$. Following Kearns et al. (1994), we refer to this version as *agnostic PAC learning*. Theorem 6 implies that these versions are essentially equivalent. In *distribution-specific* versions of this model for every $(D, \phi) \in \mathcal{A}$, D equals to some fixed distribution known in advance.

We also note that the agnostic PAC learning model can also be thought of as a model of adversarial classification noise. By definition, a Boolean function g differs from some function $f \in \mathcal{C}$ on $\Delta(g, \mathcal{C})$ fraction of the domain. Therefore g can be thought of as f corrupted by noise of rate $\Delta_D(f, \mathcal{C})$. Unlike in the random classification noise model, the points on which a concept can be corrupted are unrestricted and therefore the noise is referred to as adversarial.

2.2.1 UNIFORM CONVERGENCE

A natural approach to agnostic learning is to first draw a sample of fixed size and then choose a hypothesis that best fits the observed labels. The conditions in which this approach is successful were studied in works of Dudley (1978), Pollard (1984), Haussler (1992), Vapnik (1998) and others. They give a number of conditions on the hypothesis class $\mathcal{H}$ that guarantee *uniform convergence* of empirical error to the true error. That is, existence of a function $m_{\mathcal{H}}(\varepsilon, \delta)$ such that for every distribution A over examples, every $h \in \mathcal{H}$, $\varepsilon > 0$, $\delta > 0$, the empirical error of h on sample of $m_{\mathcal{H}}(\varepsilon, \delta)$ examples randomly chosen from A is, with probability at least $1 - \delta$, within ε of $\Delta(A, h)$. We denote the empirical error of h on sample S by $\Delta(S, h)$. In the Boolean case, the following result of Vapnik and Chervonenkis (1971) will be sufficient for our purposes.

Theorem 3 *Let $\mathcal{H}$ be a concept class over X of VC dimension d. Then for every distribution A over $X \times \{-1, 1\}$, every $h \in \mathcal{H}$, $\varepsilon > 0$, $\delta > 0$, and sample S of size $m = O((d \log(d/\varepsilon) + \log(1/\delta))/\varepsilon^2)$ randomly drawn with respect to A,*

$$\mathbf{Pr}[|\Delta(A, h) - \Delta(S, h)| \geq \varepsilon] \leq \delta.$$

In fact a simple uniform convergence result based on the cardinality of the function class follows easily from Chernoff bounds (Haussler, 1992). That is Theorem 3 holds for $m = O(\log|\mathcal{H}|/\varepsilon^2 \cdot \log(1/\delta))$. This result would also be sufficient for our purposes but might give somewhat weaker bounds.

2.3 Membership Queries

A membership oracle for a function f is the oracle that, given any point $z \in \{0, 1\}^n$, returns the value $f(z)$ (Valiant, 1984). We denote it by $\mathrm{MEM}(f)$. We refer to agnostic PAC learning with access to $\mathrm{MEM}(f)$, where f is the unknown function that labels the examples, as *agnostic PAC+MQ* learning. Similarly, one can extend the definition of a membership oracle to fully agnostic learning. For a distribution A over $X \times \{-1, 1\}$, let $\mathrm{MEM}(A)$ be the oracle that, upon query z, returns $b \in \{-1, 1\}$ with probability $\mathbf{Pr}_A[(x, b) \mid x = z]$. We say that $\mathrm{MEM}(A)$ is *persistent* if given the same query the oracle responds with the same label. When learning with persistent membership queries the learning algorithm is allowed to fail with some negligible probability over the answers of $\mathrm{MEM}(A)$. This is necessary to account for probability that the answers of $\mathrm{MEM}(A)$ might be not "representative" of A (a more formal argument can be found for example in the work of Goldman et al. 2001).

2.4 List-Decodable Codes

As we have mentioned earlier, agnostic learning can be seen as recovery of an unknown concept from possibly malicious errors. Therefore, encoding of information that allows recovery from errors, or error-correcting codes, can be useful in the design of agnostic learning algorithms. In our construction we will use binary list-decodable error-correcting codes. A list-decodable code is a code that allows recovery from errors when the number of errors is larger than the distance of the code, and hence there is more than one valid way to decode the corrupted encoding, each giving a different message (see for example the book of van Lint 1998). List-decoding of the code gives the list of all the messages corresponding to the valid ways to decode the corrupt encoding. Formally, let $C : \{0,1\}^u \to \{0,1\}^v$ be a binary code of message length u and block length v. Our construction requires efficient encoding and efficient list-decoding from $1/2-\gamma$ fraction of errors for a $\gamma > 0$ that we will define later. Specifically,

- Efficient encoding algorithm. For any $z \in \{0,1\}^u$ and $j \le v$, $C(z)_j$ (the j^{th} bit of $C(z)$) is computable in time polynomial in u and $\log v$.
- Efficient list-decoding from $(1/2-\gamma')v$ errors in time polynomial in u and $1/\gamma'$ for any $\gamma' \ge \gamma$. That is, an algorithm that given oracle access to the bits of string $y \in \{0,1\}^v$, produces the list of all messages z such that $\mathbf{Pr}_{j\in[v]}[C(z)_j \ne y_j] \le 1/2-\gamma'$ (in time polynomial in u and $1/\gamma'$).

Our main result is achieved using the Reed-Solomon code concatenated with the Hadamard code for which the list-decoding algorithm was given by Guruswami and Sudan (2000). Their code has the desired properties for $v = O(u^2/\gamma^4)$. In the description of our construction, for simplicity, we use the more familiar but exponentially longer Hadamard code.

2.4.1 HADAMARD CODE

The Hadamard code encodes a vector $a \in \{0,1\}^n$ as the values of the parity function χ_a on all the points in $\{0,1\}^n$ (that is the length of the encoding is 2^n). It is convenient to describe list-decoding of the Hadamard code using Fourier analysis over $\{0,1\}^n$ that is commonly used in the context of learning with respect to the uniform distribution (Linial, Mansour, and Nisan, 1993). We now briefly review a number of simple facts on the Fourier representation of functions over $\{0,1\}^n$ and refer the reader to a survey by Mansour (1994) for more details. In the discussion below all probabilities and expectations are taken with respect to the uniform distribution U unless specifically stated otherwise.

Define an inner product of two real-valued functions over $\{0,1\}^n$ to be $\langle f,g\rangle = \mathbf{E}_x[f(x)g(x)]$. The technique is 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 functions over $\{0,1\}^n$ with the above inner product. 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 $\mathbf{E}_x[f(x)\chi_a(x)]$; a is called the *index* of $\hat{f}(a)$. We say that a Fourier coefficient $\hat{f}(a)$ is θ-heavy if $|\hat{f}(a)| \ge \theta$. Let $L_2(f) = E_x[(f(x))^2]^{1/2}$. Parseval's identity states that

$$(L_2(f))^2 = \mathbf{E}_x[(f(x))^2] = \sum_a \hat{f}^2(a) .$$

Let $A = (U,\phi)$ be a distribution over $\{0,1\}^n \times \{-1,1\}$ with uniform marginal distribution over $\{0,1\}^n$. Fourier coefficient $\hat{\phi}(a)$ can be easily related to the error of $\chi_a(x)$ on A. That is,

$$\mathbf{Pr}_{\langle x,b\rangle\sim A}[b \ne \chi_a(x)] = (1-\hat{\phi}(a))/2. \tag{1}$$

Therefore, both list-decoding of the Hadamard code and agnostic learning of parities amount to finding the largest (within 2ϵ) Fourier coefficient of $\phi(x)$. The first algorithm for this task was given by Goldreich and Levin (1989). Given access to a membership oracle, for every $\epsilon > 0$ their algorithm can efficiently find all ϵ-heavy Fourier coefficients.

Theorem 4 (Goldreich and Levin, 1989) *There exists an algorithm* GL *that for every distribution* $A = (U, \phi)$ *and every* $\epsilon, \delta > 0$, *given access to MEM*(A), $\mathtt{GL}(\epsilon, \delta)$ *returns, with probability at least* $1 - \delta$, *a set of indices* $T \subseteq \{0,1\}^n$ *that contains all a such that* $|\hat{\phi}(a)| \geq \epsilon$ *and for all* $a \in T$, $|\hat{\phi}(a)| \geq \epsilon/2$. *Furthermore, the algorithm runs in time polynomial in n,*$1/\epsilon$ *and* $\log(1/\delta)$.

Note that by Parseval's identity, the condition $|\hat{\phi}(a)| \geq \epsilon/2$ implies that there are at most $4/\epsilon^2$ elements in T.

2.5 Pseudo-random Function Families

A key part of our construction in Section 4 will be based on the use of pseudorandom functions families defined by Goldreich, Goldwasser, and Micali (1986).

Definition 5 *A function family* $\mathcal{F} = \{F_n\}_{n=1}^{\infty}$ *where* $F_n = \{\pi_z\}_{z \in \{0,1\}^n}$ *is a* pseudorandom function family *of Boolean functions over* $\{0,1\}^n$ *if*

- *There exists a polynomial time algorithm that for every n, given* $z \in \{0,1\}^n$ *and* $x \in \{0,1\}^n$ *computes* $\pi_z(x)$.
- *Any adversary M whose resources are bounded by a polynomial in n can distinguish between a function* π_z *(where* $z \in \{0,1\}^n$ *is chosen randomly and kept secret) and a totally random function from* $\{0,1\}^n$ *to* $\{-1,1\}$ *only with negligible probability. That is, for every probabilistic polynomial time M with an oracle access to a function from* $\{0,1\}^n$ *to* $\{-1,1\}$ *there exists a negligible function* $\nu(n)$,

$$|\mathbf{Pr}[M^{\pi_z}(1^n) = 1] - \mathbf{Pr}[M^{\rho}(1^n) = 1]| \leq \nu(n),$$

where π_z *is a function randomly and uniformly chosen from* F_n *and* ρ *is a randomly chosen function from* $\{0,1\}^n$ *to* $\{-1,1\}$. *The probability is taken over the random choice of* π_z *or* ρ *and the coin flips of M.*

Results of Håstad et al. (1999) and Goldreich et al. (1986) give a construction of pseudorandom function families based on the existence of one-way functions.

3. Distribution-Independent Agnostic Learning

In this section we show that in distribution-independent agnostic learning membership queries do not help. In addition, we prove that fully agnostic learning is equivalent to agnostic PAC learning. Our proof is based on two simple observations about agnostic learning via empirical error minimization. Values of the unknown function on points outside of the sample can be set to any value without changing the best fit by a function from the touchstone class. Therefore membership queries do not make empirical error minimization easier. In addition, points with contradicting labels do not influence the complexity of empirical error minimization since any function has the same error on pairs of contradicting labels. We will now provide the formal statement of this result.

Theorem 6 *Let* `Alg` *be an algorithm that agnostically PAC+MQ learns a concept class $\mathcal{C}$ in time $T(\sigma,\varepsilon,\delta)$ and outputs a hypothesis in a representation class $\mathcal{H}(\sigma,\varepsilon)$. Then $\mathcal{C}$ is (fully) agnostically learnable by $\mathcal{H}(\sigma,\varepsilon/2)$ in time $T(\sigma,\varepsilon/2,\delta/2)+O(d\cdot\log(d/\varepsilon)+\log(1/\delta))/\varepsilon^2)$, where d is the VC dimension of $\mathcal{H}(\sigma,\varepsilon/2)\cup\mathcal{C}$.*

Proof Let $A=(D,\phi)$ be a distribution over $X\times\{-1,1\}$. Our reduction works as follows. Start by drawing m examples from A for m to be defined later. Denote this sample by S. Let S' be S with all contradicting pairs of examples removed, that is for each example $\langle x,1\rangle$ we remove it together with one example $\langle x,-1\rangle$. Every function has the same error rate of $1/2$ on examples in $S\setminus S'$. Therefore for every function h,

$$\Delta(S,h)=\frac{\Delta(S',h)|S'|+|S\setminus S'|/2}{|S|}=\Delta(S',h)\frac{|S'|}{m}+\frac{m-|S'|}{2m} \tag{2}$$

and hence

$$\Delta(S,\mathcal{C})=\Delta(S',\mathcal{C})\frac{|S'|}{m}+\frac{m-|S'|}{2m}. \tag{3}$$

Let $f(x)$ denote the function equal to b if $\langle x,b\rangle\in S'$ and equal to 1 otherwise. Let $U_{S'}$ denote the uniform distribution over S'. Given the sample S' we can easily simulate the example oracle $\mathrm{EX}(U_{S'},f)$ and $\mathrm{MEM}(f)$. We run `Alg`$(\varepsilon/2,\delta/2)$ with theses oracles and denote its output by h. Note, that this simulates $\mathcal{A}$ in the agnostic PAC+MQ setting over distribution $(U_{S'},f)$.

By the definition of $U_{S'}$, for any Boolean function $g(x)$,

$$\mathbf{Pr}_{U_{S'}}[f(x)\neq g(x)]=\frac{1}{|S'|}\left|\{x\in S'\mid f(x)\neq g(x)\}\right|=\Delta(S',g)\ .$$

That is, the error of any function g on $U_{S'}$ is exactly the empirical error of g on sample S'. Thus $\Delta((U_{S'},f),h)=\Delta(S',h)$ and $\Delta((U_{S'},f),\mathcal{C})=\Delta(S',\mathcal{C})$. By the correctness of `Alg`, with probability at least $1-\delta/2$, $\Delta(S',h)\leq\Delta(S',\mathcal{C})+\varepsilon/2$. By Equations (2) and (3) we thus obtain that

$$\Delta(S,h)=\Delta(S',h)\frac{|S'|}{m}+\frac{m-|S'|}{2m}\leq(\Delta(S',\mathcal{C})+\frac{\varepsilon}{2})\frac{|S'|}{m}+\frac{m-|S'|}{2m}=\Delta(S,\mathcal{C})+\frac{\varepsilon}{2}\frac{|S'|}{m}.$$

Therefore $\Delta(S,h)\leq\Delta(S,\mathcal{C})+\varepsilon/2$. We can apply the VC dimension-based uniform convergence results for $\mathcal{H}(\sigma,\varepsilon/2)\cup\mathcal{C}$ (Theorem 3) to conclude that for

$$m(\varepsilon/4,\delta/4)=O\left(\frac{d\log(d/\varepsilon)+\log(1/\delta)}{\varepsilon^2}\right),$$

with probability at least $1-\delta/2$, $\Delta(A,h)\leq\Delta(S,h)+\frac{\varepsilon}{4}$ and $\Delta(S,\mathcal{C})+\frac{\varepsilon}{4}\leq\Delta(A,\mathcal{C})$. Finally, we obtain that with probability at least $1-\delta$,

$$\Delta(A,h)\leq\Delta(S,h)+\frac{\varepsilon}{4}\leq\Delta(S,\mathcal{C})+\frac{3\varepsilon}{4}\leq\Delta(A,\mathcal{C})+\varepsilon.$$

It easy to verify that the running time and hypothesis space of this algorithm are as claimed. ■

Note that if `Alg` is efficient then $d(\sigma,\varepsilon/2)$ is polynomial in σ and $1/\varepsilon$ and, in particular, the obtained algorithm is efficient. In addition, in place of VC-dim one can use the uniform convergence result based on the cardinality of the hypothesis space. The description length of a hypothesis output by `Alg` is polynomial in σ and $1/\varepsilon$ and hence in this case a polynomial number of samples will be required to simulate `Alg`.

Remark 7 *We note that while this proof is given for the strongest version of agnostic learning in which the error of an agnostic algorithm is bounded by $\Delta(A,\mathcal{C})+\varepsilon$, it can be easily extended to weaker forms of agnostic learning, such as algorithms that only guarantee error bounded by $\alpha\cdot\Delta(A,\mathcal{C})+\beta+\varepsilon$ for some $\alpha\geq 1$ and $\beta\geq 0$. This is true since the reduction adds at most $\varepsilon/2$ to the error of the original algorithm.*

4. Learning with Respect to the Uniform Distribution

In this section we show that when learning with respect to the uniform distribution over $\{0,1\}^n$, membership queries are helpful. Specifically, we show that if one-way functions exist, then there exists a concept class $\mathcal{C}$ that is not agnostically PAC learnable (even weakly) with respect to the uniform distribution but is agnostically learnable over the uniform distribution given membership queries. Our agnostic learning algorithm is successful only when $\varepsilon\geq 1/p(n)$ for a polynomial p fixed in advance (the definition of $\mathcal{C}$ depends on p). While this is slightly weaker than required by the definition of the model it still exhibits the gap between agnostic learning with and without membership queries. We remark that a number of known PAC and agnostic learning algorithms are efficient only for restricted values of ε (O'Donnell and Servedio, 2006; Gopalan et al., 2008; Kalai et al., 2008a).

Theorem 8 *For every polynomial $p(\cdot)$, there exists a concept class $\mathcal{C}^p$ over $\{0,1\}^n$ such that,*

1. *there exists no efficient algorithm that weakly PAC learns $\mathcal{C}^p$ with respect to the uniform distribution over $\{0,1\}^n$;*
2. *there exists a randomized algorithm* `AgnLearn` *that for every distribution $A=(U,\phi)$ over $\{0,1\}^n\times\{-1,1\}$ and every $\varepsilon\geq 1/p(n),\delta>0$, given access to MEM$(A)$, with probability at least $1-\delta$, finds h such that $\Delta(A,h)\leq\Delta(A,\mathcal{C}^p_n)+\varepsilon$. The probability is taken over the coin flips of MEM(A) and* `AgnLearn`. `AgnLearn` *runs in time polynomial in n and $\log(1/\delta)$.*

4.1 Background

We first show why some of the known separation results will not work in the agnostic setting. It is well-known that the PAC model with membership queries is strictly stronger than the PAC model without membership queries (under the same cryptographic assumption). The separation result is obtained by using a concept class $\mathcal{C}$ that is not PAC learnable and augmenting each concept $f\in\mathcal{C}$ with the encoding of f in a fixed part of the domain. This encoding is readable using membership queries and therefore an MQ algorithm can "learn" the augmented $\mathcal{C}$ by querying the points that contain the encoding. On the other hand, with overwhelming probability this encoding will not be observed in random examples and therefore does not help learning from random examples. This simple approach would fail in the agnostic setting. The unknown function might be random on the part of the domain that contains the encoding and equal to a concept from $\mathcal{C}$ elsewhere. The agreement of the unknown function with a concept from $\mathcal{C}$ is almost 1 but membership queries on the points of encoding will not yield any useful information.

A similar problem arises with encoding schemes used in the separation results of Elbaz et al. (2007) and Feldman and Shah (2009). There too the secret encoding can be rendered unusable by a function that agrees with a concept in $\mathcal{C}$ on a significant fraction of the domain.

4.2 Outline

We start by presenting some of the intuition behind our construction. As in most other separation results our goal is to create a concept class that is not learnable from uniform examples but includes an encoding of the unknown function that is readable using membership queries. We first note that in order for this approach to work in the agnostic setting the secret encoding has to be "spread" over at least $1-2\varepsilon$ fraction of $\{0,1\}^n$. To see this let f be a concept and let $S \subseteq \{0,1\}^n$ be the subset of the domain where the encoding of f is contained. Assume, for simplicity, that without the encoding the learning algorithm cannot predict f on $\overline{S} = \{0,1\}^n \setminus S$ with any significant advantage over random guessing. Let f' be a function equal to f on $\overline{S}$ and truly random on S. Then

$$\mathbf{Pr}[f = f'] \approx (|\overline{S}| + |S|/2)/2^n = 1/2 + \frac{|\overline{S}|}{2^{n+1}} .$$

On the other hand, f' does not contain any information about the encoding of f and therefore, by our assumption, no efficient algorithm can produce a hypothesis with advantage significantly higher than $1/2$ on both S and $\overline{S}$. This means that the error of any efficient algorithm will be higher by at least $|\overline{S}|/2^{n+1}$ than the best possible. To ensure that $|\overline{S}|/2^{n+1} \leq \varepsilon$, we need $|S| \geq (1-2\varepsilon)2^n$.

Another requirement that the construction has to satisfy is that the encoding of the secret has to be resilient to almost any amount of noise. In particular, since the encoding is a part of the function, we also need to be able to reconstruct an encoding that is close to the best possible. An encoding with this property is in essence a list-decodable binary code. In order to achieve the strongest separation result we will use the code of Guruswami and Sudan (2000) that is the concatenation of Reed-Solomon code with the binary Hadamard code. However, to simplify the presentation, we will use the more familiar binary Hadamard code in our construction. In Section 4.6 we provide the details on the use of the Guruswami-Sudan code in place of the Hadamard code.

The Hadamard code is equivalent to encoding a vector $a \in \{0,1\}^n$ as the values of the parity function χ_a on all points in $\{0,1\}^n$. That is, n bit vector a is encoded into 2^n bits given by $\chi_a(x)$ for every $x \in \{0,1\}^n$. This might appear quite inefficient since a learning algorithm will not be able to read all the bits of the encoding. However the Goldreich-Levin algorithm provides an efficient way to recover the indices of all the parities that agree with a given function with probability significantly higher than $1/2$ (Goldreich and Levin, 1989). Therefore the Hadamard code can be decoded by reading the code in only a polynomial number of (appropriately-chosen) locations.

The next problem that arises is that the encoding should not be readable from random examples. As we have observed earlier, we cannot simply "hide" it on a negligible fraction of the domain. Specifically, we need to make sure that our Hadamard encoding is not recoverable from random examples. Our solution to this problem is to use a pseudo-random function to make values on random examples indistinguishable from random coin flips in the following manner. Let $a \in \{0,1\}^n$ be the vector we want to encode and let $\pi_d : \{0,1\}^n \to \{-1,1\}$ be a pseudo-random function from some pseudorandom function family $\mathcal{F} = \{\pi_b\}_{b \in \{0,1\}^n}$. We define a function $g : \{0,1\}^n \times \{0,1\}^n \to \{-1,1\}$ as

$$g(z,x) = \pi_d(z) \oplus \chi_a(x)$$

($\oplus$ is simply the product in $\{-1,1\}$). The label of a random point $(z,x) \in \{0,1\}^{2n}$ is a XOR of a pseudorandom bit with an independent bit and therefore is pseudorandom. Values of a pseudorandom function b on any polynomial set of distinct points are pseudorandom and therefore random points will have pseudorandom labels as long as their z parts are distinct. In a sample of polynomial in n size of random and uniform points from $\{0,1\}^{2n}$ this happens with overwhelming probability and therefore $g(z,x)$ is not learnable from random examples. On the other hand, for a fixed z, $\pi_d(z) \oplus \chi_a(x)$ gives a Hadamard encoding of a or its negation. Hence it is possible to find a using membership queries with the same prefix. A construction based on a similar idea was used by Elbaz et al. (2007) in their separation result.

Finally, the problem with the construction we have so far is that while a membership query learning algorithm can find the secret a, it cannot predict $g(z,x)$ without knowing d. This means that we also need to provide d to the learning algorithm. It is tempting to use the Hadamard code to encode d together with a. However, a bit of the encoding of d is no longer independent of π_d, and therefore the previous argument does not hold. We are unaware of any constructions of pseudorandom functions that would remain pseudorandom when the value of the function is "mixed" with the description of the function (see the work of Halevi and Krawczyk (2007) for a discussion of this problem). An identical problem also arises in the construction of Elbaz et al. (2007). They used another pseudorandom function π_{d^1} to "hide" the encoding of d, then used another pseudorandom function π_{d^2} to "hide" the encoding of d^1 and so on. The fraction of the domain used up for the encoding of d^i is becoming progressively smaller as i grows. In their construction a PAC learning algorithm can recover as many of the encodings as is required to reach accuracy ε. This method would not be effective in our case. First, in the agnostic setting all the encodings but the one using the largest fraction of the domain can be "corrupted". This makes the largest encoding unrecoverable and implies that the best ε achievable is at most half of the fraction of the domain used by the largest encoding. In addition, in the agnostic setting the encoding of d^i for every odd i can be completely "corrupted" making all the other encodings unrecoverable. To solve these problems in our construction we split the domain into p equal parts and on part i we use a pseudorandom function π_{d^i} to "hide" the encoding of d^j for all $j < i$. In Figure 1 we provide a schematic view of a concept that we construct (for $p = 4$).

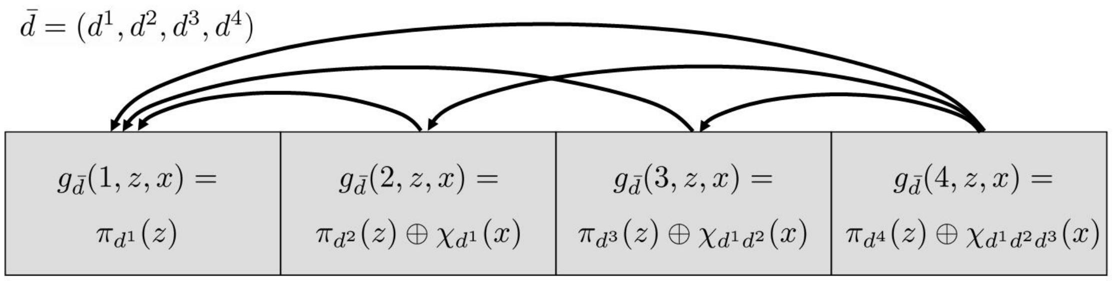

$\bar{d} = (d^1, d^2, d^3, d^4)$
$g_{\bar{d}}(1,z,x) = \pi_{d^1}(z)$
$g_{\bar{d}}(2,z,x) = \pi_{d^2}(z) \oplus \chi_{d^1}(x)$
$g_{\bar{d}}(3,z,x) = \pi_{d^3}(z) \oplus \chi_{d^1 d^2}(x)$
$g_{\bar{d}}(4,z,x) = \pi_{d^4}(z) \oplus \chi_{d^1 d^2 d^3}(x)$

Figure 1: Structure of a concept in $\mathcal{C}_n^p$ for $p = 4$. Arrow from part i to part j indicates that the secret key to part j is encoded using the Hadamard code in part i.

The crucial property of this construction is that the unknown concept can be "recovered" on all but one part of the domain. Specifically, the only part where the unknown concept cannot be "recovered" agnostic is the part i such for all $j > i$ agreement of the target function with every $g_{\bar{d}} \in \mathcal{C}_n^p$ on part j is close to $1/2$ and hence d^j cannot be recovered. Therefore, by making the number of parts p larger than $1/\varepsilon$, we can make sure that there exists an efficient algorithm that finds a hypothesis with the error within ε of the optimum.

4.3 The Construction

We will now describe the construction formally and give a proof of its correctness. Let $p = p(n)$ be a polynomial, let $\ell = \log p(n)$ (we assume for simplicity that $p(n)$ is a power of 2) and let $m = \ell + n \cdot p$. We refer to an element of $\{0,1\}^m$ by triple $(k, z, \bar{x})$ where $k \in [p]$, $z \in \{0,1\}^n$, and

$$\bar{x} = (x^1, x^2, \dots, x^{p-1}) \in \{0,1\}^{n \times (p-1)}.$$

Here k indexes the encodings, z is the input to the k-th pseudorandom function and $\bar{x}$ is the input to a parity function on $n(p-1)$ variables that encodes the secret keys for all pseudorandom functions used for encodings 1 through $k-1$. Formally, let

$$\overline{d} = (d^1, d^2, \dots, d^{p-1})$$

be a vector in $\{0,1\}^{n \times (p-1)}$ (where each $d^i \in \{0,1\}^n$) and for $k \in [p]$ let

$$\overline{d}(k) = (d^1, d^2, \dots, d^{k-1}, 0^n, \dots, 0^n).$$

Let $\mathcal{F} = \{\pi_y\}_{y \in \{0,1\}^*}$ be a pseudorandom function family (Definition 5). We define $g_{\overline{d}} : \{0,1\}^m \to \{-1,1\}$ as follows:

$$g_{\overline{d}}(k, z, \bar{x}) = \pi_{d^k}(z) \oplus \chi_{\overline{d}(k)}(\bar{x}) \ .$$

Finally, we define

$$\mathcal{C}_n^p = \left\{ g_{\overline{d}} \mid \overline{d} \in \{0,1\}^{n \times (p-1)} \right\} \ .$$

4.4 Hardness of Learning $\mathcal{C}_n^p$ From Random Examples

We start by showing that $\mathcal{C}_n^p$ is not agnostically learnable from random and uniform examples only. In fact, we will show that it is not even weakly PAC learnable. Our proof is similar to the proof by Elbaz et al. (2007) who show that the same holds for the concept class they define.

Theorem 9 *There exists no efficient algorithm that weakly PAC learns $\mathcal{C}_n^p$ with respect to the uniform distribution over $\{0,1\}^m$.*

Proof In order to prove the claim we show that a weak PAC learning algorithm for $\mathcal{C}_n^p$ can be used to distinguish a pseudorandom function family from a truly random function. A weak learning algorithm for $\mathcal{C}_n^p$ implies that every function in $\mathcal{C}_n^p$ can be distinguished from a truly random function on $\{0,1\}^m$. If, on the other hand, in the computation of $g_{\overline{d}}(k, z, \bar{x})$ we used a truly random function in place of each $\pi_{d^k}(z)$ then the resulting labels would be truly random and, in particular, unpredictable.

Formally, let `Alg` be a weak learning algorithm for $\mathcal{C}_n^p$ that, with probability at least $1-\delta$, produces a hypothesis with error of at most $1/2 - 1/q(m)$ and runs in time $t(m, 1/\delta)$ for some polynomials $t(\cdot,\cdot)$ and $q(\cdot)$. Our concept class $\mathcal{C}_n^p$ uses numerous pseudorandom functions from F_n and therefore we use a so-called "hybrid" argument to show that one can replace a single $\pi_{d^k}(z)$ with a truly random function to cause `Alg` to fail.

For $0 \le i \le p$, let $\mathbb{O}(i)$ denote an oracle randomly chosen according to the following procedure. First choose randomly and uniformly $\pi_{d^1}, \pi_{d^2}, \dots, \pi_{d^i} \in F_n$ and then choose randomly and uniformly $\rho_{i+1}, \rho_{i+2}, \dots, \rho_p$ from the set of all Boolean functions over $\{0,1\}^n$. Upon request such an oracle returns an example $\langle (k, z, \bar{x}), b \rangle$ where $(k, z, \bar{x})$ is chosen randomly and uniformly from $\{0,1\}^m$ and

$$b = \begin{cases} \pi_{d^k}(z) \oplus \chi_{\overline{d}(k)}(\bar{x}) & 0 \le k \le i, \\ \rho_k(z) & i < k \le p. \end{cases}$$

We note that in order to simulate such an oracle it is not needed to explicitly choose $\rho_{i+1}, \rho_{i+2}, \ldots, \rho_p$ (and, indeed that would not be possible in polynomial time). Instead their values can be generated upon request by flipping a fair coin. This means that for every i, $\mathbb{O}(i)$ can be chosen and then simulated in time polynomial in m and the number of examples requested. Let $M(i)$ denote the algorithm that performs the following steps.

- Choose $\mathbb{O}(i)$ randomly according to the above procedure.
- Simulate `Alg` with random examples from $\mathbb{O}(i)$ and $\delta = 1/2$. Let h be the output of `Alg`.
- Produce an estimate $\tilde{e_h}$ of the error of h on distribution defined by $\mathbb{O}(i)$ that, with probability at least $7/8$, is within $1/(3q(m))$ of the true error. Chernoff bounds imply that this can be done using an empirical estimate on $O(q^2(m))$ random samples.
- Output 1 if $\tilde{e_h} \leq 1/2 - 2/(3q(m))$ and 0 otherwise.

We denote by δ_i the probability that $M(i)$ outputs 1. The probability is taken over all the random choices made by $M(i)$: the random choice and simulation of $\mathbb{O}(i)$, the coin flips of `Alg` and the estimation of the error of h.

Claim 10 $\delta_p - \delta_0 \geq 1/4$.

Proof To see this we first observe that $\mathbb{O}(0)$ is defined using p truly random functions and therefore, the probability that there exists a hypothesis of size at most $t(m,2)$ that has error less than $1/2 - 1/3q(m)$ is some negligible function $\nu(n)$. In particular, the error of the hypothesis produced by `Alg` is at least $1/2 - 1/3q(m)$ (with probability at least $1 - \nu(n)$). This means that $\tilde{e_h} \leq 1/2 - 2/(3q(m))$ only if the estimation fails. By the definition of our error estimation procedure, this happens with probability at most $1/8$ and therefore $\delta_0 \leq 1/8 + \nu(n)$. On the other hand, $\mathbb{O}(p)$ is equivalent to $\mathrm{EX}(U, g_{\bar{d}})$ for some randomly chosen $\bar{d}$. This implies that with probability at least $1/2$, `Alg` outputs a hypothesis with error of at most $1/2 - 1/q(m)$. With probability at least $7/8$, $\tilde{e_h} \leq 1/2 - 2/(3q(m))$, and hence $\delta_p \geq 7/16$. This implies our claim. ■

We now describe our distinguisher M^π, where π denotes the function given to M as an oracle. Let $\mathbb{O}^\pi(i)$ denote the example oracle generated by using π in place of π_{d^i} in the definition of $\mathbb{O}(i)$. That is, first choose randomly and uniformly $\pi_{d^1}, \pi_{d^2}, \ldots, \pi_{d^{i-1}} \in F_n$ and then choose randomly and uniformly $\rho_{i+1}, \rho_{i+2}, \ldots, \rho_p$ from the set of all Boolean functions over $\{0,1\}^n$. Upon request $\mathbb{O}^\pi(i)$ returns an example $\langle (k,z,\bar{x}), b\rangle$ where $(k,z,\bar{x})$ is chosen randomly and uniformly from $\{0,1\}^m$ and

$$b = \begin{cases} \pi_{d^k}(z) \oplus \chi_{\bar{d}(k)}(\bar{x}) & k < i, \\ \pi(z) \oplus \chi_{\bar{d}(k)}(\bar{x}) & k = i, \\ \rho_k(z) & k > i. \end{cases}$$

Similarly, we denote by $M^\pi(i)$ the algorithm that is the same as $M(i)$ but chooses a random $\mathbb{O}^\pi(i)$ in place of $\mathbb{O}(i)$. The distinguishing test M^π chooses a random $i \in [p]$ and runs $M^\pi(i)$.

We first observe that if π is chosen randomly from F_n then choosing and simulating a random $\mathbb{O}^\pi(i)$ is equivalent to choosing and simulating a random $\mathbb{O}(i)$. Therefore for every $i \in [p]$, $M^\pi(i)$ is equivalent to $M(i)$. This implies that in this case M^π will output 1 with probability

$$\frac{1}{p}\sum_{i \in [p]} \delta_i .$$

On the other hand, if π is chosen randomly from the set of all Boolean function over $\{0,1\}^n$ then $\mathbb{O}^{\pi}(i)$ is equivalent to $\mathbb{O}(i-1)$. Therefore in this case M^{π} will output 1 with probability

$$\frac{1}{p}\sum_{i\in[p]}\delta_{i-1}.$$

Therefore, by Claim 10 this implies that the difference in the probability that M outputs 1 in these two cases is

$$\frac{1}{p}\sum_{i\in[p]}\delta_i - \frac{1}{p}\sum_{i\in[p]}\delta_{i-1} = \frac{1}{p}(\delta_p - \delta_0) \geq 1/(4p),$$

that is, non-negligible.

The efficiency of M follows readily from the efficiency of `Alg` and the efficiency of the steps we described. This gives us the contradiction to the pseudorandomness property of function family $\mathcal{F}$. ∎

4.5 Agnostic Learning of $\mathcal{C}_n^p$ with Membership Queries

We now describe a (fully) agnostic learning algorithm for $\mathcal{C}_n^p$ that uses membership queries and is successful for any $\varepsilon \geq 1/p(n)$.

Theorem 11 *There exists a randomized algorithm* `AgnLearn` *that for every distribution $A = (U,\phi)$ over $\{0,1\}^m \times \{-1,1\}$ and every $\varepsilon \geq 1/p(n), \delta > 0$, given access to MEM$(A)$, with probability at least $1-\delta$, finds h such that $\Delta(A,h) \leq \Delta(A,\mathcal{C}_n^p)+\varepsilon$. The probability is taken over the coin flips of MEM(A) and* `AgnLearn`. `AgnLearn` *runs in time polynomial in m and $\log(1/\delta)$.*

Proof Let $g_{\overline{e}}$ for $\overline{e} = (e^1,e^2,\ldots,e^{p-1}) \in \{0,1\}^{(p-1)\times n}$ be the function for which $\Delta(A,g_{\overline{e}}) = \Delta(A,\mathcal{C}_n^p)$. The goal of our algorithm is to find the largest j such that on random examples from the j-th part of the domain (i.e., for $k=j$) A agrees with the encoding of $\overline{e}(j) = (e^1,e^2,\ldots,e^{j-1},0^n,\ldots,0^n)$ with probability at least $1/2+\varepsilon/4$. Using Goldreich-Levin algorithm such j can be used to recover $\overline{e}(j)$ and therefore will allow us to reconstruct $g_{\overline{e}}$ on all points $(k,z,\overline{x})$ for $k<j$. For points with $k \geq j$, our hypothesis will be either constant 1 or constant -1, whichever has the higher agreement with A. This guarantees that the error on this part is at most $1/2$. By the definition of j, $g_{\overline{e}}$ has error of at least $1/2-\varepsilon/4-1/(2p) \geq 1/2-\varepsilon$ on this part of the domain and therefore our hypothesis has error close to that of $g_{\overline{e}}$.

We now describe `AgnLearn` formally. For every $i\in[p]$ and $y\in\{0,1\}^n$, let $A_{i,y}$ be A restricted to points in $\{0,1\}^m$ with prefix i,y. That is $A = (U_{(p-1)\times n}, \phi_{i,y})$ where $\phi_{i,y}(\overline{x}) \equiv \phi(i,y,\overline{x})$ and $U_{(p-1)\times n}$ is the uniform distribution over $\{0,1\}^{(p-1)\times n}$. Note that MEM$(A_{i,y})$ can be simulated using MEM(A): when queried on a point $\overline{x} \in \{0,1\}^{(p-1)\times n}$ MEM$(A_{i,y})$ returns the answer of MEM(A) on point $(i,y,\overline{x})$. Further, for each vector $\overline{d} \in \{0,1\}^{(p-1)\times n}$ and $b\in\{-1,1\}$, let $h_{\overline{d},i,b}$ be defined as

$$h_{\overline{d},i,b}(k,z,\overline{x}) = \begin{cases} \pi_{d^k}(z) \oplus \chi_{\overline{d}(k)}(\overline{x}) & k<i, \\ b & k \geq i. \end{cases} \tag{4}$$

(Here π_{d^k} is an element of the pseudorandom function family F_n used in the construction.)

`AgnLearn` performs the following steps.

1. Initializes $H = \{h_1, h_{-1}\}$, where $h_1 \equiv 1$ and $h_{-1} \equiv -1$.
2. For each $2 \leq i \leq p$:
   (a) Chooses r independent random and uniform points in $\{0,1\}^n$, for r to be defined later. Denote the obtained set of points by Y_i.
   (b) For each $y \in Y_i$:
      i. Runs `GL`$(\varepsilon/4, 1/2)$ over $\{0,1\}^{(p-1)\times n}$ using MEM$(A_{i,y})$. Let T denote the set of indices of heavy Fourier coefficients returned by `GL`.
      ii. For each vector $\overline{d} \in T$ and $b \in \{-1,1\}$ adds $h_{\overline{d},i,b}$ to the set of hypotheses H.
3. For each $h \in H$, estimates $\Delta(A,h)$ to within accuracy $\varepsilon/8$ and with overall confidence $1-\delta/2$ using the empirical error on random samples from A. Chernoff bounds imply that this can be done using samples of size $O(\log(|H|/\delta)/\varepsilon^2)$. Denote the estimate obtained for a hypothesis $h_{\overline{d},i,b} \in H$ by $\tilde{\Delta}_{\overline{d},i,b}$.
4. Returns $h \in H$ with the lowest empirical error.

Claim 12 *For $r = O(\log(1/\delta)/\varepsilon)$, with probability at least $1-\delta$,* `AgnLearn` *returns h such that $\Delta(A,h) \leq \Delta(A, \mathcal{C}_n^p) + \varepsilon$.*

Proof We show that in the set H of hypotheses considered by `AgnLearn` there will be a hypothesis h' such that $\Delta(A,h') \leq \Delta(A, g_{\overline{e}}) + 3\varepsilon/4$ (with sufficiently high probability). The estimates of the error of each hypothesis are within $\varepsilon/8$ of the true error and therefore the hypothesis h with the smallest empirical error will satisfy

$$\Delta(A,h) \leq \Delta(A,h') + \varepsilon/4 \leq \Delta(A, g_{\overline{e}}) + \varepsilon \ .$$

For $i \in [p]$, denote

$$\Delta_i = \mathbf{Pr}_A[b \neq g_{\overline{e}}(k,z,\overline{x}) \mid k = i]$$

(here and below by probability with respect to A we mean that a labeled example $\langle (k,z,\overline{x}), b\rangle$ is chosen randomly according to A). By the definition,

$$\frac{1}{p}\sum_{i\in[p]} \Delta_i = \Delta(A, g_{\overline{e}}). \tag{5}$$

Let j be the largest $i \in [p]$ that satisfies $\Delta_i \leq 1/2 - \varepsilon/4$. If j is undefined (when no i satisfies the condition) then by Equation (5), $\Delta(A, g_{\overline{e}}) > 1/2 - \varepsilon/4$. Either h_1 or h_{-1} has error of at most $1/2$ on A and therefore there exists $h' \in H$ such that $\Delta(A,h') \leq \Delta(A, g_{\overline{e}}) + 3\varepsilon/4$.

We can now assume that j is well-defined. For $i \in [p]$ and $y \in \{0,1\}^n$ denote

$$\Delta_{i,y} = \mathbf{Pr}_A[b \neq g_{\overline{e}}(k,z,\overline{x}) \mid k = i,\ z = y] = \mathbf{Pr}_{\langle \overline{x},b\rangle \sim A_{i,y}}[b \neq g_{\overline{e}}(i,y,\overline{x})].$$

The function $g_{\overline{e}}(j,y,\overline{x})$ equals $\pi_{d^j}(y) \cdot \chi_{\overline{e}(j)}(\overline{x})$. If $\pi_{d^j}(y) = 1$ then by Equation (1) and the definition of $A_{j,y}$,

$$\Delta_{j,y} = \frac{1 - \widehat{\phi_{j,y}}(\overline{e}(j))}{2} \ ,$$

and therefore $\widehat{\phi_{j,y}}(\bar{e}(j)) = 1 - 2\Delta_{j,y}$. If $\pi_{d^j}(y) = -1$ then

$$\Delta_{j,y} = 1 - \frac{1 - \widehat{\phi_{j,y}}(\bar{e}(j))}{2} = \frac{1 + \widehat{\phi_{j,y}}(\bar{e}(j))}{2}$$

and thus $\widehat{\phi_{j,y}}(\bar{e}(j)) = -(1 - 2\Delta_{j,y})$. In either case

$$|\widehat{\phi_{j,y}}(\bar{e}(j))| \geq 1 - 2\Delta_{j,y}. \tag{6}$$

By the definition,

$$\mathbf{E}_{y\in\{0,1\}^n}[\Delta_{i,y}] = \Delta_i.$$

This implies that for a randomly and uniformly chosen y, with probability at least $\varepsilon/4$, $\Delta_{j,y} \leq 1/2 - \varepsilon/8$. This is true since otherwise

$$\Delta_j \geq (1 - \frac{\varepsilon}{4})(\frac{1}{2} - \frac{\varepsilon}{8}) > \frac{1}{2} - \frac{\varepsilon}{4},$$

contradicting the choice of j.

Together with Equation (6) we obtain that for a randomly chosen y, with probability at least $\varepsilon/4$, $|\widehat{\phi_{j,y}}(\bar{e}(j))| \geq \varepsilon/4$. In this case, by Theorem 4, $\texttt{GL}(\varepsilon/4, 1/2)$ with access to $\mathrm{MEM}(A_{j,y})$ will return $\bar{e}(j)$ with probability at least $1/2$ (possibly, among other vectors). This means that $\bar{e}(j)$ will be found with probability at least $\varepsilon/8$. By taking $r = 8\ln(2/\delta)/\varepsilon$ we ensure that `AgnLearn` finds $\bar{e}(j)$ with probability at least $1 - \delta/2$.

Now let b_j be the constant with the lowest error on examples from A for which $k \geq j$, that is

$$b_j = \texttt{sign}(\mathbf{E}_A[b \mid k \geq j]).$$

Clearly, the error of b_j on A when $k \geq j$ is at most $1/2$. By the definition of $h_{\bar{e}(j),j,b_j}$ (Equation 4), $h_{\bar{e}(j),j,b_j}$ equals $g_{\bar{e}}$ on points for which $k < j$ and equals b_j on the rest of the domain. Therefore

$$\begin{aligned}\Delta(A, h_{\bar{e}(j),j,b_j}) &= \frac{j-1}{p}\mathbf{Pr}_A[b \neq g_{\bar{e}}(k,z,\bar{x}) \mid k < j] + \frac{p-j+1}{p}\mathbf{Pr}_A[b \neq b_j \mid k \geq j] \\ &\leq \frac{1}{p}\left(\sum_{i<j}\Delta_i + \frac{p-j+1}{2}\right).\end{aligned}$$

On the other hand, by the properties of j, for all $i > j$, $\Delta_i \geq 1/2 - \varepsilon/4$ and thus

$$\Delta(A, g_{\bar{e}}) = \frac{1}{p}\left(\sum_{i\in[p]}\Delta_i\right) \geq \frac{1}{p}\left(\sum_{i<j}\Delta_i + (p-j)\left(\frac{1}{2} - \frac{\varepsilon}{4}\right)\right).$$

By combining these equations we obtain that

$$\Delta(A, h_{\bar{e}(j),j,b_j}) - \Delta(A, g_{\bar{e}}) \leq \frac{1}{2p} + \frac{\varepsilon}{4} \leq \frac{3\varepsilon}{4}.$$

As noted before, this implies the claim. ∎

Given Claim 12, we only need to check that the running time of `AgnLearn` is polynomial in m and $\log(1/\delta)$. By Parseval's identity there are $O(1/\varepsilon^2)$ elements in each set of vectors returned by `GL` and $r = 8\ln(2/\delta)/\varepsilon$. Therefore the efficiency of `GL` implies that H is found in polynomial time and the size of H is $O(p \cdot \log(1/\delta)/\varepsilon^3)$. This implies that the error estimation step of `AgnLearn` takes polynomial time. ■

Remark 13 *In Theorem 11 we assumed that MEM(A) is not persistent. If MEM(A) is persistent then executions of* `GL` *for different y's are not completely independent and* `GL` *might fail with some negligible probability. A simple and standard modification of the analysis (as in the work of Bshouty et al. (2004) for example) can be used to show that the probability of failure of* `AgnLearn` *in this case is negligible. This implies that* `AgnLearn` *agnostically learns C_n^p from persistent membership queries.*

4.6 Bounds on ε

In Theorem 11 C_n^p is defined over $\{0,1\}^m$ for $m = n \cdot p(n) + \log p(n)$ and is learnable agnostically for any $\varepsilon \geq 1/p(n)$. This means that this construction cannot achieve dependence on ε beyond $1/m$. To improve this dependence we use a more efficient list-decodable code in place of the Hadamard code. Specifically, we need a list-decodable code $C : \{0,1\}^u \rightarrow \{0,1\}^v$ that can be list-decoded from $(1/2 - \gamma')v$ errors in time polynomial in u and $1/\gamma'$ for any $\gamma' \geq \varepsilon/8$. Guruswami and Sudan (2000) gave a list-decoding algorithm for the Reed-Solomon code concatenated with the Hadamard code that has the desired properties for $v = O(u^2/\varepsilon^4)$. Note that this is exponentially more efficient than the Hadamard code for which $v = 2^u$. In fact for this code we can afford to read the whole corrupt message in polynomial time. This means that we can assume that the output of the list-decoding algorithm is exact (and not approximate as in the case of list-decoding using the Hadamard code using the Goldreich-Levin algorithm).

In our construction $u = n(p(n) - 1)$. To apply the above code we index a position in the code using $\log v = O(\log(n/\varepsilon))$ bits. Further we can use pseudorandom functions over $\{0,1\}^{n/2}$ instead of $\{0,1\}^n$ in the definition of C_n^p. We would then obtain that the dimension of C_n^p is $m = n/2 + \log v + \log p(n) \leq n$ for any polynomial $p(n)$ and $\varepsilon \geq 1/p(n)$. This implies that our learning algorithm is successful for every $\varepsilon \geq 1/p(n) \geq 1/p(m)$. It is easy to verify that Theorems 9 and 11 still hold for this variant of the construction and imply Theorem 8.

5. Discussion

Our results clarify the role of membership queries in agnostic learning. They imply that in order to extract any meaningful information from membership queries the learner needs to have significant prior knowledge about the distribution of examples. Specifically, either the set of possible classification functions has to be restricted (as in the PAC model) or the set of possible marginal distributions (as in distribution-specific agnostic learning).

A interesting result in this direction would be a demonstration that membership queries are useful for distribution-specific agnostic learning of a natural concept class such as halfspaces.

Finally, we would be interested to see a proof that membership queries are useful in distribution-specific agnostic learning that places no restriction on ε.

Acknowledgments

I thank Parikshit Gopalan, Salil Vadhan and David Woodruff for valuable discussions and comments on this research. I also thank anonymous reviewers of COLT 2008 and JMLR for their numerous helpful comments.

References

D. Angluin. Queries and concept learning. *Machine Learning*, 2:319–342, 1988.

A. Blum, M. Furst, M. Kearns, and R. J. Lipton. Cryptographic primitives based on hard learning problems. In *Proceedings of International Cryptology Conference on Advances in Cryptology (CRYPTO)*, pages 278–291, 1993.

A. Blum, A. Kalai, and H. Wasserman. Noise-tolerant learning, the parity problem, and the statistical query model. *Journal of the ACM*, 50(4):506–519, 2003.

N. Bshouty. Exact learning via the monotone theory. *Information and Computation*, 123(1):146–153, 1995.

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, J. Jackson, and C. Tamon. More efficient PAC-learning of DNF with membership queries under the uniform distribution. *Journal of Computer and System Sciences*, 68(1):205–234, 2004.

R. Dudley. Central limit theorems for empirical measures. *Annals of Probability*, 6(6):899–929, 1978.

A. Elbaz, H. Lee, R. Servedio, and A. Wan. Separating models of learning from correlated and uncorrelated data. *Journal of Machine Learning Research*, 8:277–290, 2007.

V. Feldman. Optimal hardness results for maximizing agreements with monomials. In *Proceedings of Conference on Computational Complexity (CCC)*, pages 226–236, 2006.

V. Feldman. Attribute efficient and non-adaptive learning of parities and DNF expressions. *Journal of Machine Learning Research*, (8):1431–1460, 2007.

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.

V. Feldman and S. Shah. Separating models of learning with faulty teachers. *Theoretical Computer Science*, doi:10.1016/j.tcs.2009.01.017, 2009.

S. A. Goldman, S. Kwek, and S. D. Scott. Agnostic learning of geometric patterns. *Journal of Computer and System Sciences*, 62(1):123–151, 2001.

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.

P. Gopalan, A. Kalai, and A. Klivans. Agnostically learning decision trees. In *Proceedings of STOC*, pages 527–536, 2008.

G. Guruswami and M. Sudan. List decoding algorithms for certain concatenated codes. In *Proceedings of STOC*, pages 181–190, 2000.

V. Guruswami and P. Raghavendra. Hardness of learning halfspaces with noise. In *Proceedings of FOCS*, pages 543–552, 2006.

S. Halevi and H. Krawczyk. Security under key-dependent inputs. In *CCS '07: Proceedings of the 14th ACM conference on Computer and communications security*, pages 466–475, 2007.

J. Håstad. Some optimal inapproximability results. *Journal of the ACM*, 48(4):798–859, 2001. ISSN 0004-5411.

J. Håstad, R. Impagliazzo, L. Levin, and M. Luby. A pseudorandom generator from any one-way function. *SIAM Journal on Computing*, 28(4):1364–1396, 1999.

D. Haussler. Decision theoretic generalizations of the PAC model for neural net and other learning applications. *Information and Computation*, 100(1):78–150, 1992. ISSN 0890-5401.

A. Kalai, A. Klivans, Y. Mansour, and R. Servedio. Agnostically learning halfspaces. *SIAM Journal on Computing*, 37(6):1777–1805, 2008a.

A. Kalai, Y. Mansour, and E. Verbin. Agnostic boosting and parity learning. In *Proceedings of STOC*, pages 629–638, 2008b.

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

M. Kharitonov. Cryptographic lower bounds for learnability of boolean functions on the uniform distribution. *Journal of Computer and System Sciences*, 50:600–610, 1995.

E. Kushilevitz and Y. Mansour. Learning decision trees using the Fourier spectrum. *SIAM Journal on Computing*, 22(6):1331–1348, 1993.

W. S. Lee, P. L. Bartlett, and R. C. Williamson. On efficient agnostic learning of linear combinations of basis functions. In *Proceedings of COLT*, pages 369–376, 1995.

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

N. Linial, Y. Mansour, and N. Nisan. Constant depth circuits, Fourier transform and learnability. *Journal of the ACM*, 40(3):607–620, 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.

R. O'Donnell and R. Servedio. Learning monotone decision trees in polynomial time. In *Proceedings of IEEE Conference on Computational Complexity*, pages 213–225, 2006.

D. Pollard. *Convergence of Stochastic Processes*. Springer-Verlag, 1984.

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

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

V. Vapnik. *Statistical Learning Theory*. Wiley-Interscience, New York, 1998.

V. Vapnik and A. Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probab. and its Applications*, 16(2):264–280, 1971.

Journal of Machine Learning Research 10 (2009) 183-206 Submitted 1/08; Revised 9/08; Published 2/09

Using Local Dependencies within Batches to Improve Large Margin Classifiers

Volkan Vural VVURAL@ECE.NEU.EDU
Department of Electrical and Computer Engineering
Northeastern University
Boston, MA 02115

Glenn Fung GLENN.FUNG@SIEMENS.COM
Balaji Krishnapuram BALAJI.KRISHNAPURAM@SIEMENS.COM
Computer Aided Diagnosis and Therapy
Siemens Medical Solutions
Malvern, PA 19355

Jennifer G. Dy JDY@ECE.NEU.EDU
Department of Electrical and Computer Engineering
Northeastern University
Boston, MA 02115

Bharat Rao BHARAT.RAO@SIEMENS.COM
Computer Aided Diagnosis and Therapy
Siemens Medical Solutions
Malvern, PA 19355

Editor: Lyle Ungar

Abstract

Most classification methods assume that the samples are drawn independently and identically from an unknown data generating distribution, yet this assumption is violated in several real life problems. In order to relax this assumption, we consider the case where batches or groups of samples may have internal correlations, whereas the samples from different batches may be considered to be uncorrelated. This paper introduces three algorithms for classifying all the samples in a batch jointly: one based on a probabilistic formulation, and two based on mathematical programming analysis. Experiments on three real-life *computer aided diagnosis* (CAD) problems demonstrate that the proposed algorithms are significantly more accurate than a naive support vector machine which ignores the correlations among the samples.

Keywords: batch-wise classification, support vector machine, linear programming, machine learning, statistical methods, unconstrained optimization

1. Introduction

Most classification systems assume that the data used to train and test the classifier are drawn from an independent and identically distributed underlying distribution. For example, samples are classified one at a time in a *support vector machine* (SVM), thus the classification of a particular test sample does not depend on the features from any other test sample. Nevertheless, this assumption

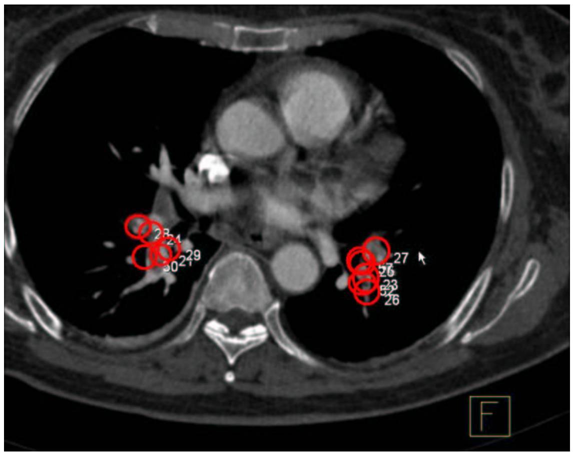


Figure 1: Two pulmonary emboli (blood clots) as they are detected by the Candidate Generation algorithm in a CT image. The candidates are shown as five circles for the left embolus & six circles for the right embolus. The disease status of spatially overlapping or proximate candidates is highly correlated.

is commonly violated in many real-life problems where sub-groups of samples have a high degree of correlation amongst both their features and their labels.

A sample problem with the characteristics described above is *computer aided diagnosis* (CAD), where the goal is to detect structures of interest to physicians in medical images: for example, to identify potentially malignant tumors in computed tomography (CT) scans, X-ray images, etc. In an almost universal paradigm for CAD algorithms, this problem is addressed by a three-stage system: (1) identification of potentially unhealthy candidates *regions of interest* (ROI) from a medical image, (2) computation of descriptive features for each candidate, and (3) classification of each candidate (e.g., normal or diseased) based on its features.

Figure 1 displays a CT image of a lung showing circular marks that point to potential diseased candidate regions that are detected by a CAD algorithm. There are five candidates on the left and six candidates on the right (marked by circles) in Figure 1. Descriptive features are extracted for each candidate and each candidate region is classified as normal or abnormal. In this setting, correlations exist among the candidates belonging to the same batch (image, in the case of CAD) both in the training data set and in the unseen testing data. Further, the level of correlation is a function of the pairwise distance between candidates: the disease status (class-label) of a candidate is highly correlated with the status of other spatially proximate candidates, but the correlations decrease as the distance is increased. Most conventional CAD algorithms classify one candidate at a time, ignoring the correlations amongst the candidates in an image. By explicitly accounting for the correlation structure between the labels of the test samples, classification accuracy can be improved.

In this paper, we present two large margin classifiers and a batch-wise version of logistic regression that take into account the local correlations among samples in a batch. We use the term *batch-wise classification* to mean training and testing classifiers in *batches* or groups (i.e., samples in the same batch are learned and classified collectively). Our experiments on three real CAD problems show that indeed incorporating the local dependencies among samples within a batch improve the classification accuracy significantly compared to standard SVMs.

Beyond the domain of CAD applications, our algorithms are quite general and may be used for batch-wise classification problems in many other contexts. In general, the proposed classifiers can be used whenever data samples are presented in independent batches. In the CAD example, a batch corresponds to an image and strong correlations among candidate samples are based on spatial proximity within an image. A similar relationship structure is true with geographical classification of regions in satellite images; an image would be a batch and spatial proximity corresponds to correlations among the pixel instances. In other contexts, a batch may correspond to data from the same hospital, the patients treated by the same doctor or nurse, etc. In non-medical contexts, a batch can be transactions from the same person, or documents from the same author. Another example is in credit card transaction approval where a single transaction has to be classified by taking into account both the entire set of available training transactions and other transactions made by the same person recently during the testing phase. In this credit card example, it might be preferable to test the recent transaction of this same person as a batch collectively rather than singly.

1.1 Related Work

We are not aware of previous work that exploit internal correlations among the samples to improve the performance of standard classifiers (such as, SVM). However, there exist classification algorithms that incorporate special relationships among the samples (the Markov property) into their model formulation. In natural language processing (NLP), *conditional random fields* (CRF) (Lafferty et al., 2001) and recently *maximum margin Markov* (MMM) networks (Taskar et al., 2004) are used to identify part-of-speech information about words by using the context of nearby words. CRF are also used in similar applications in spoken word recognition. For images, Markov random fields (MRF) (Besag, 1974; Geman and Geman, 1984) are applied to capture correlation among neighborhood pixels.

However, while these Markov models are also fairly general algorithms, they are both computationally demanding and are not easy to implement for problems where the relationship structure among the samples is in any form other than a linear chain (as in the text and speech processing applications). Certainly, their application would be difficult in many large-scale medical applications where run time requirements would be quite severe. For example, in the CAD applications shown in our experiments, the run-time of the testing phase usually has to be less than a second in order that the end user's (radiologist's) time would not be wasted.

Our algorithm is also related to the *multiple instance learning* (MIL) problem, where one is given bags (batches) of samples; class labels are provided only for the bags, not for the individual samples. A bag is labeled positive if we know that at least one sample from it is positive, and a negative bag is known to not contain any positive sample. In this manner, the MIL problem also encodes a form of prior knowledge about correlations between the labels of the training instances.

There are two differences between our algorithm and MIL. First, we want to classify each instance (candidate) in our algorithm; unlike MIL, we are not only trying to label a bag of related instances. Second, unlike the MIL problem which treats all the instances in a bag as equally related to each other, we account for more fine grained differences in the level of correlation between samples (via the similarity matrix $\mathbf{R}$).

Local learning algorithms such as, Wu and Schlkopf (2007) and Bottou and Vapnik (1992), also capture the correlation among data points. However, one major difference between our method and local learning based methods is that our methods build a global classifier that combines the global information extracted from the entire training data with the additional side-information extracted from correlations in the same batch, whereas the local learning algorithms rely on the local information only. Another major difference is that local learning algorithms classify a particular point based on its neighbors in the feature space, therefore the captured correlation is limited to the neighbors of that particular point. On the other hand, the methods that we propose in this paper can incorporate *any* type of correlation information that is not fully encoded in the features. The same difference is also valid between our method and semi-supervised algorithms such as, Vural et al. (2008) and Zhou et al. (2003) that exploit the idea of using local information in semi-supervised learning. The graph based semi-supervised algorithms such as, Krishnapuram et al. (2005) and Zhu et al. (2003), can be considered to be related to the methods introduced in this paper in the sense that they build weighted graphs based on the correlation among data points. However, the targeted problems in semi-supervised learning and this paper are substantially different. In semi-supervised learning problems, the goal is to make use of unlabeled data to improve classification accuracy. In our methods, on the other hand, we investigate the relations among the data points that are in the same batch only. These data points may not constitute a neighborhood in the feature space and every data point is labeled in our problem.

1.2 Organization of the Paper

After briefly describing the notation in Section 2, we build two SVM based methods for classifying batches of correlated samples in Section 3. We also provide a probabilistic interpretation of our approach in Section 4.

Unlike the previous methods such as CRF, MMM and MRF, the proposed algorithms are easy to implement for arbitrary relationships between samples, and further we are able to run these fast enough to be viable in commercial CAD products. In Section 5, we provide experimental evidence from three different CAD problems to show that the proposed algorithms are more accurate in terms of the metrics appropriate to CAD as compared to a naive SVM which is routinely used for these problems as the state-of-the-art in the current literature and commercial products. We conclude with a review of our contributions in Section 6.

2. Notations

Throughout this paper, we will use the following notations. The capital letters shown in bold font represent matrices whereas lower case letters in bold correspond to vectors. The notation $\mathbf{A} \in \mathbb{R}^{m \times n}$ will signify a real $m \times n$ matrix. For such a matrix, $\mathbf{A}'$ will denote the transpose of $\mathbf{A}$ and $\mathbf{A}_i$ will denote the i-th row of $\mathbf{A}$. All vectors will be column vectors.

For $\mathbf{x} \in \mathbb{R}^n$, $\|\mathbf{x}\|_p$ denotes the p-norm, $p = 1, 2, \infty$. A vector of ones in a real space of arbitrary dimension will be denoted by $\mathbf{e}$. Thus, for $\mathbf{e} \in \mathbb{R}^m$ and $\mathbf{y} \in \mathbb{R}^m$, $\mathbf{e}'\mathbf{y}$ is the sum of the components of $\mathbf{y}$. A vector of zeros in a real space of arbitrary dimension will be denoted by 0. The identity matrix of arbitrary dimension will be denoted by $\mathbf{I}$. A *separating hyperplane*, with respect to two given point sets $\mathbf{A}$ and $\mathbf{B}$, is a plane that attempts to separate $\mathbb{R}^n$ into two half spaces such that each open half space contains points mostly of $\mathbf{A}$ or $\mathbf{B}$.

3. A Mathematical Programming Approach

In this section, we formulate the problem of learning for batch-wise prediction as an SVM-like mathematical program.

3.1 Standard Support Vector Machine

In a standard SVM hyperplane classifier, $f(\mathbf{x}) = \mathbf{x}'\mathbf{w} - \gamma$ is learned from the training instances individually, ignoring the correlations among them. Consider the problem of classifying m points in the n-dimensional real space $\mathbb{R}^n$, represented by the $m \times n$ matrix $\mathbf{A}$, according to class membership of each point $\mathbf{x}_i$ (i^{th} row of $\mathbf{A}$) in the classes $\mathbf{A}+$, $\mathbf{A}-$ as specified by a given $m \times m$ diagonal matrix $\mathbf{D}$ with $+1$ or -1 along its diagonal. The standard 1-norm support vector machine with a linear kernel (Vapnik, 1995; Bradley and Mangasarian, 1998) is given by the following linear program with parameter $\nu > 0$:

$$\begin{aligned} \min_{(\mathbf{w},\gamma,\xi,\mathbf{v})\in\mathbb{R}^{n+1+m+n}} \quad & \nu\mathbf{e}'\xi + \mathbf{e}'\mathbf{v} \\ \text{s.t.} \quad \mathbf{D}(\mathbf{A}\mathbf{w} - \mathbf{e}\gamma) + \xi &\geq \mathbf{e} \\ \mathbf{v} \geq \mathbf{w} &\geq -\mathbf{v} \\ \xi &\geq 0 \end{aligned} \tag{1}$$

where, $\xi \in \mathbb{R}^m$ is a vector of slack variables, ν is the cost parameter, and at a solution, $\mathbf{v} = |\mathbf{w}|$ is the absolute value of $\mathbf{w}$. We used the 1-norm linear programming formulation instead of the more popular 2-norm quadratic programming (QP) formulation since the 1-norm formulation is known to produce solutions that depend on fewer features. In other words, by solving the 1-norm formulation, an implicit feature selection is obtained automatically. This is an advantage in CAD applications because the initial pool typically contains a large set of experimental features, although often only a small subset of these features is typically found to be very useful for the classification task at the end of design (i.e., training) stage. It is important to note that all the batch-based classification approaches presented in this paper are equally applicable even when the standard QP SVM formulation is considered.

3.2 Batch Support Vector Machine

The idea in our model is to encode that the classification of any sample (i.e., data point) depends on two factors: (a) the features that encode information about each candidate sample in a batch; and (b) additional side-information—not already fully encoded in the features—that is available to describe the level of correlations among the set of samples within a batch.[1]

Let the covariance matrix $\mathbf{R}^j$ represent the correlations among the samples withing the j^{th} batch. As mentioned above, in CAD applications, $\mathbf{R}^j$ can be defined in terms of $\mathbf{S^j}$, the matrix of Euclidean distances between candidates inside a medical image (from a patient) as:

1. We define a batch to be a set of possibly correlated samples that occur together naturally, for example, the set of all candidate locations from the image of the same patient.

$$\mathbf{R}^j_{pq} = \begin{cases} 0 & ,p=q \\ \exp(-\alpha \mathbf{S}^j_{pq}) & ,o.w. \end{cases}$$

with $\alpha > 0$ as a model parameter. Accordingly, $\mathbf{B}^j \in \mathbb{R}^{m_j \times n}$ represents the m_j training points that belong to the j^{th} batch and the labels of these training points are represented by the $m_j \times m_j$ diagonal matrix $\mathbf{D}^j$ with positive or negative ones along its diagonal. Using the new notations, the standard SVM set of constraints: $\mathbf{D}(\mathbf{Aw} - \mathbf{e}\gamma) + \xi \geq \mathbf{e}$ can be presented as:

$$\mathbf{D}^j(\mathbf{B}^j\mathbf{w} - \mathbf{e}\gamma) + \xi^j \geq \mathbf{e}, \text{ for } j = 1, \ldots, k. \tag{2}$$

Here, k is defined as the number of batches in $\mathbf{A}$ and the vector of slack variables, ξ, in Equation (1) corresponds to $\xi = [\xi^{1'}, \xi^{2'}, \ldots, \xi^{k'}]'$. In Equation (2), every data point in $\mathbf{B}^j$ sets a constraint on $\mathbf{w}$ and γ *independently*, that is, the i^{th} constraint is set by only the i^{th} data point, $\mathbf{B}^j_i$, ignoring the correlations between $\mathbf{B}^j_i$ and the rest of the samples in $\mathbf{B}^j$. Therefore, we modify the standard SVM set of constraints in order to take into account the correlations among samples in the same batches, using the $\mathbf{R}^j$ matrix as:

$$\mathbf{D}^j\left[\left(\mathbf{I} + \theta\mathbf{R}^j\right)\left(\mathbf{B}^j\mathbf{w} - \mathbf{e}\gamma\right)\right] + \xi^j \geq \mathbf{e}, \text{ for } j = 1, \ldots, k. \tag{3}$$

Here, θ is a parameter that helps us control the contribution of the correlation information encoded by $\mathbf{R}^j$. We can find the optimum value of θ by tuning over a validation set. However, we also present a method where we can learn θ automatically in Section 3.3.

In Equation (3), the class membership prediction for any single sample in batch j is a weighted average of the batch members' prediction vector $\mathbf{B}^j\mathbf{w} - \mathbf{e}\gamma$, and the weighting coefficients depend on the pairwise similarity between samples. Using constraints (3) in the SVM formulation (1), we obtain the optimization problem for learning *BatchSVM* with parameters ν and θ:

$$\begin{aligned} \min_{(\mathbf{w},\gamma,\xi,\mathbf{v}) \in \mathbb{R}^{n+1+m+n}} \quad & \nu\mathbf{e}'\xi + \mathbf{e}'\mathbf{v} \qquad (4) \\ \text{s.t.} \quad \mathbf{D}^j\left[\left(\theta\mathbf{R}^j + \mathbf{I}\right)\left(\mathbf{B}^j\mathbf{w} - \mathbf{e}\gamma\right)\right] + \xi^j \geq \mathbf{e}, & \quad \text{for } j = 1, \ldots, k \\ \mathbf{v} \geq \mathbf{w} \geq -\mathbf{v} & \\ \xi \geq 0. & \end{aligned}$$

When $\theta = 0$, the constraints in Equation (4) become standard SVM constraints. Therefore, if the information provided in $\mathbf{R}^j$ matrix does not help our algorithm improve the accuracy performance of SVM, we expect our method to turn into the standard SVM.

Unlike standard SVMs, the hyperplane ($f(\mathbf{x}) = \mathbf{w}'\mathbf{x} - \gamma$) produced by *BatchSVM* is *not* the final decision function. We refer to $f(\mathbf{x})$ as a pre-classifier that will be used in the next stage to make the final decision on a batch of instances. While testing an arbitrary data point $\mathbf{x}^j_i$ in batch $\mathbf{B}^j$, the *BatchSVM* algorithm accounts for the pre-classifier prediction $\mathbf{w}'\mathbf{x}^j_p - \gamma$ for every member in the batch. The final prediction $\hat{f}(\mathbf{x}^j_i)$ is given by:

$$sign(\hat{f}(\mathbf{x}^j_i)) = sign(\mathbf{w}'\mathbf{x}^j_i - \mathbf{e}\gamma + \theta\mathbf{R}^j_i\left[\mathbf{B}^j\mathbf{w} - \mathbf{e}\gamma\right]). \tag{5}$$

Consider the two dimensional example in Figure 2, showing batches of training points. The data points that belong to the same batch are indicated by the elliptical boundaries in the figure. Figure 2b displays the correlations amongst the training points given in Figure 2a using an edge.

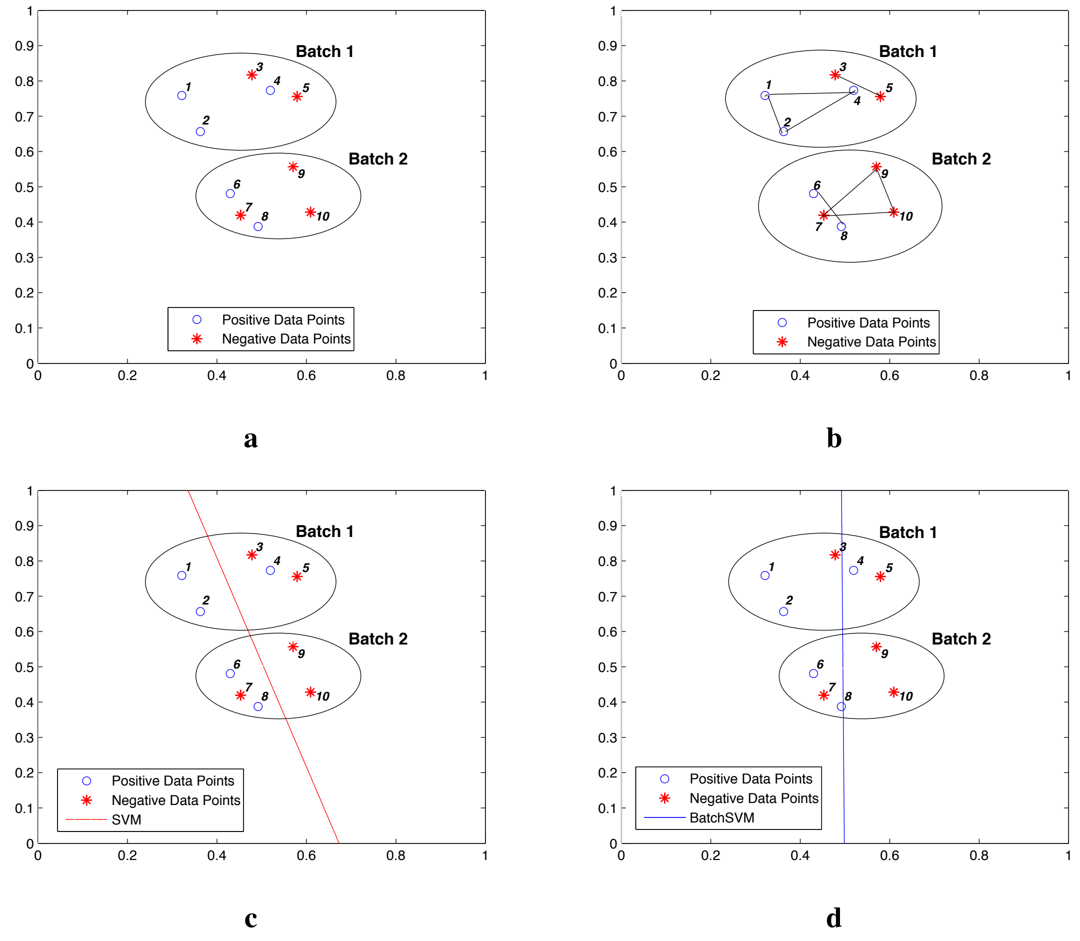


Figure 2: An illustrative example for batch learning. **a**) Training data points are displayed in batches. **b**) Relations within training points are displayed as a linked graph. **c**) Classifier produced by SVM. **d**) Pre-classifier produced by $BatchSVM$. Unlike standard SVMs, the hyperplane, $f(\mathbf{x})$, produced by $BatchSVM$ (preclassifier) is not the decision function. Instead, the decision of each test sample $\mathbf{x}_i$, is based on a weighted average of the $f(\mathbf{x})$ values for the points linked to $\mathbf{x}_i$.

In Figure 2c, the hyperplane $f_{svm}(\mathbf{x})$ is the final decision function for standard SVM and gives the results displayed in Table 1 where we observe that the fourth and the seventh instances are misclassified. In Figure 2d, the pre-classifier produced by $BatchSVM$, $f_{batch}(\mathbf{x})$ gives the results displayed in the fifth column of Table 1 for the training data. If this pre-classifier were to be considered as the decision function, then three training points would be misclassified. However, during batch-testing (Equation 5), the predictions of those points are corrected as seen in the sixth column of Table 1.

Point	Batch	Label	SVM	Pre-classifier	Final classifier
1	1	+	0.2826	0.1723	0.1918
2	1	+	0.2621	0.1315	0.2122
3	1	-	-0.2398	**0.0153**	-0.0781
4	1	+	**-0.3188**	**-0.0259**	0.2909
5	1	-	-0.4787	-0.0857	-0.0276
6	2	+	0.2397	0.0659	0.0372
7	2	-	**0.2329**	**0.0432**	-0.0888
8	2	+	0.1490	0.0042	0.0680
9	2	-	-0.2525	-0.0752	-0.1079
10	2	-	-0.2399	-0.1135	-0.1671

Table 1: Outputs of the classifier produced by SVM, pre-classifier and the final classifier produced by *BatchSVM*. The outputs are calculated for the data points presented in Figure 2. The first column of the table indicates the order of the data points as they are presented in Figure 2a and the second column specifies the corresponding labels. Misclassified points are displayed in bold. Notice that the combination of the pre-classifier outputs at the final stage corrects the mistakes.

3.3 A Faster Proximal Batch Approach

Support vector machines are known to be slow to train. In Fung and Mangasarian (2001), Fung and Mangasarian introduced proximal support vector machine which is a fast and simple algorithm for generating a linear or nonlinear classifier that merely requires the solution of a single system of linear equations. To yield a faster training method, we present a batch approach to proximal support vector machine in this section.

Similar to the proximal support vector machine formulation (PSVM) in Fung and Mangasarian (2001), we can slightly modify the set of inequalities (3) and consider a Gaussian prior (instead of a Laplacian) for both the error vector ξ and the regularization term, to obtain the following unconstrained formulation:

$$\min_{(\mathbf{w},\gamma)\in\mathbb{R}^{n+1}} \nu\frac{1}{2}\sum_{j=1}^{k} \|\mathbf{e}-\mathbf{D}^j\left[\left(\theta\mathbf{R}^j+\mathbf{I}\right)\left(\mathbf{B}^j\mathbf{w}-\mathbf{e}\gamma\right)\right]\|^2+\frac{1}{2}(\mathbf{w}'\mathbf{w}+\gamma^2). \tag{6}$$

Note that formulation (6) is a strongly convex (quadratic) unconstrained optimization problem and it has a unique solution. Then, obtaining the solution to formulation (6) requires only solving a single system of linear equations; therefore, it should be substantially faster than the linear programming approach of the standard QP approach while maintaining approximate accuracy results (Fung and Mangasarian, 2001). Moreover, this simple formulation allows us to introduce a very small modification to learn the parameter θ in a very simple but effective manner:

$$\min_{(\mathbf{w},\gamma,\theta)\in\mathbb{R}^{n+1+1}} \nu\frac{1}{2}\sum_{j=1}^{k} \|\mathbf{e}-\mathbf{D}^j\left[\left(\theta\mathbf{R}^j+\mathbf{I}\right)\left(\mathbf{B}^j\mathbf{w}-\mathbf{e}\gamma\right)\right]\|^2+\frac{1}{2}(\mathbf{w}'\mathbf{w}+\gamma^2+\theta^2). \tag{7}$$

This formulation is strongly convex in all its variables $(\mathbf{w}, \gamma, \theta)$ and it has a unique solution as well. The optimization problem (7) can be solved in many different ways, however we chose to use an alternating optimization approach similar to the one used in Fung et al. (2004), since it suits the structure of the problem perfectly and it converged rapidly in our experiments. It took nine iterations on average for the algorithm to converge in our experiments. The alternate optimization method used can be summarized as follows:

0 *initialization*: Given an initial θ_0, the parameter ν, the training data $A = \bigcup_{j=1}^{k} \mathbf{B}^j$ and the corresponding labels $\mathbf{D}$.

1. Solve optimization problem (6) for $\theta = \theta^{i-1}$ obtain $\mathbf{w}^i$ and γ^i.

2. For the obtained $\mathbf{w}^i$ and γ^i, solve the following linear system of equations:

$$\theta^i = \frac{-\sum_{j=1}^{k} M^j}{2\sum_{j=1}^{k} N^j}$$

where

$$M^j \quad = \mathbf{S}^{j\prime}\mathbf{Z}_j\mathbf{S}^j + 2\mathbf{e}'\mathbf{D}^j\mathbf{R}^j\mathbf{S}^j \text{ , for } \mathbf{S}^j = \left(\mathbf{B}^j\mathbf{w} - \mathbf{e}\gamma\right) \text{ and } \mathbf{Z}_j = (\mathbf{R}^{j\prime} + \mathbf{R}^j),$$

$$N^j \quad = \mathbf{Y}^{j\prime}\mathbf{Y}^j + \tfrac{1}{\nu} \text{ , for } \mathbf{Y}^j = \mathbf{R}^j\left(\mathbf{B}^j\mathbf{w} - \mathbf{e}\gamma\right).$$

This is an iterative method with two steps. At the first step, it finds the optimum values of $\mathbf{w}$ and γ ($\mathbf{w}_{opt}, \gamma_{opt}$) for a given θ. At the second step, for the given $\mathbf{w}_{opt}$ and γ_{opt}, it finds the optimum value of θ (θ_{opt}), which will be used at the first leg of the next iteration. Appendix A.2 provides a derivation of the update equations. Initially, we assign a random value for θ, the procedure is repeated until it converges to the global optimum solution for $\mathbf{w}$, γ and θ.

4. A Probabilistic Batch Classification Model

Let $\mathbf{x}_i^j \in \mathbb{R}^n$ represent the n features for the i^{th} candidate in the j^{th} image, and let $\mathbf{w} \in \mathbb{R}^n$ be the weight parameters and $\gamma \in \mathbb{R}^1$ be the offset of some hyperplane classifier. Traditionally, classifiers label samples one at a time (i.e., independently) based only on the features that describe a given sample $\mathbf{x}_i^j$:

$$z_i^j \quad = \mathbf{w}'\mathbf{x}_i^j - \gamma \quad = (\mathbf{x}_i^j)'\mathbf{w} - \gamma \quad , z_i^j \in \mathbb{R}^1.$$

For example, in logistic regression, the posterior probability of the sample $\mathbf{x}_i^j$ belonging to class $+1$ is obtained using the sigmoid function $P(y_i^j = 1|\mathbf{x}_i^j) = \frac{1}{1+\exp(-\mathbf{w}'\mathbf{x}_i^j+\gamma)}$.

We claim that z_i^j is only a noisy observation of the underlying, unobserved variable $u_i^j \in \mathbb{R}^1$ that actually influences classification (as opposed to the traditional classification approach, where classification directly depends on z_i^j). Beside the features $\mathbf{x}_i^j$ that are available to influence the classification of a sample point (via u_i^j), we have side information about the extent of correlation amongst the quantities u_i^j. We mathematically model the side information as an a-priori guess or intuition about u_i^j even before we observe any $\mathbf{x}_i^j$ (therefore before z_i^j).

For example, in the context of CAD, features describe the shape, size, texture, and intensity profile of a candidate (sample) in an image. However, besides the above information encoded in the form of features that describe each candidate, we also know that correlation amongst the samples in an image (batch) purely based on the proximity of the spatial locations of samples in the j^{th} image. In CAD applications, this spatial adjacency induces correlations in the predictions for the labels, but in other applications the nature of side-information that accounts for the correlations would be different. Nevertheless, in general (in a domain independent way) we model this side-information as a Gaussian prior on u_i^j.

$$P(\mathbf{u}^j \in \mathbb{R}^{n_j}) \quad = N(\mathbf{u}^j | 0, \mathbf{R}^j) \tag{8}$$

where n_j is the number of the candidates in the j^{th} image, and the covariance matrix $\mathbf{R}^j$ encodes the correlations. As mentioned above, in CAD applications, $\mathbf{R}^j$ can be defined in terms of $\mathbf{S}$, the matrix of Euclidean distances between candidates inside a medical image (from a patient) as $\mathbf{R}^j = \exp(-\alpha\mathbf{S})$, with $\alpha > 0$ as a model parameter.

Having defined a prior, next we define the likelihood as follows:

$$P(z_i^j | u_i^j) \quad = N(z_i^j | u_i^j, \sigma^2). \tag{9}$$

After observing $\mathbf{x}_i^j$ and therefore z_i^j, we can modify our prior intuition about $\mathbf{u}^j$ in (8), based on our observations from (9) to obtain the Bayesian posterior:

$$P(\mathbf{u}^j | \mathbf{z}^j) \quad = N\left(\mathbf{u}^j | (\mathbf{R}^{j^{-1}}\sigma^2 + \mathbf{I})^{-1}\mathbf{z}^j; (\mathbf{R}^{j^{-1}} + \tfrac{1}{\sigma^2}\mathbf{I})^{-1}\right) \tag{10}$$

The class-membership prediction for the i^{th} candidate in the j^{th} image is controlled exclusively by u_i^j. Let $\mathbf{B}^j \in \mathbb{R}^{m_j \times n}$ represent the m_j training points that belong to the j^{th} batch. Exact computation of the integral required for $P(\mathbf{y}^j = 1 | \mathbf{B}^j, \mathbf{w}, \gamma, \alpha, \sigma^2)$ is analytically infeasible because of the nonlinear sigmoid terms. However, by using the *maximum-a-posteriori* (MAP) estimate of u_i^j, the prediction probability for class label, y_i^j is then determined approximately as:

$$\begin{gathered} P(y_i^j = 1 | \mathbf{B}^j, \mathbf{w}, \gamma, \alpha, \sigma^2) \approx 1/\left(1 + \exp\left(-\mathbf{V}_i^j[\mathbf{B}^j\mathbf{w} - \mathbf{e}\gamma]\right)\right), \\ \text{where} \\ \mathbf{V}^j = \left[\mathbf{R}^{j^{-1}}\sigma^2 + \mathbf{I}\right]^{-1}. \end{gathered} \tag{11}$$

Note, however, that for each batch j, the probabilistic method requires calculating two matrix inversions to compute $\left(\mathbf{R}^{j^{-1}}\sigma^2 + \mathbf{I}\right)^{-1}$. Hence, training and testing using this method can be time consuming for large batch sizes. On the other hand, $\left(\mathbf{R}^j\theta + \mathbf{I}\right)$ that we introduced in Equation (3) is relatively less time consuming to compute.

In general, $\left(\mathbf{R}^{j^{-1}}\sigma^2 + \mathbf{I}\right)^{-1} \neq \left(\mathbf{R}^j\theta + \mathbf{I}\right)$. However, the intuition behind both are similar: *the class prediction for each sample is a weighted sum of the prediction values of the test sample and the samples that are correlated with the test instance*. In the appendix A.1, we show that when σ is small and we are considering a least squared loss function as in Section 3.3, the constraint (3) results in an approximation to the following constraint:

$$\mathbf{D}^j \left[\left(\mathbf{R}^{j^{-1}}\sigma^2 + \mathbf{I}\right)^{-1}(\mathbf{B}^j\mathbf{w} - \mathbf{e}\gamma)\right] + \xi^j \geq \mathbf{e}, \ \text{ for } j = 1, \ldots, k \tag{12}$$

when the sum of the labels of neighboring data points according to each row of $\mathbf{R}^j$ is the same or they have similar values. Note that, this is particularly true when $\mathbf{R}^j$ only contains information about correlations for a constant number of nearest neighbors for each sample (k-nearest neighbors) and when the assumption that neighboring segments should have similar labels holds.

4.1 Learning in this Model

For batch-wise prediction using (11), $\mathbf{w}, \alpha$ and σ^2 can be learned from a set of N training images via *maximum-a-posteriori* (MAP) estimation of $\mathbf{w}$ and *maximum-likelihood* estimation of α, σ^2 as follows:

$$[\hat{\mathbf{w}}, \hat{\gamma}, \hat{\alpha}, \hat{\sigma^2}] \quad = \arg\max_{\mathbf{w},\gamma,\alpha,\sigma^2} P(\mathbf{w}) \prod_{j=1}^{N} P(\mathbf{y}^j | \mathbf{B}^j, \mathbf{w}, \gamma, \alpha, \sigma^2)$$

where, $P(\mathbf{y}^j | \mathbf{B}^j, \mathbf{w}, \gamma, \alpha, \sigma^2)$ is defined as in (11) and $P(\mathbf{w})$ may be assumed to be Gaussian $N(\mathbf{w}|0, \lambda)$. The regularization parameter λ is typically chosen by cross-validation. General purpose numerical optimizers such as conjugate gradient algorithms may be used for the above optimization problem.

4.2 Intuition about Batch Classification

Equations (10) and (11) imply that $\mathbb{E}[\mathbf{u}^j | \mathbf{z}^j] = (\mathbf{R}^{j^{-1}} \sigma^2 + \mathbf{I})^{-1} \mathbf{z}^j$. In other words, the class membership prediction for any single sample is a weighted average of the noisy prediction quantity $\mathbf{z}^j$ (distance to the hyperplane), where the weighting coefficients depend on the pairwise Euclidean distances between samples. Hence, the intuition presented above is that we predict the classes for all the n_j candidates in the j^{th} image together, as a function of the features for all the candidates in the batch (here a batch corresponds to an image from a patient). In every test image, *each* of the candidates is classified using the features from *all* the samples in the image. Note that although we classify each instance in a batch at the same time, our algorithm outputs a class for each instance and the class membership for these samples need not be the same.

5. Experiments

In this section, we compare regular SVM, probabilistic batch learning (*BatchProb*), and *BatchSVM*. Note that in order to make the comparison consistent, *BatchProb* was implemented by using the same linear programming formulation as *BatchSVM* but using Equation (12) instead of Equation (3). We also compare PSVM based techniques introduced in Section 3: (1) the proximal version of batch learning presented in (6) where θ is determined as a parameter via cross-validation ($BatchPSVM - cv$), and (2) the alternating optimization method presented in (7), where θ is learned automatically ($BatchPSVM - learned$).

5.1 Implementation Details

We present the details of the similarity function that we used in our experiments and how we tuned the parameters of our methods in the next sections.

5.1.1 THE SIMILARITY FUNCTION

As mentioned earlier, the matrix $\mathbf{R}^j$ represents the level of correlation between all pairs of candidates from a batch (an image in our case) and it is a function of the pairwise-similarity between the candidates. One needs to produce the $\mathbf{R}^j$ matrix using the appropriate similarity metric that represents the pairwise-similarities the best for a particular application.

In our CAD applications, we have used the Euclidean distances between candidates inside a medical image as our similarity metric to define the covariance matrix $\mathbf{R}^j$. Let $\mathbf{r}_p$ and $\mathbf{r}_q$ represent the coordinates of two candidates, $\mathbf{B}_p^j$ and $\mathbf{B}_q^j$ on the j^{th} image. For our experiments, we used the Euclidean distance between $\mathbf{r}_p$ and $\mathbf{r}_q$ to define the pairwise-similarity, $s(p,q)$, between $\mathbf{B}_p^j$ and $\mathbf{B}_q^j$ as following:

$$s(p,q) = e^{-\frac{\|\mathbf{r}_p - \mathbf{r}_q\|^2}{\varsigma^2}}.$$

If the matrices $s(p,q)$ are sparse, it aids the computational efficiency of our algorithms, both in terms of memory use and speed of implementation. Note also that the matrix is naturally sparse, due to the rapid decrease of the exponential function. Nevertheless, to make the implementation stable and memory efficient (by avoiding storing all the elements of these matrices), we found it useful to discretize the continuous similarity function, $s(p,q)$ to the binary similarity function, $s^*(p,q)$ by applying a threshold as following:

$$s^*(p,q) = \begin{cases} 0 & s(p,q) < e^{-4}, \\ 1 & s(p,q) \geq e^{-4}. \end{cases} \tag{13}$$

In all experiments, we set the threshold at e^{-4} to provide us with a similarity of one if the neighbor is at a 95% confidence level of belonging to the same density as the candidate assuming that the neighborhood is a Gaussian distribution with mean equal to the candidate and variance $\varsigma^2 = \frac{1}{\alpha}$. Each element of the matrix $\mathbf{R}$ is given by:

$$\mathbf{R}_{pq} = s^*(p,q).$$

In order to observe the effects of binarization of the $\mathbf{R}$ matrix, we also run experiments on *BatchSVM* without applying the threshold in Equation (13), that is, $\mathbf{R} = s(p,q)$. We refer to this version of *BatchSVM* as $BatchSVM_{cont}$ in our experiments.

5.1.2 PARAMETER TUNING

All our parameters in these experiments are tuned by 10-fold patient cross-validation on the training data (i.e., the training data is split into ten folds). During cross-validation, a range of parameters $(\theta, \sigma, \varsigma)$ were evaluated for the proposed methods. In $BatchPSVM - learned$, the parameter θ is learned automatically through equation (7).

5.2 Evaluation Method

As we explained earlier, $\mathbf{R}$ used in the proposed methods is produced from the spatial locations of the candidates. These spatial locations can be considered as extra information provided to our methods. In order to make fair comparisons between our methods and standard SVM and PSVM, we added the spatial locations (x,y,z coordinates of the candidates in 3D image) to the feature vectors of the candidates.

Receiver Operating Characteristic (ROC) plots are used to study the classification accuracy of these techniques on three CAD applications for detecting pulmonary embolism, colon cancer, and

	PE		Colon		Lung	
	train	test	train	test	train	test
SVM	16.87	0.00	7.22	0.00	1.58	0.00
$BatchProb$	19.26	4.71	105.14	18.63	46.42	30.02
$BatchSVM$	11.39	0.23	13.45	0.91	11.73	2.81
$BatchSVM_{cont}$	13.42	0.74	61.09	5.93	14.69	3.01
PSVM	0.22	0.00	0.13	0.00	0.05	0.00
$BatchPSVM-cv$	1.02	0.23	3.67	0.89	4.80	2.82
$BatchPSVM-learned$	1.39	0.33	4.08	0.89	5.44	2.79

Table 2: Training and testing times for each classifier are displayed in seconds.

lung cancer. In clinical practice, CAD systems are evaluated on the basis of a somewhat domain-specific metric: to maximize the fraction of *positives* that are correctly identified by the system while displaying at most a clinically acceptable number of false-marks per image. We report this domain-specific metric in an ROC plot, where the y-axis is a measure of sensitivity and the x-axis is the number of false-marks per patient (in our case, per image is also per patient). Sensitivity is the number of patients diagnosed as having the disease divided by the number of patients that has the disease. High sensitivity and low false-marks are desired. All classification algorithms are trained on a training data set and evaluated on a sequestered (held-out) test set.

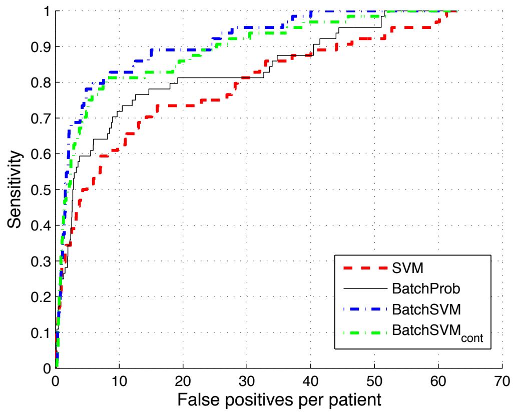
The area under the ROC curve is a commonly used criteria to evaluate the performance of an algorithm. Table 3 displays the area under the ROC curve of each classifier. It is important to note that the areas under the curve presented in Table 3 are calculated considering the entire false positive range which tends to underestimate the superiority of our proposed approach with respect to traditional approaches around the range of false positives acceptable for CAD applications.

PSVM, $BatchPSVM-cv$ and $BatchPSVM-learned$ algorithms require solving a set of linear equations while training, therefore we expect them to be faster than the other methods mentioned in this paper, namely SVM, $BatchProb$ and $BatchSVM$. In order to compare the time performances of the methods, we measured the CPU time consumed by each classifier. Table 2, displays the training and testing times required by the algorithms. The training times obtained for the parametric methods (SVM, $BatchProb$, $BatchSVM$, PSVM and $BatchPSVM-cv$) are the total times divided by the number of cross-validations to find the best value of θ. On the other hand, the training time for $BatchPSVM-learned$ is the total time divided by the number of iterations that $BatchPSVM-learned$ required while converging to the optimum value of θ. We measured the training times this way to be able to make a fair comparison between the algorithms.

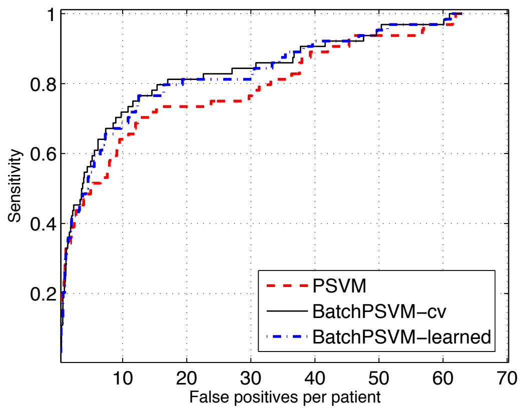
We ran our experiments on an Intel Pentium 4 CPU with 2.8 GHz clock frequency and 1GB RAM and the algorithms were implemented in Matlab.

5.3 Data Sources and Domain Description

When searching for descriptive features, researchers often deploy a large amount of experimental image features to describe the identified candidates, which consequently introduces irrelevant or redundant features. The initial large set of features is based on medically relevant characteristics:



a



b

Figure 3: Experimental results for PE data. ROC curves obtained by (a) SVM, *BatchProb*, *BatchSVM* and $BatchSVM_{cont}$ (b) PSVM, $BatchPSVM-cv$ and $BatchPSVM-learned$

the pixel intensity descriptive statistics, texture, contrast, 2D and 3D shape descriptors, and the location of the corresponding structure.

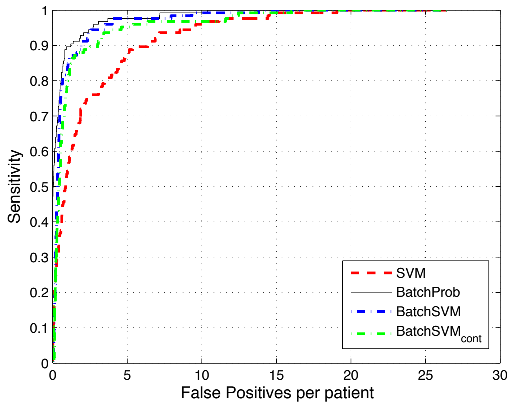
5.3.1 EXAMPLE: PULMONARY EMBOLISM

Pulmonary embolism (PE), a potentially life-threatening condition, is a result of underlying venous thromboembolic disease. An early and accurate diagnosis is the key to survival. Computed tomography angiography (CTA) has emerged as an accurate diagnostic tool for PE. However, there are hundreds of CT slices in each CTA study. Manual reading is laborious, time consuming and complicated by various PE look-alikes (false positives) including respiratory motion artifact, flow-related artifact, streak artifact, partial volume artifact, stair step artifact, lymph nodes, vascular bifurcation among many others (Masutani et al., 2002; Wittram et al., 2004). Several CAD systems are developed to assist radiologists in this process by helping them detect and characterize emboli in an accurate, efficient and reproducible way (Quist et al., 2004; Zhou et al., 2003).

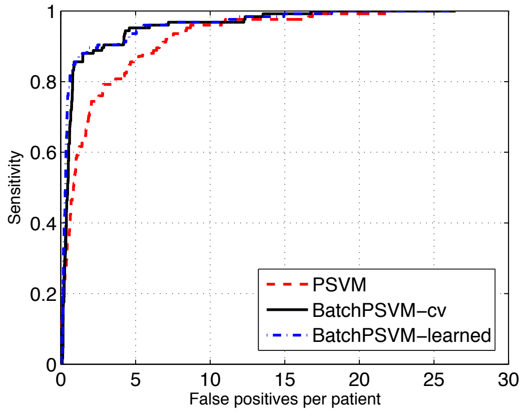
We have collected 72 cases with 242 PEs marked by expert chest radiologists at four different institutions (two North American sites and two European sites). For our experiments, they were randomly divided into two sets: a training and a testing set. The training set was used to train and validate the classifiers and consists of 48 cases with 173 PEs and a total of 3655 candidates. The testing set consists of 24 cases with 69 true PEs out of a total of 1857 candidates. This set was only used to evaluate the performance of the final system. A combined total of 70 features were extracted to describe the shape, texture, size and intensity profile of each candidate.

5.3.2 EXAMPLE: COLON CANCER DETECTION

Colorectal cancer is the third most common cancer in both men and women. It is estimated that in 2004, nearly 147,000 cases of colon and rectal cancer will be diagnosed in the US, and more than 56,730 people would die from colon cancer (Jemal et al., 2004). In over 90% of the cases colon



a



b

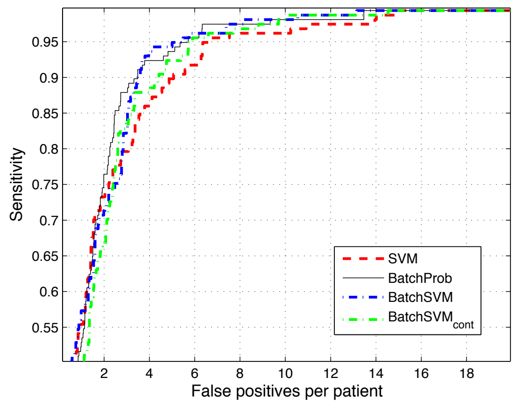
Figure 4: Experimental results for Colon Cancer data. ROC curves obtained by (a) SVM, $BatchProb$, $BatchSVM$ and $BatchSVM_{cont}$ (b) PSVM, $BatchPSVM-cv$ and $BatchPSVM-learned$

cancer progressed rapidly is from local (polyp adenomas) to advanced stages (colorectal cancer), which has very poor survival rates. However, identifying (and removing) lesions (polyp) when still in a local stage of the disease, has very high survival rates, thus illustrating the critical need for early diagnosis. The sizes of a polyp in the colon can vary from 1mm all the way up to 10cm. Most polyps no matter how small they are, are represented by two candidates; one obtained from the prone view and the other one from the supine view. Moreover, for large polyps or so called masses a typical candidate generation algorithm generates several candidates across the polyp surface. Therefore most polyps in the training data are inherently represented by multiple candidates.

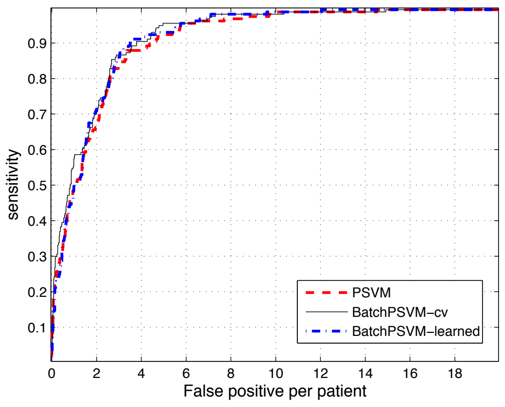
For the sake of clinical acceptability, it is sufficient to detect one of these candidates during classification. Unlike a standard classification algorithm where the emphasis is to accurately classify each and every candidate, here we seek to classify at least one of the candidates representing a polyp accurately. The database of high-resolution CT images used in this study were obtained from seven different sites across US, Europe and Asia. The 188 patients were randomly partitioned into a training and a test set. The training set consists of 65 cases containing 127 volumes. Fifty polyps were identified in this set out of a total of 6748 candidates. The testing set consists of 123 cases containing 237 volumes. There are 103 polyps in this set from a total of 12984 candidates. A total of 75 features were extracted to describe the shape, size, texture and intensity profile of each candidate.

5.3.3 EXAMPLE: LUNG CANCER

LungCAD is a computer aided detection system for detecting potentially cancerous pulmonary nodules from thin slice multi-detector computed tomography (CT) scans. The final output of LungCAD is provided by a classifier that classifies a set of candidates as positive or negative. This is a very hard classification problem: most patient lung CTs contain a few thousand structures (candidates),



a



b

Figure 5: Experimental results for Lung Cancer data. ROC curves obtained by (a) SVM, $BatchProb$, $BatchSVM$ and $BatchSVM_{cont}$ (b) PSVM, $BatchPSVM-cv$ and $BatchPSVM-learned$

and only a few (≤ 5 on average) of which are potential nodules that should be identified as positive by LungCAD, all within the run-time requirements of completing the classification on-line during the time the physician completes their manual review.

The training set consists of 60 patients. The number of candidates labeled as nodules in the training set are 157 and the total number of candidates is 9987. The testing set consists of 26 patients. In this testing set, there are 79 candidates labeled as nodules out of 6159 generated candidates. A total of 15 features were used to describe the shape, size, texture and intensity profile of each candidate region.

5.4 Discussion and Results

In our experiments, we would like to find out whether the batch methods are better than the standard SVM, whether $BatchSVM$ is better than $BatchProb$ in terms of accuracy and speed, whether B$atchPSVM$ is as accurate as $BatchSVM$, whether $BatchPSVM$ is faster than $BatchSVM$, and whether the automated tuning works by comparing $BatchPSVM-learned$ with $BatchPSVM-cv$.

5.4.1 SVM versus Batch Methods

Figures 3, 4, and 5 show the ROC curves for pulmonary embolism, colon cancer, and lung cancer data respectively. In our medical applications high-sensitivity is critical as early detection of lung and colon cancer is believed to greatly improve the chances of successful treatment (Bogoni et al., 2005). Furthermore, high specificity is also critical, as a large number of false positives will vastly increase physician load and lead (ultimately) to loss of physician confidence.

In Figure 3a, corresponding to the comparison of the ROC curves on the PE data set, we observe that standard SVM can only achieve 53% sensitivity for six false positives. However, $BatchSVM$ achieves 80% with a remarkable improvement of 27%. $BatchProb$ also outperforms SVM with a

	PE	Colon	Lung
SVM	0.79	0.91	0.98
BatchProb	0.83	**0.98**	**0.99**
BatchSVM	**0.89**	0.96	**0.99**
$BatchSVM_{cont}$	0.88	0.95	**0.99**
PSVM	0.77	0.91	**0.99**
$BatchPSVM-cv$	0.82	0.96	**0.99**
$BatchPSVM-learned$	0.83	0.96	**0.99**

Table 3: Area under the ROC curves for each classifier. (Best in bold).

64% sensitivity. As seen from the figure, the two proposed methods are substantially more accurate than standard SVMs at any specificity level. When we compare the performances of PSVM and $BatchPSVM-learned$ in Figure 3b, we observe that the proposed batch method ($BatchPSVM-learned$) dominates over PSVM. However, they do not improve the accuracy of PSVM as much as *BatchProb* and *BatchSVM* do.

Colon cancer data is a relatively easier data set than pulmonary embolism since standard SVM can achieve 54.5% sensitivity at one false positive level as illustrated in Figure 4a. However, *BatchSVM* improved SVM's performance to 84% sensitivity for the same number of false positives. Note that *BatchProb* improved the sensitivity further, giving 89.6% for the same specifity. In one to ten false positives region which constitutes the region of interest in our applications, our proposed methods outperform standard SVM significantly. PSVM's performance is substantially worse than that of our proposed batch methods, $BatchPSVM-cv$ and $BatchPSVM-learned$.

Although SVM is very accurate for the lung cancer application, Figure 5a shows that *BatchProb* and *BatchSVM* could still improve SVM's performance further. *BatchProb* method is superior to the other methods at two and three false positives per image. Both *BatchProb* and *BatchSVM* outperform SVM in the 2-6 false positives per image region, which is the region of interest for commercial clinical lung CAD systems. All three of the methods are comparable at other specificity levels. On the other hand, we observe in Figure 5b that $BatchPSVM-cv$ and $BatchPSVM-learned$ methods did not achieve significant improvements, yet they are still comparable to PSVM.

When we consider the area under the ROC curve for each of the methods in Table 3, we observe that our proposed methods are better than SVM and PSVM in every case except one where PSVM performs the same as our proposed methods in the lung cancer data.

In order to understand if the proposed methods are statistically significantly better than standard SVM and PSVM, we applied Z-test to compare the areas under the ROC curves derived from the same set of patients as explained in Hanley and McNeil (1983). The corresponding P-values that indicate the level of confidence of the paired comparisons are presented in Table 4. The smaller the P-values, the higher the level of confidence. Looking at the results in Table 4, we can say that the improvements provided by the proposed methods are statistically significant for PE and Colon data sets.

	SVM vs *BatchProb*	SVM vs *BatchSVM*	PSVM vs *BatchPSVM − cv*	PSVM vs *BatchPSVM − learned*
PE	0.05	0.01	0.06	0.05
Colon	0.01	0.01	0.05	0.05
Lung	0.81	0.81	1	1

Table 4: P-values that indicate the confidence level of significance when the proposed algorithms are compared to SVM and PSVM in terms of the area under the ROC curves.

5.4.2 *BatchSVM* VERSUS *BatchProb*

As we discussed in Section 3, *BatchProb* requires calculating two matrix inversions, which can be time consuming for large sizes of batches, whereas *BatchSVM* avoids matrix inversions, therefore *BatchSVM* should be faster than *BatchProb* in both training and testing. As expected, our experiments show us that *BatchSVM* is faster than *BatchProb* in every case. Note that the difference in run times is significant for testing in PE and Colon data sets, where *BatchSVM* is twenty times faster than *BatchProb*. On the other hand, *BatchProb* produced a better ROC curve in Colon data set. Therefore, although *BatchProb* is slow compared to *BatchSVM*, it may be preferable with respect to accuracy depending on the application.

5.4.3 *BatchPSVM − learned* VERSUS *BatchPSVM − cv*

BatchPSVM − cv requires cross-validated tuning over the parameter θ to achieve the optimum accuracy. One problem with tuning is that we can only tune the parameter in a limited range. Therefore a good guess of the range of the parameter values is crucial. Another problem is that we can only tune the parameter over discrete values which may not include the best parameter value. One way to obtain a close approximation of the best parameter value is to divide the range of the parameter into small steps. However this method will increase the training time since cross-validation will be repeated as many times as the possible parameter values (in our experiments, we cross-validated θ over 21 values). On the other hand, automated tuning (*BatchPSVM − learned*) is quite feasible and avoids the problems stated above. In our experiments, *BatchPSVM − learned* was able to automatically converge to the optimum value of θ in only nine iterations on average while producing comparable ROC plots with cross-validated tuning (*BatchPSVM − cv*) as seen in Figures 3b, 4b and 5b. The optimum θ values obtained by cross-validated tuning and automated tuning are presented in Table 5. As observed in the table, the optimum θ values found by both algorithms are very similar.

When we compare the two methods with respect to training times, we observe that *BatchPSVM − cv* is faster than *BatchPSVM − learned* for a fixed value of θ. However, since we repeated cross-validation 21 times for *BatchPSVM − cv* and *BatchPSVM − learned* required nine iterations on average, *BatchPSVM − learned* is actually faster than *BatchPSVM − cv* in overall while training.

5.4.4 *BatchPSVM* VERSUS *BatchSVM*

BatchSVM results in a linear programming problem, whereas *BatchPSVM* requires solving a set of linear equations. Therefore, we expect *BatchPSVM* to be faster than *BatchSVM* while training. Experimental results on training times displayed in Table 2 confirm that it takes less time for

	PE	Colon	Lung
$BatchPSVM-cv$	1	0.6	0.1
$BatchPSVM-learned$	1.11	0.68	0.03

Table 5: θ values obtained by $BatchPSVM-cv$ and $BatchPSVM-learned$.

$BatchPSVM$ in the training phase. However, despite the training time performance, $BatchSVM$ produced better ROC curves than that of $BatchPSVM$ in PE and lung cancer data sets as illustrated in Figures 3 and 5 respectively. The area under the ROC curve produced by $BatchSVM$ is also better than $BatchPSVM$ for these two data sets. For Colon data set, in Figure 4b, we observe that $BatchPSVM-cv$ and $BatchSVM$ performed slightly better than $BatchSVM-learned$ where both achieve 86% sensitivity at one false positive level. Furthermore, the total area under the ROC curves produced by $BatchSVM$ and $BatchPSVM$ are the same (0.96). From the experimental results, we can draw the conclusion that $BatchPSVM$ is a fast alternative to $BatchSVM$ and can be preferable in some applications.

6. Conclusions

Three related algorithms have been proposed for classifying batches of correlated data samples. Although primarily motivated by real-life CAD applications, the problem occurs commonly in many situations; our algorithms are sufficiently general to be applied in other contexts. Experimental results indicate that the proposed method can substantially improve the diagnosis of (a) early stage cancer in the Lung & Colon, and (b) pulmonary embolisms (which may result in strokes). With the increasing adoption of these systems in routine clinical practice, these experimental results demonstrate the potential of our methods to impact a large cross-section of the population.

In this paper, we validated the improvement in diagnosis sensitivity that batch-wise classification provides to CAD problems. The algorithms presented here are general and can be applied to other domains. In other domains, one may apply other similarity metric appropriate for the application. For data with no clear batch information, clustering can be performed to create the batches. Extending the batch algorithms to other data and automatically learning the similarity metric are topics for future work.

Acknowledgments

This research is supported by Siemens Medical Solutions USA, Inc. and NSF CAREER IIS-0347532.

Appendix A.

In this section, we present the connection between Equations (3) and (12), and the derivation steps for automatically tuning the parameters in BatchPSVM.

A.1 Connection between Equations (3) and (12)

We will show the connection between the optimization problems that result by considering the constraints (3) and (12) by proving that the resulting objective functions are closely related to each other.

First, let us show that the inverse $\left(\mathbf{I}+\mathbf{R}^{j^{-1}}\sigma^2\right)^{-1}$ that appears in Equation (12) can be approximated by the expression $\mathbf{I}-\sigma^2\mathbf{R}^{j^{-1}}$.

By replacing $\mathbf{A}=\sigma^2\mathbf{R}^{j^{-1}}$, in the identity:

$$(\mathbf{I}+\mathbf{A})^{-1}=\mathbf{I}-\mathbf{A}+\mathbf{A}^2-\mathbf{A}^3+\ldots+,\forall\mathbf{A}$$

we have:

$$\left(\mathbf{I}+\mathbf{R}^{j^{-1}}\sigma^2\right)^{-1}=\mathbf{I}-\sigma^2\mathbf{R}^{j^{-1}}+\sigma^4\mathbf{R}^{j^{-1}}\mathbf{R}^{j^{-1}}-\ldots$$

where the terms $\sigma^4\mathbf{R}^{j^{-1}}\mathbf{R}^{j^{-1}}-\sigma^6\mathbf{R}^{j^{-1}}\mathbf{R}^{j^{-1}}\mathbf{R}^{j^{-1}}+\ldots$ can be ignored if σ is sufficiently small.

Then, we have:

$$\begin{aligned}\left(\mathbf{I}+\mathbf{R}^{j^{-1}}\sigma^2\right)^{-1} &\cong \mathbf{I}-\sigma^2\mathbf{R}^{j^{-1}}\\ &= -\sigma^2\mathbf{R}^{j^{-1}}\left(\mathbf{I}+\theta\mathbf{R}^j\right)\end{aligned}\tag{14}$$

where $\theta=\frac{-1}{\sigma^2}$ provided that $\sigma\neq 0$.

Let us recall that the quadratic unconstrained objective function that results from the constraint (12) is:

$$\min_{(\mathbf{w},\gamma,\theta)\in\mathbb{R}^{n+1+1}}\nu\frac{1}{2}\sum_{j=1}^{k}\left\|\mathbf{e}-\mathbf{D}^j\left[\left(\sigma^2\mathbf{R}^{j^{-1}}+\mathbf{I}\right)^{-1}(\mathbf{B}^j\mathbf{w}-\mathbf{e}\gamma)\right]\right\|^2+\frac{1}{2}(\mathbf{w}'\mathbf{w}+\gamma^2+\theta^2)\tag{15}$$

and that the quadratic unconstrained objective function that results from the constraint (3) is given by:

$$\min_{(\mathbf{w},\gamma,\theta)\in\mathbb{R}^{n+1+1}}\nu\frac{1}{2}\sum_{j=1}^{k}\left\|\mathbf{e}-\mathbf{D}^j\left[\left(\theta\mathbf{R}^j+\mathbf{I}\right)(\mathbf{B}^j\mathbf{w}-\mathbf{e}\gamma)\right]\right\|^2+\frac{1}{2}(\mathbf{w}'\mathbf{w}+\gamma^2+\theta^2).\tag{16}$$

Next, we will show that the objective functions (15) and (16) are related to each other under certain assumptions. In order to ease the notation, let's now define

$$g(\mathbf{w},\gamma)=\sum_{j=1}^{k}\left\|\mathbf{e}-\mathbf{D}^j\left[\left(\sigma^2\mathbf{R}^{j^{-1}}+\mathbf{I}\right)^{-1}(\mathbf{B}^j\mathbf{w}-\mathbf{e}\gamma)\right]\right\|^2.$$

Then if σ^2 is small we have from (14) that:

$$\begin{aligned}g(\mathbf{w},\gamma) &\approx \textstyle\sum_{j=1}^{k}\left\|\mathbf{e}-\mathbf{D}^j\left[-\sigma^2\mathbf{R}^{j^{-1}}\left(\theta\mathbf{R}^j+\mathbf{I}\right)(\mathbf{B}^j\mathbf{w}-\mathbf{e}\gamma)\right]\right\|^2\\ &= \textstyle\sum_{j=1}^{k}\left\|\left(-\sigma^2\mathbf{R}^{j^{-1}}\right)\left(\left(-\frac{1}{\sigma^2}\mathbf{R}^j\right)\mathbf{e}-\mathbf{D}^j\left[\left(\theta\mathbf{R}^j+\mathbf{I}\right)(\mathbf{B}^j\mathbf{w}-\mathbf{e}\gamma)\right]\right)\right\|^2\\ &\leq \textstyle\sum_{j=1}^{k}\left\|\sigma^2\mathbf{R}^{j^{-1}}\right\|^2\left\|\left(-\frac{1}{\sigma^2}\mathbf{R}^j\right)\mathbf{e}-\mathbf{D}^j\left[\left(\theta\mathbf{R}^j+\mathbf{I}\right)(\mathbf{B}^j\mathbf{w}-\mathbf{e}\gamma)\right]\right\|^2\\ &\leq \Psi\textstyle\sum_{j=1}^{k}\left\|-\frac{1}{\sigma^2}\mathbf{R}^j\mathbf{e}-\mathbf{D}^j\left[\left(\theta\mathbf{R}^j+\mathbf{I}\right)(\mathbf{B}^j\mathbf{w}-\mathbf{e}\gamma)\right]\right\|^2=f(\mathbf{w},\gamma)\end{aligned}$$

where $\Psi = max_j\{\left\|\sigma^2\mathbf{R}^{j^{-1}}\right\|^2\}$, then, if σ is small we can conclude that $f(\mathbf{w},\gamma)$ is approximately a "good" upper bound for $g(\mathbf{w},\gamma)$. Furthermore if $h(\mathbf{w},\gamma,\theta) = \mathbf{w}'\mathbf{w}+\gamma^2+\theta^2$,then $F(\mathbf{w},\gamma,\theta) = \nu\frac{1}{2}f(\mathbf{w},\gamma)+\frac{1}{2}h(\mathbf{w},\gamma,\theta)$ is also a "good" upper bound for $G(\mathbf{w},\gamma,\theta) = \nu\frac{1}{2}g(\mathbf{w},\gamma)+\frac{1}{2}h(\mathbf{w},\gamma,\theta)$.

So intuitively, since $G(\mathbf{w},\gamma,\theta)$ and $F(\mathbf{w},\gamma,\theta)$ are non-negative convex quadratic functions, minimizing $F(\mathbf{w},\gamma,\theta)$ should provide good approximate solutions to the problem of minimizing $G(\mathbf{w},\gamma,\theta)$.

Let us assume now that the number of neighboring points that affect the classification for every point is more or less constant. This is true when $\mathbf{R}$ is binary and calculated only for the k nearest neighbors or when the rows of $\mathbf{R}$ are normalized and approximately true when $\mathbf{R}$ is defined as in (13). This is equivalent to assuming that the summation of all the rows of $\mathbf{R}$ is constant, or equivalently, for every batch j:

$$-\frac{1}{\sigma^2}\mathbf{R}^j\mathbf{e} \approx k\mathbf{e}$$

where k is a constant. hence:

$$f(\mathbf{w},\gamma) = \sum_{j=1}^{k}\left\|k\mathbf{e}-\mathbf{D}^j\left[\left(\theta\mathbf{R}^j+\mathbf{I}\right)\left(\mathbf{B}^j\mathbf{w}-\mathbf{e}\gamma\right)\right]\right\|^2.$$

Then, solving the problem:

$$min_{\mathbf{w},\gamma,\theta}F(\mathbf{w},\gamma,\theta)$$

is equivalent to solving the proximal batch formulation presented in Equation (7) for some $\nu = \bar{\nu}$ (the constant k only rescales the problem in the variables w and γ).

A.2 Derivation for Automatically Tuning the Parameters in BatchPSVM

The optimization problem in (7) is a bi-convex problem and can be solved in an iterative fashion with two steps. At the first step, we find the optimum values of $\mathbf{w}$ and γ ($\mathbf{w}_{opt},\gamma_{opt}$) for a given θ. At the second step, for $\mathbf{w}_{opt}$ and γ_{opt} held fixed, we find the optimum value of θ (θ_{opt}), which will be used at the first leg of the next iteration.

A.2.1 OPTIMIZING FOR w AND γ WITH θ FIXED

Given θ is held fixed, we can optimize (7) with respect to $\mathbf{w}$ and γ, and rewrite the equation as follows:

$$\min_{(\mathbf{w},\gamma)\in\mathbb{R}^{n+1}}\frac{1}{2}\mu\sum_{j=1}^{k}\mathbf{x}'\mathbf{P}^j\mathbf{x}+\mathbf{q}^j\mathbf{x}+K_1,$$

where

$$\mathbf{x} = \begin{bmatrix}\mathbf{w}\\ \gamma\end{bmatrix},$$

$$\mathbf{P}^j = \begin{bmatrix}\frac{1}{\mu}\mathbf{I}+\theta^2\mathbf{B}^{j\prime}\mathbf{R}^{j\prime}\mathbf{R}^j\mathbf{B}^j+\theta\mathbf{B}^{j\prime}(\mathbf{R}^{j\prime}+\mathbf{R}^j)\mathbf{B}^j+\mathbf{B}^{j\prime}\mathbf{B}^j & -\theta^2\mathbf{B}^{j\prime}\mathbf{R}^{j\prime}\mathbf{R}^j\mathbf{e}-\theta\mathbf{B}^{j\prime}(\mathbf{R}^{j\prime}+\mathbf{R}^j)\mathbf{e}-\mathbf{B}^{j\prime}\mathbf{e}\\ -\theta^2\mathbf{e}'\mathbf{R}^{j\prime}\mathbf{R}^j\mathbf{B}^j-\theta\mathbf{e}'(\mathbf{R}^{j\prime}+\mathbf{R}^j)\mathbf{B}^j-\mathbf{e}'\mathbf{B}^j & \theta^2\mathbf{e}'\mathbf{R}^{j\prime}\mathbf{R}^j\mathbf{e}+\theta\mathbf{e}'(\mathbf{R}^{j\prime}+\mathbf{R}^j)\mathbf{e}+\mathbf{e}'\mathbf{e}+\frac{1}{\mu}\end{bmatrix},$$

$$\mathbf{q}^j = 2\begin{bmatrix}-\theta\mathbf{e}'\mathbf{D}^j\mathbf{R}^j\mathbf{B}^j-\mathbf{e}'\mathbf{D}^j\mathbf{B}^j\\ -\theta\mathbf{e}'\mathbf{D}^j\mathbf{R}^j\mathbf{e}+\mathbf{e}'\mathbf{D}^j\mathbf{e}\end{bmatrix}$$

and K_1 is a constant term with respect to $\mathbf{w}$ and γ.

After taking the derivative of the objective function with respect to $\mathbf{x}$ and equating to zero, we can obtain the solution as following:

$$x = -(\sum_{j=1}^{k} \mathbf{P}^j)^{-1}(\sum_{j=1}^{k} \mathbf{q}^j)$$

A.2.2 OPTIMIZING FOR θ WITH $\mathbf{w}$ AND γ FIXED

When $\mathbf{w}$ and γ have fixed values, we can rewrite Equation (7) considering θ is the only variable as following:

$$\min_{(\mathbf{w},\gamma)\in\mathbb{R}^{n+1}} \frac{1}{2}\mu \sum_{j=1}^{k} \theta^2 N^j + \theta M^j + K_2$$

where K_2 is a constant with respect to θ and

$$N^j = \quad \mathbf{w}'\mathbf{B}^{j'}\mathbf{R}^{j'}\mathbf{R}^j\mathbf{B}^j\mathbf{w} - \mathbf{w}'\mathbf{B}^{j'}\mathbf{R}^{j'}\mathbf{R}^j\mathbf{e}\gamma - \gamma\mathbf{e}'\mathbf{R}^{j'}\mathbf{R}^j\mathbf{B}^j\mathbf{w} + \gamma^2\mathbf{e}'\mathbf{R}^{j'}\mathbf{R}^j\mathbf{e} + \frac{1}{\mu}$$

and

$$\begin{aligned} M^j = \quad & -\mathbf{e}'\mathbf{D}^j\mathbf{R}^j\mathbf{B}^j\mathbf{w} + \mathbf{e}'\mathbf{D}^j\mathbf{R}^j\mathbf{e}\gamma - \mathbf{w}'\mathbf{B}^{j'}\mathbf{R}^{j'}\mathbf{D}^{j'}\mathbf{e} + \mathbf{w}'\mathbf{B}^{j'}\mathbf{R}^{j'}\mathbf{B}^j\mathbf{w} - \mathbf{w}'\mathbf{B}^{j'}\mathbf{R}^{j'}\mathbf{e}\gamma \\ & +\gamma\mathbf{e}'\mathbf{R}^{j'}\mathbf{D}^{j'}\mathbf{e} - \gamma\mathbf{e}'\mathbf{R}^{j'}\mathbf{B}^j\mathbf{w} + \gamma\mathbf{e}'\mathbf{R}^{j'}\mathbf{e}\gamma + \mathbf{w}'\mathbf{B}^{j'}\mathbf{R}^j\mathbf{B}^j\mathbf{w} - \mathbf{w}'\mathbf{B}^{j'}\mathbf{R}^j\mathbf{e}\gamma \\ & -\gamma\mathbf{e}'\mathbf{R}^j\mathbf{B}^j\mathbf{w} + \gamma\mathbf{e}'\mathbf{R}^j\mathbf{e} \end{aligned} .$$

M^j can be further simplified as:

$$\begin{aligned} M^j = \quad & \mathbf{w}'\mathbf{B}^{j'}(\mathbf{R}^{j'} + \mathbf{R}^j)\mathbf{B}^j\mathbf{w} - \mathbf{w}'\mathbf{B}^{j'}(\mathbf{R}^{j'} + \mathbf{R}^j)\mathbf{e}\gamma - \gamma\mathbf{e}'(\mathbf{R}^{j'} + \mathbf{R}^j)\mathbf{B}^j\mathbf{w} \\ & +\gamma\mathbf{e}'(\mathbf{R}^{j'} + \mathbf{R}^j)\mathbf{e}\gamma - 2\mathbf{e}'\mathbf{D}^j\mathbf{R}^j\mathbf{B}^j\mathbf{w} + 2\mathbf{e}'\mathbf{D}^j\mathbf{R}^j\mathbf{e}\gamma \end{aligned}$$

using the property that $A' = A$, if A is a scalar.

References

J. Besag. Spatial interaction and the statistical analysis of lattice systems. *Journal of the Royal Statistical Society, Series B*, 36:192–236, 1974.

L. Bogoni, P. Cathier, M. Dundar, A. Jerebko, S. Lakare, J. Liang, S. Periaswamy, M. Baker, and M. Macari. CAD for colonography: A tool to address a growing need. *British Journal of Radiology*, 78:57–62, 2005.

L. Bottou and V.N. Vapnik. Local learning algorithms. *Neural Computation*, 4(6):888–900, 1992. URL `citeseer.ist.psu.edu/bottou92local.html`.

P. S. Bradley and O. L. Mangasarian. Feature selection via concave minimization and support vector machines. In J. Shavlik, editor, *Proc. 15th International Conf. on Machine Learning*, pages 82–90, San Francisco, CA, 1998. Morgan Kaufmann.

G. Fung and O. L. Mangasarian. Proximal support vector machine classifiers. In *Knowledge Discovery and Data Mining*, pages 77–86, 2001.

G. Fung, M. Dundar, J. Bi, and B. Rao. A fast iterative algorithm for fisher discriminant using heterogeneous kernels. In *ICML '04: Proceedings of the twenty-first international conference on Machine learning*, page 40, New York, NY, USA, 2004. ACM Press. ISBN 1-58113-828-5. doi: http://doi.acm.org/10.1145/1015330.1015409.

S. Geman and D. Geman. Stochastic relaxation, Gibbs distribution and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6(6):721–741, 1984.

J.A. Hanley and B.J. McNeil. A method of comparing the areas under receiver operating characteristic curves derived from the same cases. *Radiology*, 148(3):839–843, 1983. URL `http://radiology.rsnajnls.org/cgi/content/abstract/148/3/839`.

D. Jemal, R. Tiwari, T. Murray, A. Ghafoor, A. Samuels, E. Ward, E. Feuer, and M. Thun. Cancer statistics. *CA Cancer J. Clin.*, 54:8–29, 2004.

B. Krishnapuram, D. Williams, Y. Xue, A. Hartemink, L. Carin, and M. Figueiredo. On semi-supervised classification. In *NIPS 17*, pages 721–728. MIT Press, 2005.

J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. 18th International Conf. on Machine Learning*, pages 282–289, San Francisco, CA, 2001. Morgan Kaufmann.

Y. Masutani, H. MacMahon, and K. Doi. Computerized detection of pulmonary embolism in spiral CT angiography based on volumetric image analysis. *IEEE Transactions on Medical Imaging*, 21:1517–1523, 2002.

M. Quist, H. Bouma, C. Van Kuijk, O. Van Delden, and F. Gerritsen. Computer aided detection of pulmonary embolism on multi-detector CT. In *Proceedings of the 90th Meeting of the Radiological Society of North America (RSNA)*, 2004.

B. Taskar, C. Guestrin, and D. Koller. Max-margin Markov networks. In Sebastian Thrun, Lawrence Saul, and Bernhard Schölkopf, editors, *Advances in Neural Information Processing Systems 16*. MIT Press, Cambridge, MA, 2004.

V. N. Vapnik. *The Nature of Statistical Learning Theory*. Springer, New York, 1995.

V. Vural, G. Fung, J. Dy, and Rao B. Fast semi-supervised svm classifiers using a-priori metric information. *Optimization Methods and Software (OMS)*, 23(4):521–532, 2008.

C. Wittram, M. Maher, A. Yoo, M. Kalra, O. Jo-Anne, M. Shepard, and T. McLoud. CT angiography of pulmonary embolism: Diagnostic criteria and causes of misdiagnosis. *RadioGraphics*, 24: 1219–1238, 2004.

M. Wu and B. Schlkopf. Transductive classification via local learning regularization. In *11th International Conference on Artificial Intelligence and Statistics*, pages 628–635, Brookline, MA, USA, 03 2007. Microtome. URL `http://jmlr.csail.mit.edu/proceedings/papers/v2/`.

C. Zhou, L. M. Hadjiiski, B. Sahiner, H.-P. Chan, S. Patel, P. Cascade, E. A. Kazerooni, and J. Wei. Computerized detection of pulmonary embolism in 3D computed tomographic (CT) images: vessel tracking and segmentation techniques. In *Medical Imaging 2003: Image Processing. Edited by Sonka, Milan; Fitzpatrick, J. Michael. Proceedings of the SPIE, Volume 5032, pp. 1613-1620 (2003).*, pages 1613–1620, May 2003. doi: 10.1117/12.481369.

D. Zhou, O. Bousquet, T. Navin Lal, Jason Weston, and Bernhard Schölkopf. Learning with local and global consistency. In *Advances in Neural Information Processing Systems*. MIT Press, 2003.

X. Zhu, Z. Ghahramani, and J. D. Lafferty. Semi-supervised learning using gaussian fields and harmonic functions. In *Proc. 20th Int. Conf. on Machine Learning*, pages 912–919, 2003.

Journal of Machine Learning Research 10 (2009) 207-244 Submitted 12/07; Revised 9/08; Published 2/09

Distance Metric Learning for Large Margin Nearest Neighbor Classification

Kilian Q. Weinberger KILIAN@YAHOO-INC.COM
Yahoo! Research
2821 Mission College Blvd
Santa Clara, CA 9505

Lawrence K. Saul SAUL@CS.UCSD.EDU
Department of Computer Science and Engineering
University of California, San Diego
9500 Gilman Drive, Mail Code 0404
La Jolla, CA 92093-0404

Editor: Sam Roweis

Abstract

The accuracy of k-nearest neighbor (kNN) classification depends significantly on the metric used to compute distances between different examples. In this paper, we show how to learn a Mahalanobis distance metric for kNN classification from labeled examples. The Mahalanobis metric can equivalently be viewed as a global linear transformation of the input space that precedes kNN classification using Euclidean distances. In our approach, the metric is trained with the goal that the k-nearest neighbors always belong to the same class while examples from different classes are separated by a large margin. As in support vector machines (SVMs), the margin criterion leads to a convex optimization based on the hinge loss. Unlike learning in SVMs, however, our approach requires no modification or extension for problems in multiway (as opposed to binary) classification. In our framework, the Mahalanobis distance metric is obtained as the solution to a semidefinite program. On several data sets of varying size and difficulty, we find that metrics trained in this way lead to significant improvements in kNN classification. Sometimes these results can be further improved by clustering the training examples and learning an individual metric within each cluster. We show how to learn and combine these local metrics in a globally integrated manner.

Keywords: convex optimization, semi-definite programming, Mahalanobis distance, metric learning, multi-class classification, support vector machines

1. Introduction

One of the oldest and simplest methods for pattern classification is the k-nearest neighbors (kNN) rule (Cover and Hart, 1967). The kNN rule classifies each unlabeled example by the majority label of its k-nearest neighbors in the training set. Despite its simplicity, the kNN rule often yields competitive results and in certain domains, when cleverly combined with prior knowledge, it has significantly advanced the state-of-the-art (Belongie et al., 2002; Simard et al., 1993).

By the very nature of its decision rule, the performance of kNN classification depends crucially on the way that distances are computed between different examples. When no prior knowledge is available, most implementations of kNN compute simple Euclidean distances (assuming the examples are represented as vector inputs). Unfortunately, Euclidean distances ignore any statistical regularities that might be estimated from a large training set of labeled examples. Ideally, one would like to adapt the distance metric to the application at hand. Suppose, for example, that we are using kNN to classify images of faces by age and gender. It can hardly be optimal to use the same distance metric for age and gender classification, even if in both tasks, distances are computed between the same sets of extracted features (e.g., pixels, color histograms).

Motivated by these issues, a number of researchers have demonstrated that kNN classification can be greatly improved by learning an appropriate distance metric from labeled examples (Chopra et al., 2005; Goldberger et al., 2005; Shalev-Shwartz et al., 2004; Shental et al., 2002). This is the so-called problem of *distance metric learning*. Recently, it has been shown that even a simple linear transformation of the input features can lead to significant improvements in kNN classification (Goldberger et al., 2005; Shalev-Shwartz et al., 2004). Our work builds in a novel direction on the success of these previous approaches.

In this paper, we show how to learn a Mahalanobis distance metric for kNN classification. The algorithm that we propose was described at a high level in earlier work (Weinberger et al., 2006) and later extended in terms of scalability and accuracy (Weinberger and Saul, 2008). Intuitively, the algorithm is based on the simple observation that the kNN decision rule will correctly classify an example if its k-nearest neighbors share the same label. The algorithm attempts to increase the number of training examples with this property by learning a linear transformation of the input space that precedes kNN classification using Euclidean distances. The linear transformation is derived by minimizing a loss function that consists of two terms. The first term penalizes large distances between examples in the same class that are desired as k-nearest neighbors, while the second term penalizes small distances between examples with non-matching labels. Minimizing these terms yields a linear transformation of the input space that increases the number of training examples whose k-nearest neighbors have matching labels. The Euclidean distances in the transformed space can equivalently be viewed as Mahalanobis distances in the original space. We exploit this equivalence to cast the problem of distance metric learning as a problem in convex optimization.

Our approach is largely inspired by recent work on neighborhood component analysis (Goldberger et al., 2005) and metric learning in energy-based models (Chopra et al., 2005). Despite similar goals, however, our method differs significantly in the proposed optimization. We formulate the problem of distance metric learning as an instance of semidefinite programming. Thus, the optimization is convex, and its global minimum can be efficiently computed. There have been other studies in distance metric learning based on eigenvalue problems (Shental et al., 2002; De Bie et al., 2003) and semidefinite programming (Globerson and Roweis, 2006; Shalev-Shwartz et al., 2004; Xing et al., 2002). These previous approaches, however, essentially attempt to learn distance metrics that cluster together *all* similarly labeled inputs, even those that are not k-nearest neighbors. This objective is far more difficult to achieve than what we propose. Moreover, it does not leverage the full power of kNN classification, whose accuracy does not require that all similarly labeled inputs be tightly clustered.

There are many parallels between our method and classification by support vector machines (SVMs)—most notably, a convex objective function based on the hinge loss, and the potential to work in nonlinear feature spaces by using the "kernel trick". In light of these parallels, we describe our approach as *large margin nearest neighbor* (LMNN) classification. Our framework can be viewed as the logical counterpart to SVMs in which kNN classification replaces linear classification.

Our framework contrasts with classification by SVMs, however, in one intriguing respect: it requires no modification for multiclass problems. Extensions of SVMs to multiclass problems typically involve combining the results of many binary classifiers, or they require additional machinery that is elegant but non-trivial (Crammer and Singer, 2001). In both cases the training time scales at least linearly in the number of classes. By contrast, our framework has no explicit dependence on the number of classes.

We also show how to extend our framework to learn multiple Mahalanobis metrics, each of them associated with a different class label and/or region of the input space. The multiple metrics are trained simultaneously by minimizing a single loss function. While the loss function couples metrics in different parts of the input space, the optimization remains an instance of semidefinite programming. The globally integrated training of local distance metrics distinguishes our approach from earlier work on discriminant adaptive kNN classification (Hastie and Tibshirani, 1996)

Our paper is organized as follows. Section 2 introduces the general problem of distance metric learning for *k*NN classification and reviews previous approaches that motivated our work. Section 3 describes our model for LMNN classification and formulates the required optimization as an instance of semidefinite programming. Section 4 presents experimental results on several data sets. Section 5 discusses several extensions to LMNN classification, including iterative re-estimation of target neighbors, locally adaptive Mahalanobis metrics in different parts of the input space, and "kernelization" of the basic algorithm. Section 6 describes faster implementations for training and testing in LMNN classification using ball trees. Section 7 concludes by summarizing our main contributions and sketching several directions of ongoing research. Finally, appendix A describes the special-purpose solver that we implemented for large scale problems in LMNN classification.

2. Background

In this section, we introduce the general problem of distance metric learning (section 2.1) and review a number of previously studied approaches. Broadly speaking, these approaches fall into three categories: eigenvector methods based on second-order statistics (section 2.2), convex optimizations over the space of positive semidefinite matrices (section 2.3), and fully supervised algorithms that directly attempt to optimize *k*NN classification error (section 2.4) .

2.1 Distance Metric Learning

We begin by reviewing some basic terminology. A mapping $D : \mathcal{X} \times \mathcal{X} \rightarrow \Re_0^+$ over a vector space $\mathcal{X}$ is called a **metric** if for all vectors $\forall \vec{x}_i, \vec{x}_j, \vec{x}_k \in \mathcal{X}$, it satisfies the properties:

1. $D(\vec{x}_i,\vec{x}_j) + D(\vec{x}_j,\vec{x}_k) \geq D(\vec{x}_i,\vec{x}_k)$ (triangular inequality).
2. $D(\vec{x}_i,\vec{x}_j) \geq 0$ (non-negativity).
3. $D(\vec{x}_i,\vec{x}_j) = D(\vec{x}_j,\vec{x}_i)$ (symmetry).
4. $D(\vec{x}_i,\vec{x}_j) = 0 \iff \vec{x}_i = \vec{x}_j$ (distinguishability).

Strictly speaking, if a mapping satisfies the first three properties but not the fourth, it is called a **pseudometric**. However, to simplify the discussion in what follows, we will often refer to pseudometrics as metrics, pointing out the distinction only when necessary.

We obtain a family of metrics over $\mathcal{X}$ by computing Euclidean distances after performing a linear transformation $\vec{x}' = \mathbf{L}\vec{x}$. These metrics compute squared distances as:

$$D_{\mathbf{L}}(\vec{x}_i,\vec{x}_j) = \|\mathbf{L}(\vec{x}_i - \vec{x}_j)\|_2^2, \tag{1}$$

where the linear transformation in Eq. (1) is parameterized by the matrix $\mathbf{L}$. It is simple to show that Eq. (1) defines a valid metric if $\mathbf{L}$ is full rank and a valid pseudometric otherwise.

It is common to express squared distances under the metric in Eq. (1) in terms of the square matrix:

$$\mathbf{M} = \mathbf{L}^\top \mathbf{L}. \tag{2}$$

Any matrix $\mathbf{M}$ formed in this way from a real-valued matrix $\mathbf{L}$ is guaranteed to be positive semidefinite (i.e., to have no negative eigenvalues). In terms of the matrix $\mathbf{M}$, we denote squared distances by

$$\mathcal{D}_{\mathbf{M}}(\vec{x}_i, \vec{x}_j) = (\vec{x}_i - \vec{x}_j)^\top \mathbf{M} (\vec{x}_i - \vec{x}_j), \tag{3}$$

and we refer to pseudometrics of this form as **Mahalanobis** metrics. Originally, this term was used to describe the quadratic forms in Gaussian distributions, where the matrix $\mathbf{M}$ played the role of the inverse covariance matrix. Here we allow $\mathbf{M}$ to denote any positive semidefinite matrix. The distances in Eq. (1) and Eq. (3) can be viewed as generalizations of Euclidean distances. In particular, Euclidean distances are recovered by setting $\mathbf{M}$ to be equal to the identity matrix.

A Mahalanobis distance metric can be parameterized in terms of the matrix $\mathbf{L}$ or the matrix $\mathbf{M}$. Note that the matrix $\mathbf{L}$ uniquely defines the matrix $\mathbf{M}$, while the matrix $\mathbf{M}$ defines $\mathbf{L}$ up to rotation (which does not affect the computation of distances). This equivalence suggests two different approaches to distance metric learning. In particular, we can either estimate a linear transformation $\mathbf{L}$, or we can estimate a positive semidefinite matrix $\mathbf{M}$. Note that in the first approach, the optimization is unconstrained, while in the second approach, it is important to enforce the constraint that the matrix $\mathbf{M}$ is positive semidefinite. Though generally more complicated to solve a constrained optimization, this second approach has certain advantages that we explore in later sections.

Many researchers have proposed ways to estimate Mahalanobis distance metrics for the purpose of computing distances in *k*NN classification. In particular, let $\{(\vec{x}_i, y_i)\}_{i=1}^n$ denote a training set of n labeled examples with inputs $\vec{x}_i \in \Re^d$ and discrete (but not necessarily binary) class labels $y_i \in \{1, 2, \ldots, \mathcal{C}\}$. For *k*NN classification, one seeks a linear transformation such that nearest neighbors computed from the distances in Eq. (1) share the same class labels. We review several previous approaches to this problem in the following section.

2.2 Eigenvector Methods

Eigenvector methods have been widely used to discover informative linear transformations of the input space. As discussed in section 2.1, these linear transformations can be viewed as inducing a Mahalanobis distance metric. Popular eigenvector methods for linear preprocessing are principal component analysis, linear discriminant analysis, and relevant component analysis. These methods differ in the way that they use labeled or unlabeled data to derive linear transformations of the input space. These methods can also be "kernelized" to work in a nonlinear feature space (Müller et al., 2001; Schölkopf et al., 1998; Tsang et al., 2005), though we do not discuss such formulations here.

2.2.1 PRINCIPAL COMPONENT ANALYSIS

We briefly review principal component analysis (PCA) (Jolliffe, 1986) in the context of distance metric learning. Essentially, PCA computes the linear transformation $\vec{x}_i \rightarrow \mathbf{L}\vec{x}_i$ that projects the training inputs $\{\vec{x}_i\}_{i=1}^n$ into a variance-maximizing subspace. The variance of the projected inputs can be written in terms of the covariance matrix:

$$\mathbf{C} = \frac{1}{n}\sum_{i=1}^{n}(\vec{x}_i - \vec{\mu})^\top(\vec{x}_i - \vec{\mu}),$$

where $\vec{\mu} = \frac{1}{n}\sum_i \vec{x}_i$ denotes the sample mean. The linear transformation $\mathbf{L}$ is chosen to maximize the variance of the projected inputs, subject to the constraint that $\mathbf{L}$ defines a projection matrix. In terms of the input covariance matrix, the required optimization is given by:

$$\max_{\mathbf{L}} \mathrm{Tr}(\mathbf{L}^\top \mathbf{C}\mathbf{L}) \text{ subject to: } \mathbf{L}\mathbf{L}^\top = \mathbf{I}. \tag{4}$$

The optimization in Eq. (4) has a closed-form solution; the standard convention equates the rows of $\mathbf{L}$ with the leading eigenvectors of the covariance matrix. If $\mathbf{L}$ is a rectangular matrix, the linear transformation projects the inputs into a lower dimensional subspace. If $\mathbf{L}$ is a square matrix, then the transformation does not reduce the dimensionality, but this solution still serves to rotate and re-order the input coordinates by their respective variances.

Note that PCA operates in an unsupervised setting without using the class labels of training inputs to derive informative linear projections. Nevertheless, PCA still has certain useful properties as a form of linear preprocessing for kNN classification. For example, PCA can be used for "de-noising": projecting out the components of the bottom eigenvectors often reduces kNN error rate. PCA can also be used to accelerate neighbor nearest computations in large data sets. The linear preprocessing from PCA can significantly reduce the amount of computation either by explicitly reducing the dimensionality of the inputs, or simply by re-ordering the input coordinates in terms of their variance (as discussed further in section 6).

2.2.2 LINEAR DISCRIMINANT ANALYSIS

We briefly review linear discriminant analysis (LDA) (Fisher, 1936) in the context of distance metric learning. Let Ω_c denote the set of indices of examples in the cth class (with $y_i = c$). Essentially, LDA computes the linear projection $\vec{x}_i \to \mathbf{L}\vec{x}_i$ that maximizes the amount of between-class variance relative to the amount of within-class variance. These variances are computed from the between-class and within-class covariance matrices, defined by:

$$\begin{aligned} \mathbf{C}_b &= \frac{1}{\mathcal{C}}\sum_{c=1}^{\mathcal{C}} \vec{\mu}_c \vec{\mu}_c^\top, \\ \mathbf{C}_w &= \frac{1}{n}\sum_{c=1}^{\mathcal{C}}\sum_{i\in\Omega_c}(\vec{x}_i - \vec{\mu}_c)(\vec{x}_i - \vec{\mu}_c)^\top, \end{aligned} \tag{5}$$

where $\vec{\mu}_c$ denotes the sample mean of the c^{th} class; we also assume that the data is globally centered. The linear transformation $\mathbf{L}$ is chosen to maximize the ratio of between-class to within-class variance, subject to the constraint that $\mathbf{L}$ defines a projection matrix. In terms of the above covariance matrices, the required optimization is given by:

$$\max_{\mathbf{L}} \mathrm{Tr}\left(\frac{\mathbf{L}^\top \mathbf{C}_b \mathbf{L}}{\mathbf{L}^\top \mathbf{C}_w \mathbf{L}}\right) \text{ subject to: } \mathbf{L}\mathbf{L}^\top = \mathbf{I}. \tag{6}$$

The optimization in Eq. (6) has a closed-form solution; the standard convention equates the rows of $\mathbf{L}$ with the leading eigenvectors of $\mathbf{C}_w^{-1}\mathbf{C}_b$.

LDA is widely used as a form of linear preprocessing for pattern classification. Unlike PCA, LDA operates in a supervised setting and uses the class labels of the inputs to derive informative linear projections. Note that the between-class covariance matrix $\mathbf{C}_b$ in Eq. (5) has at most rank $\mathcal{C}$, where $\mathcal{C}$ is the number of classes. Thus, up to $\mathcal{C}$ linear projections can be extracted from the eigenvalue problem in LDA. Because these projections are based on second-order statistics, they work well to separate classes whose conditional densities are multivariate Gaussian. When this assumption does not hold, however, LDA may extract spurious features that are not well suited to kNN classification.

2.2.3 RELEVANT COMPONENT ANALYSIS

Finally, we briefly review relevant component analysis (RCA) (Shental et al., 2002; Bar-Hillel et al., 2006) in the context of distance metric learning. RCA is intermediate between PCA and LDA in its use of labeled data. Specifically, RCA makes use of so-called "chunklet" information, or subclass membership assignments. A chunklet is essentially a subset of a class. Inputs in the same chunklet belong to the same class, but inputs in different chunklets do not necessarily belong to different classes. Essentially, RCA computes the linear projection $\vec{x}_i \longrightarrow \mathbf{L}\vec{x}_i$ that "whitens" the data with respect to the averaged within-chunklet covariance matrix. In particular, let Ω_ℓ denote the set of indices of examples in the ℓth chunklet, and let $\vec{\mu}_\ell$ denote the mean of these examples. The averaged within-chunklet covariance matrix is given by:

$$\mathbf{C}_w = \frac{1}{n}\sum_{l=1}^{L}\sum_{i\in\Omega_l}(\vec{x}_i - \vec{\mu}_l)(\vec{x}_i - \vec{\mu}_l)^\top .$$

RCA uses the linear transformation $\vec{x}_i \longrightarrow \mathbf{L}\vec{x}_i$ with $\mathbf{L} = \mathbf{C}_w^{-1/2}$. This transformation acts to normalize the within-chunklet variance. An unintended side effect of this transformation may be to amplify noisy directions in the data. Thus, it is recommended to de-noise the data by PCA before computing the within-chunklet covariance matrix.

2.3 Convex Optimization

Recall that the goal of distance metric learning can be stated in two ways: to learn a linear transformation $\vec{x}_i \longrightarrow \mathbf{L}\vec{x}_i$ or, equivalently, to learn a Mahalanobis metric $\mathbf{M} = \mathbf{L}\mathbf{L}^\top$. It is possible to formulate certain types of distance metric learning as convex optimizations over the cone of positive semidefinite matrices $\mathbf{M}$. In this section, we review two previous approaches based on this idea.

2.3.1 MAHALANOBIS METRIC FOR CLUSTERING

A convex objective function for distance metric learning was first proposed by Xing et al. (2002). The goal of this work was to learn a Mahalanobis metric for clustering (MMC) with side-information. MMC shares a similar goal as LDA: namely, to minimize the distances between similarly labeled inputs while maximizing the distances between differently labeled inputs. MMC differs from LDA in its formulation of distance metric learning as an convex optimization problem. In particular, whereas LDA solves the eigenvalue problem in Eq. (6) to compute the linear transformation $\mathbf{L}$, MMC solves a convex optimization over the matrix $\mathbf{M} = \mathbf{L}^\top\mathbf{L}$ that directly represents the Mahalanobix metric itself.

To state the optimization for MMC, it is helpful to introduce further notation. From the class labels y_i, we define the $n \times n$ binary association matrix with elements $y_{ij} = 1$ if $y_i = y_j$ and $y_{ij} = 0$ otherwise. In terms of this notation, MMC attempts to maximize the distances between pairs of inputs with different labels ($y_{ij} = 0$), while constraining the sum over squared distances of pairs of similarly labeled inputs ($y_{ij} = 1$). In particular, MMC solves the following optimization:

$$\boxed{\begin{array}{l} \textbf{Maximize } \sum_{ij}(1-y_{ij})\sqrt{\mathcal{D}_{\mathbf{M}}(\vec{x}_i,\vec{x}_j)} \textbf{ subject to:} \\ \quad\quad \textbf{(1) } \sum_{ij} y_{ij}\mathcal{D}_{\mathbf{M}}(\vec{x}_i,\vec{x}_j) \ \leq\ 1 \\ \quad\quad \textbf{(2) } \mathbf{M} \succeq 0\textbf{.} \end{array}}$$

The first constraint is required to make the problem feasible and bounded; the second constraint enforces that $\mathbf{M}$ is a positive semidefinite matrix. The overall optimization is convex. The square root in the objective function ensures that MMC leads to generally different results than LDA.

MMC was designed to improve the performance of iterative clustering algorithms such as k-means. In these algorithms, clusters are generally modeled as normal or unimodal distributions. MMC builds on this assumption by attempting to minimize distances between all pairs of similarly labeled inputs; this objective is only sensible for unimodal clusters. For this reason, however, MMC is not especially appropriate as a form of distance metric learning for kNN classification. One of the major strengths of kNN classification is its non-parametric framework. Thus a different objective for distance metric learning is needed to preserve this strength of kNN classification—namely, that it does not implicitly make parametric (or other limiting) assumptions about the input distributions.

2.3.2 ONLINE LEARNING OF MAHALANOBIS DISTANCES

Convex optimizations over the cone of positive semidefinite matrices have also been proposed for perceptron-like approaches to distance metric learning. The Pseudometric Online Learning Algorithm (POLA) (Shalev-Shwartz et al., 2004) combines ideas from convex optimization and large margin classification. Like LDA and MMC, POLA attempts to learn a metric that shrinks distances between similarly labeled inputs and expands distances between differently labeled inputs. POLA differs from LDA and MMC, however, in explicitly encouraging a finite margin that separates differently labeled inputs. POLA was also conceived in an online setting.

The online version of POLA works as follows. At time t, the learning environment presents a tuple $(\vec{x}_t, \vec{x}'_t, y_t)$, where the binary label y_t indicates whether the two inputs $\vec{x}_t$ and $\vec{x}'_t$ belong to the same ($y_t = 1$) or different ($y_t = -1$) classes. From streaming tuples of this form, POLA attempts to learn a Mahalanobis metric $\mathbf{M}$ and a scalar threshold b such that similarly labeled inputs are *at most* a distance of $b - 1$ apart, while differently labeled inputs are *at least* a distance of $b + 1$ apart. These constraints can be expressed by the single inequality:

$$y_t \left[b - \left(\vec{x}_t - \vec{x}'_t\right)^{\top} \mathbf{M} \left(\vec{x}_t - \vec{x}'_t\right) \right] \geq 1. \tag{7}$$

The distance metric $\mathbf{M}$ and threshold b are updated after each tuple $(\vec{u}_t, \vec{v}_t, y_t)$ to correct any violation of this inequality. In particular, the update computes a positive semidefinite matrix $\mathbf{M}$ that satisfies (7). The required optimization can be performed by an alternating projection algorithm, similar to the one described in appendix A. The algorithm extends naturally to problems with more than two classes.

POLA can also be implemented on a data set of fixed size. In this setting, pairs of inputs are repeatedly processed until no pair violates its margin constraints by more than some constant $\beta > 0$. Moreover, as in perceptron learning, the number of iterations over the data set can be bounded above (Shalev-Shwartz et al., 2004).

In many ways, POLA exhibits the same strengths and weaknesses as MMC. Both algorithms are based on convex optimizations that do not have spurious local minima. On the other hand, both algorithms make implicit assumptions about the distributions of inputs and class labels. The margin constraints enforced by POLA are designed to learn a distance metric under which all pairs of similarly labeled inputs are closer than all pairs of differently labeled inputs. This type of learning may often be unrealizable, however, even in situations where *k*NN classification is able to succeed. For this reason, a different framework is required to learn distance metrics for *k*NN classification.

2.4 Neighborhood Component Analysis

Recently, Goldberger et al. (2005) considered how to learn a Mahalnobis distance metric especially for *k*NN classification. They proposed a novel supervised learning algorithm known as *Neighborhood Component Analysis* (NCA). The algorithm computes the expected leave-one-out classification error from a stochastic variant of *k*NN classification. The stochastic classifier uses a Mahalanobis distance metric parameterized by the linear transformation $\vec{x} \rightarrow \mathbf{L}\vec{x}$ in Eqs. (1–3). The algorithm attempts to estimate the linear transformation $\mathbf{L}$ that minimizes the expected classification error when distances are computed in this way.

The stochastic classifier in NCA is used to label queries by the majority vote of nearby training examples, but not necessarily the k nearest neighbors. In particular, for each query, the reference examples in the training set are drawn from a softmax probability distribution that favors nearby examples over faraway ones. The probability of drawing $\vec{x}_j$ as a reference example for $\vec{x}_i$ is given by:

$$p_{ij} = \begin{cases} \frac{\exp\left(-\|\mathbf{L}x_i - \mathbf{L}x_j\|^2\right)}{\sum_{k\neq i}\exp\left(-\|\mathbf{L}x_i - \mathbf{L}x_k\|^2\right)} & \text{if } i \neq j \\ 0 & \text{if } i = j. \end{cases} \tag{8}$$

Note that there is no free parameter k for the number of nearest neighbors in this stochastic classifier. Instead, the scale of $\mathbf{L}$ determines the size of neighborhoods from which nearby training examples are sampled. On average, though, this sampling procedure yields similar results as a deterministic *k*NN classifier (for some value of k) with the same Mahalanobis distance metric.

Under the softmax sampling scheme in Eq. (8), it is simple to compute the expected leave-one-out classification error on the training examples. As in section 2.3.1, we define the $n \times n$ binary matrix with elements $y_{ij} = 1$ if $y_i = y_j$ and $y_{ij} = 0$ otherwise. The expected error computes the fraction of training examples that are (on average) misclassified:

$$\varepsilon_{\text{NCA}} = 1 - \frac{1}{n}\sum_{ij} p_{ij} y_{ij}. \tag{9}$$

The error in Eq. (9) is a continuous, differentiable function of the linear transformation $\mathbf{L}$ used to compute Mahalanobis distances in Eq. (8).

Note that the differentiability of Eq. (9) depends on the stochastic neighborhood assignment of the NCA decision rule. By contrast, the leave-one-out error of a deterministic *k*NN classifier is neither continuous nor differentiable in the parameters of the distance metric. For distance metric learning, the differentiability of Eq. (9) is a key advantage of stochastic neighborhood assignment, making it possible to minimize this error measure by gradient descent. It would be much more difficult to minimize the leave-one-out error of its deterministic counterpart.

The objective function for NCA differs in one important respect from other algorithms reviewed in this section. Though continuous and differentiable with respect to the parameters of the distance metric, Eq. (9) is not convex, nor can it be minimized using eigenvector methods. Thus, the optimization in NCA can suffer from spurious local minima. In practice, the results of the learning algorithm depend on the initialization of the distance metric.

The linear transformation in NCA can also be used to project the inputs into a lower dimensional Euclidean space. Eqs. (8–9) remain valid when $\mathbf{L}$ is a rectangular as opposed to square matrix. Lower dimensional projections learned by NCA can be used to visualize class structure and/or to accelerate *k*NN search.

Recently, Globerson and Roweis (2006) proposed a related model known as Metric Learning by Collapsing Classes (MLCC). The goal of MLCC is to find a distance metric that (like LDA) shrinks the within-class variance while maintaining the separation between different classes. MLCC uses a similar rule as NCA for stochastic classification, so as to yield a differentiable objective function. Compared to NCA, MLCC has both advantages and disadvantages for distance metric learning. The main advantage is that distance metric learning in MLCC can be formulated as a convex optimization over the space of positive semidefinite matrices. The main disadvantage is that MLCC implicitly assumes that the examples in each class have a unimodal distribution. In this sense, MLCC shares the same basic strengths and weaknesses of the methods described in section 2.3.

3. Model

The model we propose for distance metric learning builds on the algorithms reviewed in section 2. In common with all of them, we attempt to learn a Mahalanobis distance metric of the form in Eqs. (1–3). Other key aspects of our model build on the particular strengths of individual approaches. As in MMC (see section 2.3.1), we formulate the parameter estimation in our model as a convex optimization over the space of positive semidefinite matrices. As in POLA (see section 2.3.2), we attempt to maximize the margin by which the model correctly classifies labeled examples in the training set. Finally, as in NCA (see section 2.4), our model was conceived specifically to learn a Mahalanobis distance metric that improves the accuracy of *k*NN classification. Indeed, the three essential ingredients of our model are (i) its convex loss function, (ii) its goal of margin maximization, and (iii) the constraints on the distance metric imposed by accurate *k*NN classification.

3.1 Intuition and Terminology

Our model is based on two simple intuitions (and idealizations) for robust *k*NN classification: first, that each training input $\vec{x}_i$ should share the same label y_i as its k nearest neighbors; second, that training inputs with different labels should be widely separated. We attempt to learn a linear transformation of the input space such that the training inputs satisfy these properties. In fact, these objectives are neatly balanced by two competing terms in our model's loss function. Specifically, one term penalizes large distances between nearby inputs with the same label, while the other term penalizes small distances between inputs with different labels. To make precise these relative notions of "large" and "small", however, we first need to introduce some new terminology.

Learning in our framework requires auxiliary information beyond the label y_i of each input $\vec{x}_i$ in the training set. Recall that the goal of learning is to estimate a distance metric under which each input $\vec{x}_i$ has k nearest neighbors that share its same label y_i. We facilitate this goal by identifying *target neighbors* for each input $\vec{x}_i$ at the outset of learning. The target neighbors of $\vec{x}_i$ are those that we desire to be closest to $\vec{x}_i$; in particular, we attempt to learn a linear transformation of the input space such that the resulting nearest neighbors of $\vec{x}_i$ are indeed its target neighbors. We emphasize that target neighbors are fixed a priori and do not change during the learning process. This step significantly simplifies the learning process by specifying a priori which similarly labeled inputs to cluster together. In many applications, there may be prior knowledge or auxiliary information (e.g., a similarity graph) that naturally identifies target neighbors. In the absence of prior knowledge, the simplest prescription is to compute the k nearest neighbors with the same class label, as determined by Euclidean distance. This was done for all the experiments in this paper. We use the notation $j \rightsquigarrow i$ to indicate that input $\vec{x}_j$ is a target neighbor of input $\vec{x}_i$. Note that this relation is not symmetric: $j \rightsquigarrow i$ does not imply $i \rightsquigarrow j$.

For kNN classification to succeed, the target neighbors of each input $\vec{x}_i$ should be closer than all differently labeled inputs. In particular, for each input $\vec{x}_i$, we can imagine the target neighbors as establishing a perimeter that differently labeled inputs should not invade. We refer to the differently labeled inputs in the training set that invade this perimeter as *impostors*; the goal of learning (roughly speaking) is to minimize the number of impostors.

In fact, to increase the robustness of kNN classification, we adopt an even more stringent goal for learning—namely to maintain a large (finite) distance between impostors and the perimeters established by target neighbors. By maintaining a *margin* of safety around the kNN decision boundaries, we ensure that the model is robust to small amounts of noise in the training inputs. This robustness criterion also gives rise to the name of our approach: *large margin nearest neighbor* (LMNN) classification.

In mathematical terms, impostors are defined by a simple inequality. For an input $\vec{x}_i$ with label y_i and target neighbor $\vec{x}_j$, an impostor is any input $\vec{x}_l$ with label $\vec{y}_l \neq \vec{y}_i$ such that

$$\|\mathbf{L}(\vec{x}_i - \vec{x}_l)\|^2 \leq \|\mathbf{L}(\vec{x}_i - \vec{x}_j)\|^2 + 1. \tag{10}$$

In other words, an impostor $\vec{x}_l$ is any differently labeled input that invades the perimeter plus unit margin defined by any target neighbor $\vec{x}_j$ of the input $\vec{x}_i$.

Figure 1 illustrates the main idea behind LMNN classification. Before learning, a training input has both target neighbors and impostors in its local neighborhood. During learning, the impostors are pushed outside the perimeter established by the target neighbors. After learning, there exists a finite margin between the perimeter and the impostors. The figure shows the idealized scenario where kNN classification errors in the original input space are corrected by learning an appropriate linear transformation.

3.2 Loss Function

With the intuition and terminology from the previous section, we can now construct a loss function for LMNN classification. The loss function consists of two terms, one which acts to *pull* target neighbors closer together, and another which acts to *push* differently labeled examples further apart.

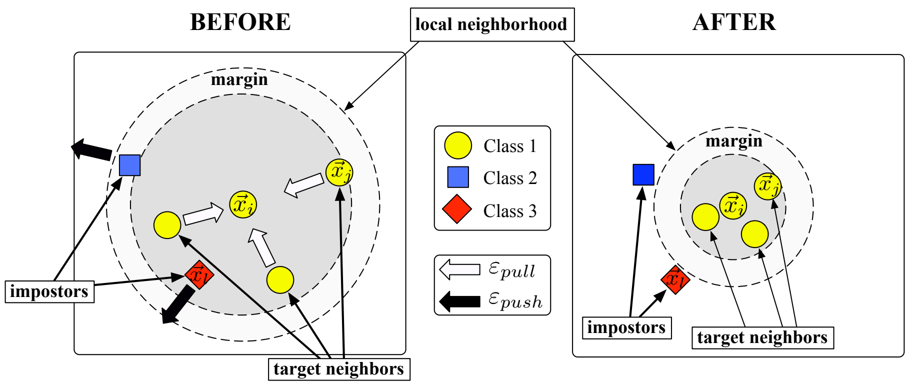


Figure 1: Schematic illustration of one input's neighborhood before training (*left*) versus after training (*right*). The distance metric is optimized so that: (i) its $k=3$ target neighbors lie within a smaller radius after training; (ii) differently labeled inputs lie outside this smaller radius by some finite margin. Arrows indicate the gradients on distances arising from different terms in the cost function.

These two terms have competing effects, since the first is reduced by shrinking the distances between examples while the second is generally reduced by magnifying them. We discuss each term in turn.

The first term in the loss function penalizes large distances between each input and its target neighbors. In terms of the linear transformation $\mathbf{L}$ of the input space, the sum of these squared distances is given by:

$$\varepsilon_{\text{pull}}(\mathbf{L}) = \sum_{j \rightsquigarrow i} \|\mathbf{L}(\vec{x}_i - \vec{x}_j)\|^2. \tag{11}$$

The gradient of this term generates a pulling force that attracts target neighbors in the linearly transformed input space. It is important that Eq. (11) only penalizes large distances between inputs and their *target neighbors*; in particular, it does not penalize large distances between all similarly labeled inputs. We purposefully do not penalize the latter because accurate kNN classification does not require that all similarly labeled inputs be tightly clustered. Our approach is distinguished in this way from many previous approaches to distance metric learning; see section 2. By only penalizing large distances between neighbors, we build models that leverage the full power of kNN classification.

The second term in the loss function penalizes small distances between differently labeled examples. In particular, the term penalizes violations of the inequality in Eq. (10). To simplify notation, we introduce a new indicator variable $y_{il}=1$ if and only if $y_i=y_l$, and $y_{il}=0$ otherwise. In terms of this notation, the second term of the loss function ε_{push} is given by:

$$\varepsilon_{\text{push}}(\mathbf{L}) = \sum_{i,j \rightsquigarrow i} \sum_{l} (1-y_{il}) \left[1 + \|\mathbf{L}(\vec{x}_i - \vec{x}_j)\|^2 - \|\mathbf{L}(\vec{x}_i - \vec{x}_l)\|^2\right]_+ \tag{12}$$

where the term $[z]_+ = \max(z, 0)$ denotes the standard hinge loss. The hinge loss monitors the inequality in Eq. (10). If the inequality does not hold (i.e., the input $\vec{x}_l$ lies a safe distance away from $\vec{x}_i$), then its hinge loss has a negative argument and makes no contribution to the overall loss function. The (sub-)gradient of Eq. (12) generates a pushing force that repels imposters away from the perimeter established by each example's k nearest (similarly labeled) neighbors; see Fig. 1.

The choice of unit margin is an arbitrary convention that sets the scale for the linear transformation $\mathbf{L}$ (which enters every other term in the loss function). If a margin $c > 0$ was enforced instead of the unit margin, the loss function would be minimized by the same linear transformation up to an overall scale factor $\sqrt{c}$.

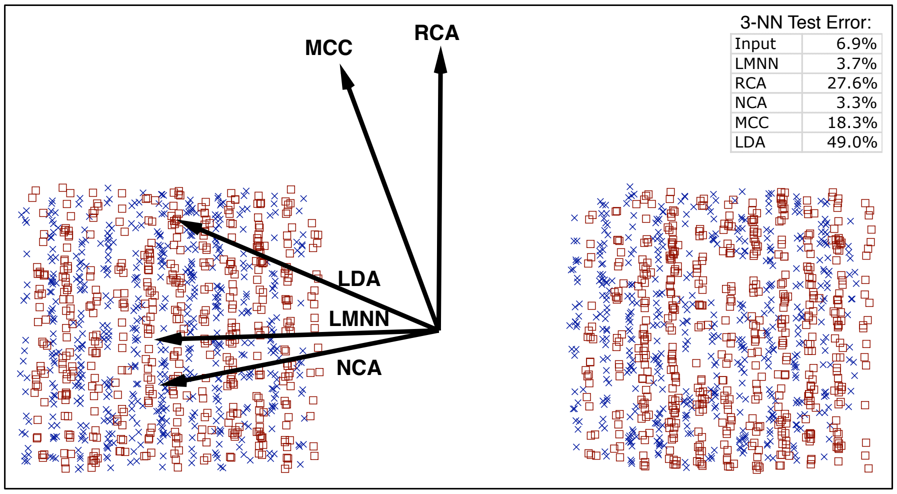
Finally, we combine the two terms $\varepsilon_{\text{pull}}(\mathbf{L})$ and $\varepsilon_{\text{push}}(\mathbf{L})$ into a single loss function for distance metric learning. The two terms can have competing effects—to attract target neighbors on one hand, to repel impostors on the other. A weighting parameter $\mu \in [0,1]$ balances these goals:

$$\varepsilon(\mathbf{L}) = (1-\mu)\,\varepsilon_{pull}(\mathbf{L}) + \mu\,\varepsilon_{push}(\mathbf{L}). \tag{13}$$

Generally, the parameter μ can be tuned via cross validation, though in our experience, the results from minimizing the loss function in Eq. (13) did not depend sensitively on the value of μ. In practice, the value $\mu = 0.5$ worked well.

The competing terms in Eq. (13) are analogous to those in the loss function for learning in SVMs (Schölkopf and Smola, 2002). In both loss functions, one term penalizes the norm of the "parameter" vector (i.e., the weight vector of the maximum margin hyperplane, or the linear transformation in the distance metric), while the other incurs the hinge loss. Just as the hinge loss in SVMs is only triggered by examples near the decision boundary, the hinge loss in Eq. (13) is only triggered by differently labeled examples that invade each other's neighborhoods. Both loss functions in SVMs and LMNN can be rewritten to depend on the input vectors only through their inner products. Working with the inner product matrix directly allows the application of the *kernel trick*; see section 5.3. Finally, as in SVMs, we can formulate the minimization of the loss function in Eq. (13) as a convex optimization. This last point will be developed further in section 3.4.

Our framework for distance metric learning provides an alternative to the earlier approach of NCA (Goldberger et al., 2005) described in section 2.4. We briefly compare the two approaches at a high level. Both LMNN and NCA are designed to learn a Mahalanobis distance metric over the input space that improves kNN classification at test time. Though test examples are not available during training, the learning algorithms for LMNN and NCA are based on training in "simulated" test conditions. Neither approach directly minimizes the leave-one-out error[1] for kNN classification over the training set. The leave-one-out error is a piecewise constant but non-smooth function of the linear transformation $\mathbf{L}$, making it difficult to minimize directly. NCA uses stochastic neighborhood assignment to construct a smooth loss function, thus circumventing this problem. LMNN uses the hinge loss to construct an upper bound on the leave-one-out error for kNN classification; this upper bound is continuous and similarly well behaved for standard gradient-based methods. In NCA, it is not necessary to select a fixed number k of target neighbors in advance of the optimization. Because the objective function for NCA is not convex, however, the initial conditions for the Mahalanobis metric implicitly favor the preservation of certain neighborhoods over others. By contrast, in LMNN, the target neighborhoods must be explicitly specified. A potential advantage of LMNN is that the required optimization can be formulated as an instance of semidefinite programming.



3-NN Test Error:

Input	6.9%
LMNN	3.7%
RCA	27.6%
NCA	3.3%
MCC	18.3%
LDA	49.0%

Figure 2: A toy data set for distance metric learning, with $n = 2000$ data points sampled from a bi-modal distribution. Within each mode, examples from two classes are distributed in alternating vertical stripes. The figure shows the dominant axis extracted by several different algorithms for distance metric learning. Only NCA and LMNN reduce the 1-NN classification error on this data set; the other algorithms actually increase the error by focusing on global versus local distances.

3.3 Local Versus Global Distances

We emphasize that the loss function for LMNN classification only penalizes large distances between target neighbors as opposed to all examples in the same class. The toy data set in Fig. 2 illustrates the potential advantages of this approach. The data was generated by sampling $n = 2000$ data points from two classes in a zebra striped pattern; additionally, the data for each class was generated in two sets of stripes displaced by a large horizontal offset. As a result, this data set has the property that within-class variance is much larger in the horizontal direction than the vertical direction; however, local class membership is much more reliably predicted by examples that are nearby in the vertical direction.

Algorithms such as LMNN and NCA perform very differently on this data set than algorithms such as LDA, RCA, and MCC. In particular, LMNN and NCA adapt to the local striped structure in the data set and learn distance metrics that significantly reduce the *k*NN error rate. By contrast, LDA, RCA, and MCC attempt to shrink distances between all examples in the same class and actually increase the *k*NN error rate as a result. Though this data set is especially contrived, it illustrates in general the problems posed by classes with multimodal support. Such classes violate a basic assumption behind metric learning algorithms that attempt to shrink global distances between all similarly labeled examples.

1. This is the number of training examples that *would have* been mislabeled by *k*NN classification if their label was in fact unknown.

3.4 Convex Optimization

The loss function in Eq. (13) is not convex in the matrix elements of the linear transformation $\mathbf{L}$. To minimize this loss function, one straightforward approach is gradient descent in the elements of $\mathbf{L}$. However, such an approach is prone to being trapped in local minima. The results of this form of gradient descent will depend in general on the initial estimates for $\mathbf{L}$. Thus they may not be reproducible across different problems and applications.

We can overcome these difficulties by reformulating the optimization of Eq. (13) as an instance of semidefinite programming (Boyd and Vandenberghe, 2004). A semidefinite program (SDP) is a linear program that incorporates an additional constraint on a symmetric matrix whose elements are linear in the unknown variables. This additional constraint requires the matrix to be positive semidefinite, or in other words, to only have nonnegative eigenvalues. This matrix constraint is nonlinear but convex, so that the overall optimization remains convex. There exist provably efficient algorithms to solve SDPs (with polynomial time convergence guarantees).

We begin by reformulating Eq. (13) as an optimization over positive semidefinite matrices. Specifically, as described in Eq. (2), we work in terms of the new variable $\mathbf{M} = \mathbf{L}^\top\mathbf{L}$. With this change of variable, we can rewrite the squared distances that appear in the loss function using Eq. (3). Recall that $\mathcal{D}_\mathbf{M}(\vec{x}_i,\vec{x}_j)$ denotes the squared distance with respect to the Mahalanobis metric $\mathbf{M}$. As shown in section 2.1, this distance is equivalent to the Euclidean distance after the mapping $\vec{x}_i \to \mathbf{L}\vec{x}_i$. Substituting Eq. (3) into Eq. (13), we obtain the loss function:

$$\varepsilon(\mathbf{M}) = (1-\mu)\sum_{i,j\rightsquigarrow i}\mathcal{D}_\mathbf{M}(\vec{x}_i,\vec{x}_j) + \mu\sum_{i,j\rightsquigarrow i}\sum_{l}(1-y_{il})\left[1+\mathcal{D}_\mathbf{M}(\vec{x}_i,\vec{x}_j)-\mathcal{D}_\mathbf{M}(\vec{x}_i,\vec{x}_l)\right]_+ . \tag{14}$$

With this substitution, the loss function is now expressed over positive semidefinite matrices $\mathbf{M} \succeq 0$, as opposed to real-valued matrices $\mathbf{L}$. Note that the constraint $\mathbf{M} \succeq 0$ must be added to the optimization to ensure that we learn a well-defined pseudometric.

The loss function in Eq. (14) is a piecewise linear, convex function of the elements in the matrix $\mathbf{M}$. In particular, the first term in the loss function (penalizing large distances between target neighbors) is linear in the elements of $\mathbf{M}$, while the second term (penalizing impostors) is derived from the convex hinge loss. To formulate the optimization of Eq. (14) as an SDP, however, we need to convert it into a more standard form.

An SDP is obtained by introducing slack variables which mimic the effect of the hinge loss. In particular, we introduce nonnegative slack variables $\{\xi_{ijl}\}$ for all triplets of target neighbors ($j \rightsquigarrow i$) and impostors $\vec{x}_l$. The slack variable $\xi_{ijl} \geq 0$ is used to measure the amount by which the large margin inequality in Eq. (10) is violated. Using the slack variables to monitor these margin violations, we obtain the SDP:

Minimize $(1-\mu)\sum_{i,j\rightsquigarrow i}(\vec{x}_i-\vec{x}_j)^\top\mathbf{M}(\vec{x}_i-\vec{x}_j) + \mu\sum_{i,j\rightsquigarrow i,l}(1-y_{il})\xi_{ijl}$ **subject to:**

(1) $(\vec{x}_i-\vec{x}_l)^\top\mathbf{M}(\vec{x}_i-\vec{x}_l) - (\vec{x}_i-\vec{x}_j)^\top\mathbf{M}(\vec{x}_i-\vec{x}_j) \;\geq\; 1-\xi_{ijl}$

(2) $\xi_{ijl} \geq 0$

(3) $\mathbf{M} \succeq 0$**.**

While SDPs in this form can be solved by standard solver packages, general-purpose solvers tend to scale poorly in the number of constraints. For this work, we implemented our own special-purpose solver, exploiting the fact that most of the slack variables $\{\xi_{ijl}\}$ never attain positive values. The slack variables $\{\xi_{ijl}\}$ are sparse because most inputs $\vec{x}_i$ and $\vec{x}_l$ are well separated relative to the distance between $\vec{x}_i$ and any of its target neighbors $\vec{x}_j$. Such triplets do not incur a positive hinge loss, resulting in very few *active* constraints in the SDP. Thus, a great speedup can be achieved by solving an SDP that only monitors a fraction of the margin constraints, then using the resulting solution as a starting point for the actual SDP of interest.

Our solver was based on a combination of sub-gradient descent in both the matrices **L** and **M**, the latter used mainly to verify that we had reached the global minimum. We projected updates in **M** back onto the positive semidefinite cone after each step. Alternating projection algorithms provably converge (Vandenberghe and Boyd, 1996), and in this case our implementation[2] worked much faster than generic solvers. For a more detailed description of the solver please see appendix A.

3.5 Energy Based Classification

The matrix **M** that minimizes the loss function in Eq. (14) can be used as a Mahalanobis distance metric for *k*NN classification. However, it is also possible to use the loss function directly as a so-called "energy-based" classifier. This use is inspired by previous work on energy-based models (Chopra et al., 2005).

Energy-based classification of a test example is done by considering it as an extra training example and computing the loss function in Eq. (14) for every possible label y_t. In particular, for a test example $\vec{x}_t$ with hypothetical label y_t, we locate k (similarly labeled) target neighbors (as determined by Euclidean distance to $\vec{x}_t$ or other a priori considerations) and then compute both terms in Eq. (14) given the already estimated Mahalanobis metric **M**. For the first term, we accumulate the squared distances to the k target neighbors of $\vec{x}_t$. For the second term, we accumulate the hinge loss over all impostors (i.e., differently labeled examples) that invade the perimeter around $\vec{x}_t$ as determined by its target neighbors; we also accumulate the hinge loss for differently labeled examples whose perimeters are invaded by $\vec{x}_t$. Finally, the test example is classified by the hypothetical label that minimizes the combination of these terms:

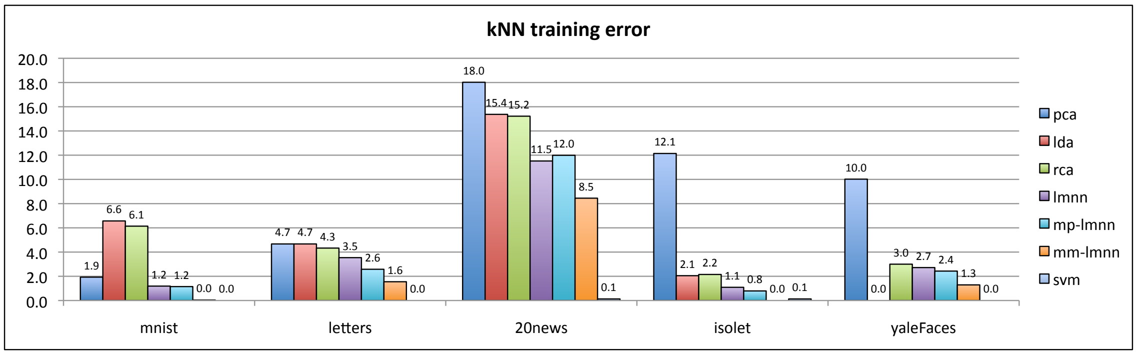
$$\begin{aligned} y_t \;=\; \operatorname{argmin}_{y_t} \Bigg\{ & (1-\mu) \sum_{j \rightsquigarrow t} \mathcal{D}_{\mathbf{M}}(\vec{x}_t, \vec{x}_j) + \mu \sum_{j \rightsquigarrow t, l} (1 - y_{tl}) \left[1 + \mathcal{D}_{\mathbf{M}}(\vec{x}_t, \vec{x}_j) - \mathcal{D}_{\mathbf{M}}(\vec{x}_t, \vec{x}_l)\right]_+ \\ & + \mu \sum_{i, j \rightsquigarrow i} (1 - y_{it}) \left[1 + \mathcal{D}_{\mathbf{M}}(\vec{x}_i, \vec{x}_j) - \mathcal{D}_{\mathbf{M}}(\vec{x}_i, \vec{x}_t)\right]_+ \Bigg\}. \end{aligned} \tag{15}$$

Note that the relation $j \rightsquigarrow t$ in this criterion depends on the value of y_t. As shown in Fig. 3, energy-based classification with this assignment rule generally leads to further improvements in test error rates. Often these improvements are significantly beyond those already achieved by adopting the Mahalanobis distance metric **M** for *k*NN classification.

4. Results

We evaluated LMNN classification on nine data sets of varying size and difficulty. Some of these data sets were derived from collections of images, speech, and text, yielding very high dimensional inputs. In these cases, we used PCA to reduce the dimensionality of the inputs before training LMNN classifiers. Pre-processing the inputs with PCA helped to reduce computation time and avoid overfitting. Table 1 compares the different data sets in detail.

2. A matlab implementation is currently available at `http://www.weinbergerweb.net/Downloads/LMNN.html`.



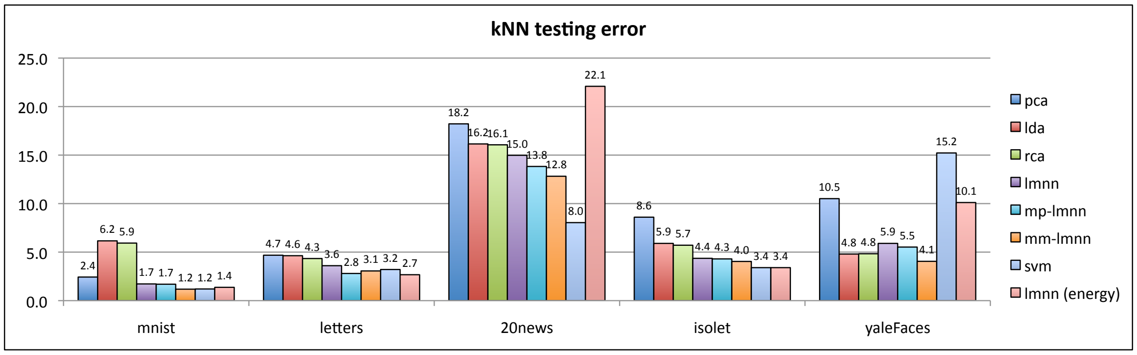


Figure 3: Training and test results on the five largest data sets, preprocessed in different ways, and using different variants of kNN classification. We compared principal component analysis (pca), linear discriminant analysis (lda), relevant component analysis (rca), large margin nearest neighbor classification (lmnn), lmnn with multiple passes (mp-lmnn), lmnn with multiple metrics (mm-lmnn), multi-class support vector machines (svm), lmnn classification with the energy based decision rule (lmnn (energy)). All variations of lmnn, rca and lda were applied after pre-processing with pca for general noise reduction. See text and Table 1 for details. The lmnn results consistently outperform pca and lda. The multiple metrics version of lmnn (mm-lmnn) is comparable with multiclass svm on most data sets (with 20news and yaleFaces as only exceptions).

Experimental results were obtained by averaging over multiple runs on randomly generated 70/30 splits of each data set. This procedure was followed with two exceptions: no averaging was done for the Isolet and MNIST data sets, which have pre-defined training/test splits. For all experiments reported in this paper, the number of target neighbors k was set to $k = 3$, and the weighting parameter μ in Eqs. (14-15) was set to $\mu = 0.5$. Though we experimented with different settings, the results from LMNN classification appeared fairly insensitive to the values of these parameters.

The main results on the five largest data sets are shown in Fig. 3. (See Table 1 for a complete listing of results, including those for various extensions of LMNN classification described in section 5.) All training error rates reported are leave-one-out estimates. To break ties among different classes from the *k*NN decision rule, we repeatedly reduced the neighborhood size, ultimately classifying (if necessary) by just the $k = 1$ nearest neighbor. We begin by reporting overall trends, then discuss the results on individual data sets in more detail.

The first general trend is that LMNN classification using Mahalanobis distances consistently improves on *k*NN classification using Euclidean distances. In general, the Mahalanobis metrics learned by semidefinite programming led to significant improvements in *k*NN classification, both in training and testing.

A second general trend is that the energy-based decision rule described in section 3.5 leads to further improvements over the (already improved) results from *k*NN classification using Mahalanobis distances. In particular, better performance was observed on most of the large data sets. The results are shown in Fig. 3.

A third general trend is that LMNN classification works better with PCA than LDA when some form of dimensionality reduction is required for preprocessing. Table 1 shows the results of LMNN classification on inputs whose dimensionality was reduced by LDA. While pre-processing by LDA helps on some data sets (e.g., wine, yale faces), it generally leads to worse results than pre-processing by PCA. On some data sets, moreover, it leads to drastically worse results (e.g., olivetti faces, MNIST). Consequently we used PCA as a pre-processing step for all subsequent experiments throughout this paper.

A fourth general trend is that LMNN classification yields larger improvements on larger data sets. Though we do not have a formal analysis that accounts for this observation, we can provide the following intuitive explanation. One crucial aspect of the optimization in LMNN classification is the choice of the *target neighbors*. In all of our experiments, we chose the target neighbors based on Euclidean distance in the input space (after dimensionality reduction by PCA or LDA). This choice was a simple heuristic used in the absence of prior knowledge. However, the quality of this choice presumably depends on the sample density of the data set. In particular, as the sample density increases, we suspect that more reliable discriminative signals can be learned from target neighbors chosen in this way. The experimental results bear this out.

Finally, we compare our results to those of competing methods. We take multi-class SVMs (Crammer and Singer, 2001) as providing a fair representation of the state-of-the-art. On each data set (except MNIST), we trained multi-class SVMs using linear, polynomial and RBF kernels and chose the best kernel with cross validation. On MNIST, we used a non-homogeneous polynomial kernel of degree four, which gave us our best results, as also reported in LeCun et al. (1995). The results of the energy-based LMNN classifier are very close to those of state-of-the-art multi-class SVMs: better on some data sets, worse on others. However, consistent improvement over multi-class SVMs was obtained by a multiple-metric variant of LMNN, discussed in section 5.2. This multi-metric extension outperformed SVMs on three of the five large data sets; see Fig. 3. On the only data set with a large performance difference, 20-newsgroups, the multi-class SVMs benefited from training in the original $d = 20000$ dimensional input space, whereas the LMNN classifiers were trained only on the input's leading $d = 200$ principal components. Based on these results, in section 7, we suggest some applications that seem particularly well suited to LMNN classification, though poorly suited to SVMs. These are applications with moderate input dimensionality, but large numbers of classes.

To compare with previous work, we also evaluated RCA (Shental et al., 2002), LDA (Fisher, 1936) and NCA (Goldberger et al., 2005) on the same data sets. For NCA and RCA, we used the code provided by the authors; however, the NCA code ran out of memory on the larger data sets.

Table 1 shows the results of all algorithms on small and larger data sets. LMNN outperforms these other methods for distance metric learning on the four largest data sets. In terms of running times, RCA is by far the fastest method (since its projections can be computed in closed form), while NCA is the slowest, mainly due to the $O(n^2)$ normalization of its softmax probability distributions. Although the optimization in LMNN naively scales as $O(n^2)$, in practice it can be accelerated by various efficiency measures: Appendix A discusses our semidefinite programming solver in detail. We did also include the results of MCC (Xing et al., 2002); however, the code provided by the authors could only handle a few of the small data sets. As shown in Table 1, on those data sets it resulted in classification rates generally higher than NCA.

The results of experiments on particular data sets provide additional insight into the performance of LMNN classification versus competing methods. We give a more detailed overview of these experiments in what follows.

4.1 Small Data Sets with Few Classes

The wine, iris, and bal data sets are small in size, with less than 500 training examples. Each of these data sets has three classes. The data sets are available from the UCI Machine Learning Repository.[3] On data sets of this size, a distance metric can be learned in a matter of seconds. The results in Table 1 were averaged over 100 experiments with different random 70/30 splits of each data set.

On these data sets, LMNN classification improves on kNN classification with a Euclidean distance metric. These results could potentially be improved further with better measures against overfitting (such as regularization). Table 1 also compares the results from LMNN classification to other competing methods. Here, the results are somewhat variable; compared to NCA, RCA, LDA, and multiclass SVMs, LMNN fares better in some cases, worse in others. We mainly report these results to facilitate direct comparisons with previously published work. However, the small size of these data sets makes it difficult to assess the significance of these results. Moreover, these data sets do not represent the regime in which we expect LMNN classification to be most useful.

4.2 Face Recognition

The Olivetti face recognition data set[4] contains 400 grayscale images of 40 subjects in 10 different poses. We downsampled the images to 38×31 pixels and used PCA to further reduce the dimensionality, projecting the images into the subspace spanned by the first 200 eigenfaces (Turk and Pentland, 1991). Training and test sets were created by randomly sampling 7 images of each subject for training and 3 images for testing. The task involved 40-way classification—essentially, recognizing a face from an unseen pose. Table 1 shows the improvements due to LMNN classification. Fig. 4 illustrates the improvements more graphically by showing how the $k=3$ nearest neighbors change as a result of learning a Mahalanobis metric. (Although the algorithm operated on downsampled, projected images, for clarity the figure shows the original images.)

The (extended) Yale face data set contains $n = 2414$ frontal images of 38 subjects. For each subject, there are 64 images taken under extreme illumination conditions. (A few subjects are represented with fewer images.) As for the Olivetti data set, we preprocessed the images by downsampling and projecting them onto their leading 200 principal components. To reduce the impact of the very high variance in illumination, we followed the standard practice of discarding the leading 5

3. Available at `http://www.ics.uci.edu/$\sim$mlearn/MLRepository.html`.
4. Available at `http://www.uk.research.att.com/facedatabase.html`.

eigenvectors. Results from LMNN classification were averaged over 10 runs of 70/30 splits. Each split was obtained by randomly selecting 45 images of each subject for training and 19 images for testing. This protocol ensured that the training examples were evenly distributed across the relatively large number of classes. To guard against overfitting, we employed a validation set consisting of 30% of the training data and stopped the training early when the lowest classification error on the validation set was reached. On this data set, Fig. 3 shows that the LMNN metric outperforms the Euclidean metric and even improves on multiclass SVMs. (Particularly effective on this data set, though, is the simple strategy of LDA.)

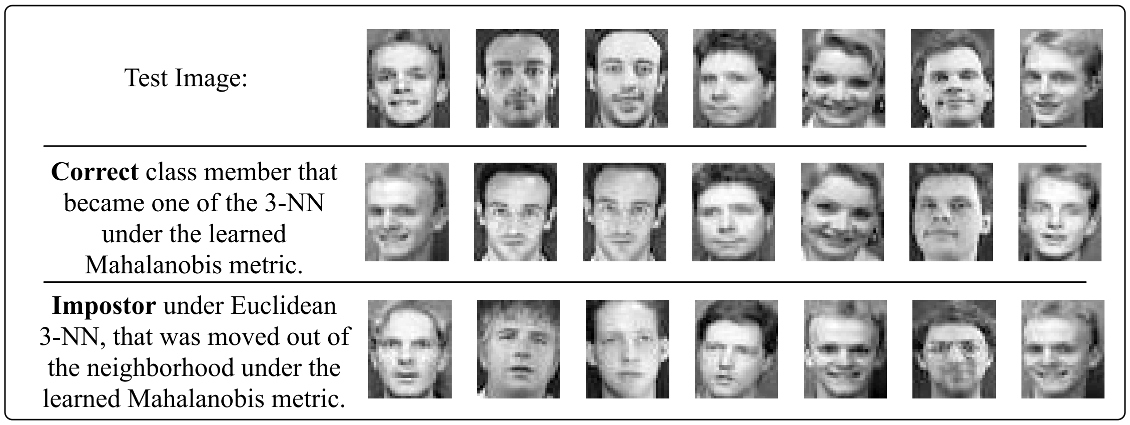

Test Image:
Correct class member that became one of the 3-NN under the learned Mahalanobis metric.
Impostor under Euclidean 3-NN, that was moved out of the neighborhood under the learned Mahalanobis metric.

Figure 4: Test images from the Olivetti face recognition data set (*top row*). The middle row shows images from the same class that were among the 3-NN under the learned Mahalanobis metric (after training) but not among the original 3-NN under the Euclidean metric (before training). The bottom row shows impostors under the Euclidean metric that were no longer inside the local neighborhoods under the Mahalanobis metric.

4.3 Spoken Letter Recognition

The Isolet data set from the UCI Machine Learning Repository contains 6238 examples and 26 classes corresponding to letters of the alphabet. We reduced the input dimensionality (originally at 617) by projecting the data onto its leading 172 principal components—enough to account for 95% of its total variance. On this data set, Dietterich and Bakiri report test error rates of 4.2% using nonlinear backpropagation networks with 26 output units (one per class) and 3.3% using nonlinear backpropagation networks with a 30-bit error correcting code (Dietterich and Bakiri, 1995). LMNN with energy-based classification obtains a test error rate of 3.4%.

4.4 Letter Recognition

The letter recognition data set was also taken from the UCI Machine Learning Repository. It contains randomly distorted images of the 26 letters in the English alphabet in 20 different fonts. The features consist of 16 attributes, such as height, width, correlations of axes and others.[5] It is inter-

5. Full details on the data set can be found at `http://www.ics.uci.edu/$\sim$mlearn/databases/letter-recognition/letter-recognition.names`.

esting that LMNN with energy-based classification significantly outperforms other variants of *k*NN classification on this data set.

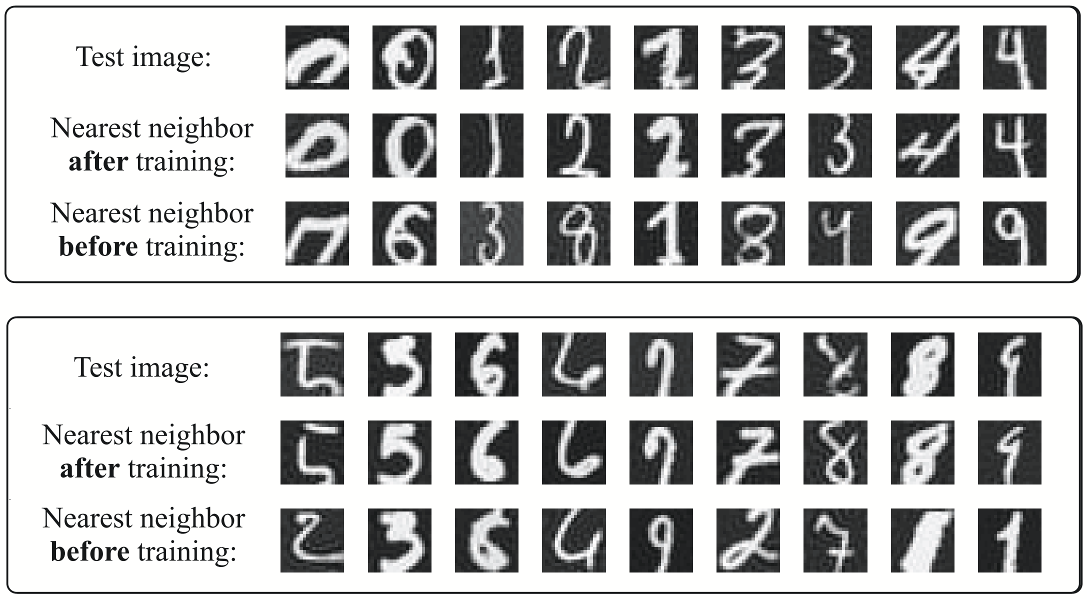

Test image:
Nearest neighbor **after** training:
Nearest neighbor **before** training:
Test image:
Nearest neighbor **after** training:
Nearest neighbor **before** training:

Figure 5: Images from the MNIST data set, along with nearest neighbors before and after training.

4.5 Text Categorization

The 20-newsgroups data set consists of posted articles from 20 newsgroups, with roughly 1000 articles per newsgroup. We used the 18828-version of the data set[6] in which cross-postings are removed and some headers stripped out. The data set was tokenized using the rainbow package (McCallum, 1996). Each article was initially represented by a word-count vector for the 20,000 most common words in the vocabulary. These word-count vectors were then reduced in dimensionality by projecting them onto their leading 200 principal components. The results in Fig. 3 were obtained by averaging over 10 runs with 70/30 splits for training and test data. The best result for LMMN on this data set improved significantly over *k*NN classification using Euclidean distances and PCA (with 14.98% versus 48.57% and 18.22% test error rates). LMNN was outperformed by multiclass SVM (Crammer and Singer, 2001), which obtained a 8.0% test error rate using a linear kernel and 20000 dimensional inputs.[7]

4.6 Handwritten Digit Recognition

The MNIST data set of handwritten digits[8] has been extensively benchmarked (LeCun et al., 1995). We deskewed the original 28×28 grayscale images, then reduced their dimensionality by projecting them onto their leading 164 principal components (enough to capture 95% of the data's overall

6. Available at `http://people.csail.mit.edu/jrennie/20Newsgroups/`.
7. Results vary from previous work (Weinberger et al., 2006) due to different pre-processing.
8. Available at `http://yann.lecun.com/exdb/mnist/`.

Benchmark test error rates

statistics	**mnist**	**letters**	**20news**	**isolet**	**yFaces**	**bal**	**oFaces**	**wine**	**iris**
# inputs	70000	20000	18827	7797	2414	535	400	152	128
# features	784	16	20000	617	8064	4	200	13	4
# reduced dimensions	164	16	200	172	300	4	200	13	4
# training examples	60000	14000	13179	6238	1690	375	280	106	90
# testing examples	10000	6000	5648	1559	724	161	120	46	38
# classes	10	26	20	26	38	3	40	3	3
# of train/test splits	1	10	10	1	10	100	100	100	100
% validation	0	0	0	0	30	0	30	0	30
kNN									
Euclidean	2.12	4.68	48.57	8.98	29.19	18.33	6.03	25.00	4.87
PCA	2.43	4.68	18.22	8.60	10.79	18.33	2.80	25.00	4.87
LDA	6.16	4.63	16.15	5.90	4.80	10.82	10.01	2.17	4.00
RCA	5.93	4.34	16.06	5.71	4.83	12.31	10.02	2.28	3.71
MCC	N/A	N/A	N/A	N/A	N/A	15.66	15.91	30.96	3.55
NCA	N/A	N/A	N/A	N/A	N/A	5.33	2.60	28.67	4.32
LMNN									
PCA	1.72	3.60	14.98	4.36	5.90	11.16	3.28	8.72	4.37
LDA	6.16	3.61	16.98	5.84	5.08	10.84	40.72	**2.11**	3.79
LMNN (energy)	1.37	**2.67**	22.09	**3.40**	10.11	9.14	3.16	7.67	3.68
LMNN (multiple passes)	1.69	2.80	13.83	4.30	5.52	5.86	4.83	7.59	4.26
LMNN (multiple metrics)	**1.18**	3.06	12.82	4.04	**4.05**	10.72	3.11	8.72	4.66
solver statistics									
CPU time (1M)	3h 25m	2m	70m	20m	8m	6s	66s	14s	2s
CPU time (MM)	8h 43m	14m	74m	84m	14m	8s	149s	16s	5s
# active constraints (1M)	540037	135715	676482	64396	86994	41522	3843	10194	574
# active constraints (MM)	305114	18588	101803	135832	30135	31717	70	748	1548
multiclass SVM	1.20	3.21	**8.04**	**3.40**	15.22	**1.92**	**1.90**	22.24	**3.45**

larger data sets (mnist–yFaces) **smaller data sets** (bal–iris)

Table 1: Results and statistics from all experiments. The data sets are sorted by largest to smallest from left to right. The table shows data statistics and error rates from different variants of LMNN training (single-pass, multi-pass, multi-metric), testing (*k*NN decision rule, energy-based classification), and preprocessing (PCA, LDA). Results from RCA, NCA and multiclass support vector machines (SVMs) are also provided for comparison. See section 5 for discussion of multi-pass and multi-metric LMNN training.

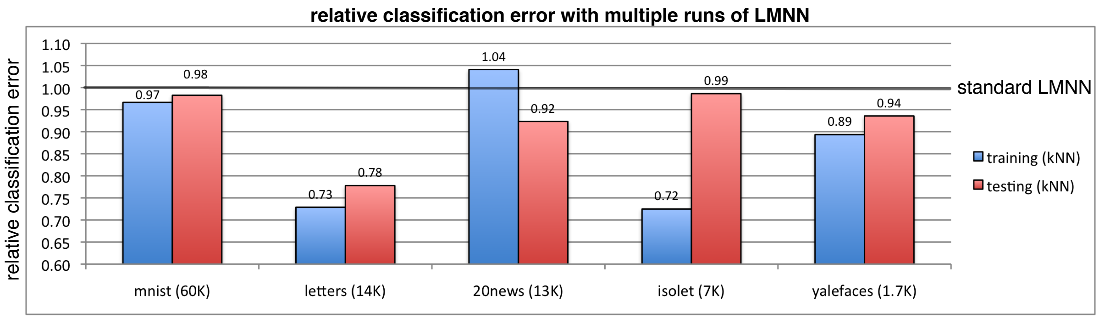


Figure 6: The relative change of the 3-NN classification error after multiple runs of LMNN over a single run of LMNN.

variance). Energy-based LMNN classification yielded a test error rate at 1.4%, cutting the baseline *k*NN error rate by over one-third. Other comparable benchmarks (LeCun et al., 1995) (not exploiting additional prior knowledge) include multilayer neural nets at 1.6% and SVMs at 1.2%. Fig. 5 shows some digits whose nearest neighbor changed as a result of learning, from a mismatch using Euclidean distances to a match using Mahalanobis distances. Table 1 reveals that the LMNN error can be further reduced by learning a different distance metric for each digit class. This is discussed further in section 5.2.

5. Extensions

In this section, we investigate four extensions designed to improve LMNN classification. Section 5.1 examines the impact of multiple consecutive applications of LMNN on one data set. Section 5.2 shows how to learn multiple (locally linear) metrics instead of a single global metric. Section 5.3 discusses how to "kernelize" the algorithm for LMNN classification and reviews complementary work by Torresani and Lee (2007). Finally, section 5.4 investigates the use of LMNN as a method for supervised dimensionality reduction.

5.1 Multi-pass LMNN

One potential weakness of LMNN is that target neighbors must be a priori specified. In the absence of prior knowledge, a default choice is to use Euclidean distances to determine target nearest neighbors. While the target nearest neighbors are fixed during learning, however, the actual nearest neighbors may change as a result of the linear transformation of the input space. These changes suggest an iterative approach, in which the Mahalanobis distances learned in one application (or "pass") of LMNN are used to determine the target nearest neighbors in a subsequent run of the algorithm. More formally, let $\mathbf{L}_p$ be the transformation matrix obtained from the p^{th} pass of LMNN. For the $(p+1)^{th}$ pass, we can assign target neighbors using the Euclidean distance metric after the linear transformation $\vec{x}_i \rightarrow \mathbf{L}_p\mathbf{L}_{p-1}...\mathbf{L}_1\mathbf{L}_0\vec{x}_i$ (with $\mathbf{L}_0 = \mathbf{I}$).

To evaluate this approach, we performed multiple passes of LMNN on all the data sets from Table 1. The parameter k was set to $k = 3$. Figure 6 shows the relative improvements in *k*NN classi-

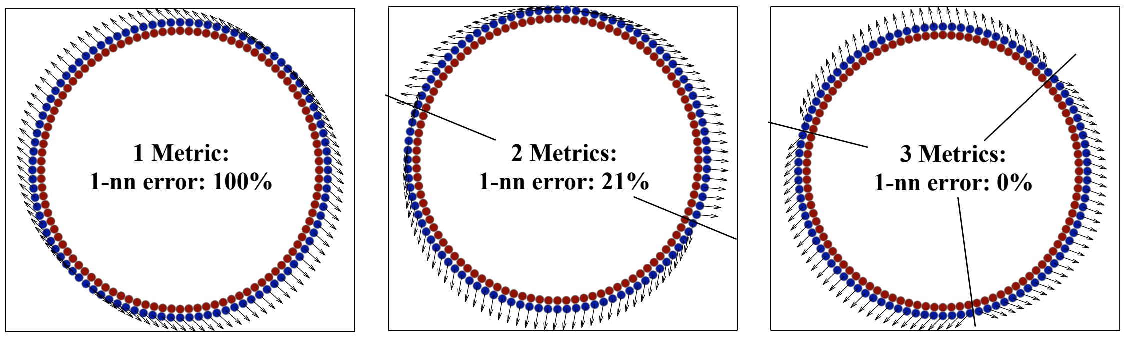


Figure 7: A synthetic data set to illustrate the potential of multiple metrics. The data set consists of inputs sampled from two concentric circles, each of which defines a different class membership. LMNN training was used to estimate one global metric, as well as multiple local metrics. *Left:* a single linear metric cannot model the non-linear decision boundary. The leave-one-out (LOO) error is 100%. *Middle:* if the data set is divided into two clusters (by k-means), and a local metric learned within each cluster, the error rate drops drastically. *Right:* the use of three metrics reduces the LOO-error on the training set to zero. The principal directions of individual distance metrics are indicated by arrows.

fication error rates on the five largest data sets. (Here, a value of one indicates that multiple passes of LMNN did not change the error rate, while a value less than one indicates an improvement.) On these data sets, multiple passes of LMMN were generally helpful, sometimes significantly improving the results. On smaller data sets, though, the multi-pass strategy seemed prone to overfit. Table 1 shows the absolute results on all data sets from multiple passes of LMNN (indicated by MP-LMNN).

A better strategy for choosing target neighbors remains an open question. This aspect of LMNN classification differs significantly from NCA, which does not require the choice of target neighbors. In fact, NCA also determines the effective neighborhood size as part of its optimization. On the other hand, the optimization in NCA is not convex; as such, the initial conditions implicitly specify a basin of attraction that determines the final result. In LMNN classification, the target neighbors are fixed in order to obtain a convex optimization. This trade-off is reminiscent of other convex relaxations of computationally hard problems in machine learning.

5.2 Multi-metric LMNN

On some data sets, a global linear transformation of the input space may not be sufficiently powerful to improve *k*NN classification. Figure 7 shows an example of a synthetic data set for which a single metric is not sufficient. The data set consists of inputs sampled from two concentric circles, each of which defines a different class membership. Global linear transformations cannot improve the accuracy of kNN classification of this data set. In general, highly nonlinear multiclass decision boundaries may not be well modeled by a single Mahalanobis distance metric.

In these situations, one useful extension of LMNN is to learn multiple locally linear transformations instead of a single global linear transformation. In this section, we show how to learn different Mahalanobis distance metrics for different examples in the input space. The idea of learning locally linear distance metrics for kNN classification is at least a decade old (Hastie and Tibshirani, 1996). It has also been explored more recently in the context of metric learning for semi-supervised clustering (Bilenko et al., 2004). The novelty of our approach lies in learning these metrics specifically to maximize the margin of correct kNN classification. As a first step, we partition the training data into disjoint clusters using k-means, spectral clustering (Shi and Malik, 2000), or label information. (In our experience, the latter seems to work best.) We then learn a Mahalanobis distance metric for each cluster. While the training procedure couples the distance metrics in different clusters, the optimization remains a convex problem in semidefinite programming. The globally integrated training of local distance metrics also distinguishes our approach from earlier work (Hastie and Tibshirani, 1996).

Before developing this idea more formally, we first illustrate its potential in a toy setting—namely, on the data set in Fig. 7. For this data set, LMNN training was used to estimate one global metric, as well as multiple local metrics (as described below). Cluster boundaries in the input space were determined by the k-means algorithm. We measured the leave-one-out (LOO) training error (with $k=1$ nearest neighbors) after learning one, two and three metrics. With one metric, the error was 100%; with two metrics, it dropped to 21%; finally, with three metrics, it vanished altogether. The figure illustrates how the multiple metrics adapt to the local structure of the class decision boundaries.

In order to learn different Mahalanobis metrics in different parts of the input space, we minimize a variation of the objective function in Eq. (14). We denote the different metrics by $\mathbf{M}^1,\ldots,\mathbf{M}^c$, where c is the number of clusters. If we partition the training examples by their class labels, then c also coincides with the number of classes; this was done for the remaining experiments in this section. In this case, as the cluster that contains $\vec{x}_i$ is indexed by its label y_i, we can refer to its metric as $\mathbf{M}^{y_i}$. We further define the cluster-dependent distance between two vectors $\vec{x}_i$ and $\vec{x}_j$ as:

$$\hat{\mathcal{D}}(\vec{x}_i,\vec{x}_j) = (\vec{x}_i-\vec{x}_j)^\top \mathbf{M}^{y_j}(\vec{x}_i-\vec{x}_j). \tag{16}$$

Note that this cluster-dependent measure of distance $\hat{\mathcal{D}}(\vec{x}_i,\vec{x}_j)$ is not symmetric with respect to its input arguments. In a slight abuse of terminology, however, we will continue to refer to Eq. (16) as a distance metric; the symmetry is not required for its use in kNN classification. To learn these metrics from data, we solve a modified version of the original SDP:

Minimize $(1-\mu)\sum_{i,j\rightsquigarrow i}(\vec{x}_i-\vec{x}_j)^\top \mathbf{M}^{y_j}(\vec{x}_i-\vec{x}_j)+\mu\sum_{j\rightsquigarrow i,l}(1-y_{il})\xi_{ijl}$
subject to:
(1) $(\vec{x}_i-\vec{x}_l)^\top \mathbf{M}^{y_l}(\vec{x}_i-\vec{x}_l)-(\vec{x}_i-\vec{x}_j)^\top \mathbf{M}^{y_j}(\vec{x}_i-\vec{x}_j) \;\geq\; 1-\xi_{ijl}$
(2) $\xi_{ijl}\geq 0$
(3) $\mathbf{M}^i\succeq 0$ **for** $i=1,\ldots,c$**.**

Note that all the matrices $\mathbf{M}^i$ are learned simultaneously by solving a single SDP. This approach ensures the consistency of distance computations in different clusters: for example, the distance from a test example to training examples with different labels. The integrated learning of different metrics is necessary to calibrate these distances on the same scale; if the metrics were learned independently, then the distances computed by different metrics could not be meaningfully compared—obviously, a crucial requirement for *k*NN classification.

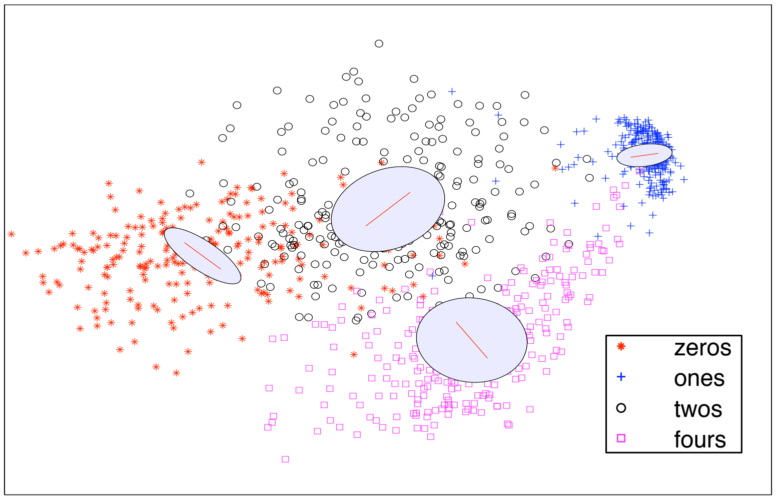


Figure 8: Multiple local distance metrics learned for a data set consisting of handwritten digits *four*, *two*, *one* and *zero*.

Fig. 8 illustrates the multiple metrics learned from an image data set of four different handwritten digits: *zero*, *one*, *four*, and *two*. The plot shows the first two principal components of the data. Only these principal components were used in training in order to yield an easily visualized solution. The solution can be visualized by illustrating the metrics as ellipsoids centered at the class means. The ellipsoids show the effect on a unit circle of each local linear transformation learned by LMNN. The line inside each ellipsoid indicates its principal axis.

We experimented with this multi-metric version of LMNN on all of the data sets from section 4. To avoid overfitting, we held out 30% of each data set's training examples and used them as a validation set. We learned one metric per class. To speed up training, we initialized the multi-metric optimization by setting each class-dependent metric to the solution from LMNN classification with a single global distance metric. Table 1 reports the error rates and other results from all these experiments (under "MM-LMNN"). The training times for MM-LMNN include the time required to compute the initial metric settings from the optimization in Eq. (14).

Fig. 9 shows the relative improvement in 3-NN classification error rates from multi-metric LMNN over standard LMNN on the five largest data sets. The multiple metrics variant improves over standard LMNN on every data set. The best result occurs on the MNIST handwritten digits data set, where MM-LMNN obtained a 1.18% *k*NN classification error rate, slightly outperforming multi-class SVMs. However, the improvement from multi-metric LMNN is not as consistently observed when the energy-based decision rule is used for classification.

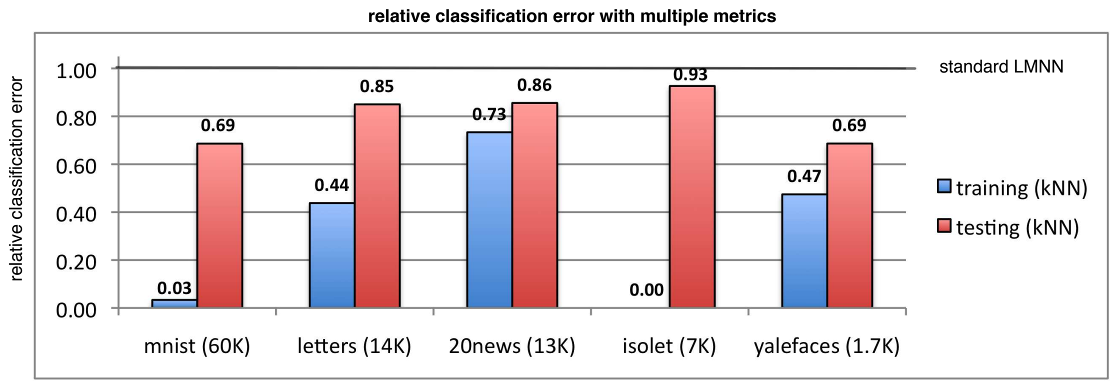


Figure 9: Relative improvement in $k-$NN classification error rates using multiple metrics over error rates using a single metric.

5.3 Kernel Version

LMNN can also be extended by using kernel methods (Schölkopf and Smola, 2002) to work in a nonlinear feature space, as opposed to the original input space. The idea of learning a kernel matrix has been explored in other contexts (Kwok and Tsang, 2003; Lanckriet et al., 2004; Varma and Ray, 2007), particularly large margin classification by support vector machines. This idea for LMNN has been investigated in detail by Torresani and Lee (2007). The "kernel trick" is used to map the inputs $\vec{x}_i$ into higher (possibly infinite) dimensional feature vectors $\Phi(\vec{x}_i)$. To avoid the computational cost of working directly with these feature vectors, they are only accessed through their inner products, which are pre-computed and stored in the kernel matrix:

$$\mathbf{K}_{ij} = \Phi(\vec{x}_i)^\top \Phi(\vec{x}_j).$$

Note how in Eq. (14), the inputs $\vec{x}_i$ are only accessed in terms of the distances in Eq. (3). Torresani and Lee (2007) considered Mahalanobis metrics of the form $\mathbf{M} = \sum_{lm} \mathbf{A}_{lm} \Phi(\vec{x}_l)\Phi(\vec{x}_m)^\top$, where the matrix $\mathbf{A}$ is constrained to be positive semidefinite. They showed that the gradient of Eq. (14) with respect to the matrix $\mathbf{A}$ can be written entirely in terms of the elements of the kernel matrix. Thus, a "kernelized" version of LMNN can be implemented efficiently in the same way as kernel PCA (Schölkopf et al., 1998), without ever working directly in the high dimensional feature space.

Torresani and Lee (2007) show that the kernelized version of LMNN can lead to significant further improvements, but at the cost of increased computation. The increased computation is due to the size of the matrix that must be learned in this setting: the matrix $\mathbf{A}$ has $O(n^2)$ elements instead of $O(d^2)$. (However, the kernel version could require less computation in applications where $n < d$.) More details on the kernelized version of LMNN can be found in their paper.

5.4 Dimensionality Reduction

Often it is useful to generate low dimensional representations of high dimensional data. These representations can be used to visualize the data and/or to accelerate algorithms whose time complexity scales with the input dimensionality. In section 6, for example, we will investigate how to accelerate the *k*NN search in LMNN classification by mapping the training data into a low dimensional subspace.

Low dimensional representations of inputs can be derived from the linear transformation $\vec{x}_i \rightarrow \mathbf{L}\vec{x}_i$ in LMNN classification. This can be done in two ways. The first way is to project the *transformed* inputs onto their leading principal components. Note that if the inputs are whitened prior to optimizing Eq. (13), then these principal components are given simply by the leading eigenvectors of the square matrix $\mathbf{L}$. Another way to derive low dimensional representations is to build this goal explicitly into the optimization for LMNN classification. In particular, we can attempt to minimize Eq. (13) with respect to $\mathbf{L}$ (rather than with respect to $\mathbf{M} = \mathbf{L}^\top \mathbf{L}$) and constrain $\mathbf{L}$ to be rectangular of size $r \times d$, where r is the desired output dimensionality (presumed to be much smaller than the input dimensionality, d). The optimization in terms of $\mathbf{L}$ is not convex, but in practice (Torresani and Lee, 2007), it does not appear to suffer from very poor local minima. In the following section, we use and compare both these methods to build efficient tree data structures for LMNN classification.

6. Metric Trees

One inherent disadvantage of *k*NN search is its relatively high computational complexity at test time. The simplest brute-force way to locate a test example's nearest neighbors is to compute its distance to all the training examples. Such a naïve implementation has a test time-complexity of $O(nd)$, where n is the number of training examples, and d is the input dimensionality.

One way to accelerate *k*NN search is to rotate the input space such that the coordinate axes are aligned with the data's principal components. Such a rotation sorts the input coordinates by decreasing variance; see Section 2.2.1. This ordering can be used to prune unnecessary computations in *k*NN search. In particular, for any test example, a nearest neighbor query consists of computing the distance to each training example and comparing this distance to the k closest examples already located. The distance computation to a particular training example can be aborted upon determining that it lies further away than the k closest examples already located. When the coordinate axes are aligned with the principal components, this determination can often be made after examining just a few of the leading, load-bearing dimensions. We have used this optimization in our baseline implementation of *k*NN search.

Generally there are two major approaches to gain additional speed-ups. The first approach is to reduce the input dimensionality d. The Johnson-Lindenstrauss Lemma (Dasgupta and Gupta, 1999) states that n points can be mapped into a space of dimensionality $O(\frac{\log(n)}{\varepsilon^2})$ such that the distances between any two points changes only by a factor of $(1 \pm \varepsilon)$. Thus we can often reduce the dimensionality of the input data without distorting the nearest neighbor relations. (Note also that for *k*NN classification, we may tolerate inexact nearest neighbor computations if they do not lead to significant errors in classification.) The second approach to speed up *k*NN search is to build a sophisticated tree-based data structure for storing training examples. Such a data structure can reduce the nearest neighbor test time complexity in practice to $O(d \log n)$ (Beygelzimer et al., 2006). This latter method works best for low dimensional data. Fig. 10 compares a baseline implementation of *k*NN search versus one based on ball trees (Liu et al., 2005; Omohundro, 1987). Note how the speed-up from the ball trees is magnified by dimensionality reduction of the inputs.

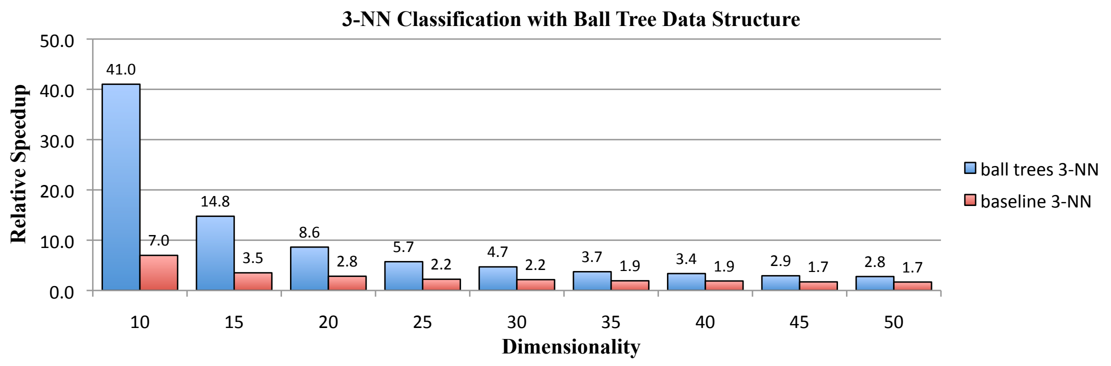


Figure 10: Relative speed-up for 3NN classification obtained from different orthogonal projections of MNIST handwritten digits onto their leading principal components. For these experiments, the $d = 784$ dimensional inputs from the raw images were projected onto the number of principal components shown on the x-axis. The figure compares the speed-ups when ball trees are used (blue) versus when ball trees are not used (red) in the lower dimensional space. Note how the gains from ball trees diminish with increasing dimensionality. All the NN computations in these experiments were accelerated by aligning the coordinate axes along principal components, as described in section 6.

In this section, we explore the use of ball trees for LMNN classification and dimensionality reduction. We find that ball trees can be used for both faster training and testing of LMNN classifiers.

6.1 Review of Ball Trees

Several authors have proposed tree-based data structures to speed up *k*NN search. Examples are kd-trees (Friedman et al., 1977), ball trees (Liu et al., 2005; Omohundro, 1987) and cover-trees (Beygelzimer et al., 2006). All these data structures exploit the same idea: to partition the input space data into hierarchically nested bounding regions. The bounding regions are set up to guarantee that the distance from a test example to a training example inside the bounding region is at least as large as the distance from the test example's to the region's boundary. Thus, for each test example, the training examples inside the bounding region can be ruled out as k nearest neighbors if k training examples have already been found that are closer than the region's boundary. In this case, the *k*NN search can proceed without explicitly computing the distances to training examples in the bounding region. This "pruning" of distance computations often leads to a significant speedup in *k*NN computation time.

We experimented with ball trees (Liu et al., 2005), in which the bounding regions are hyperspheres. Fig. 11 illustrates the basic idea behind ball trees. If a set S of training examples is encapsulated inside a ball with center $\vec{c}$ and radius r, such that $\forall \vec{x} \in S:\ \|\vec{x}-\vec{c}\| \leq r$, then for any test example $\vec{x}_t$ we can bound the distance to any training example inside the ball by the following expression:

$$\forall \vec{x}_i \in S \quad \|\vec{x}_t - \vec{x}_i\| \geq \max(\|\vec{x}_t - \vec{c}\|_2 - r, 0). \tag{17}$$

Ball trees exploit this inequality to build a hierarchical data structure. The data structure is based

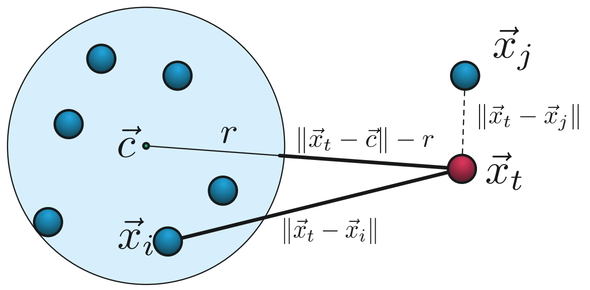

$\vec{x}_j$
$\|\vec{x}_t - \vec{x}_j\|$
$\vec{c}$
r
$\|\vec{x}_t - \vec{c}\| - r$
$\vec{x}_t$
$\vec{x}_i$
$\|\vec{x}_t - \vec{x}_i\|$

Figure 11: The basic idea behind ball trees: for any training example $\vec{x}_i$ inside the ball we can bound the distance $\|\vec{x}_t - \vec{x}_i\|_2$ from below using (17). If another training example $\vec{x}_j$ outsider the ball is already known to be closer than this bound to the test example $\vec{x}_t$, then the training examples inside the ball can be ruled out as nearest neighbors.

on recursively splitting the training examples into two disjoint sets. The sets are encapsulated by hyperspheres (or "balls") which may be partially overlapping. The training examples are recursively divided into smaller and smaller sets until no leaf set contains more than some predefined number of examples.

From this hierarchical data structure, the k-nearest neighbors of a test example can be found by a standard depth-first tree-based search. Recall that each node in the tree has an associated hypersphere that encloses the training examples stored by its descendants. The kNN search proceeds by traversing the tree and computing a test example's distance to the center of each node's hypersphere. The tree is traversed by greedily descending sub-trees in order of this distance. Before descending a subtree, however, Eq. (17) is checked to determine whether training examples in the subtree lie further away than the currently estimated k-nearest neighbors. If this is true, the sub-tree is pruned from the search without further computation. When a leaf node is reached, all the training examples at the leaf node are compared to the currently estimated k-nearest neighbors, and the estimates are updated as necessary. Note that ball trees support exact queries for kNN search.

As pointed out earlier, and as illustrated by Fig. 10, ball trees yield the largest gains in kNN search time for low dimensional data. When the data is high dimensional, the search is plagued by the so-called "curse of dimensionality" (Indyk and Motwani, 1998). In particular, the distances between high dimensional points tend to be more uniform, thereby reducing the opportunities for pruning subtrees in the depth-first search.

6.2 Ball Trees for LMNN Training

The most computationally intensive part of LMNN training is computing the gradient of the penalty for margin violations in Eq. (12). The gradient computation requires a search over all pairs of differently labeled examples to determine if any of them are "impostors" (see section 3.1) that incur margin violations. The solver described in appendix A reduces the number of these searches by maintaining an active list of previous margin violations. Nevertheless, this search scales $O(n^2d)$, which is very computationally intensive for large data sets.

Ball trees can be used to further speed up the search for impostors. Recall how impostors were defined in section 3.1. For any training example $\vec{x}_i$, and for any similarly labeled example $\vec{x}_j$ that

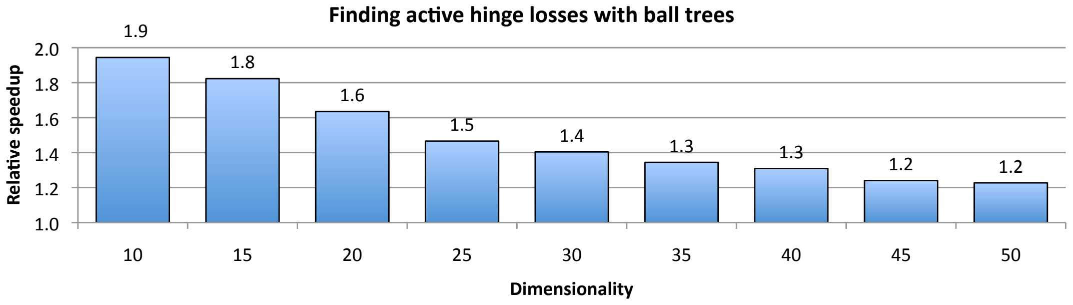


Figure 12: The relative speed-up obtained using ball trees to search for margin violations. The speed-up was measured on the MNIST data set of handwritten digits, with inputs of varying dimensionality derived from PCA. Note how the gains from ball trees diminish with increasing input dimensionality.

is one of its target k-nearest neighbors (with $j \rightsquigarrow i$), the impostors consist of all differently labeled examples $\vec{x}_l$ (with $y_{il} = 0$) that satisfy Eq. (10). Ball trees can be used to search for all training examples that meet this criterion. As in their use for kNN search, many subtrees in the depth-first search for impostors can be pruned: if for some ball the lower bound distances between examples is already greater than the right hand side of Eq. (10), then all the examples stored in the subtree can be ruled out as impostors. Note that for each training example $\vec{x}_i$, we only need to search for impostors among other training examples $\vec{x}_l$ that have a different class label (with $y_{il} = 0$). Thus, we build one ball tree data structure per class and perform a separate search for impostors in each class.

Fig. 12 shows the relative speed-up when ball trees are used to search for margin violations in LMNN classification. The figure shows results from experiments with the MNIST images of handwritten digits. For these experiments, the images were projected into subspaces of varying dimensionality using PCA. The gains from ball trees in this context are significant, though not as dramatic as those in Fig. 10 for simple kNN search. The lesser gains for LMNN classification can be attributed to the minimum enforced margin of unit distance, which sometimes causes a high number of sub-trees to be traversed. This effect is controlled by the relative magnitude of the unit margin; it can be partially offset by scaling the input data by a constant factor before training.

6.3 Ball Trees for LMNN Testing

Ball trees can also be used to accelerate kNN search at test time. We have observed earlier, though, that the speed-up from ball trees diminishes quickly as the input dimensionality increases; see Fig. 10. If very fast kNN classification using ball trees is desired on a large data set, then often it is necessary to work with a lower dimensional representation of the training examples.

The most commonly used methods for dimensionality reduction in ball trees are random projections and PCA. Neither of these methods, however, is especially geared to preserve the accuracy of kNN classification. There is an inherent trade-off between dimensionality reduction and nearest

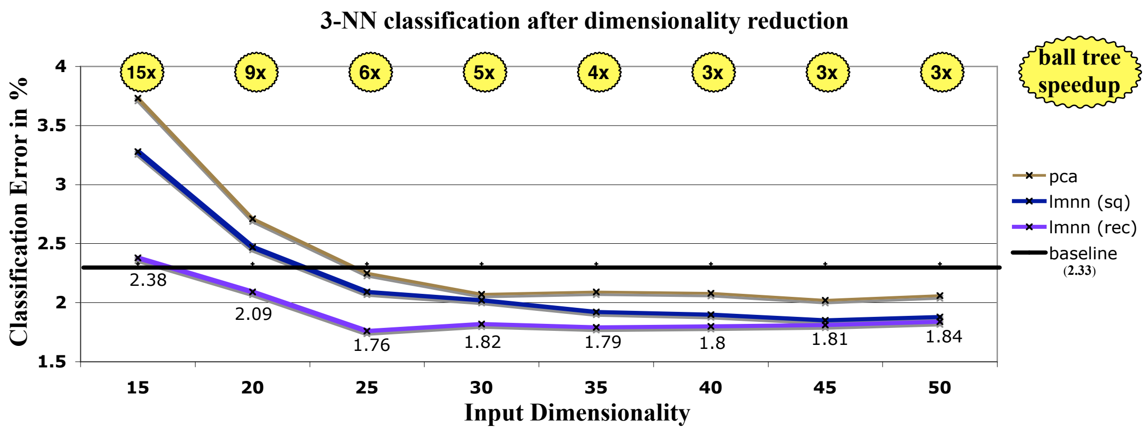


Figure 13: Graph of *k*NN classification error (with $k = 3$) on different low dimensional representations of the MNIST data set; see text for details. The speed-up from ball-trees is shown at the top of the graph.

neighbor preservation. Nearest neighbor relationships can change when the training examples are projected into a lower dimensional space, resulting in significantly worse *k*NN classification.

In this section, we explore how the distance metric learned for LMNN classification can be used for more effective dimensionality reduction in ball trees. In section 5.4, we described two different ways to derive low dimensional representations for LMNN classification. The first computed a low-rank approximation to the (generally full rank) matrix $\mathbf{L}$; the second directly learned a low-rank rectangular matrix $\mathbf{L}$ by optimizing the non-convex loss function in Eq. (13). For shorthand, we refer to these approaches for dimensionality reduction as LMNN-S and LMNN-R, denoting whether a square (S) or rectangular (R) matrix is learned to minimize the LMNN cost function. Fig. 13 shows the results of *k*NN classification from both these methods on the MNIST data set of handwritten digits. For these experiments, the raw MNIST images (of size 28×28) were projected onto their 350 leading principal components before any training for LMNN classification. Also shown in the figure are the results from further dimensionality reduction using PCA, as well as the baseline *k*NN error rate in the original (high dimensional) input space. The square matrix in LMNN-S was of size 350×350, and for dimensionality reduction, the data was projected onto the r leading eigenvectors of linear transformation $\mathbf{L}$. The rectangular matrix in LMNN-R was of size $r \times 350$, where r varied from 15 to 50. The speed-up from ball trees is shown at the top of the graph. The amount of speed-up depends significantly on the amount of dimensionality reduction, but very little on the particular method of dimensionality reduction.

The results show that LMNN can be used effectively for dimensionality reduction. For example, LMNN-R achieves a *k*NN test error rate of 2.38% in 15 dimensions, only slightly higher than the baseline error rate of 2.33% in the original input space. In this space, moreover, ball trees yield a 15x speedup over baseline *k*NN search. In 25 dimensions, the LMNN-R error rate drops further to 1.76% while still yielding a 5.7x speed-up. Of the three methods compared in Fig. 13, LMNN-R is the most effective. In fact, though working in many fewer dimensions, LMNN-R obtains results very close to the best results reported in section 4. It is interesting that LMNN-R outperforms LMNN-S, though (as expected) their results converge as the rectangular matrix in LMNN-R becomes more square. These results show that aggressive dimensionality reduction can be combined with highly accurate *k*NN classification.

7. Discussion

In this paper, we have introduced a new framework for large margin nearest neighbor (LMNN) classification. From labeled training examples, we have shown how to learn a Mahalanobis distance metric for *k*NN classification. The required optimization was formulated as an instance of semidefinite programming. Our framework makes no parametric assumptions about the structure or distribution of the data and scales naturally to problems with large number of classes. On multiple data sets, we have demonstrated that we can significantly improve the accuracy of *k*NN classification by learning a metric in this way. We have also shown that an alternative energy-based decision rule typically leads to further improvements over traditional *k*NN classification.

Beyond the basic framework for LMNN classification, we described several useful and complementary extensions. These included: iterative re-estimation of target neighbor assignments, globally integrated learning of multiple locally linear metrics, kernel methods for LMNN classification, low-rank distance metrics for dimensionality reduction, and ball trees for more efficient gradient computations (in training) and *k*NN search (in testing). These extensions can be adapted and combined to meet the demands of particular applications. For example, to build a highly accurate classifier without regard to the actual computation at test time, our results suggest to train multiple locally linear metrics. At the other extreme, to build a *k*NN classifier that is as fast as possible at test time, our results suggest to combine low-rank distance metrics with ball trees.

Taken as a whole, our results demonstrate the promise and widespread applicability of LMNN classification. Perhaps the greatest promise lies in problems with very large numbers of classes, such as face and identity recognition. The number of classes in these problems can be in the hundreds, thousands, or more. Nearest neighbor methods handle this regime more transparently than other leading methods, such as SVMs. The ideas behind LMNN classification have also been extended by others in various ways (Torresani and Lee, 2007; Kumar et al., 2007). In the appendix, we describe a simple solver that scales well to problems with tens of thousands of examples. A MATLAB implementation of the algorithm is also freely available with this paper.

Future work will concentrate on several open problems. The improved performance with multiple metrics suggests that LMNN classification could benefit from even more adaptive transformations of the input space. It would also be useful to study LMMN classification in the semi-supervised, transductive setting, where only a few labeled inputs are available for training but the unlabeled test set is known in advance. Finally, for many real-world applications in computer vision and information retrieval, the data sets can be much larger than the ones we have studied. For very large data sets, our current implementation for LMNN does not scale as well as simpler eigenvector methods such as PCA, LDA, and RCA. It remains an interesting challenge to scale LMNN to even larger data sets with millions or more training examples.

Acknowledgments

We especially thank John C. Blitzer for his many suggestions to improve the algorithm and his generous help with various data sets. We also thank Koby Crammer for many useful comments and suggestions. This work was supported by NSF Award 0238323.

Appendix A. Solver

We implemented our own special-purpose solver for large-scale problems in LMNN classification. Our solver was designed to exploit the particular structure of the cost function in Eq. (13). The solver iteratively re-estimates the Mahalanobis distance metric as it attempts to minimize the cost function for LMNN classification. The amount of computation is minimized by careful book-keeping from one iteration to the next. The speed-ups from these optimizations enabled us to work comfortably on data sets with up to $n = 60,000$ training examples.

Our solver implements an iterative sub-gradient projection method to optimize Eq. (14) in terms of the positive semidefinite matrix $\mathbf{M}$. We refer to the Mahalanobis distance metric at the tth iteration as $\mathbf{M}_t$ and to its squared Mahalanobis distance in Eq. (3) as $\mathcal{D}_t$. At each iteration, the optimization takes a step along the sub-gradient to reduce the loss function and then projects $\mathbf{M}_t$ onto the feasible set. In our case, the feasible set is the cone of all positive semidefinite matrices $\mathcal{S}_+$. The following sections derive the gradient and describe the projection onto $\mathcal{S}_+$.

It is worth emphasizing that although we can phrase the optimization of Eq. (14) as a semidefinite program (by introducing nonnegative slack variables to model the hinge loss), in practice our large-scale solver works directly to minimize Eq. (14). The hinge losses that appear in this loss function are not differentiable at all points. Nevertheless, because the loss function is convex, we can compute its sub-gradient and use standard hill-climbing algorithms to find its minimum. It has been shown that such sub-gradient methods converge to the correct solution, provided that the gradient step-size is sufficiently small (Boyd and Vandenberghe, 2004).

A.1 Gradient Computation

The gradient computation can be done most efficiently by careful book-keeping from one iteration to the next. As simplifying notation, let $\mathbf{C}_{ij} = (\vec{x}_i - \vec{x}_j)(\vec{x}_i - \vec{x}_j)^\top$. It is straightforward to express the distances, as defined in Eq. (3), in terms of this notation. In particular, at the tth iteration, we have $\mathcal{D}_t(\vec{x}_i, \vec{x}_j) = \mathrm{tr}(\mathbf{M}_t\mathbf{C}_{ij})$. Consequently, we can rewrite the loss function in Eq. (14) as:

$$\varepsilon(\mathbf{M}_t) \;=\; (1-\mu)\sum_{i,j\rightsquigarrow i}\mathrm{tr}(\mathbf{M}_t\mathbf{C}_{ij}) + \mu\sum_{j\rightsquigarrow i,l}(1-y_{il})\left[1+\mathrm{tr}(\mathbf{M}_t\mathbf{C}_{ij}) - \mathrm{tr}(\mathbf{M}_t\mathbf{C}_{il})\right]_+ \tag{18}$$

Note that Eq. (18) is piecewise linear with respect to $\mathbf{M}_t$. Let us define a set of triples $\mathcal{N}^t$, such that $(i,j,l) \in \mathcal{N}^t$ if and only if the indices (i,j,l) trigger the hinge loss in the second part of Eq. (18). With this definition, we can write the gradient $\mathbf{G}_t$ of $\varepsilon(\mathbf{M}_t)$ as:

$$\mathbf{G}_t = \frac{\partial\varepsilon}{\mathbf{M}_t} = (1-\mu)\sum_{i,j\rightsquigarrow i}\mathbf{C}_{ij} + \mu\sum_{(i,j,l)\in\mathcal{N}^t}(\mathbf{C}_{ij} - \mathbf{C}_{il}).$$

Computing the gradient requires computing the outer products in $\mathbf{C}_{ij}$; it thus scales quadratically in the input dimensionality. As the set $\mathcal{N}_t$ is potentially very large, a naïve computation of the gradient would be extremely expensive. However, we can exploit the fact that the gradient contribution from each active triplet (i,j,l) does not depend on the degree of its margin violation. Thus, the changes in the gradient from one iteration to the next are determined entirely by the differences between the sets $\mathcal{N}_t$ and $\mathcal{N}_{t+1}$. We can use this fact to derive an extremely efficient update that relates the gradient $\mathbf{G}_{t+1}$ at iteration $t+1$ from the gradient $\mathbf{G}_t$ at iteration t. The update simply subtracts the contributions from triples that are no longer active and adds the contributions of those that just became active:

$$\mathbf{G}_{t+1} = \mathbf{G}_t - \mu \sum_{(i,j,l)\in\mathcal{N}_t - \mathcal{N}_{t+1}} (\mathbf{C}_{ij} - \mathbf{C}_{il}) + \mu \sum_{(i,j,l)\in\mathcal{N}_{t+1} - \mathcal{N}_t} (\mathbf{C}_{ij} - \mathbf{C}_{il}). \tag{19}$$

For small gradient step sizes, the set $\mathcal{N}_t$ changes very little from one iteration to the next. In this case, computing the right hand side of Eq. (19) is extremely fast.

To accelerate the solver even further, we adopt an active set method. Note that computing the set $\mathcal{N}_t$ at each iteration requires checking every triplet (i,j,l) with $j \rightsquigarrow i$ for a potential margin violation. This computation scales as $O(nd^2 + kn^2d)$, making it impractical for large data sets. To avoid this computational burden, we exploit the fact that the great majority of triples do not incur margin violations: in particular, for each training example, only a very small fraction of differently labeled examples typically lie nearby in the input space. Consequently, a useful approximation is to check only a subset of likely triples for margin violations per gradient computation. We initialize the training procedure by checking all triples and maintaining an active list of those with margin violations; however, a full re-check is only made every 10-20 iterations, depending on fluctuations of the set $\mathcal{N}_t$. For intermediate iterations, we only check for margin violations from among those active triples accumulated over previous iterations. When the optimization converges, we verify that the working set $\mathcal{N}_t$ does contain all active triples that incur margin violations. This final check is needed to ensure convergence to the correct minimum. If the check is not satisfied, the optimization restarts with the newly expanded active set.

A.2 Projection

The minimization of Eq. (18) must enforce the constraint that the matrix $\mathbf{M}_t$ remains positive semidefinite. To enforce this constraint, we project $\mathbf{M}_t$ onto the cone of all positive semidefinite matrices $\mathcal{S}_+$ after each gradient step. This projection is computed from the diagonalization of $\mathbf{M}_t$. Let $\mathbf{M}_t = \mathbf{V}\Delta\mathbf{V}^\top$ denote the eigendecomposition of $\mathbf{M}_t$, where $\mathbf{V}$ is the orthonormal matrix of eigenvectors and Δ is the diagonal matrix of corresponding eigenvalues. We can further decompose $\Delta = \Delta^- + \Delta^+$, where $\Delta^+ = \max(\Delta, 0)$ contains all the positive eigenvalues and $\Delta^- = \min(\Delta, 0)$ contains all the negative eigenvalues. The projection of $\mathbf{M}_t$ onto the cone of positive semidefinite matrices is given by:

$$\mathcal{P}_{\mathcal{S}}(\mathbf{M}_t) = \mathbf{V}\Delta^+\mathbf{V}^\top. \tag{20}$$

The projection effectively truncates any negative eigenvalues from the gradient step, setting them equal to zero.

A.3 Algorithm

Our gradient projection algorithm combined the update rules for the gradient in Eq. (19) and the projection in Eq. (20). A simplified pseudo-code implementation is shown in Algorithm 1. We denote the gradient step-size by $\alpha > 0$. In practice, it worked best to start with a small value of α. Then, at each iteration, we increased α by a factor of 1.01 if the loss function decreased and decreased α by a factor of 0.5 if the loss function increased.

Algorithm 1 A simple gradient projection pseudo-code implementation.

```
 1: $\mathbf{M}_0 := \mathbf{I}$ {Initialize with the identity matrix}
 2: $t := 0$ {Initialize counter}
 3: $\mathcal{N}^{(0)}, \mathcal{N}_0 := \{\}$ {Initialize active sets}
 4: $\mathbf{G}_0 := (1-\mu)\sum_{i,j\rightsquigarrow i}\mathbf{C}_{ij}$ {Initialize gradient}
 5: while (not converged) do
 6:     if mod$(t, someconstant) = 0 \vee$ (almost converged){we used $someconstant$=10} then
 7:         compute $\mathcal{N}_{t+1}$ exactly
 8:         $\mathcal{N}^{(t+1)} := \mathcal{N}^{(t)} \cup \mathcal{N}_{t+1}$ {Update active set}
 9:     else
10:         compute $\mathcal{N}_{t+1} \approx \mathcal{N}_{t+1} \cap \mathcal{N}^{(t)}$ { Only search active set}
11:         $\mathcal{N}^{(t+1)} := \mathcal{N}^{(t)}$ {Keep active set untouched}
12:     end if
13:     $\mathbf{G}_{t+1} := \mathbf{G}_t - \mu\sum_{(i,j,l)\in\mathcal{N}_t - \mathcal{N}_{t+1}}(\mathbf{C}_{ij} - \mathbf{C}_{il}) + \mu\sum_{(i,j,l)\in\mathcal{N}_{t+1} - \mathcal{N}_t}(\mathbf{C}_{ij} - \mathbf{C}_{il})$
14:     $\mathbf{M}_{t+1} := \mathcal{P}_{\mathcal{S}}(\mathbf{M}_t - \alpha\mathbf{G}_{t+1})$ {Take gradient step and project onto SDP cone}
15:     $t := t + 1$
16: end while
17: Output $\mathbf{M}_t$
```

References

A. Bar-Hillel, T. Hertz, N. Shental, and D. Weinshall. Learning a Mahalanobis metric from equivalence constraints. *Journal of Machine Learning Research*, 6(1):937–965, 2006.

S. Belongie, J. Malik, and J. Puzicha. Shape matching and object recognition using shape contexts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24 (4):509–522, 2002.

A. Beygelzimer, S. Kakade, and J. Langford. Cover trees for nearest neighbor. In *Proceedings of the Twenty Third International Conference on Machine Learning*, pages 97–104, Pittsburgh, PA, 2006.

M. Bilenko, S. Basu, and R.J. Mooney. Integrating constraints and metric learning in semi-supervised clustering. In *Proceedings of the Twenty First International Conference on Machine Learning (ICML-04)*, pages 839–846, Banff, Canada, 2004.

S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

S. Chopra, R. Hadsell, and Y. LeCun. Learning a similiarty metric discriminatively, with application to face verification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR-05)*, pages 349–356, San Diego, CA, 2005.

T. Cover and P. Hart. Nearest neighbor pattern classification. In *IEEE Transactions in Information Theory, IT-13*, pages 21–27, 1967.

K. Crammer and Y. Singer. On the algorithmic implementation of multiclass kernel-based vector machines. *Journal of Machine Learning Research*, 2:265–292, 2001.

S. Dasgupta and A. Gupta. An elementary proof of the Johnson-Lindenstrauss lemma. Technical Report 99–006, International Computer Science Institute, UC Berkeley, 1999.

T. De Bie, M. Momma, and N. Cristianini. Efficiently Learning the Metric with Side-Information. *Lecture Notes in Computer Science*, pages 175–189, 2003.

T. G. Dietterich and G. Bakiri. Solving multiclass learning problems via error-correcting output codes. In *Journal of Artificial Intelligence Research*, volume 2, pages 263–286, 1995.

R. A. Fisher. The use of multiple measures in taxonomic problems. *Annals of Eugenics*, 7:179–188, 1936.

J. H. Friedman, J. L. Bentley, and R. A. Finkel. An algorithm for finding best matches in logarithmic expected time. *ACM Transactions on Mathematical Software*, 3(3):209–226, 1977.

A. Globerson and S. T. Roweis. Metric learning by collapsing classes. In *Advances in Neural Information Processing Systems 18*, 2006.

J. Goldberger, S. Roweis, G. Hinton, and R. Salakhutdinov. Neighbourhood components analysis. In L. K. Saul, Y. Weiss, and L. Bottou, editors, *Advances in Neural Information Processing Systems 17*, pages 513–520, Cambridge, MA, 2005. MIT Press.

T. Hastie and R. Tibshirani. Discriminant adaptive nearest neighbor classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 18:607–616, 1996.

P. Indyk and R. Motwani. Approximate nearest neighbors: towards removing the curse of dimensionality. *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pages 604–613, 1998.

I. T. Jolliffe. *Principal Component Analysis*. Springer-Verlag, New York, 1986.

M. P. Kumar, P. H. S. Torr, and A. Zisserman. An invariant large margin nearest neighbour classifier. In *Proceedings of the Eleventh IEEE International Conference on Computer Vision (ICCV-07)*, pages 1–8, Rio de Janeiro, Brazil, 2007.

J.T. Kwok and I.W. Tsang. Learning with idealized kernels. In *Proceedings of the Twentieth International Conference on Machine Learning (ICML-03)*, pages 400–407, Washington, D.C., 2003.

G. R. G. Lanckriet, N. Cristianini, P. Bartlett, L. El Ghaoui, and M. I. Jordan. Learning the kernel matrix with semidefinite programming. *Journal of Machine Learning Research*, 5:27–72, 2004.

Y. LeCun, L. Jackel, L. Bottou, A. Brunot, C. Cortes, J. Denker, H. Drucker, I. Guyon, U. Muller, E. Sackinger, P. Simard, and V. Vapnik. A comparison of learning algorithms for handwritten digit recognition. In F. Fogelman and P. Gallinari, editors, *Proceedings of the 1995 International Conference on Artificial Neural Networks (ICANN-95)*, pages 53–60, Paris, 1995.

T. Liu, A. W. Moore, A. Gray, and K. Yang. An investigation of practical approximate nearest neighbor algorithms. In L. K. Saul, Y. Weiss, and L. Bottou, editors, *Advances in Neural Information Processing Systems 17*, pages 825–832. MIT Press, Cambridge, MA, 2005.

A. Kachites McCallum. Bow: A toolkit for statistical language modeling, text retrieval, classification and clustering. http://www.cs.cmu.edu/ mccallum/bow, 1996.

K.-R. Müller, S. Mika, G. Räsch, K. Tsuda, and B. Schökopf. An introduction to kernel-based learning algorithms. *IEEE Transactions on Neural Networks*, 12(2):181–201, 2001.

S. Omohundro. Efficient algorithms with neural network behavior. *Complex Systems*, 1:273–347, 1987.

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

B. Schölkopf, A. J. Smola, and K.-R. Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 10:1299–1319, 1998.

S. Shalev-Shwartz, Y. Singer, and A. Y. Ng. Online and batch learning of pseudo-metrics. In *Proceedings of the Twenty First International Conference on Machine Learning (ICML-04)*, pages 94–101, Banff, Canada, 2004.

N. Shental, T. Hertz, D. Weinshall, and M. Pavel. Adjustment learning and relevant component analysis. In *Proceedings of the Seventh European Conference on Computer Vision (ECCV-02)*, volume 4, pages 776–792, London, UK, 2002. Springer-Verlag.

J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, pages 888–905, August 2000.

P. Y. Simard, Y. LeCun, and J. Decker. Efficient pattern recognition using a new transformation distance. In S. Hanson, J. Cowan, and L. Giles, editors, *Advances in Neural Information Processing Systems 6*, pages 50–58, San Mateo, CA, 1993. Morgan Kaufman.

L. Torresani and K.C. Lee. Large margin component analysis. In B. Schölkopf, J. Platt, and T. Hofmann, editors, *Advances in Neural Information Processing Systems 19*, pages 1385–1392. MIT Press, Cambridge, MA, 2007.

I.W. Tsang, P.M. Cheung, and J.T. Kwok. Kernel relevant component analysis for distance metric learning. In *Proceedings of the IEEE International Joint Conference on Neural Networks (IJCNN-05)*, volume 2, pages 954–959, Montreal, Canada, 2005.

M. Turk and A. Pentland. Eigenfaces for recognition. *Journal of Cognitive Neuroscience*, 3(1): 71–86, 1991.

L. Vandenberghe and S. P. Boyd. Semidefinite programming. *SIAM Review*, 38(1):49–95, March 1996.

M. Varma and D. Ray. Learning the discriminative power-invariance trade-off. In *Proceedings of the Eleventh IEEE International Conference on Computer Vision (ICCV-07)*, pages 1–8, 2007.

K. Q. Weinberger and L. K. Saul. Fast solvers and efficient implementations for distance metric learning. In *Proceedings of the Twenty Fifth International Conference on Machine learning*, pages 1160–1167, Helsinki, Finland, 2008.

K. Q. Weinberger, J. Blitzer, and L. Saul. Distance metric learning for large margin nearest neighbor classification. In Y. Weiss, B. Schölkopf, and J. Platt, editors, *Advances in Neural Information Processing Systems 18*, pages 1473–1480. MIT Press, Cambridge, MA, 2006.

E. P. Xing, A. Y. Ng, M. I. Jordan, and S. Russell. Distance metric learning, with application to clustering with side-information. In T. G. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems 14*, pages 521–528, Cambridge, MA, 2002. MIT Press.

Journal of Machine Learning Research 10 (2009) 245-279 Submitted 3/08; Revised 9/08; Published 2/09

Data-driven Calibration of Penalties for Least-Squares Regression

Sylvain Arlot SYLVAIN.ARLOT@ENS.FR
CNRS ; Willow Project-Team
Laboratoire d'Informatique de l'Ecole Normale Superieure
(CNRS/ENS/INRIA UMR 8548)
45, rue d'Ulm, 75230 Paris, France
Pascal Massart PASCAL.MASSART@MATH.U-PSUD.FR
Univ Paris-Sud, UMR 8628
Laboratoire de Mathematiques
Orsay, F-91405

Editor: John Lafferty

Abstract

Penalization procedures often suffer from their dependence on multiplying factors, whose optimal values are either unknown or hard to estimate from data. We propose a completely data-driven calibration algorithm for these parameters in the least-squares regression framework, without assuming a particular shape for the penalty. Our algorithm relies on the concept of minimal penalty, recently introduced by Birgé and Massart (2007) in the context of penalized least squares for Gaussian homoscedastic regression. On the positive side, the minimal penalty can be evaluated from the data themselves, leading to a data-driven estimation of an optimal penalty which can be used in practice; on the negative side, their approach heavily relies on the homoscedastic Gaussian nature of their stochastic framework.

The purpose of this paper is twofold: stating a more general heuristics for designing a data-driven penalty (the *slope heuristics*) and proving that it works for penalized least-squares regression with a random design, even for heteroscedastic non-Gaussian data. For technical reasons, some exact mathematical results will be proved only for regressogram bin-width selection. This is at least a first step towards further results, since the approach and the method that we use are indeed general.

Keywords: data-driven calibration, non-parametric regression, model selection by penalization, heteroscedastic data, regressogram

1. Introduction

In the last decades, model selection has received much interest, commonly through penalization. In short, penalization chooses the model minimizing the sum of the empirical risk (how well the algorithm fits the data) and of some measure of complexity of the model (called penalty); see FPE (Akaike, 1970), AIC (Akaike, 1973), Mallows' C_p or C_L (Mallows, 1973). Many other penalization procedures have been proposed since, among which Rademacher complexities (Koltchinskii, 2001; Bartlett et al., 2002), local Rademacher complexities (Bartlett et al., 2005; Koltchinskii, 2006), bootstrap penalties (Efron, 1983), resampling and V-fold penalties (Arlot, 2008b,c).

Model selection can target two different goals. On the one hand, a procedure is *efficient* (or asymptotically optimal) when its quadratic risk is asymptotically equivalent to the risk of the oracle.

On the other hand, a procedure is *consistent* when it chooses the smallest true model asymptotically with probability one. This paper deals with *efficient* procedures, without assuming the existence of a true model.

A huge amount of literature exists about efficiency. First Mallows' C_p, Akaike's FPE and AIC are asymptotically optimal, as proved by Shibata (1981) for Gaussian errors, by Li (1987) under suitable moment assumptions on the errors, and by Polyak and Tsybakov (1990) under sharper moment conditions, in the Fourier case. Non-asymptotic oracle inequalities (with some leading constant $C > 1$) have been obtained by Barron et al. (1999) and by Birgé and Massart (2001) in the Gaussian case, and by Baraud (2000, 2002) under some moment assumptions on the errors. In the Gaussian case, non-asymptotic oracle inequalities with leading constant C_n tending to 1 when n tends to infinity have been obtained by Birgé and Massart (2007).

However, from the practical point of view, both AIC and Mallows' C_p still present serious drawbacks. On the one hand, AIC relies on a strong asymptotic assumption, so that for small sample sizes, the optimal multiplying factor can be quite different from one. Therefore, corrected versions of AIC have been proposed (Sugiura, 1978; Hurvich and Tsai, 1989). On the other hand, the optimal calibration of Mallows' C_p requires the knowledge of the noise level σ^2, assumed to be constant. When real data are involved, σ^2 has to be estimated separately and independently from any model, which is a difficult task. Moreover, the best estimator of σ^2 (say, with respect to the quadratic error) quite unlikely leads to the most efficient model selection procedure. Contrary to Mallows' C_p, the data-dependent calibration rule defined in this article is not a "plug-in" method; it focuses directly on efficiency, which can improve significantly the performance of the model selection procedure.

Existing penalization procedures present similar or stronger drawbacks than AIC and Mallows' C_p, often because of a gap between theory and practice. For instance, oracle inequalities have only been proved for (global) Rademacher penalties multiplied by a factor two (Koltchinskii, 2001), while they are used without this factor (Lozano, 2000). As proved by Arlot (2007, Chapter 9), this factor is necessary in general. Therefore, the optimal calibration of these penalties is really an issue. The calibration problem is even harder for local Rademacher complexities: theoretical results hold only with large calibration constants, particularly the multiplying factor, and no optimal values are known. One of the purposes of this paper is to address the issue of optimizing the multiplying factor for general-shape penalties.

Few automatic calibration algorithms are available. The most popular ones are certainly cross-validation methods (Allen, 1974; Stone, 1974), in particular V-fold cross-validation (Geisser, 1975), because these are general-purpose methods, relying on a widely valid heuristics. However, their computational cost can be high. For instance, V-fold cross-validation requires the entire model selection procedure to be performed V times for each candidate value of the constant to be calibrated. For penalties proportional to the dimension of the models, such as Mallows' C_p, alternative calibration procedures have been proposed by George and Foster (2000) and by Shen and Ye (2002).

A completely different approach has been proposed by Birgé and Massart (2007) for calibrating dimensionality-based penalties. Since this article extends their approach to a much wider range of applications, let us briefly recall their main results. In Gaussian homoscedastic regression with a fixed design, assume that each model is a finite-dimensional vector space. Consider the penalty $\mathrm{pen}(m) = KD_m$, where D_m is the dimension of the model m and $K > 0$ is a positive constant, to be calibrated. First, there exists a *minimal* constant $K_{\min}$, such that the ratio between the quadratic risk of the chosen estimator and the quadratic risk of the oracle is asymptotically infinite if $K < K_{\min}$, and finite if $K > K_{\min}$. Second, when $K = K^\star := 2K_{\min}$, the penalty KD_m yields an efficient model selection procedure. In other words, *the optimal penalty is twice the minimal penalty*. This relationship characterizes the "slope heuristics" of Birgé and Massart (2007).

A crucial fact is that the minimal constant $K_{\min}$ can be estimated from the data, since large models are selected if and only if $K < K_{\min}$. This leads to the following strategy for choosing K from the data. For every $K \geq 0$, let $\widehat{m}(K)$ be the model selected by minimizing the empirical risk penalized by $\operatorname{pen}(D_m) = KD_m$. First, compute $K_{\min}$ such that $D_{\widehat{m}(K)}$ is "huge" for $K < K_{\min}$ and "reasonably small" when $K \geq K_{\min}$; explicit values for "huge" and "small" are proposed in Section 3.3. Second, define $\widehat{m} := \widehat{m}(2K_{\min})$. Such a method has been successfully applied for multiple change points detection by Lebarbier (2005).

From the theoretical point of view, the issue for understanding and validating this approach is the existence of a minimal penalty. This question has been addressed for Gaussian homoscedastic regression with a fixed design by Birgé and Massart (2001, 2007) when the variance is known, and by Baraud et al. (2007) when the variance is unknown. Non-Gaussian or heteroscedastic data have never been considered. This article contributes to fill this gap in the theoretical understanding of penalization procedures.

The calibration algorithm proposed in this article relies on a generalization of Birgé and Massart's slope heuristics (Section 2.3). In Section 3, the algorithm is defined in the least-squares regression framework, for general-shape penalties. The shape of the penalty itself can be estimated from the data, as explained in Section 3.4.

The theoretical validation of the algorithm is provided in Section 4, from the *non-asymptotic point of view*. Non-asymptotic means in particular that the collection of models is allowed to depend on n: in practice, it is usual to allow the number of explanatory variables to increase with the number of observations. Considering models with a large number of parameters (for example of the order of a power of the sample size n) is also necessary to approximate functions belonging to a general approximation space. Thus, the non-asymptotic point of view allows us not to assume that the regression function is described with a small number of parameters.

The existence of minimal penalties for *heteroscedatic regression with a random design* (Theorem 2) is proved in Section 4.3. In Section 4.4, by proving that twice the minimal penalty has some optimality properties (Theorem 3), we extend the so-called slope heuristics to heteroscedatic regression with a random design. Moreover, neither Theorem 2 nor Theorem 3 assume the data to be Gaussian; only mild moment assumptions are required.

For proving Theorems 2 and 3, each model is assumed to be the vector space of piecewise constant functions on some partition of the feature space. This is indeed a restriction, but we conjecture that it is mainly technical, and that the slope heuristics remains valid at least in the general least-squares regression framework. We provide some evidence for this by proving two key concentration inequalities without the restriction to piecewise constant functions. Another argument supporting this conjecture is that recently several simulation studies have shown that the slope heuristics can be used in several frameworks: mixture models (Maugis and Michel, 2008), clustering (Baudry, 2007), spatial statistics (Verzelen, 2008), estimation of oil reserves (Lepez, 2002) and genomics (Villers, 2007). Although the slope heuristics has not been formally validated in these frameworks, this article is a first step towards such a validation, by proving that the slope heuristics can be applied whatever the shape of the ideal penalty.

This paper is organized as follows. The framework and the slope heuristics are described in Section 2. The resulting algorithm is defined in Section 3. The main theoretical results are stated in Section 4. All the proofs are given in Appendix A.

2. Framework

In this section, we describe the framework and the general slope heuristics.

2.1 Least-squares Regression

Suppose we observe some data $(X_1,Y_1),\dots(X_n,Y_n) \in \mathcal{X} \times \mathbb{R}$, independent with common distribution P, where the feature space $\mathcal{X}$ is typically a compact set of $\mathbb{R}^d$. The goal is to predict Y given X, where $(X,Y) \sim P$ is a new data point independent of $(X_i,Y_i)_{1 \leq i \leq n}$. Denoting by s the regression function, that is $s(x) = \mathbb{E}[Y \mid X = x]$ for every $x \in \mathcal{X}$, we can write

$$Y_i = s(X_i) + \sigma(X_i)\varepsilon_i \tag{1}$$

where $\sigma : \mathcal{X} \mapsto \mathbb{R}$ is the heteroscedastic noise level and ε_i are i.i.d. centered noise terms, possibly dependent on X_i, but with mean 0 and variance 1 conditionally to X_i.

The quality of a predictor $t : \mathcal{X} \mapsto \mathcal{Y}$ is measured by the (quadratic) prediction loss

$$\mathbb{E}_{(X,Y)\sim P}[\gamma(t,(X,Y))] =: P\gamma(t) \qquad \text{where} \quad \gamma(t,(x,y)) = (t(x) - y)^2$$

is the least-squares contrast. The minimizer of $P\gamma(t)$ over the set of all predictors, called Bayes predictor, is the regression function s. Therefore, the excess loss is defined as

$$\ell(s,t) := P\gamma(t) - P\gamma(s) = \mathbb{E}_{(X,Y)\sim P}(t(X) - s(X))^2 \quad .$$

Given a particular set of predictors S_m (called a *model*), we define the best predictor over S_m as

$$s_m := \arg\min_{t \in S_m} \{ P\gamma(t) \} \quad ,$$

with its empirical counterpart

$$\widehat{s}_m := \arg\min_{t \in S_m} \{ P_n\gamma(t) \}$$

(when it exists and is unique), where $P_n = n^{-1} \sum_{i=1}^n \delta_{(X_i,Y_i)}$. This estimator is the well-known *empirical risk minimizer*, also called least-squares estimator since γ is the least-squares contrast.

2.2 Ideal Model Selection

Let us assume that we are given a family of models $(S_m)_{m \in \mathcal{M}_n}$, hence a family of estimators $(\widehat{s}_m)_{m \in \mathcal{M}_n}$ obtained by empirical risk minimization. The model selection problem consists in looking for some data-dependent $\widehat{m} \in \mathcal{M}_n$ such that $\ell(s, \widehat{s}_{\widehat{m}})$ is as small as possible. For instance, it would be convenient to prove some oracle inequality of the form

$$\ell(s, \widehat{s}_{\widehat{m}}) \leq C \inf_{m \in \mathcal{M}_n} \{ \ell(s, \widehat{s}_m) \} + R_n$$

in expectation or on an event of large probability, with leading constant C close to 1 and $R_n = o(n^{-1})$.

General penalization procedures can be described as follows. Let $\mathrm{pen} : \mathcal{M}_n \mapsto \mathbb{R}^+$ be some penalty function, possibly data-dependent, and define

$$\widehat{m} \in \arg\min_{m \in \mathcal{M}_n} \{ \mathrm{crit}(m) \} \qquad \text{with} \qquad \mathrm{crit}(m) := P_n\gamma(\widehat{s}_m) + \mathrm{pen}(m) \quad . \tag{2}$$

Since the ideal criterion $\operatorname{crit}(m)$ is the true prediction error $P\gamma\left(\widehat{s}_m\right)$, the *ideal penalty* is

$$\operatorname{pen}_{\mathrm{id}}(m) := P\gamma(\widehat{s}_m) - P_n\gamma(\widehat{s}_m) \enspace .$$

This quantity is unknown because it depends on the true distribution P. A natural idea is to choose $\operatorname{pen}(m)$ as close as possible to $\operatorname{pen}_{\mathrm{id}}(m)$ for every $m \in \mathcal{M}_n$. We will show below, in a general setting, that when pen is a good estimator of the ideal penalty $\operatorname{pen}_{\mathrm{id}}$, then $\widehat{m}$ satisfies an oracle inequality with leading constant C close to 1.

By definition of $\widehat{m}$,

$$\forall m \in \mathcal{M}_n, \quad P_n\gamma(\widehat{s}_{\widehat{m}}) \leq P_n\gamma(\widehat{s}_m) + \operatorname{pen}(m) - \operatorname{pen}(\widehat{m}) \enspace .$$

For every $m \in \mathcal{M}_n$, we define

$$p_1(m) = P\left(\gamma(\widehat{s}_m) - \gamma(s_m)\right) \quad p_2(m) = P_n\left(\gamma(s_m) - \gamma(\widehat{s}_m)\right) \quad \delta(m) = (P_n - P)\left(\gamma(s_m)\right)$$

so that

$$\begin{aligned} & \operatorname{pen}_{\mathrm{id}}(m) = p_1(m) + p_2(m) - \delta(m) \\ \text{and} \qquad & \ell\left(s, \widehat{s}_m\right) = P_n\gamma(\widehat{s}_m) + p_1(m) + p_2(m) - \delta(m) - P\gamma(s) \enspace . \end{aligned}$$

Hence, for every $m \in \mathcal{M}_n$,

$$\ell\left(s, \widehat{s}_{\widehat{m}}\right) + (\operatorname{pen} - \operatorname{pen}_{\mathrm{id}})(\widehat{m}) \leq \ell\left(s, \widehat{s}_m\right) + (\operatorname{pen} - \operatorname{pen}_{\mathrm{id}})(m) \enspace . \tag{3}$$

Therefore, in order to derive an oracle inequality from (3), it is sufficient to show that for every $m \in \mathcal{M}_n$, $\operatorname{pen}(m)$ is close to $\operatorname{pen}_{\mathrm{id}}(m)$.

2.3 The Slope Heuristics

If the penalty is too big, the left-hand side of (3) is larger than $\ell\left(s, \widehat{s}_{\widehat{m}}\right)$ so that (3) implies an oracle inequality, possibly with large leading constant C. On the contrary, if the penalty is too small, the left-hand side of (3) may become negligible with respect to $\ell\left(s, \widehat{s}_{\widehat{m}}\right)$ (which would make C explode) or—worse—may be nonpositive. In the latter case, no oracle inequality may be derived from (3). We shall see in the following that $\ell\left(s, \widehat{s}_{\widehat{m}}\right)$ blows up if and only if the penalty is smaller than some “minimal penalty”.

Let us consider first the case $\operatorname{pen}(m) = p_2(m)$ in (2). Then, $\mathbb{E}\left[\operatorname{crit}(m)\right] = \mathbb{E}\left[P_n\gamma\left(s_m\right)\right] = P\gamma\left(s_m\right)$, so that $\widehat{m}$ approximately minimizes its bias. Therefore, $\widehat{m}$ is one of the more complex models, and the risk of $\widehat{s}_{\widehat{m}}$ is large. Let us assume now that $\operatorname{pen}(m) = Kp_2(m)$. If $0 < K < 1$, $\operatorname{crit}(m)$ is a decreasing function of the complexity of m, so that $\widehat{m}$ is again one of the more complex models. On the contrary, if $K > 1$, $\operatorname{crit}(m)$ increases with the complexity of m (at least for the largest models), so that $\widehat{m}$ has a small or medium complexity. This argument supports the conjecture that the “minimal amount of penalty” required for the model selection procedure to work is $p_2(m)$.

In many frameworks such as the one of Section 4.1, it turns out that

$$\forall m \in \mathcal{M}_n, \qquad p_1(m) \approx p_2(m) \enspace .$$

Hence, the ideal penalty $\operatorname{pen}_{\mathrm{id}}(m) \approx p_1(m) + p_2(m)$ is close to $2p_2(m)$. Since $p_2(m)$ is a “minimal penalty”, the optimal penalty is close to twice the minimal penalty:

$$\operatorname{pen}_{\mathrm{id}}(m) \approx 2\operatorname{pen}_{\min}(m) \enspace .$$

This is the so-called "slope heuristics", first introduced by Birgé and Massart (2007) in a Gaussian homoscedastic setting. Note that a formal proof of the validity of the slope heuristics has only been given for Gaussian homoscedastic least-squares regression with a fixed design (Birgé and Massart, 2007); up to the best of our knowledge, the present paper yields the second theoretical result on the slope heuristics.

This heuristics has some applications because the minimal penalty can be estimated from the data. Indeed, when the penalty smaller than $\mathrm{pen}_{\min}$, the selected model $\widehat{m}$ is among the more complex. On the contrary, when the penalty is larger than $\mathrm{pen}_{\min}$, the complexity of $\widehat{m}$ is much smaller. This leads to the algorithm described in the next section.

3. A Data-driven Calibration Algorithm

Now, a data-driven calibration algorithm for penalization procedures can be defined, generalizing a method proposed by Birgé and Massart (2007) and implemented by Lebarbier (2005).

3.1 The General Algorithm

Assume that the shape $\mathrm{pen}_{\mathrm{shape}} : \mathcal{M}_n \mapsto \mathbb{R}^+$ of the ideal penalty is known, from some prior knowledge or because it had first been estimated, see Section 3.4. Then, the penalty $K^\star \mathrm{pen}_{\mathrm{shape}}$ provides an approximately optimal procedure, for some unknown constant $K^\star > 0$. The goal is to find some $\widehat{K}$ such that $\widehat{K}\,\mathrm{pen}_{\mathrm{shape}}$ is approximately optimal.

Let D_m be some known complexity measure of the model $m \in \mathcal{M}_n$. Typically, when the models are finite-dimensional vector spaces, D_m is the dimension of S_m. According to the "slope heuristics" detailed in Section 2.3, the following algorithm provides an optimal calibration of the penalty $\mathrm{pen}_{\mathrm{shape}}$.

Algorithm 1 (Data-driven penalization with slope heuristics)

1. *Compute the selected model $\widehat{m}(K)$ as a function of $K > 0$*

$$\widehat{m}(K) \in \arg\min_{m \in \mathcal{M}_n} \left\{ P_n\gamma(\widehat{s}_m) + K\,\mathrm{pen}_{\mathrm{shape}}(m) \right\} \quad .$$

2. *Find $\widehat{K}_{\min} > 0$ such that $D_{\widehat{m}(K)}$ is "huge" for $K < \widehat{K}_{\min}$ and "reasonably small" for $K > \widehat{K}_{\min}$.*

3. *Select the model $\widehat{m} := \widehat{m}\left(2\widehat{K}_{\min}\right)$.*

A computationally efficient way to perform the first step of Algorithm 1 is provided in Section 3.2. The accurate definition of $\widehat{K}_{\min}$ is discussed in Section 3.3, including explicit values for "huge" and "reasonably small"). Then, once $P_n\gamma(\widehat{s}_m)$ and $\mathrm{pen}_{\mathrm{shape}}(m)$ are known for every $m \in \mathcal{M}_n$, the complexity of Algorithm 1 is $O(\mathrm{Card}(\mathcal{M}_n)^2)$ (see Algorithm 2 and Proposition 1). This can be a decisive advantage compared to cross-validation methods, as discussed in Section 4.6.

3.2 Computation of $(\widehat{m}(K))_{K \geq 0}$

Step 1 of Algorithm 1 requires to compute $\widehat{m}(K)$ for every $K \in (0, +\infty)$. A computationally efficient way to perform this step is described in this subsection.

We start with some notations:

$$\forall m \in \mathcal{M}_n, \qquad f(m) = P_n\gamma(\widehat{s}_m) \qquad\qquad g(m) = \mathrm{pen}_{\mathrm{shape}}(m)$$
$$\text{and} \qquad \forall K \geq 0, \qquad \widehat{m}(K) \in \arg\min_{m\in\mathcal{M}_n} \{ f(m) + Kg(m) \} \enspace .$$

Since the latter definition can be ambiguous, let us choose any total ordering $\preceq$ on $\mathcal{M}_n$ such that g is non-decreasing, which is always possible if $\mathcal{M}_n$ is at most countable. Then, $\widehat{m}(K)$ is defined as the smallest element of

$$E(K) := \arg\min_{m\in\mathcal{M}_n} \{ f(m) + Kg(m) \}$$

for $\preceq$. The main reason why the whole trajectory $(\widehat{m}(K))_{K\geq 0}$ can be computed efficiently is its particular shape.

Indeed, the proof of Proposition 1 shows that $K \mapsto \widehat{m}(K)$ is piecewise constant, and non-increasing for $\preceq$. Then, the whole trajectory $(\widehat{m}(K))_{K\geq 0}$ can be summarized by

- the number of jumps $i_{\max} \in \{0,\dots,\mathrm{Card}(\mathcal{M}_n)-1\}$,
- the location of the jumps: an increasing sequence of nonnegative reals $(K_i)_{0\leq i\leq i_{\max}+1}$, with $K_0 = 0$ and $K_{i_{\max}+1} = +\infty$,
- a non-increasing sequence of models $(m_i)_{0\leq i\leq i_{\max}}$,

$$\text{with} \qquad \forall i \in \{0,\dots,i_{\max}\}, \quad \forall K \in [K_i, K_{i+1}), \qquad \widehat{m}(K) = m_i \enspace .$$

Algorithm 2 (Step 1 of Algorithm 1) *For every $m \in \mathcal{M}_n$, define $f(m) = P_n\gamma(\widehat{s}_m)$ and $g(m) = \mathrm{pen}_{\mathrm{shape}}(m)$. Choose $\preceq$ any total ordering on $\mathcal{M}_n$ such that g is non-decreasing.*

- *Init: $K_0 := 0$, $m_0 := \arg\min_{m\in\mathcal{M}_n}\{f(m)\}$ (when this minimum is attained several times, m_0 is defined as the smallest one with respect to $\preceq$).*
- *Step i, $i \geq 1$: Let*

$$G(m_{i-1}) := \{ m \in \mathcal{M}_n \text{ s.t. } f(m) > f(m_{i-1}) \quad \text{and} \quad g(m) < g(m_{i-1}) \} \enspace .$$

If $G(m_{i-1}) = \emptyset$, then put $K_i = +\infty$, $i_{\max} = i-1$ and stop. Otherwise,

$$K_i := \inf\left\{ \frac{f(m) - f(m_{i-1})}{g(m_{i-1}) - g(m)} \text{ s.t. } m \in G(m_{i-1}) \right\} \tag{4}$$
$$\text{and} \qquad m_i := \min_{\preceq} F_i \qquad \text{with} \quad F_i := \arg\min_{m\in G(m_{i-1})} \left\{ \frac{f(m) - f(m_{i-1})}{g(m_{i-1}) - g(m)} \right\} \enspace .$$

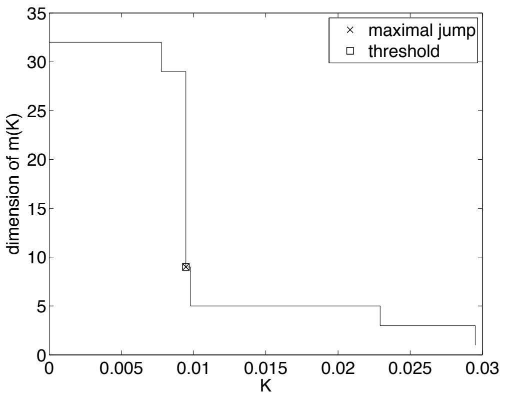
Proposition 1 (Correctness of Algorithm 2) *If $\mathcal{M}_n$ is finite, Algorithm 2 terminates and $i_{\max} \leq \mathrm{Card}(\mathcal{M}_n) - 1$. With the notations of Algorithm 2, let $\widehat{m}(K)$ be the smallest element of*

$$E(K) := \arg\min_{m\in\mathcal{M}_n} \{ f(m) + Kg(m) \} \qquad \text{with respect to } \preceq .$$

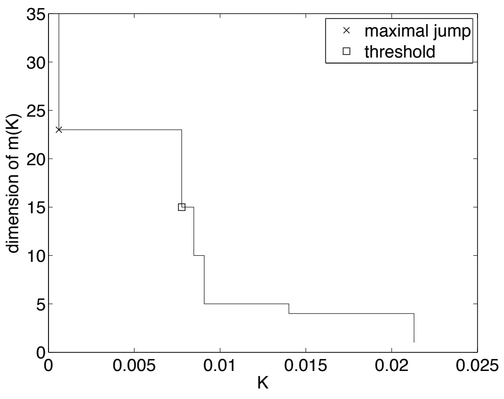
Then, $(K_i)_{0\leq i\leq i_{\max}+1}$ is increasing and $\forall i \in \{0,\dots,i_{\max}-1\}$, $\forall K \in [K_i, K_{i+1})$, $\widehat{m}(K) = m_i$.

It is proved in Section A.2. In the change-point detection framework, a similar result has been proved by Lavielle (2005).

Proposition 1 also gives an upper bound on the computational complexity of Algorithm 2; since the complexity of each step is $O(\mathrm{Card}\,\mathcal{M}_n)$, Algorithm 2 requires less than $O(i_{\max}\,\mathrm{Card}\,\mathcal{M}_n) \leq O((\mathrm{Card}\,\mathcal{M}_n)^2)$ operations. In general, this upper bound is pessimistic since $i_{\max} \ll \mathrm{Card}\,\mathcal{M}_n$.



(a) One clear jump.



(b) Two jumps, two values for $\widehat{K}_{\min}$.

Figure 1: $D_{\widehat{m}(K)}$ as a function of K for two different samples. Data are simulated according to (1) with $n = 200$, $X_i \sim \mathcal{U}([0,1])$, $\varepsilon_i \sim \mathcal{N}(0,1)$, $s(x) = \sin(\pi x)$ and $\sigma \equiv 1$. The models $(S_m)_{m \in \mathcal{M}_n}$ are the sets of piecewise constant functions on regular partitions of $[0,1]$, with dimensions between 1 and $n/(\ln(n))$. The penalty shape is $\mathrm{pen}_{\mathrm{shape}}(m) = D_m$ and the dimension threshold is $D_{\mathrm{thresh}} = 19 \approx n/(2\ln(n))$. See experiment S1 by Arlot (2008c, Section 6.1) for details.

3.3 Definition of $\widehat{K}_{\min}$

Step 2 of Algorithm 1 estimates $\widehat{K}_{\min}$ such that $\widehat{K}_{\min}\,\mathrm{pen}_{\mathrm{shape}}$ is the minimal penalty. The purpose of this subsection is to define properly $\widehat{K}_{\min}$ as a function of $(\widehat{m}(K))_{K>0}$.

According to the slope heuristics described in Section 2.3, $\widehat{K}_{\min}$ corresponds to a "complexity jump". If $K < \widehat{K}_{\min}$, $\widehat{m}(K)$ has a large complexity, whereas if $K > \widehat{K}_{\min}$, $\widehat{m}(K)$ has a small or medium complexity. Therefore, the two following definitions of $\widehat{K}_{\min}$ are natural.

Let D_{thresh} be the largest "reasonably small" complexity, meaning the models with larger complexities should not be selected. When D_m is the dimension of S_m as a vector space, $D_{\mathrm{thresh}} \propto n/(\ln(n))$ or $n/(\ln(n))^2$ are natural choices since the dimension of the oracle is likely to be of order n^{α} for some $\alpha \in (0,1)$. Then, define

$$\widehat{K}_{\min} := \inf\left\{ K > 0 \text{ s.t. } D_{\widehat{m}(K)} \leq D_{\mathrm{thresh}} \right\} \quad . \tag{5}$$

With this definition, Algorithm 2 can be stopped as soon as the threshold is reached.

Another idea is that $\widehat{K}_{\min}$ should match with the largest complexity jump:

$$\widehat{K}_{\min} := K_{i_{\mathrm{jump}}} \quad \text{with} \quad i_{\mathrm{jump}} = \arg \max_{i \in \{0,\dots,i_{\max}-1\}} \left\{ D_{m_{i+1}} - D_{m_i} \right\} \quad . \tag{6}$$

In order to ensure that there is a clear jump in the sequence $(D_{m_i})_{i \geq 0}$, it may be useful to add a few models of large complexity.

As an illustration, we compared the two definitions above (5 and 6) on 1 000 simulated samples. The exact simulation framework is described below Figure 1. Three cases occured:

1. There is one clear jump. Both definitions give the same value for $\widehat{K}_{\min}$. This occured for about 85% of the samples; an example is given on Figure 1a.

2. There are several jumps corresponding to close values of K. Definitions (5) and (6) give slightly different values for $\widehat{K}_{\min}$, but the selected models $\widehat{m}\left(2\widehat{K}_{\min}\right)$ are equal. This occured for about 8.5% of the samples.

3. There are several jumps corresponding to distant values of K. Definitions (5) and (6) strongly disagree, giving different selected models $\widehat{m}\left(2\widehat{K}_{\min}\right)$ at final. This occured for about 6.5% of the samples; an example is given on Figure 1b.

The only problematic case is the third one, in which an arbitrary choice has to be made between definitions (5) and (6).

With the same simulated data, we have compared the prediction errors of the two methods by estimating the constant C_{or} that would appear in some oracle inequality,

$$C_{\text{or}} := \frac{\mathbb{E}\left[\ell\left(s, \widehat{s}_{\widehat{m}}\right)\right]}{\mathbb{E}\left[\inf_{m \in \mathcal{M}_n}\left\{\ell\left(s, \widehat{s}_m\right)\right\}\right]} \quad .$$

With definition (5) $C_{\text{or}} \approx 1.88$; with definition (6) $C_{\text{or}} \approx 2.01$. For both methods, the standard error of the estimation is 0.04. As a comparison, Mallows' C_p with a classical estimator of the variance σ^2 has an estimated performance $C_{\text{or}} \approx 1.93$ on the same data.

The overall conclusion of this simulation experiment is that Algorithm 1 can be competitive with Mallows' C_p in a framework where Mallows' C_p is known to be optimal. Definition (5) for $\widehat{K}_{\min}$ seems slightly more efficient than (6), but without convincing evidence. Indeed, both definitions depend on some arbitrary choices: the value of the threshold D_{thresh} in (5), the maximal complexity among the collection of models $(S_m)_{m \in \mathcal{M}_n}$ in (6). When n is small, say $n = 200$, choosing D_{thresh} is tricky since $n/(2\ln(n))$ and $\sqrt{n}$ are quite close. Then, the difference between (5) and (6) is likely to come mainly from the particular choice $D_{\text{thresh}} = 19$ than from basic differences between the two definitions.

In order to estimate $\widehat{K}_{\min}$ as automatically as possible, we suggest to combine the two definitions; when the selected models $\widehat{m}(2\widehat{K}_{\min})$ differ, send a warning to the final user advising him to look at the curve $K \mapsto D_{\widehat{m}(K)}$ himself; otherwise, remain confident in the automatic choice of $\widehat{m}(2\widehat{K}_{\min})$.

3.4 Penalty Shape

For using Algorithm 1 in practice, it is necessary to know *a priori*, or at least to estimate, the optimal shape $\text{pen}_{\text{shape}}$ of the penalty. Let us explain how this can be achieved in different frameworks.

The first example that comes to mind is $\text{pen}_{\text{shape}}(m) = D_m$. It is valid for homoscedastic least-squares regression on linear models, as shown by several papers mentioned in Section 1. Indeed, when $\text{Card}(\mathcal{M}_n)$ is smaller than some power of n, Mallows' C_p penalty—defined by $\text{pen}(m) = 2\mathbb{E}\left[\sigma^2(X)\right] n^{-1} D_m$—is well known to be asymptotically optimal. For larger collections $\mathcal{M}_n$, more elaborate results (Birgé and Massart, 2001, 2007) have shown that a penalty proportional to $\ln(n)\mathbb{E}\left[\sigma^2(X)\right] n^{-1} D_m$ and depending on the size of $\mathcal{M}_n$ is asymptotically optimal.

Algorithm 1 then provides an alternative to plugging an estimator of $\mathbb{E}\left[\sigma^2(X)\right]$ into the above penalties. Let us detail two main advantages of our approach. First, we avoid the difficult task of estimating $\mathbb{E}\left[\sigma^2(X)\right]$ without knowing in advance some model to which the true regression function belongs. Algorithm 1 provides a model-free estimation of the factor multiplying the penalty. Second, the estimator $\widehat{\sigma^2}$ of $\mathbb{E}\left[\sigma^2(X)\right]$ with the smallest quadratic risk is certainly far from being the optimal one for model selection. For instance, underestimating the multiplicative factor is well-known to lead to poor performances, whereas overestimating the multiplicative factor does not increase much the prediction error in general. Then, a good estimator of $\mathbb{E}\left[\sigma^2(X)\right]$ for model selection should overestimate it with a probability larger than $1/2$. Algorithm 1 satisfies this property automatically because $\widehat{K}_{\min}$ so that the selected model cannot be too large.

In short, *Algorithm 1 with* $\mathrm{pen}_{\mathrm{shape}}(m) = D_m$ *is quite different from a simple plug-in version of Mallows' C_p*. It leads to a really *data-dependent penalty, which may perform better in practice than the best deterministic penalty $K^\star D_m$*.

In a more general framework, Algorithm 1 allows to choose a different shape of penalty $\mathrm{pen}_{\mathrm{shape}}$. For instance, in the heteroscedastic least-squares regression framework of Section 2.1, the optimal penalty is no longer proportional to the dimension D_m of the models. This can be shown from computations made by (Arlot, 2008c, Proposition 1) when S_m is assumed to be the vector space of piecewise constant functions on a partition $(I_\lambda)_{\lambda \in \Lambda_m}$ of $\mathcal{X}$:

$$\mathbb{E}\left[\mathrm{pen}_{\mathrm{id}}(m)\right] = \mathbb{E}\left[(P - P_n)\gamma(\widehat{s}_m)\right] \approx \frac{2}{n} \sum_{\lambda \in \Lambda_m} \mathbb{E}\left[\sigma(X)^2 \,\middle|\, X \in I_\lambda\right] \quad . \tag{7}$$

An exact result has been proved by Arlot (2008c, Proposition 1). Moreover, Arlot (2008a) gave an example of model selection problem in which no penalty proportional to D_m can be asymptotically optimal.

A first way to estimate the shape of the penalty is simply to use (7) to compute $\mathrm{pen}_{\mathrm{shape}}$, when both the distribution of X and the shape of the noise level σ are known. In practice, one has seldom such a prior knowledge.

We suggest in this situation to use *resampling penalties* (Efron, 1983; Arlot, 2008c), or *V-fold penalties* (Arlot, 2008b) which have much smaller computational costs. Up to a multiplicative factor (automatically estimated by Algorithm 1), these penalties should estimate correctly $\mathbb{E}\left[\mathrm{pen}_{\mathrm{id}}(m)\right]$ in any framework. In particular, resampling and V-fold penalties are asymptotically optimal in the heteroscedastic least-squares regression framework (Arlot, 2008b,c).

3.5 The General Prediction Framework

Section 2 and definition of Algorithm 1 have restricted ourselves to the least-squares regression framework. Actually, this is not necessary at all to make Algorithm 1 well-defined, so that it can naturally be extended to the general prediction framework. More precisely, the (X_i, Y_i) can be assumed to belong to $\mathcal{X} \times \mathcal{Y}$ for some general $\mathcal{Y}$, and $\gamma : S \times (\mathcal{X} \times \mathcal{Y}) \mapsto [0; +\infty)$ any contrast function. In particular, $\mathcal{Y} = \{0, 1\}$ leads to the binary classification problem, for which a natural contrast function is the 0–1 loss $\gamma(t; (x, y)) = \mathbb{1}_{t(x) \neq y}$. In this case, the shape of the penalty $\mathrm{pen}_{\mathrm{shape}}$ can for instance be estimated with the global or local Rademacher complexities mentioned in Section 1.

However, a natural question is whether the slope heuristics of Section 2.3, upon which Algorithm 1 relies, can be extended to the general framework. Several concentration results used to prove the validity of the slope heuristics in the least-squares regression framework in this article are valid in a general setting including binary classification. Even if the factor 2 coming from the closeness of $\mathbb{E}[p_1]$ and $\mathbb{E}[p_2]$ (see Section 2.3) may not be universally valid, we conjecture that Algorithm 1 can be used in other settings than least-squares regression. Moreover, as already mentioned at the end of Section 1, empirical studies have shown that Algorithm 1 can be successfully applied to several problems, with different shapes for the penalty. To our knowledge, to give a formal proof of this fact remains an interesting open problem.

4. Theoretical Results

Algorithm 1 mainly relies on the "slope heuristics", developed in Section 2.2. The goal of this section is to provide a theoretical justification of this heuristics.

It is split into two main results. First, Theorem 2 provides lower bounds on $D_{\widehat{m}}$ and the risk of $\widehat{s}_{\widehat{m}}$ when the penalty is smaller than $\mathrm{pen}_{\min}(m) := \mathbb{E}[p_2(m)]$. Second, Theorem 3 is an oracle inequality with leading constant almost one when $\mathrm{pen}(m) \approx 2\mathbb{E}[p_2(m)]$, relying on (3) and the comparison $p_1 \approx p_2$.

In order to prove both theorems, two probabilistic results are necessary. First, p_1, p_2 and δ concentrate around their expectations; for p_2 and δ, it is proved in a general framework in Appendix A.6. Second, $\mathbb{E}[p_1(m)] \approx \mathbb{E}[p_2(m)]$ for every $m \in \mathcal{M}_n$. The latter point is quite hard to prove in general, so that we must make an assumption on the models. Therefore, in this section, we restrict ourselves to the regressogram case, assuming that for every $m \in \mathcal{M}_n$, S_m is the set of piecewise constant functions on some fixed partition $(I_\lambda)_{\lambda\in\Lambda_m}$ of $\mathcal{X}$. This framework is described precisely in the next subsection. Although we do not consider regressograms as a final goal, the theoretical results proved for regressograms help to understand better how to use Algorithm 1 in practice.

4.1 Regressograms

Let S_m be the the set of piecewise constant functions on some partition $(I_\lambda)_{\lambda\in\Lambda_m}$ of $\mathcal{X}$. The empirical risk minimizer $\widehat{s}_m$ on S_m is called a *regressogram*. S_m is a vector space of dimension $D_m = \mathrm{Card}(\Lambda_m)$, spanned by the family $(\mathbb{1}_{I_\lambda})_{\lambda\in\Lambda_m}$. Since this basis is orthogonal in $L^2(\mu)$ for any probability measure μ on $\mathcal{X}$, computations are quite easy. In particular, we have:

$$s_m = \sum_{\lambda\in\Lambda_m} \beta_\lambda \mathbb{1}_{I_\lambda} \quad \text{and} \quad \widehat{s}_m = \sum_{\lambda\in\Lambda_m} \widehat{\beta}_\lambda \mathbb{1}_{I_\lambda} \enspace ,$$

where

$$\beta_\lambda := \mathbb{E}_P[Y \mid X \in I_\lambda] \qquad \widehat{\beta}_\lambda := \frac{1}{n\widehat{p}_\lambda} \sum_{X_i\in I_\lambda} Y_i \qquad \widehat{p}_\lambda := P_n(X \in I_\lambda) \enspace .$$

Note that $\widehat{s}_m$ is uniquely defined if and only if each I_λ contains at least one of the X_i. Otherwise, $\widehat{s}_m$ is not uniquely defined and we consider that the model m cannot be chosen.

4.2 Main Assumptions

In this section, we make the following assumptions. First, each model S_m is a set of piecewise constants functions on some fixed partition $(I_\lambda)_{\lambda\in\Lambda_m}$ of $\mathcal{X}$. Second, the family $(S_m)_{m\in\mathcal{M}_n}$ satisfies:

(**P1**) Polynomial complexity of $\mathcal{M}_n$: $\mathrm{Card}(\mathcal{M}_n) \leq c_{\mathcal{M}} n^{\alpha_{\mathcal{M}}}$.

(**P2**) Richness of $\mathcal{M}_n$: $\exists m_0 \in \mathcal{M}_n$ s.t. $D_{m_0} \in [\sqrt{n}, c_{\mathrm{rich}}\sqrt{n}]$.

Assumption (**P1**) is quite classical for proving the asymptotic optimality of a model selection procedure; it is for instance implicitly assumed by Li (1987) in the homoscedastic fixed-design case. Assumption (**P2**) is merely technical and can be changed if necessary; it only ensures that $(S_m)_{m\in\mathcal{M}_n}$ does not contain only models which are either too small or too large.

For any penalty function pen : $\mathcal{M}_n \mapsto \mathbb{R}^+$, we define the following model selection procedure:

$$\widehat{m} \in \arg \min_{m \in \mathcal{M}_n,\, \min_{\lambda \in \Lambda_m} \{ \widehat{p}_\lambda \} > 0} \{ P_n \gamma(\widehat{s}_m) + \operatorname{pen}(m) \} \quad . \tag{8}$$

Moreover, the data $(X_i, Y_i)_{1 \le i \le n}$ are assumed to be i.i.d. and to satisfy:

($\mathbf{Ab}$) The data are bounded: $\|Y_i\|_\infty \le A < \infty$.

($\mathbf{An}$) Uniform lower-bound on the noise level: $\sigma(X_i) \ge \sigma_{\min} > 0$ a.s.

($\mathbf{Ap_u}$) The bias decreases as a power of D_m: there exist some $\beta_+, C_+ > 0$ such that

$$\ell(s, s_m) \le C_+ D_m^{-\beta_+} \quad .$$

($\mathbf{Ar}_\ell^{\mathbf{X}}$) Lower regularity of the partitions for $\mathcal{L}(X)$: $D_m \min_{\lambda \in \Lambda_m} \{ \mathbb{P}(X \in I_\lambda) \} \ge c_{\mathrm{r},\ell}^X$.

Further comments are made in Sections 4.3 and 4.4 about these assumptions, in particular about their possible weakening.

4.3 Minimal Penalties

Our first result concerns the existence of a minimal penalty. In this subsection, ($\mathbf{P2}$) is replaced by the following strongest assumption:

($\mathbf{P2}+$) $\exists c_0, c_{\mathrm{rich}} > 0$ s.t. $\forall l \in [\sqrt{n}, c_0 n / (c_{\mathrm{rich}} \ln(n))]$, $\exists m \in \mathcal{M}_n$ s.t. $D_m \in [l, c_{\mathrm{rich}} l]$.

The reason why ($\mathbf{P2}$) is not sufficient to prove Theorem 2 below is that at least one model of dimension of order $n/\ln(n)$ should belong to the family $(S_m)_{m \in \mathcal{M}_n}$; otherwise, it may not be possible to prove that such models are selected by penalization procedures beyond the minimal penalty.

Theorem 2 *Suppose all the assumptions of Section 4.2 are satisfied. Let $K \in [0;1)$, $L > 0$, and assume that an event of probability at least $1 - Ln^{-2}$ exists on which*

$$\forall m \in \mathcal{M}_n, \quad 0 \le \operatorname{pen}(m) \le K \mathbb{E}[P_n(\gamma(s_m) - \gamma(\widehat{s}_m))] \quad . \tag{9}$$

Then, there exist two positive constants K_1, K_2 such that, with probability at least $1 - K_1 n^{-2}$,

$$D_{\widehat{m}} \ge K_2 n \ln(n)^{-1} \quad ,$$

where $\widehat{m}$ is defined by (8). *On the same event,*

$$\ell(s, \widehat{s}_{\widehat{m}}) \ge \ln(n) \inf_{m \in \mathcal{M}_n} \{ \ell(s, \widehat{s}_m) \} \quad . \tag{10}$$

The constants K_1 and K_2 may depend on K, L and constants in ($\mathbf{P1}$), ($\mathbf{P2}+$), ($\mathbf{Ab}$), ($\mathbf{An}$), ($\mathbf{Ap_u}$) *and* ($\mathbf{Ar}_\ell^{\mathbf{X}}$)*, but do not depend on n.*

This theorem thus validates the first part of the heuristics of Section 2.3, proving that a minimal amount of penalization is required; when the penalty is smaller, the selected dimension $D_{\widehat{m}}$ and the quadratic risk of the final estimator $\ell(s, \widehat{s}_{\widehat{m}})$ blow up. This coupling is quite interesting, since the dimension $D_{\widehat{m}}$ is known in practice, contrary to $\ell(s, \widehat{s}_{\widehat{m}})$. It is then possible to detect from the data whether the penalty is too small, as proposed in Algorithm 1.

The main interest of this result is its combination with Theorem 3 below. Nevertheless Theorem 2 is also interesting by itself for understanding the theoretical properties of penalization procedures. Indeed, it generalizes the results of Birgé and Massart (2007) on the existence of minimal penalties to heteroscedastic regression with a random design, even if we have to restrict to regressograms. Moreover, we have a general formulation for the minimal penalty

$$\mathrm{pen}_{\min}(m) := \mathbb{E}\left[P_n \left(\gamma(s_m) - \gamma(\widehat{s}_m) \right) \right] = \mathbb{E}\left[p_2(m) \right] \quad ,$$

which can be used in frameworks situations where it is not proportional to the dimension D_m of the models (see Section 3.4 and references therein).

In addition, assumptions $(\mathbf{Ab})$ and $(\mathbf{An})$ on the data are much weaker than the Gaussian homoscedastic assumption. They are also much more realistic, and moreover can be strongly relaxed. Roughly speaking, boundedness of data can be replaced by conditions on moments of the noise, and the uniform lower bound $\sigma_{\min}$ is no longer necessary when σ satisfies some mild regularity assumptions. We refer to Arlot (2008c, Section 4.3) for detailed statements of these assumptions, and explanations on how to adapt proofs to these situations.

Finally, let us comment on conditions $(\mathbf{Ap_u})$ and $(\mathbf{Ar}_{\ell}^{\mathbf{X}})$. The upper bound $(\mathbf{Ap_u})$ on the bias occurs in the most reasonable situations, for instance when $\mathcal{X} \subset \mathbb{R}^k$ is bounded, the partition $(I_\lambda)_{\lambda \in \Lambda_m}$ is regular and the regression function s is α-Hölderian for some $\alpha > 0$ (β_+ depending on α and k). It ensures that medium and large models have a significantly smaller bias than smaller ones; otherwise, the selected dimension would be allowed to be too small with significant probability. On the other hand, $(\mathbf{Ar}_{\ell}^{\mathbf{X}})$ is satisfied at least for "almost regular" partitions $(I_\lambda)_{\lambda \in \Lambda_m}$, when X has a lower bounded density w.r.t. the Lebesgue measure on $\mathcal{X} \subset \mathbb{R}^k$.

Theorem 2 is stated with a general formulation of $(\mathbf{Ap_u})$ and $(\mathbf{Ar}_{\ell}^{\mathbf{X}})$, instead of assuming for instance that s is α-Hölderian and X has a lower bounded density w.r.t Leb, in order to point out the *generality* of the "minimal penalization" phenomenon. It occurs as soon as the models are not too much pathological. In particular, we do not make any assumption on the distribution of X itself, but only that the models are not too badly chosen according to this distribution. Such a condition can be checked in practice if some prior knowledge on $\mathcal{L}(X)$ is available; if part of the data are unlabeled—a usual case,—classical density estimation procedures can be applied for estimating $\mathcal{L}(X)$ from unlabeled data (Devroye and Lugosi, 2001).

4.4 Optimal Penalties

Algorithm 1 relies on a link between the minimal penalty pointed out by Theorem 2 and some optimal penalty. The following result is a formal proof of this link in the framework we consider: penalties close to twice the minimal penalty satisfy an oracle inequality with leading constant approximately equal to one.

Theorem 3 *Suppose all the assumptions of Section 4.2 are satisfied together with*

(**Ap**) *The bias decreases like a power of D_m: there exist $\beta_- \geq \beta_+ > 0$ and $C_+, C_- > 0$ such that*

$$C_- D_m^{-\beta_-} \leq \ell(s, s_m) \leq C_+ D_m^{-\beta_+} \ .$$

Let $\delta \in (0,1)$, $L > 0$, and assume that an event of probability at least $1 - Ln^{-2}$ exists on which for every $m \in \mathcal{M}_n$,

$$(2-\delta)\mathbb{E}[P_n(\gamma(s_m) - \gamma(\widehat{s}_m))] \leq \mathrm{pen}(m) \leq (2+\delta)\mathbb{E}[P_n(\gamma(s_m) - \gamma(\widehat{s}_m))] \ . \tag{11}$$

Then, for every $0 < \eta < \min\{\beta_+; 1\}/2$, there exist a constant K_3 and a sequence ε_n tending to zero at infinity such that, with probability at least $1 - K_3 n^{-2}$,

$$\begin{aligned} & D_{\widehat{m}} \leq n^{1-\eta} \\ \text{and} \qquad & \ell(s, \widehat{s}_{\widehat{m}}) \leq \left(\frac{1+\delta}{1-\delta} + \varepsilon_n \right) \inf_{m \in \mathcal{M}_n} \{ \ell(s, \widehat{s}_m) \} \ , \end{aligned} \tag{12}$$

where $\widehat{m}$ is defined by (8)*. Moreover, we have the oracle inequality*

$$\mathbb{E}[\ell(s, \widehat{s}_{\widehat{m}})] \leq \left(\frac{1+\delta}{1-\delta} + \varepsilon_n \right) \mathbb{E}\left[\inf_{m \in \mathcal{M}_n} \{ \ell(s, \widehat{s}_m) \} \right] + \frac{A^2 K_3}{n^2} \ .$$

The constant K_3 may depend on L, δ, η and the constants in (**P1**)*,* (**P2**)*,* (**Ab**)*,* (**An**)*,* (**Ap**) *and* ($\mathbf{Ar}_\ell^{\mathbf{X}}$)*, but not on n. The term ε_n is smaller than $\ln(n)^{-1/5}$; it can be made smaller than $n^{-\delta}$ for any $\delta \in (0; \delta_0(\beta_-, \beta_+))$ at the price of enlarging K_3.*

This theorem shows that twice the minimal penalty $\mathrm{pen}_{\min}$ pointed out by Theorem 2 satisfies an oracle inequality with leading constant almost one. In other words, the slope heuristics of Section 2.3 is valid. The consequences of the *combination* of Theorems 2 and 3 are detailed in Section 4.5.

The oracle inequality (12) remains valid when the penalty is only close to twice the minimal one. In particular, *the shape of the penalty can be estimated by resampling* as suggested in Section 3.4.

Actually, Theorem 3 above is a corollary of a more general result stated in Appendix A.3, Theorem 5. If

$$\mathrm{pen}(m) \approx K\mathbb{E}[P_n(\gamma(s_m) - \gamma(\widehat{s}_m))] \tag{13}$$

instead of (11), under the same assumptions, an oracle inequality with leading constant $C(K) + \varepsilon_n$ instead of $1 + \varepsilon_n$ holds with large probability. The constant $C(K)$ is equal to $(K-1)^{-1}$ when $K \in (1,2]$ and to $C(K) = K - 1$ when $K > 2$. Therefore, for every $K > 1$, the penalty defined by (13) is efficient up to a multiplicative constant. This result is new in the heteroscedastic framework.

Let us comment the additional assumption (**Ap**), that is the lower bound on the bias. Assuming $\ell(s, s_m) > 0$ for every $m \in \mathcal{M}_n$ is classical for proving the asymptotic optimality of Mallows' C_p (Shibata, 1981; Li, 1987; Birgé and Massart, 2007). (**Ap**) has been made by Stone (1985) and Burman (2002) in the density estimation framework, for the same technical reasons as ours. Assumption (**Ap**) is satisfied in several frameworks, such as the following: $(I_\lambda)_{\lambda \in \Lambda_m}$ is "regular", X has a lower-bounded density w.r.t. the Lebesgue measure on $\mathcal{X} \subset \mathbb{R}^k$, and s is non-constant and α-hölderian (w.r.t. $\|\cdot\|_\infty$), with

$$\beta_1 = k^{-1} + \alpha^{-1} - (k-1)k^{-1}\alpha^{-1} \qquad \text{and} \qquad \beta_2 = 2\alpha k^{-1} \ .$$

We refer to Arlot (2007, Section 8.10) for a complete proof.

When the lower bound in (**Ap**) is no longer assumed, (12) holds with two modifications in its right-hand side (for details, see Arlot, 2008c, Remark 9): the inf is restricted to models of dimension larger than $\ln(n)^{\gamma_1}$, and there is a remainder term $\ln(n)^{\gamma_2} n^{-1}$, where $\gamma_1, \gamma_2 > 0$ are numerical constants. This is equivalent to (12), unless there is a model of small dimension with a small bias. The lower bound in (**Ap**) ensures that it cannot happen. Note that if there is a small model close to s, it is hopeless to obtain an oracle inequality with a penalty which estimates $\mathrm{pen}_{\mathrm{id}}$, simply because deviations of $\mathrm{pen}_{\mathrm{id}}$ around its expectation would be much larger than the excess loss of the oracle. In such a situation, BIC-like methods are more appropriate; for instance, Csiszár (2002) and Csiszár and Shields (2000) showed that BIC penalties are minimal penalties for estimating the order of a Markov chain.

4.5 Main Theoretical and Practical Consequences

The slope heuristics and the correctness of Algorithm 1 follow from the combination of Theorems 2 and 3.

4.5.1 OPTIMAL AND MINIMAL PENALTIES

For the sake of simplicity, let us consider the penalty $K\mathbb{E}[p_2(m)]$ with any $K > 0$; any penalty close to this one satisfies similar properties. At first reading, one can think of the homoscedastic case where $\mathbb{E}[p_2(m)] \approx \sigma^2 D_m n^{-1}$; one of the novelties of our results is that the general picture is quite similar.

According to Theorem 3, the penalization procedure associated with $K\mathbb{E}[p_2(m)]$ satisfies an oracle inequality with leading constant $C_n(K)$ as soon as $K > 1$, and $C_n(2) \approx 1$. Moreover, results proved by Arlot (2008b) imply that $C_n(K) \geq C(K) > 1$ as soon as K is not close to 2. Therefore, $K = 2$ *is the optimal multiplying factor* in front of $\mathbb{E}[p_2(m)]$.

When $K < 1$, Theorem 2 shows that no oracle inequality can hold with leading constant $C_n(K) < \ln(n)$. Since $C_n(K) \leq (K-1)^{-1} < \ln(n)$ as soon as $K > 1 + \ln(n)^{-1}$, $K = 1$ *is the minimal multiplying factor* in front of $\mathbb{E}[p_2(m)]$. More generally, $\mathrm{pen}_{\min}(m) := \mathbb{E}[p_2(m)]$ is proved to be a *minimal penalty*.

In short, Theorems 2 and 3 prove the slope heuristics described in Section 2.3:

$$\text{“optimal” penalty} \approx 2 \times \text{“minimal” penalty} \enspace .$$

Birgé and Massart (2007) have proved the validity of the slope heuristics in the Gaussian homoscedastic framework. This paper extends their result to a non-Gaussian and heteroscedastic setting.

4.5.2 DIMENSION JUMP

In addition, Theorems 2 and 3 prove the existence of a crucial phenomenon: there exists a “dimension jump”—complexity jump in the general framework—around the minimal penalty. Let us consider again the penalty $K\mathbb{E}[p_2(m)]$. As in Algorithm 1, let us define

$$\widehat{m}(K) \in \arg\min_{m \in \mathcal{M}_n} \left\{ P_n \gamma(\widehat{s}_m) + K\mathbb{E}[p_2(m)] \right\} \enspace .$$

A careful look at the proofs of Theorems 2 and 3 shows that there exist constants $K_4, K_5 > 0$ and an event of probability $1 - K_4 n^{-2}$ on which

$$\forall 0 < K < 1 - \frac{1}{\ln(n)}, D_{\widehat{m}(K)} \geq \frac{K_5 n}{(\ln(n))^2} \text{ and } \forall K > 1 + \frac{1}{\ln(n)}, D_{\widehat{m}(K)} \leq n^{1-\eta} \ . \tag{14}$$

Therefore, the dimension $D_{\widehat{m}(K)}$ of the selected model jumps around the minimal value $K = 1$, from values of order $n(\ln(n))^{-2}$ to $n^{1-\eta}$.

Let us know explain why Algorithm 1 is correct, assuming that $\mathrm{pen}_{\mathrm{shape}}(m)$ is close to $\mathbb{E}[p_2(m)]$. With definition (5) of $\widehat{K}_{\min}$ and a threshold $D_{\mathrm{thresh}} \propto n(\ln(n))^{-3}$, (14) ensures that

$$1 - \frac{1}{\ln(n)} \leq \widehat{K}_{\min} \leq 1 + \frac{1}{\ln(n)}$$

with a large probability. Then, according to Theorem 3, the output of Algorithm 1 satisfies an oracle inequality with leading constant C_n tending to one as n tends to infinity.

4.6 Comparison with Data-splitting Methods

Tuning parameters are often chosen by cross-validation or by another data-splitting method, which suffer from some drawbacks compared to Algorithm 1.

First, V-fold cross-validation, leave-p-out and repeated learning-testing methods require a larger computation time. Indeed, they need to perform the empirical risk minimization process for each model several times, whereas Algorithm 1 only needs to perform it once.

Second, V-fold cross-validation is asymptotically suboptimal when V is fixed, as shown by (Arlot, 2008b, Theorem 1). The same suboptimality result is valid for the hold-out, when the size of the training set is not asymptotically equivalent to the sample size n. On the contrary, Theorems 2 and 3 prove that Algorithm 1 is asymptotically optimal in a framework including the one used by (Arlot, 2008b, Theorem 1) for proving the suboptimality of V-fold cross-validation. Hence, the quadratic risk of Algorithm 1 should be smaller, within a factor $\kappa > 1$.

Third, hold-out with a training set of size $n_t \sim n$, for instance $n_t = n - \sqrt{n}$ or $n_t = n(1 - \ln(n)^{-1})$, is known to be unstable. The final output $\widehat{m}$ strongly depends on the choice of a particular split of the data. According to the simulation study of Section 3.3, Algorithm 1 is far more stable.

To conclude, compared to data splitting methods, Algorithm 1 is either faster to compute, more efficient in terms of quadratic risk, or more stable. Then, Algorithm 1 should be preferred each time it can be used. Another approach is to use aggregation techniques, instead of selecting one model. As shown by several results (see for instance Tsybakov, 2004; Lecué, 2007), aggregating estimators built upon a training simple of size $n_t \sim n$ can have an optimal quadratic risk. Moreover, aggregation requires approximately the same computation time as Algorithm 1, and is much more stable than the hold-out. Hence, it can be an alternative to model selection with Algorithm 1.

5. Conclusion

This paper provides mathematical evidence that the method introduced by Birgé and Massart (2007) for designing data-driven penalties remains efficient in a non-Gaussian framework. The purpose of this conclusion is to relate the slope heuristics developed in Section 2 to the well known Mallows' C_p and Akaike's criteria and to the unbiased estimation of the risk principle.

Let us come come back to Gaussian model selection in order to explain how to guess what is the right penalty from the data themselves. Let γ_n be some empirical criterion (for instance the least-squares criterion as in this paper, or the log-likelihood criterion), $(S_m)_{m\in\mathcal{M}_n}$ be a collection of models and for every $m \in \mathcal{M}_n$ s_m be some minimizer of $t \mapsto \mathbb{E}[\gamma_n(t)]$ over S_m (assuming that such a point exists). Minimizing some penalized criterion

$$\gamma_n(\widehat{s}_m) + \mathrm{pen}(m)$$

over $\mathcal{M}_n$ amounts to minimize

$$\widehat{b}_m - \widehat{v}_m + \mathrm{pen}(m) \ ,$$
$$\text{where} \quad \forall m \in \mathcal{M}_n,\ \widehat{b}_m = \gamma_n(s_m) - \gamma_n(s) \text{ and } \widehat{v}_m = \gamma_n(s_m) - \gamma_n(\widehat{s}_m) \ .$$

The point is that $\widehat{b}_m$ is an unbiased estimator of the bias term $\ell(s, s_m)$. Having concentration arguments in mind, minimizing $\widehat{b}_m - \widehat{v}_m + \mathrm{pen}(m)$ can be conjectured approximately equivalent to minimize

$$\ell(s, s_m) - \mathbb{E}[\widehat{v}_m] + \mathrm{pen}(m) \ .$$

Since the purpose of model selection is to minimize the risk $\mathbb{E}[\ell(s, \widehat{s}_m)]$, an ideal penalty would be

$$\mathrm{pen}(m) = \mathbb{E}[\widehat{v}_m] + \mathbb{E}[\ell(s_m, \widehat{s}_m)] \ .$$

In Gaussian least-squares regression with a fixed design, the models S_m are linear and $\mathbb{E}[\widehat{v}_m] = \mathbb{E}[\ell(s_m, \widehat{s}_m)]$ is explicitly computable if the noise level is constant and known; this leads to Mallows' C_p penalty. When γ_n is the log-likelihood,

$$\mathbb{E}[\widehat{v}_m] \approx \mathbb{E}[\ell(s_m, \widehat{s}_m)] \approx \frac{D_m}{2n}$$

asymptotically, where D_m stands for the number of parameters defining model S_m; this leads to Akaike's Information Criterion (AIC). Therefore, both Mallows' C_p and Akaike's criterion are based on the unbiased (or asymptotically unbiased) risk estimation principle.

This paper goes further in this direction, using that $\mathbb{E}[\widehat{v}_m] \approx \mathbb{E}[\ell(s_m, \widehat{s}_m)]$ remains a valid approximation in a non-asymptotic framework. Then, a good penalty becomes $2\mathbb{E}[\widehat{v}_m]$ or $2\widehat{v}_m$, having in mind concentration arguments. Since $\widehat{v}_m$ is the minimal penalty, this explains the slope heuristics (Birgé and Massart, 2007) and connects it to Mallows' C_p and Akaike's heuristics.

The second main idea developed in this paper is that the minimal penalty can be estimated from the data; Algorithm 1 uses the jump of complexity which occurs around the minimal penalty, as shown in Sections 3.3 and 4.5.2. Another way to estimate the minimal penalty when it is (at least approximately) of the form αD_m is to estimate α by the *slope* of the graph of $\gamma_n(\widehat{s}_m)$ for large enough values of D_m; this method can be extended to other shapes of penalties, simply by replacing D_m by some (known!) function $f(D_m)$.

The slope heuristics can even be combined with resampling ideas, by taking a function f built from a randomized empirical criterion. As shown by Arlot (2008a), this approach is much more efficient than the rougher choice $f(D_m) = D_m$ for heteroscedastic regression frameworks. The question of the optimality of the slope heuristics in general remains an open problem; nevertheless, we believe that this heuristics can be useful in practice, and that proving its efficiency in this paper helps to understand it better.

Let us finally mention that contrary to Birgé and Massart (2007), we assume in this paper that the collection of models $\mathcal{M}_n$ is "small", that is $\mathrm{Card}(\mathcal{M}_n)$ grows at most like a power of n. For several problems, such that complete variable selection, larger collections of models have to be considered; then, it is known from the homoscedastic case that the minimal penalty is much larger than $\mathbb{E}[p_2(m)]$. Nevertheless, Émilie Lebarbier has used the slope heuristics with $f(D_m) = D_m \left(2.5 + \ln \left(\frac{n}{D_m} \right) \right)$ for multiple change-points detection from n noisy data, using the results by Birgé and Massart (2007) in the Gaussian case.

Let us now explain how we expect to generalize the slope heuristics to the non-Gaussian heteroscedastic case when $\mathcal{M}_n$ is large. First, group the models according to some complexity index C_m such as their dimensions D_m; for $C \in \{1, \dots, n^k\}$, define $\widetilde{S_C} = \bigcup_{C_m = C} S_m$. Then, replace the model selection problem with the family $(S_m)_{m \in \mathcal{M}_n}$ by a "complexity selection problem", that is model selection with the family $\left(\widetilde{S_C}\right)_{1 \le C \le n^k}$. We conjecture that this grouping of the models is sufficient to take into account the richness of $\mathcal{M}_n$ for the optimal calibration of the penalty. A theoretical justification of this point could rely on the extension of our results to any kind of model, since $\widetilde{S_C}$ is not a vector space in general.

Acknowledgments

The authors gratefully acknowledge the anonymous referees for several suggestions and references. The first author was financed by Univ Paris-Sud (Laboratoire de Mathematiques, CNRS - UMR 8628, Orsay, F-91405). The second author is also affiliated to INRIA Saclay, Projet Select.

Appendix A. Proofs

This appendix is devoted to the proofs of the results stated in the paper. Proposition 1 is proved in Section A.2; Theorem 3 is proved in Sections A.3 and A.4; Theorem 2 is proved in Section A.5; the remaining sections are devoted to probabilistic results used in the main proofs and technical proofs.

A.1 Conventions and Notations

In the rest of the paper, L denotes a universal constant, not necessarily the same at each occurrence. When L is not universal, but depends on $p_1, \dots, p_k$, it is written $L_{p_1, \dots, p_k}$. Similarly, $L_{(\mathbf{SH2})}$ (resp. $L_{(\mathbf{SH5})}$) denotes a constant allowed to depend on the parameters of the assumptions made in Theorem 2 (resp. Theorem 5), including (**P1**) and (**P2**). We also make use of the following notations:

- $\forall a, b \in \mathbb{R}$, $a \wedge b$ is the minimum of a and b, $a \vee b$ is the maximum of a and b, $a_+ = a \vee 0$ is the positive part of a and $a_- = a \wedge 0$ is its negative part.
- $\forall I_\lambda \subset \mathcal{X}$, $p_\lambda := P(X \in I_\lambda)$ and $\sigma_\lambda^2 := \mathbb{E}\left[(Y - s_m(X))^2 \,\middle|\, X \in I_\lambda \right]$.
- Since $\mathbb{E}[p_1(m)]$ is not well-defined (because of the event $\{\min_{\lambda \in \Lambda_m} \{\widehat{p_\lambda}\} = 0\}$), we have to take the following convention

$$p_1(m) = \widetilde{p_1}(m) := \sum_{\lambda \in \Lambda_m \text{ s.t. } \widehat{p_\lambda} > 0} p_\lambda \left(\beta_\lambda - \widehat{\beta_\lambda} \right)^2 + \sum_{\lambda \in \Lambda_m \text{ s.t. } \widehat{p_\lambda} = 0} p_\lambda \sigma_\lambda^2 \ .$$

Remark that $p_1(m) = \widetilde{p}_1(m)$ when $\min_{\lambda \in \Lambda_m} \{ \widehat{p}_\lambda \} > 0$), so that this convention has no consequences on the final results (Theorems 2 and 5).

A.2 Proof of Proposition 1

First, since $\mathcal{M}_n$ is finite, the infimum in (4) is attained as soon as $G(m_{i-1}) \neq \emptyset$, so that m_i is well defined for every $i \leq i_{\max}$. Moreover, by construction, $g(m_i)$ decreases with i, so that all the $m_i \in \mathcal{M}_n$ are different; hence, Algorithm 2 terminates and $i_{\max} + 1 \leq \mathrm{Card}(\mathcal{M}_n)$. We now prove by induction the following property for every $i \in \{0, \dots, i_{\max}\}$:

$$\mathcal{P}_i : \qquad K_i < K_{i+1} \quad \text{and} \quad \forall K \in [K_i, K_{i+1}), \quad \widehat{m}(K) = m_i \ .$$

Notice also that K_i can always be defined by (4) with the convention $\inf \emptyset = +\infty$.

A.2.1 $\mathcal{P}_0$ HOLDS TRUE

By definition of K_1, it is clear that $K_1 > 0$ (it may be equal to $+\infty$ if $G(m_0) = \emptyset$). For $K = K_0 = 0$, the definition of m_0 is the one of $\widehat{m}(0)$, so that $\widehat{m}(K) = m_0$. For $K \in (0, K_1)$, Lemma 4 shows that either $\widehat{m}(K) = \widehat{m}(0) = m_0$ or $\widehat{m}(K) \in G(0)$. In the latter case, by definition of K_1,

$$\frac{f(\widehat{m}(K)) - f(m_0)}{g(m_0) - g(\widehat{m}(K))} \geq K_1 > K$$

hence

$$f(\widehat{m}(K)) + K g(\widehat{m}(K)) > f(m_0) + K g(m_0)$$

which is contradictory with the definition of $\widehat{m}(K)$. Therefore, $\mathcal{P}_0$ holds true.

A.2.2 $\mathcal{P}_i \Rightarrow \mathcal{P}_{i+1}$ FOR EVERY $i \in \{0, \dots, i_{\max} - 1\}$

Assume that $\mathcal{P}_i$ holds true. First, we have to prove that $K_{i+2} > K_{i+1}$. Since $K_{i_{\max}+1} = +\infty$, this is clear if $i = i_{\max} - 1$. Otherwise, $K_{i+2} < +\infty$ and m_{i+2} exists. Then, by definition of m_{i+2} and K_{i+2} (resp. m_{i+1} and K_{i+1}), we have

$$f(m_{i+2}) - f(m_{i+1}) = K_{i+2}(g(m_{i+1}) - g(m_{i+2})) \tag{15}$$

$$f(m_{i+1}) - f(m_i) = K_{i+1}(g(m_i) - g(m_{i+1})) \ . \tag{16}$$

Moreover, $m_{i+2} \in G(m_{i+1}) \subset G(m_i)$, and $m_{i+2} \prec m_{i+1}$ (because g is non-decreasing). Using again the definition of K_{i+1}, we have

$$f(m_{i+2}) - f(m_i) > K_{i+1}(g(m_i) - g(m_{i+2})) \tag{17}$$

(otherwise, we would have $m_{i+2} \in F_{i+1}$ and $m_{i+2} \prec m_{i+1}$, which is not possible). Combining the difference of (17) and (16) with (15), we have

$$K_{i+2}(g(m_{i+1}) - g(m_{i+2})) > K_{i+1}(g(m_{i+1}) - g(m_{i+2})) \ ,$$

hence $K_{i+2} > K_{i+1}$, since $g(m_{i+1}) > g(m_{i+2})$.

Second, we prove that $\widehat{m}(K_{i+1}) = m_{i+1}$. From $\mathcal{P}_i$, we know that for every $m \in \mathcal{M}_n$, for every $K \in [K_i, K_{i+1})$, $f(m_i) + K g(m_i) \leq f(m) + K g(m)$. Taking the limit when K tends to K_{i+1}, it follows that $m_i \in E(K_{i+1})$. By (16), we then have $m_{i+1} \in E(K_{i+1})$. On the other hand, if $m \in E(K_{i+1})$, Lemma 4 shows that either $f(m) = f(m_i)$ and $g(m) = g(m_i)$ or $m \in G(m_i)$. In the first case, $m_{i+1} \prec m$ (because g is non-decreasing). In the second one, $m \in F_{i+1}$, so $m_{i+1} \preceq m$. Since $\widehat{m}(K_{i+1})$ is the smallest element of $E(K_{i+1})$, we have proved that $m_{i+1} = \widehat{m}(K_{i+1})$.

Last, we have to prove that $\widehat{m}(K) = m_{i+1}$ for every $K \in (K_1, K_2)$. From the last statement of Lemma 4, we have either $\widehat{m}(K) = \widehat{m}(K_1)$ or $\widehat{m}(K_1) \in G(\widehat{m}(K))$. In the latter case (which is only possible if $K_{i+2} < \infty$), by definition of K_{i+2},

$$\frac{f(\widehat{m}(K)) - f(m_{i+1})}{g(m_{i+1}) - g(\widehat{m}(K))} \geq K_{i+2} > K$$

so that

$$f(\widehat{m}(K)) + Kg(\widehat{m}(K)) > f(m_{i+1}) + Kg(m_{i+1})$$

which is contradictory with the definition of $\widehat{m}(K)$. ■

Lemma 4 *With the notations of Proposition 1 and its proof, if* $0 \leq K < K'$, $m \in E(K)$ *and* $m' \in E(K')$, *then one of the two following statements holds true:*

(a) $f(m) = f(m')$ *and* $g(m) = g(m')$.

(b) $f(m) < f(m')$ *and* $g(m) > g(m')$.

In particular, either $\widehat{m}(K) = \widehat{m}(K')$ *or* $\widehat{m}(K') \in G(\widehat{m}(K))$.

Proof By definition of $E(K)$ and $E(K')$,

$$f(m) + Kg(m) \leq f(m') + Kg(m') \tag{18}$$
$$f(m') + K'g(m') \leq f(m) + K'g(m) \ . \tag{19}$$

Summing (18) and (19) gives $(K' - K)g(m') \leq (K' - K)g(m)$ so that

$$g(m') \leq g(m) \ . \tag{20}$$

Since $K \geq 0$, (18) and (20) give $f(m) + Kg(m) \leq f(m') + Kg(m)$, that is

$$f(m) \leq f(m') \ . \tag{21}$$

Moreover, (21) and (19) imply $g(m) = g(m')$, hence $f(m') \leq f(m)$, that is $f(m) = f(m')$ by (21). Similarly, (18) and (20) show that $f(m) = f(m')$ imply $g(m) = g(m')$. In both cases, (a) is satisfied. Otherwise, $f(m) < f(m')$ and $g(m) > g(m')$, that is the (b) statement.

The last statement follows by taking $m = \widehat{m}(K)$ and $m' = \widehat{m}(K')$, because g is non-decreasing, so that the minimum of g in $E(K)$ is attained by $\widehat{m}(K)$. ■

A.3 A General Oracle Inequality

First of all, let us state a general theorem, from which Theorem 3 is an obvious corollary.

Theorem 5 *Suppose all the assumptions of Section 4.2 are satisfied together with*

(**Ap**) *The bias decreases like a power of D_m: there exist $\beta_- \geq \beta_+ > 0$ and $C_+, C_- > 0$ such that*

$$C_- D_m^{-\beta_-} \leq \ell(s, s_m) \leq C_+ D_m^{-\beta_+} \quad .$$

Let $L, \xi, c_1, C_1, C_2 \geq 0$, $c_2 > 1$ and assume that an event of probability at least $1 - Ln^{-2}$ exists on which, for every $m \in \mathcal{M}_n$ such that $D_m \geq \ln(n)^{\xi}$,

$$\begin{aligned} &\mathbb{E}\left[c_1 P(\gamma(\widehat{s}_m) - \gamma(s_m)) + c_2 P_n(\gamma(s_m) - \gamma(\widehat{s}_m)) \right] \\ &\quad \leq \mathrm{pen}(m) \leq \mathbb{E}\left[C_1 P(\gamma(\widehat{s}_m) - \gamma(s_m)) + C_2 P_n(\gamma(s_m) - \gamma(\widehat{s}_m)) \right] \quad . \end{aligned} \tag{22}$$

Then, for every $0 < \eta < \min\{\beta_+; 1\}/2$, there exist a constant K_3 and a sequence ε_n tending to zero at infinity such that, with probability at least $1 - K_3 n^{-2}$,

$$\begin{aligned} & D_{\widehat{m}} \leq n^{1-\eta} \\ \textit{and} \qquad & \ell(s, \widehat{s}_{\widehat{m}}) \leq \left[\frac{1 + (C_1 + C_2 - 2)_+}{(c_1 + c_2 - 1) \wedge 1} + \varepsilon_n \right] \inf_{m \in \mathcal{M}_n} \{ \ell(s, \widehat{s}_m) \} \end{aligned} \tag{23}$$

where $\widehat{m}$ is defined by (8). Moreover, we have the oracle inequality

$$\mathbb{E}[\ell(s, \widehat{s}_{\widehat{m}})] \leq \left[\frac{1 + (C_1 + C_2 - 2)_+}{(c_1 + c_2 - 1) \wedge 1} + \varepsilon_n \right] \mathbb{E}\left[\inf_{m \in \mathcal{M}_n} \{ \ell(s, \widehat{s}_m) \} \right] + \frac{A^2 K_3}{n^2} \quad . \tag{24}$$

The constant K_3 may depend on L, η, ξ, c_1, c_2, C_1, C_2 and constants in (**P1**), (**P2**), (**Ab**), (**An**), (**Ap**) *and* ($\mathbf{Ar}_\ell^{\mathbf{X}}$)*, but not on n. The term ε_n is smaller than $\ln(n)^{-1/5}$; it can be made smaller than $n^{-\delta}$ for any $\delta \in (0; \delta_0(\beta_-, \beta_+))$ at the price of enlarging K_3.*

The particular form of condition (22) on the penalty is motivated by the fact that the ideal shape of penalty $\mathbb{E}[\mathrm{pen}_{\mathrm{id}}(m)]$ (or equivalently $\mathbb{E}[2p_2(m)]$) is unknown in general. Then, it has to be estimated from the data, for instance by resampling. Under the assumptions of Theorem 5, Arlot (2008b,c) has proved that resampling and V-fold penalties satisfy condition (22) with constants $c_1 + c_2 = 2 - \delta_n$, $C_1 + C_2 = 2 + \delta_n$ (for some absolute sequence δ_n tending to zero at infinity), and some numerical constant $\xi > 0$. Then, Theorem 5 shows that such a penalization procedure satisfies an oracle inequality with leading constant tending to 1 asymptotically.

The rationale behind Theorem 5 is that if $\mathrm{pen}(m)$ is close to $c_1 p_1(m) + c_2 p_2(m)$, then $\mathrm{crit}(m) \approx \ell(s, s_m) + c_1 p_1(m) + (c_2 - 1) p_2(m)$. When $c_1 = c_2 = 1$, this is exactly the ideal criterion $\ell(s, \widehat{s}_m)$. When $c_1 + c_2 = 2$ with $c_1 \geq 0$ and $c_2 > 1$, we obtain the same result because $p_1(m)$ and $p_2(m)$ are quite close, at least when D_m is large enough. The closeness between p_1 and p_2 is the keystone of the slope heuristics. Notice that if $\max_{m \in \mathcal{M}_n} D_m \leq K_3'(\ln(n))^{-1} n$ (for some constant K_3' depending only on the assumptions of Theorem 3, as K_3), one can replace the condition $c_2 > 1$ by $c_1 + c_2 > 1$ and $c_1, c_2 \geq 0$.

A.4 Proof of Theorem 5

This proof is similar to the one of Arlot (2008c, Theorem 1). We give it for the sake of completeness.

From (3), we have for each $m \in \mathcal{M}_n$ such that $A_n(m) := \min_{\lambda \in \Lambda_m} \{ n\widehat{p}_\lambda \} > 0$

$$\ell(s, \widehat{s}_{\widehat{m}}) - \left(\mathrm{pen}'_{\mathrm{id}}(\widehat{m}) - \mathrm{pen}(\widehat{m}) \right) \leq \ell(s, \widehat{s}_m) + \left(\mathrm{pen}(m) - \mathrm{pen}'_{\mathrm{id}}(m) \right) \quad . \tag{25}$$

with $\mathrm{pen}'_{\mathrm{id}}(m) := p_1(m) + p_2(m) - \overline{\delta}(m) = \mathrm{pen}(m) + (P - P_n)\gamma(s)$ and $\overline{\delta}(m) := (P_n - P)(\gamma(s_m) - \gamma(s))$. It is sufficient to control $\mathrm{pen} - \mathrm{pen}'_{\mathrm{id}}$ for every $m \in \mathcal{M}_n$.

We will thus use the concentration inequalities of Section A.6 with $x = \gamma \ln(n)$ and $\gamma = 2 + \alpha_{\mathcal{M}}$. Define $B_n(m) = \min_{\lambda \in \Lambda_m} \{ np_\lambda \}$, and Ω_n the event on which

- for every $m \in \mathcal{M}_n$, (22) holds
- for every $m \in \mathcal{M}_n$ such that $B_n(m) \geq 1$, (31) and (32) hold:

$$\widetilde{p}_1(m) \geq \mathbb{E}[\widetilde{p}_1(m)] - L_{(\mathbf{SH5})} \left[\frac{\ln(n)^2}{\sqrt{D_m}} + e^{-LB_n(m)} \right] \mathbb{E}[p_2(m)]$$
$$\widetilde{p}_1(m) \leq \mathbb{E}[\widetilde{p}_1(m)] + L_{(\mathbf{SH5})} \left[\frac{\ln(n)^2}{\sqrt{D_m}} + \sqrt{D_m} e^{-LB_n(m)} \right] \mathbb{E}[p_2(m)]$$

- for every $m \in \mathcal{M}_n$ such that $B_n(m) > 0$, (33), (30) and 28 hold:

$$\widetilde{p}_1(m) \geq \left(\frac{1}{2 + (\gamma+1)B_n(m)^{-1}\ln(n)} - \frac{L_{(\mathbf{SH5})}\ln(n)^2}{\sqrt{D_m}} \right) \mathbb{E}[p_2(m)]$$
$$|p_2(m) - \mathbb{E}[p_2(m)]| \leq \frac{L_{(\mathbf{SH5})}\ln(n)}{\sqrt{D_m}} [\ell(s, s_m) + \mathbb{E}[p_2(m)]]$$
$$\left| \overline{\delta}(m) \right| \leq \frac{\ell(s, s_m)}{\sqrt{D_m}} + L_{(\mathbf{SH5})} \frac{\ln(n)}{\sqrt{D_m}} \mathbb{E}[p_2(m)]$$

From Proposition 11 (for $\widetilde{p}_1$), Proposition 10 (for p_2) and Proposition 8 (for $\overline{\delta}(m)$),

$$\mathbb{P}(\Omega_n) \geq 1 - L \sum_{m \in \mathcal{M}_n} n^{-2-\alpha_{\mathcal{M}}} \geq 1 - L_{c_{\mathcal{M}}} n^{-2} \quad .$$

For every $m \in \mathcal{M}_n$ such that $D_m \leq L_{c^X_{\mathrm{r},\ell}} n \ln(n)^{-1}$, $(\mathbf{Ar}^{\mathbf{X}}_\ell)$ implies that $B_n(m) \geq L^{-1}\ln(n) \geq 1$. As a consequence, on Ω_n, if $\ln(n)^7 \leq D_m \leq L_{c^X_{\mathrm{r},\ell}} n \ln(n)^{-1}$:

$$\max\left\{ |\widetilde{p}_1(m) - \mathbb{E}[\widetilde{p}_1(m)]|, |p_2(m) - \mathbb{E}[p_2(m)]|, \left| \overline{\delta}(m) \right| \right\} \leq \frac{L_{(\mathbf{SH5})}\mathbb{E}[\ell(s, s_m) + p_2(m)]}{\ln(n)}$$

Using (34) (in Proposition 12) and the fact that $B_n(m) \geq L^{-1}\ln(n)$,

$$\frac{(c_1 + c_2)\left(1 - \widetilde{\delta}_n\right)}{2} \leq \mathbb{E}[\mathrm{pen}(m)] \leq \frac{(C_1 + C_2)\left(1 + \widetilde{\delta}_n\right)}{2} \mathbb{E}[\widetilde{p}_1(m) + p_2(m)]$$

with $0 \leq \widetilde{\delta_n} \leq L \ln(n)^{-1/4}$. We deduce: if $n \geq L_{(\mathbf{SH5})}$, for every $m \in \mathcal{M}_n$ such that $\ln(n)^7 \leq D_m \leq L_{c^X_{\mathrm{r},\ell}} n \ln(n)^{-1}$, on Ω_n,

$$\left[(c_1 + c_2 - 2)_- - \frac{L_{(\mathbf{SH5})}}{\ln(n)^{1/4}} \right] p_1(m) \leq (\mathrm{pen} - \mathrm{pen}'_{\mathrm{id}})(m)$$
$$\leq \left[(C_1 + C_2 - 2)_+ + \frac{L_{(\mathbf{SH5})}}{\ln(n)^{1/4}} \right] p_1(m) \enspace .$$

We need to assume that n is large enough in order to upper bound $\mathbb{E}[p_2(m)]$ in terms of $p_1(m)$, since we only have

$$p_1(m) \geq \left[1 - \frac{L_{(\mathbf{SH5})}}{\ln(n)^{1/4}} \right]_+ \mathbb{E}[p_2(m)]$$

in general. Combined with (25), this gives: if $n \geq L_{(\mathbf{SH5})}$,

$$\ell(s, \widehat{s}_{\widehat{m}}) \mathbb{1}_{\ln(n)^5 \leq D_{\widehat{m}} \leq L_{c^X_{\mathrm{r},\ell}} n \ln(n)^{-1}} \leq \left[\frac{1 + (C_1 + C_2 - 2)_+}{(c_1 + c_2 - 1) \wedge 1} + \frac{L_{(\mathbf{SH5})}}{\ln(n)^{1/4}} \right]$$
$$\times \inf_{m \in \mathcal{M}_n \text{ s.t. } \ln(n)^7 \leq D_m \leq L_{\alpha_{\mathcal{M}}, c^X_{\mathrm{r},\ell}} n \ln(n)^{-1}} \{ \ell(s, \widehat{s}_m) \} \enspace .$$

We now use Lemmas 6 and 7 below to control on Ω_n the dimensions of the selected model $\widehat{m}$ and the oracle model $m^\star \in \arg\min_{m \in \mathcal{M}_n} \{ \ell(s, \widehat{s}_m) \}$.

The result follows since $L_{(\mathbf{SH5})} \ln(n)^{-1/4} \leq \varepsilon_n = \ln(n)^{-1/5}$ for $n \geq L_{(\mathbf{SH5})}$. We finally remove the condition $n \geq n_0 = L_{(\mathbf{SH5})}$ by choosing $K_3 = L_{(\mathbf{SH5})}$ such that $K_3 n_0^{-2} \geq 1$.

A.4.1 CLASSICAL ORACLE INEQUALITY

Since (23) holds true on Ω_n,

$$\mathbb{E}[\ell(s, \widehat{s}_{\widehat{m}})] = \mathbb{E}[\ell(s, \widehat{s}_{\widehat{m}}) \mathbb{1}_{\Omega_n}] + \mathbb{E}\left[\ell(s, \widehat{s}_{\widehat{m}}) \mathbb{1}_{\Omega_n^c}\right]$$
$$\leq [2\eta - 1 + \varepsilon_n] \mathbb{E}\left[\inf_{m \in \mathcal{M}_n} \{ \ell(s, \widehat{s}_m) \} \right] + A^2 K_3 \mathbb{P}(\Omega_n^c)$$

which proves (24). ■

Lemma 6 (Control on the dimension of the selected model) *Let $c > 0$ and $\alpha > (1 - \beta_+)_+ / 2$. Then, if $n \geq L_{(\mathbf{SH5}), c, \alpha}$, on the event Ω_n defined in the proof of Theorem 5,*

$$\ln(n)^7 \leq D_{\widehat{m}} \leq n^{1/2 + \alpha} \leq c n \ln(n)^{-1} \enspace .$$

Lemma 7 (Control on the dimension of the oracle model) *Define the oracle model $m^\star \in \arg\min_{m \in \mathcal{M}_n} \{ \ell(s, \widehat{s}_m) \}$. Let $c > 0$ and $\alpha > (1 - \beta_+)_+ / 2$. Then, if $n \geq L_{(\mathbf{SH5}), c, \alpha}$, on the event Ω_n defined in the proof of Theorem 5,*

$$\ln(n)^7 \leq D_{m^\star} \leq n^{1/2 + \alpha} \leq c n \ln(n)^{-1} \enspace .$$

A.4.2 PROOF OF LEMMA 6

By definition, $\widehat{m}$ minimizes $\operatorname{crit}(m)$ over $\mathcal{M}_n$. It thus also minimizes

$$\operatorname{crit}'(m) = \operatorname{crit}(m) - P_n\gamma(s) = \ell(s, s_m) - p_2(m) + \overline{\delta}(m) + \operatorname{pen}(m)$$

over $\mathcal{M}_n$.

1. Lower bound on $\operatorname{crit}'(m)$ for small models: let $m \in \mathcal{M}_n$ such that $D_m < (\ln(n))^7$. We then have

$$\begin{aligned} \ell(s, s_m) &\geq C_- (\ln(n))^{-7\beta_-} && \text{from } (\mathbf{Ap}) \\ \operatorname{pen}(m) &\geq 0 \\ p_2(m) &\leq L_{(\mathbf{SH5})}\sqrt{\frac{\ln(n)}{n}} + L_{(\mathbf{SH5})}\frac{D_m}{n} \leq L_{(\mathbf{SH5})}\sqrt{\frac{\ln(n)}{n}} && \text{from (29)} \end{aligned}$$

   and from (28) (in Proposition 8),

$$\overline{\delta}(m) \geq -L_A\sqrt{\frac{\ell(s, s_m)\ln(n)}{n}} + L_A\frac{\ln(n)}{n} \geq -L_A\sqrt{\frac{\ln(n)}{n}} \quad .$$

   We then have
$$\operatorname{crit}'(m) \geq L_{(\mathbf{SH5})}(\ln(n))^{-L_{\beta_-}} \quad .$$

2. Lower bound for large models: let $m \in \mathcal{M}_n$ such that $D_m \geq n^{1/2+\alpha}$. From (22) and (29) (in Proposition 10),

$$\begin{aligned} \operatorname{pen}(m) - p_2(m) &\geq (c_2 - 1)\mathbb{E}[p_2(m)] - L_A\sqrt{\frac{\ln(n)}{n}} \\ &\geq \frac{(c_2-1)\sigma_{\min}^2 D_m}{n} - L_A\sqrt{\frac{\ln(n)}{n}} \end{aligned}$$

   and from (26),
$$\overline{\delta}(m) \geq -L_{(\mathbf{SH5})}\sqrt{\frac{\ln(n)}{n}} \quad .$$

   Hence, if $D_m \geq n^{1/2+\alpha}$ and $n \geq L_{(\mathbf{SH5}),\alpha}$

$$\operatorname{crit}'(m) \geq \operatorname{pen}(m) + \overline{\delta}(m) - p_2(m) \geq L_{(\mathbf{SH5}),\alpha} n^{-1/2+\alpha} \quad .$$

3. There exists a better model for $\operatorname{crit}(m)$: from $(\mathbf{P2})$, there exists $m_0 \in \mathcal{M}_n$ such that $\sqrt{n} \leq D_{m_0} \leq c_{\mathrm{rich}}\sqrt{n}$. If moreover $n \geq L_{c_{\mathrm{rich}},\alpha}$, then

$$\ln(n)^7 \leq \sqrt{n} \leq D_{m_0} \leq c_{\mathrm{rich}}\sqrt{n} \leq n^{1/2+\alpha} \quad .$$

   By (35) in Lemma 13, $A_n(m_0) \geq 1$ with probability at least $1 - Ln^{-2}$.
   Using $(\mathbf{Ap})$,
$$\ell(s, s_{m_0}) \leq C_+ c_{\mathrm{rich}}^{\beta_+} n^{-\beta_+/2}$$

so that, when $n \geq L_{(\mathbf{SH5})}$,

$$\begin{aligned}\operatorname{crit}'(m_0) &\leq \ell(s, s_{m_0}) + \left|\overline{\delta}(m)\right| + \operatorname{pen}(m) \\ &\leq L_{(\mathbf{SH5})}\left(n^{-\beta_+/2} + n^{-1/2}\right) \quad .\end{aligned}$$

If $n \geq L_{(\mathbf{SH5}),\alpha}$, this upper bound is smaller than the previous lower bounds for small and large models. ■

A.4.3 PROOF OF LEMMA 7

Recall that $m^\star$ minimizes $\ell(s, \widehat{s}_m) = \ell(s, s_m) + p_1(m)$ over $m \in \mathcal{M}_n$, with the convention $\ell(s, \widehat{s}_m) = \infty$ if $A_n(m) = 0$.

1. Lower bound on $\ell(s, \widehat{s}_m)$ for small models: let $m \in \mathcal{M}_n$ such that $D_m < (\ln(n))^7$. From (**Ap**), we have
$$\ell(s, \widehat{s}_m) \geq \ell(s, s_m) \geq C_-(\ln(n))^{-7\beta_-} \quad .$$

2. Lower bound on $\ell(s, \widehat{s}_m)$ for large models: let $m \in \mathcal{M}_n$ such that $D_m > n^{1/2+\alpha}$. From (33), for $n \geq L_{(\mathbf{SH5}),\alpha}$,
$$\widetilde{p}_1(m) \geq \left(\frac{1}{2 + (\gamma+1)\left(c_{\mathrm{r},\ell}^X\right)^{-1}\ln(n)} - \frac{L_{(\mathbf{SH5}),\alpha}}{n^{1/4}}\right)\mathbb{E}\left[\widetilde{p}_2(m)\right]$$
$$\text{so that} \quad \ell(s, \widehat{s}_m) \geq \widetilde{p}_1(m) \geq L_{(\mathbf{SH5}),\alpha} n^{-1/2+\alpha} \quad .$$

3. There exists a better model for $\ell(s, \widehat{s}_m)$: let $m_0 \in \mathcal{M}_n$ be as in the proof of Lemma 6 and assume that $n \geq L_{c_{\mathrm{rich}},\alpha}$. Then,
$$p_1(m_0) \leq L_{(\mathbf{SH5})}\mathbb{E}\left[p_2(m)\right] \leq L_{(\mathbf{SH5})} n^{-1/2}$$
and the arguments of the previous proof show that
$$\ell(s, \widehat{s}_{m_0}) \leq L_{(\mathbf{SH5})}\left(n^{-\beta_+/2} + n^{-1/2}\right)$$
which is smaller than the previous upper bounds for $n \geq L_{(\mathbf{SH5}),\alpha}$. ■

A.5 Proof of Theorem 2

Similarly to the proof of Theorem 5, we consider the event Ω'_n, of probability at least $1 - L_{c_{\mathcal{M}}} n^{-2}$, on which:

- for every $m \in \mathcal{M}_n$, (9) (for pen), (33) (for $\widetilde{p}_1$), (29)–(30) (for p_2, with $x = \gamma\ln(n)$ and $\theta = \sqrt{\ln(n)/n}$) and (26)–(28) (for $\overline{\delta}$, with $x = \gamma\ln(n)$ and $\eta = \sqrt{\ln(n)/n}$) hold true.

- for every $m \in \mathcal{M}_n$ such that $B_n(m) \geq 1$, (31) and (32) hold (for $\widetilde{p}_1$).

A.5.1 LOWER BOUND ON $D_{\widehat{m}}$

By definition, $\widehat{m}$ minimizes

$$\operatorname{crit}'(m) = \operatorname{crit}(m) - P_n\gamma(s) = \ell(s, s_m) - p_2(m) + \overline{\delta}(m) + \operatorname{pen}(m)$$

over $m \in \mathcal{M}_n$ such that $A_n(m) \geq 1$. As in the proof of Theorem 5, we define $c = L_{c^X_{\mathrm{r},\ell}} > 0$ such that for every model of dimension $D_m \leq cn\ln(n)^{-1}$, $B_n(m) \geq L^{-1}\ln(n) \geq 1$. Let $c' = \min(c, c_0)$ and $d \in (0,1)$ a constant to be chosen later.

1. Lower bound on $\operatorname{crit}'(m)$ for "small" models: assume that $m \in \mathcal{M}_n$ and $D_m \leq dc'n\ln(n)^{-1}$. Then, $\ell(s, s_m) + \operatorname{pen}(m) \geq 0$ and from (26),

   $$\overline{\delta}(m) \geq -L_A\sqrt{\frac{\ln(n)}{n}} \enspace .$$

   If $D_m \geq \ln(n)^4$, (30) implies that

   $$p_2(m) \leq \left(1 + \frac{L_{(\mathbf{SH2})}}{\ln(n)}\right)\mathbb{E}\left[p_2(m)\right] \leq \frac{L_{(\mathbf{SH2})}D_m}{n} \leq \frac{c'dL_{(\mathbf{SH2})}}{\ln(n)} \enspace .$$

   On the other hand, if $D_m < \ln(n)^4$, (29) implies that

   $$p_2(m) \leq L_{(\mathbf{SH2})}\sqrt{\frac{\ln(n)}{n}} \enspace .$$

   We then have

   $$\operatorname{crit}'(m) \geq -dL_{(\mathbf{SH2})}\left(\ln(n)\right)^{-1} \enspace .$$

2. There exists a better model for $\operatorname{crit}(m)$: let $m_1 \in \mathcal{M}_n$ such that

   $$\ln(n)^4 \leq \frac{c'dn}{c_{\mathrm{rich}}\ln(n)} \leq D_{m_1} \leq \frac{c'n}{\ln(n)} \leq n \enspace .$$

   From (**P2**+), this is possible as soon as $n \geq L_{c_{\mathrm{rich}},c',d}$. By (35) in Lemma 13, $A_n(m_0) \geq 1$ with probability at least $1 - Ln^{-2}$.
   We then have

   $$\begin{aligned}
   \ell(s, s_{m_1}) &\leq L_{(\mathbf{SH2}),c'}\ln(n)^{\beta_+}n^{-\beta_+} && \text{by } (\mathbf{Ap})\\
   p_2(m_1) &\geq \left(1 - \frac{L_{(\mathbf{SH2})}}{\ln(n)}\right)\mathbb{E}\left[p_2(m_1)\right] && \text{by (30)}\\
   \operatorname{pen}(m_1) &\leq K\mathbb{E}\left[p_2(m_1)\right] && \text{by (9)}\\
   \left|\overline{\delta}(m_1)\right| &\leq L_A\sqrt{\frac{\ln(n)}{n}} && \text{by (26)}
   \end{aligned}$$

   so that

   $$\begin{aligned}
   \operatorname{crit}'(m_1) &\leq L_{(\mathbf{SH2}),c'}\ln(n)^{\beta_+}n^{-\beta_+} + \left(K - 1 + \frac{L_{(\mathbf{SH2})}}{\ln(n)}\right)\mathbb{E}\left[p_2(m_1)\right] + L_A\sqrt{\frac{\ln(n)}{n}}\\
   &\leq \frac{(K - 1 + L_{(\mathbf{SH2})}(\ln(n))^{-1})\sigma_{\min}^2 c'}{2\ln(n)}
   \end{aligned}$$

if $n \geq L_{(\mathbf{SH2}),c'}$.

We now choose d such that the constant $dL_{(\mathbf{SH2})}$ appearing in the lower bound on $\mathrm{crit}'(m)$ for "small" models is smaller than $(1-K-L_{(\mathbf{SH2})}(\ln(n))^{-1})\sigma_{\min}^2 c'/2$, that is $d \leq L_{(\mathbf{SH2}),c'}$. Then, we assume that $n \geq n_0 = L_{(\mathbf{SH2}),c',d} = L_{(\mathbf{SH2})}$. Finally, we remove this condition as before by enlarging K_1.

A.5.2 RISK OF $D_{\widehat{m}}$

The proof of (10) is quite similar to the one of Lemma 7. First, for every model $m \in \mathcal{M}_n$ such that $A_n(m) \geq 1$ and $D_m \geq K_2 n \ln(n)^{-1}$, we have

$$\ell(s, \widehat{s}_m) \geq \widetilde{p}_1(m) \geq L_{(\mathbf{SH2})} K_2 \ln(n)^{-2} \qquad \text{by (33)} \ .$$

Then, the model $m_0 \in \mathcal{M}_n$ defined previously satisfies $A_n(m) \geq 1$, and

$$\ell(s, \widehat{s}_{m_0}) \leq L_{(\mathbf{SH2})} \left(n^{-\beta_+/2} + n^{-1/2} \right) \ .$$

If $n \geq L_{(\mathbf{SH2})}$, the ratio between these two bounds is larger than $\ln(n)$, so that (10) holds. ■

A.6 Concentration Inequalities Used in the Main Proofs

In this section, we no longer assume that each model is the set of piecewise constant functions on some partition of $\mathcal{X}$. First, we control $\overline{\delta}(m)$ with general models and bounded data.

Proposition 8 *Assume that $\|Y\|_\infty \leq A < \infty$. Then for all $x \geq 0$, on an event of probability at least $1-2e^{-x}$:*

$$\forall \eta > 0, \quad \left|\overline{\delta}(m)\right| \leq \eta \ell(s, s_m) + \left(\frac{4}{\eta} + \frac{8}{3}\right) \frac{A^2 x}{n} \ . \tag{26}$$

If moreover

$$Q_m^{(p)} := \frac{n \mathbb{E}[p_2(m)]}{D_m} > 0 \ , \tag{27}$$

on the same event,

$$\left|\overline{\delta}(m)\right| \leq \frac{\ell(s, s_m)}{\sqrt{D_m}} + \frac{20}{3} \frac{A^2}{Q_m^{(p)}} \frac{\mathbb{E}[p_2(m)]}{\sqrt{D_m}} x \ . \tag{28}$$

Remark 9 (Regressogram case) *If S_m is the set of piecewise constant functions on some partition $(I_\lambda)_{\lambda \in \Lambda_m}$ of $\mathcal{X}$,*

$$Q_m^{(p)} = \frac{1}{D_m} \sum_{\lambda \in \Lambda_m} \sigma_\lambda^2 \geq (\sigma_{\min})^2 > 0 \ .$$

Then, we derive a concentration inequality for $p_2(m)$ in the regressogram case from a general result by Boucheron and Massart (2008).

Proposition 10 *Let S_m be the model of piecewise constant functions associated with the partition $(I_\lambda)_{\lambda \in \Lambda_m}$. Assume that $\|Y\|_\infty \leq A$ and define $p_2(m) = P_n(\gamma(s_m) - \gamma(\widehat{s}_m))$.*

Then, for every $x \geq 0$, there exists an event of probability at least $1 - e^{1-x}$ on which for every $\theta \in (0;1)$,

$$|p_2(m) - \mathbb{E}[p_2(m)]| \leq L \left[\theta \ell(s, s_m) + \frac{A^2 \sqrt{D_m} \sqrt{x}}{n} + \frac{A^2 x}{\theta n} \right] \tag{29}$$

for some absolute constant L. If moreover $\sigma(X) \geq \sigma_{\min} > 0$ a.s., we have on the same event:

$$|p_2(m) - \mathbb{E}[p_2(m)]| \leq \frac{L}{\sqrt{D_m}} \left[\ell(s, s_m) + \frac{A^2 \mathbb{E}[p_2(m)]}{\sigma_{\min}^2} \left(\sqrt{x} + x \right) \right] \quad . \tag{30}$$

Finally, we recall a concentration inequality for $p_1(m)$ proved by (Arlot, 2008b, Proposition 9). Its proof is particular to the regressogram case.

Proposition 11 (Proposition 9, Arlot 2008b) *Let $\gamma > 0$ and S_m be the model of piecewise constant functions associated with the partition $(I_\lambda)_{\lambda \in \Lambda_m}$. Assume that $\|Y\|_\infty \leq A < \infty$, $\sigma(X) \geq \sigma_{\min} > 0$ a.s. and $\min_{\lambda \in \Lambda_m} \{ n p_\lambda \} \geq B_n > 0$. Then, if $B_n \geq 1$, on an event of probability at least $1 - L n^{-\gamma}$,*

$$\widetilde{p_1}(m) \geq \mathbb{E}[\widetilde{p_1}(m)] - L_{A,\sigma_{\min},\gamma} \left[\frac{\ln(n)^2}{\sqrt{D_m}} + e^{-L B_n} \right] \mathbb{E}[p_2(m)] \tag{31}$$

$$\widetilde{p_1}(m) \leq \mathbb{E}[\widetilde{p_1}(m)] + L_{A,\sigma_{\min},\gamma} \left[\frac{\ln(n)^2}{\sqrt{D_m}} + \sqrt{D_m} e^{-L B_n} \right] \mathbb{E}[p_2(m)] \quad . \tag{32}$$

If we only have a lower bound $B_n > 0$, then, with probability at least $1 - L n^{-\gamma}$,

$$\widetilde{p_1}(m) \geq \left(\frac{1}{2 + (\gamma+1) B_n^{-1} \ln(n)} - \frac{L_{A,\sigma_{\min},\gamma} \ln(n)^2}{\sqrt{D_m}} \right) \mathbb{E}[p_2(m)] \quad . \tag{33}$$

A.7 Additional Results Needed

A crucial result in the proofs of Theorems 5 and 2 is that $p_1(m)$ and $p_2(m)$ are close in expectation; the following proposition was proved by Arlot (2008b, Lemma 7).

Proposition 12 (Lemma 7, Arlot 2008b) *Let S_m be a model of piecewise constant functions adapted to some partition $(I_\lambda)_{\lambda \in \Lambda_m}$. Assume that $\min_{\lambda \in \Lambda_m} \{ n p_\lambda \} \geq B > 0$. Then,*

$$\begin{aligned} \left(1 - e^{-B}\right)^2 \mathbb{E}[p_2(m)] &\leq \mathbb{E}[\widetilde{p_1}(m)] \\ &\leq \left[2 \wedge \left(1 + 5.1 \times B^{-1/4} \right) + (B \vee 1) e^{-(B \vee 1)} \right] \mathbb{E}[p_2(m)] \quad . \end{aligned} \tag{34}$$

Finally, we need the following technical lemma in the proof of the main theorems.

Lemma 13 *Let $(p_\lambda)_{\lambda \in \Lambda_m}$ be non-negative real numbers of sum 1, $(n \widehat{p}_\lambda)_{\lambda \in \Lambda_m}$ a multinomial vector of parameters $(n; (p_\lambda)_{\lambda \in \Lambda_m})$. Then, for all $\gamma > 0$,*

$$\min_{\lambda \in \Lambda_m} \{ n \widehat{p}_\lambda \} \geq \frac{\min_{\lambda \in \Lambda_m} \{ n p_\lambda \}}{2} - 2(\gamma+1) \ln(n) \tag{35}$$

with probability at least $1 - 2n^{-\gamma}$.

Proof By Bernstein inequality (Massart, 2007, Proposition 2.9), for all $\lambda \in \Lambda_m$,

$$\mathbb{P}\left(n\widehat{p}_\lambda \geq (1-\theta) n p_\lambda - \sqrt{2npx} - \frac{x}{3} \right) \geq 1 - e^{-x} \ .$$

Take $x = (\gamma+1)\ln(n)$ above, and remark that $\sqrt{2npx} \leq \frac{np}{2} + x$. The union bound gives the result since $\operatorname{Card}(\Lambda_m) \leq n$. ■

A.8 Proof of Proposition 8

Since $\|Y\|_\infty \leq A$, we have $\|s\|_\infty \leq A$ and $\|s_m\|_\infty \leq A$. In fact, everything happens as if $S_m \cup \{s\}$ was bounded by A in L^∞.

We have

$$\overline{\delta}(m) = \frac{1}{n} \sum_{i=1}^{n} \left(\gamma(s_m, (X_i,Y_i)) - \gamma(s,(X_i,Y_i)) - \mathbb{E}\left[\gamma(s_m,(X_i,Y_i)) - \gamma(s,(X_i,Y_i)) \right] \right)$$

and assumptions of Bernstein inequality (Massart, 2007, Proposition 2.9) are fulfilled with

$$c = \frac{8A^2}{3n} \quad \text{and} \quad v = \frac{8A^2 \ell(s,s_m)}{n}$$

since

$$\left\| \gamma(s_m,(X_i,Y_i)) - \gamma(s,(X_i,Y_i)) - \mathbb{E}\left[\gamma(s_m,(X_i,Y_i)) - \gamma(s,(X_i,Y_i)) \right] \right\|_\infty \leq 8A^2$$

and

$$\begin{aligned} \operatorname{var}\left(\gamma(s_m,(X_i,Y_i)) - \gamma(s,(X_i,Y_i)) \right) &\leq \mathbb{E}\left[\left(\gamma(s_m,(X_i,Y_i)) - \gamma(s,(X_i,Y_i)) \right)^2 \right] \\ &\leq 8A^2 \ell(s,s_m) \end{aligned}$$

because $\|s_m - s\|_\infty \leq 2A$ and

$$\begin{aligned} (\gamma(t,\cdot) - \gamma(s,\cdot))^2 &= (t(X) - s(X))^2 \left(2(Y - s(X)) - t(X) + s(X) \right)^2 \\ \text{and} \quad \mathbb{E}\left[(Y - s(X))^2 \,\middle|\, X \right] &\leq \frac{(2A)^2}{4} = A \ . \end{aligned}$$

We obtain that, with probability at least $1 - 2e^{-x}$,

$$\left| \overline{\delta}(m) \right| \leq \sqrt{2vx} + c = \sqrt{\frac{16A^2 \ell(s,s_m) x}{n}} + \frac{8A^2 x}{3n}$$

and (26) follows since $2\sqrt{ab} \leq a\eta + b\eta^{-1}$ for all $\eta > 0$. Taking $\eta = D_m^{-1/2} \leq 1$ and using $Q_m^{(p)}$ defined by (27), we deduce (28). ■

A.9 Proof of Proposition 10

We apply here a result by Boucheron and Massart (2008, Theorem 2.2 in a preliminary version), in which it is only assumed that γ takes its values in $[0;1]$. This is satisfied when $\|Y\|_\infty \leq A = 1/2$. When $A \neq 1/2$, we apply this result to $(2A)^{-1}Y$ and recover the general result by homogeneity.

First, we recall this result in the bounded least-squares regression framework. For every $t : \mathcal{X} \mapsto \mathbb{R}$ and $\varepsilon > 0$, we define

$$d^2(s,t) = 2\ell(s,t) \qquad \text{and} \qquad w(\varepsilon) = \sqrt{2}\varepsilon \ .$$

Let ϕ_m belong to the class of nondecreasing and continuous functions $f : \mathbb{R}^+ \mapsto \mathbb{R}^+$ such that $x \mapsto f(x)/x$ is nonincreasing on $(0;+\infty)$ and $f(1) \geq 1$. Assume that for every $u \in S_m$ and $\sigma > 0$ such that $\phi_m(\sigma) \leq \sqrt{n}\sigma^2$,

$$\sqrt{n}\mathbb{E}\left[\sup_{t \in S_m, d(u,t) \leq \sigma} |\overline{\gamma}_n(u) - \overline{\gamma}_n(t)|\right] \leq \phi_m(\sigma) \ . \tag{36}$$

Let $\varepsilon_{\star,m}$ be the unique positive solution of the equation

$$\sqrt{n}\varepsilon_{\star,m}^2 = \phi_m(w(\varepsilon_{\star,m})) \ .$$

Then, there exists some absolute constant L such that for every real number $q \geq 2$ one has

$$\|p_2(m) - \mathbb{E}[p_2(m)]\|_q \leq \frac{L}{\sqrt{n}}\left[\sqrt{2q}\left(\sqrt{\ell(s,s_m)} \vee \varepsilon_{\star,m}\right) + q\frac{2}{\sqrt{n}}\right] \ . \tag{37}$$

Using now that S_m is the set of piecewise constant functions on some partition $(I_\lambda)_{\lambda \in \Lambda_m}$ of $\mathcal{X}$, we can take

$$\phi_m(\sigma) = 3\sqrt{2}\sqrt{D_m} \times \sigma \qquad \text{in (36).} \tag{38}$$

The proof of this statement is made below. Then, $\varepsilon_{\star,m} = 6\sqrt{D_m}n^{-1/2}$.

Combining (37) with the classical link between moments and concentration (see for instance Arlot, 2007, Lemma 8.9), the first result follows. The second result is obtained by taking $\theta = D_m^{-1/2}$, as in Proposition 8. ■

A.9.1 PROOF OF (38)

Let $u \in S_m$ and $d(u,t) = \sqrt{2}\|u(X) - t(X)\|_2$ for every $t : \mathcal{X} \mapsto \mathbb{R}$. Define $\psi : \mathbb{R}^+ \mapsto \mathbb{R}^+$ by

$$\psi(\sigma) = \mathbb{E}\left[\sup_{d(u,t) \leq \sigma, t \in S_m} |(P_n - P)(\gamma(u,\cdot) - \gamma(t,\cdot))|\right] \ .$$

We are looking for some nondecreasing and continuous function $\phi_m : \mathbb{R}^+ \mapsto \mathbb{R}^+$ such that $\phi_m(x)/x$ is nonincreasing, $\phi_m(1) \geq 1$ and for every $u \in S_m$,

$$\forall \sigma > 0 \quad \text{such that} \quad \phi_m(\sigma) \leq \sqrt{n}\sigma^2 \ , \qquad \phi_m(\sigma) \geq \sqrt{n}\psi(\sigma) \ .$$

We first look at a general upperbound on ψ.

Assume that $u = s_m$. If this is not the case, the triangular inequality shows that $\psi_{\text{general } u} \leq 2\psi_{u=s_m}$. Let us write

$$t = \sum_{\lambda \in \Lambda_m} t_\lambda \mathbb{1}_{I_\lambda} \qquad u = s_m = \sum_{\lambda \in \Lambda_m} \beta_\lambda \mathbb{1}_{I_\lambda} \ .$$

A.9.2 COMPUTATION OF $P(\gamma(t,\cdot)-\gamma(s_m,\cdot))$

for some general $t \in S_m$:

$$\begin{aligned}
P(\gamma(t,\cdot)-\gamma(s_m,\cdot)) &= \mathbb{E}\left[(t(X)-Y)^2-(s_m(X)-Y)^2\right] \\
&= \mathbb{E}\left[(t(X)-s_m(X))^2\right]+2\mathbb{E}\left[(t(X)-s_m(X))(s_m(X)-s(X))\right] \\
&= \mathbb{E}\left[(t(X)-s_m(X))^2\right] \\
&= \sum_{\lambda\in\Lambda_m} p_\lambda (t_\lambda-\beta_\lambda)^2
\end{aligned}$$

since for every $\lambda \in \Lambda_m, \mathbb{E}[s(X) \mid X\in I_\lambda] = \beta_\lambda$.

A.9.3 COMPUTATION OF $P_n(\gamma(t,\cdot)-\gamma(s_m,\cdot))$

for some general $t \in S_m$: with $\eta_i = Y_i - s_m(X_i)$, we have

$$\begin{aligned}
P_n(\gamma(t,\cdot)-\gamma(s_m,\cdot)) &= \frac{1}{n}\sum_{i=1}^n \left[(t(X_i)-Y_i)^2-(u(X_i)-Y_i)^2\right] \\
&= \frac{1}{n}\sum_{i=1}^n (t(X_i)-u(X_i))^2 - \frac{2}{n}\sum_{i=1}^n \left[(t(X_i)-u(X_i))\eta_i\right] \\
&= \frac{1}{n}\sum_{i=1}^n \sum_{\lambda\in\Lambda_m} (t_\lambda-u_\lambda)^2 \mathbb{1}_{X_i\in I_\lambda} - \frac{2}{n}\sum_{i=1}^n \sum_{\lambda\in\Lambda_m} (t_\lambda-u_\lambda)\mathbb{1}_{X_i\in I_\lambda}\eta_i \quad .
\end{aligned}$$

A.9.4 BACK TO $(P_n - P)$

We sum the two inequalities above and use the triangular inequality:

$$\begin{aligned}
\left|(P_n-P)(\gamma(t,\cdot)-\gamma(u,\cdot))\right| &\leq \left|\frac{1}{n}\sum_{i=1}^n\sum_{\lambda\in\Lambda_m}(t_\lambda-u_\lambda)^2(\mathbb{1}_{X_i\in I_\lambda}-p_\lambda)\right| \\
&\quad + \left|\frac{2}{n}\sum_{i=1}^n\sum_{\lambda\in\Lambda_m}(t_\lambda-u_\lambda)\mathbb{1}_{X_i\in I_\lambda}\eta_i\right| \\
&\leq \frac{2A}{n}\sum_{\lambda\in\Lambda_m}\left[\left(\sqrt{p_\lambda}\,|t_\lambda-u_\lambda|\right)\frac{\left|\sum_{i=1}^n(\mathbb{1}_{X_i\in I_\lambda}-p_\lambda)\right|}{\sqrt{p_\lambda}}\right] \\
&\quad + \frac{2}{n}\sum_{\lambda\in\Lambda_m}\left[\left(\sqrt{p_\lambda}\,|t_\lambda-u_\lambda|\right)\frac{\left|\sum_{i=1}^n\mathbb{1}_{X_i\in I_\lambda}\eta_i\right|}{\sqrt{p_\lambda}}\right]
\end{aligned}$$

since $|t_\lambda-u_\lambda|\leq 2A$ for every $t\in S_m$.

We now assume that $d(u,t)\leq\sigma$ for some $\sigma>0$, that is

$$d(u,t)^2 = 2\sum_{\lambda\in\Lambda_m} p_\lambda(t_\lambda-u_\lambda)^2 \leq \sigma^2 \quad .$$

From Cauchy-Schwarz inequality, we obtain for every $t \in S_m$ such that $d(u,t) \leq \sigma$

$$|(P_n - P)(\gamma(t,\cdot) - \gamma(u,\cdot))| \leq \frac{2A\sigma}{\sqrt{2}n} \sqrt{\sum_{\lambda \in \Lambda_m} \frac{\left(\sum_{i=1}^n (\mathbb{1}_{X_i \in I_\lambda} - p_\lambda)\right)^2}{p_\lambda}} + \frac{\sqrt{2}\sigma}{n} \sqrt{\sum_{\lambda \in \Lambda_m} \frac{\left(\sum_{i=1}^n \mathbb{1}_{X_i \in I_\lambda} \eta_i\right)^2}{p_\lambda}}$$

A.9.5 BACK TO ψ

The upper bound above does not depend on t, so that the left-hand side of the inequality can be replaced by a supremum over $\{t \in S_m \text{ s.t. } d(u,t) \leq \sigma\}$. Taking expectations and using Jensen's inequality ($\sqrt{\cdot}$ being concave), we obtain an upper bound on ψ:

$$\psi(\sigma) \leq \frac{2A\sigma}{\sqrt{2}n} \sqrt{\sum_{\lambda \in \Lambda_m} \mathbb{E}\left[\frac{\left(\sum_{i=1}^n (\mathbb{1}_{X_i \in I_\lambda} - p_\lambda)\right)^2}{p_\lambda}\right]} + \frac{\sqrt{2}\sigma}{n} \sqrt{\sum_{\lambda \in \Lambda_m} \mathbb{E}\left[\frac{\left(\sum_{i=1}^n \mathbb{1}_{X_i \in I_\lambda} \eta_i\right)^2}{p_\lambda}\right]} \tag{39}$$

For every $\lambda \in \Lambda_m$, we have

$$\mathbb{E}\left(\sum_{i=1}^n (\mathbb{1}_{X_i \in I_\lambda} - p_\lambda)\right)^2 = \sum_{i=1}^n \mathbb{E}\left(\mathbb{1}_{X_i \in I_\lambda} - p_\lambda\right)^2 = np_\lambda(1 - p_\lambda) \tag{40}$$

which simplifies the first term. For the second term, notice that

$$\begin{aligned} \forall i \neq j, \qquad & \mathbb{E}\left[\mathbb{1}_{X_i \in I_\lambda} \mathbb{1}_{X_j \in I_\lambda} \eta_i \eta_j\right] = \mathbb{E}\left[\mathbb{1}_{X_i \in I_\lambda} \eta_i\right] \mathbb{E}\left[\mathbb{1}_{X_j \in I_\lambda} \eta_j\right] \\ \text{and} \quad \forall i, \qquad & \mathbb{E}\left[\mathbb{1}_{X_i \in I_\lambda} \eta_i\right] = \mathbb{E}\left[\mathbb{1}_{X_i \in I_\lambda} \mathbb{E}\left[\eta_i \mid \mathbb{1}_{X_i \in I_\lambda}\right]\right] = 0 \end{aligned}$$

since η_i is centered conditionally to $\mathbb{1}_{X_i \in I_\lambda}$. Then,

$$\mathbb{E}\left(\sum_{i=1}^n \mathbb{1}_{X_i \in I_\lambda} \eta_i\right)^2 = \sum_{i=1}^n \mathbb{E}\left[\mathbb{1}_{X_i \in I_\lambda} \eta_i^2\right] \leq np_\lambda \|\eta\|_\infty^2 \leq np_\lambda (2A)^2 \ . \tag{41}$$

Combining (39) with (40) and (41), we deduce that

$$\psi(\sigma) \leq \frac{2A\sigma}{\sqrt{2}\sqrt{n}} \sqrt{D_m - 1} + \frac{2\sqrt{2}A\sigma}{\sqrt{n}} \sqrt{D_m} \leq 3A\sqrt{2} \frac{\sqrt{D_m}}{\sqrt{n}} \times \sigma \ .$$

As already noticed, we have to multiply this bound by 2 so that it is valid for every $u \in S_m$ and not only $u = s_m$.

The resulting upper bound (multiplied by $\sqrt{n}$) has all the desired properties for ϕ_m since $6A\sqrt{2}\sqrt{D_m} = 3\sqrt{2D_m} \geq 1$. The result follows. ■

References

Hirotugu Akaike. Statistical predictor identification. *Ann. Inst. Statist. Math.*, 22:203–217, 1970.

Hirotugu Akaike. Information theory and an extension of the maximum likelihood principle. In *Second International Symposium on Information Theory (Tsahkadsor, 1971)*, pages 267–281. Akadémiai Kiadó, Budapest, 1973.

David M. Allen. The relationship between variable selection and data augmentation and a method for prediction. *Technometrics*, 16:125–127, 1974.

Sylvain Arlot. *Resampling and Model Selection*. PhD thesis, University Paris-Sud 11, December 2007. oai:tel.archives-ouvertes.fr:tel-00198803_v1.

Sylvain Arlot. Suboptimality of penalties proportional to the dimension for model selection in heteroscedastic regression, December 2008a. arXiv:0812.3141.

Sylvain Arlot. V-fold cross-validation improved: V-fold penalization, February 2008b. arXiv:0802.0566v2.

Sylvain Arlot. Model selection by resampling penalization, March 2008c. oai:hal.archives-ouvertes.fr:hal-00262478_v1.

Yannick Baraud. Model selection for regression on a fixed design. *Probab. Theory Related Fields*, 117(4):467–493, 2000.

Yannick Baraud. Model selection for regression on a random design. *ESAIM Probab. Statist.*, 6: 127–146 (electronic), 2002.

Yannick Baraud, Christophe Giraud, and Sylvie Huet. Gaussian model selection with unknown variance, 2007. To appear in The Annals of Statistics. arXiv:math.ST/0701250.

Andrew Barron, Lucien Birgé, and Pascal Massart. Risk bounds for model selection via penalization. *Probab. Theory Related Fields*, 113(3):301–413, 1999.

Peter L. Bartlett, Stéphane Boucheron, and Gábor Lugosi. Model selection and error estimation. *Machine Learning*, 48:85–113, 2002.

Peter L. Bartlett, Olivier Bousquet, and Shahar Mendelson. Local Rademacher complexities. *Ann. Statist.*, 33(4):1497–1537, 2005.

Jean-Patrick Baudry. Clustering through model selection criteria, June 2007. Poster session at One Day Statistical Workshop in Lisieux. `http://www.math.u-psud.fr/∼baudry`.

Lucien Birgé and Pascal Massart. Gaussian model selection. *J. Eur. Math. Soc. (JEMS)*, 3(3): 203–268, 2001.

Lucien Birgé and Pascal Massart. Minimal penalties for Gaussian model selection. *Probab. Theory Related Fields*, 138(1-2):33–73, 2007.

Stéphane Boucheron and Pascal Massart. A poor man's wilks phenomenon, March 2008. Personal communication.

Prabir Burman. Estimation of equifrequency histograms. *Statist. Probab. Lett.*, 56(3):227–238, 2002.

Imre Csiszár. Large-scale typicality of Markov sample paths and consistency of MDL order estimators. *IEEE Trans. Inform. Theory*, 48(6):1616–1628, 2002.

Imre Csiszár and Paul C. Shields. The consistency of the BIC Markov order estimator. *Ann. Statist.*, 28(6):1601–1619, 2000.

Luc Devroye and Gábor Lugosi. *Combinatorial Methods in Density Estimation*. Springer Series in Statistics. Springer-Verlag, New York, 2001.

Bradley Efron. Estimating the error rate of a prediction rule: improvement on cross-validation. *J. Amer. Statist. Assoc.*, 78(382):316–331, 1983.

Seymour Geisser. The predictive sample reuse method with applications. *J. Amer. Statist. Assoc.*, 70:320–328, 1975.

Edward I. George and Dean P. Foster. Calibration and empirical Bayes variable selection. *Biometrika*, 87(4):731–747, 2000.

Clifford M. Hurvich and Chih-Ling Tsai. Regression and time series model selection in small samples. *Biometrika*, 76(2):297–307, 1989.

Vladimir Koltchinskii. Rademacher penalties and structural risk minimization. *IEEE Trans. Inform. Theory*, 47(5):1902–1914, 2001.

Vladimir Koltchinskii. Local Rademacher complexities and oracle inequalities in risk minimization. *Ann. Statist.*, 34(6):2593–2656, 2006.

Marc Lavielle. Using penalized contrasts for the change-point problem. *Signal Proces.*, 85(8): 1501–1510, 2005.

Émilie Lebarbier. Detecting multiple change-points in the mean of a Gaussian process by model selection. *Signal Proces.*, 85:717–736, 2005.

Guillaume Lecué. *Méthodes d'Agrégation : Optimalité et Vitesses Rapides*. PhD thesis, LPMA, University Paris VII, May 2007.

Vincent Lepez. *Some Estimation Problems Related to Oil Reserves*. PhD thesis, University Paris XI, 2002.

Ker-Chau Li. Asymptotic optimality for C_p, C_L, cross-validation and generalized cross-validation: discrete index set. *Ann. Statist.*, 15(3):958–975, 1987.

Fernando Lozano. Model selection using Rademacher penalization. In *Proceedings of the 2nd ICSC Symp. on Neural Computation (NC2000). Berlin, Germany*. ICSC Academic Press, 2000.

Colin L. Mallows. Some comments on C_p. *Technometrics*, 15:661–675, 1973.

Pascal Massart. *Concentration Inequalities and Model Selection*, volume 1896 of *Lecture Notes in Mathematics*. Springer, Berlin, 2007.

Cathy Maugis and Bertrand Michel. A non asymptotic penalized criterion for gaussian mixture model selection. Technical Report 6549, INRIA, 2008.

Boris T. Polyak and Alexandre B. Tsybakov. Asymptotic optimality of the C_p-test in the projection estimation of a regression. *Teor. Veroyatnost. i Primenen.*, 35(2):305–317, 1990.

Xiaotong Shen and Jianming Ye. Adaptive model selection. *J. Amer. Statist. Assoc.*, 97(457):210–221, 2002.

Ritei Shibata. An optimal selection of regression variables. *Biometrika*, 68(1):45–54, 1981.

Charles J. Stone. An asymptotically optimal histogram selection rule. In *Proceedings of the Berkeley Conference in Honor of Jerzy Neyman and Jack Kiefer, Vol. II (Berkeley, Calif., 1983)*, Wadsworth Statist./Probab. Ser., pages 513–520, Belmont, CA, 1985. Wadsworth.

M. Stone. Cross-validatory choice and assessment of statistical predictions. *J. Roy. Statist. Soc. Ser. B*, 36:111–147, 1974.

Nariaki Sugiura. Further analysis of the data by Akaike's information criterion and the finite corrections. *Comm. Statist. A—Theory Methods*, 7(1):13–26, 1978.

Alexandre B. Tsybakov. Optimal aggregation of classifiers in statistical learning. *Ann. Statist.*, 32 (1):135–166, 2004.

Nicolas Verzelen. *Gaussian Graphical Models and Model Selection*. PhD thesis, University Paris XI, December 2008.

Fanny Villers. *Tests et Sélection de Modèles pour l'Analyse de Données Protéomiques et Transcriptomiques*. PhD thesis, University Paris XI, December 2007.

Journal of Machine Learning Research 10 (2009) 281-299 Submitted 12/07; 12/08; Published 2/09

Analysis of Perceptron-Based Active Learning

Sanjoy Dasgupta DASGUPTA@CS.UCSD.EDU
Department of Computer Science and Engineering
University of California, San Diego
La Jolla, CA 92093-0404, USA

Adam Tauman Kalai ADUM@MICROSOFT.COM
Microsoft Research
Office 14063
One Memorial Drive
Cambridge, MA 02142, USA

Claire Monteleoni CMONTEL@CCLS.COLUMBIA.EDU
Center for Computational Learning Systems
Columbia University
Suite 850, 475 Riverside Drive, MC 7717
New York, NY 10115, USA

Editor: Manfred Warmuth

Abstract

We start by showing that in an active learning setting, the Perceptron algorithm needs $\Omega(\frac{1}{\varepsilon^2})$ labels to learn linear separators within generalization error ε. We then present a simple active learning algorithm for this problem, which combines a modification of the Perceptron update with an adaptive filtering rule for deciding which points to query. For data distributed uniformly over the unit sphere, we show that our algorithm reaches generalization error ε after asking for just $\tilde{O}(d \log \frac{1}{\varepsilon})$ labels. This exponential improvement over the usual sample complexity of supervised learning had previously been demonstrated only for the computationally more complex query-by-committee algorithm.

Keywords: active learning, perceptron, label complexity bounds, online learning

1. Introduction

In many machine learning applications, unlabeled data is abundant but labeling is expensive. This distinction is not captured in standard models of supervised learning, and has motivated the field of *active learning*, in which the labels of data points are initially hidden, and the learner must pay for each label it wishes revealed. If query points are chosen randomly, the number of labels needed to reach a target generalization error ε, at a target confidence level $1-\delta$, is similar to the sample complexity of supervised learning. The hope is that there are alternative querying strategies which require significantly fewer labels.

An early dramatic demonstration of the potential of active learning was Freund et al.'s analysis of the query-by-committee (QBC) learning algorithm (Freund et al., 1997). The analysis is with respect to the *selective sampling* model: the learner observes a stream of unlabeled data and makes spot decisions about whether or not to ask for each point's label. The paper showed that if the data is drawn uniformly from the surface of the unit sphere in $\mathbb{R}^d$, and the hidden labels correspond perfectly to a homogeneous (i.e., through the origin) linear separator from this same distribution, then it is possible to achieve generalization error ε after seeing $\tilde{O}(\frac{d}{\varepsilon}\log\frac{1}{\varepsilon})$ points and requesting just $\tilde{O}(d\log\frac{1}{\varepsilon})$ labels:[1] an exponential improvement over the usual $\tilde{O}(\frac{d}{\varepsilon})$ sample complexity of learning linear separators in a supervised setting. (An $\Omega(d\log\frac{1}{\varepsilon})$ label complexity can be seen to be optimal by counting the number of spherical caps of radius ε that can be packed onto the surface of the unit sphere in $\mathbb{R}^d$.) This remarkable result is tempered somewhat by the complexity of the QBC algorithm, which involves random sampling from intermediate version spaces; the complexity of the update step scales (polynomially) with the number of updates performed.

In this paper, we show how a simple modification of the perceptron update can be used to achieve the same sample complexity bounds (within $\tilde{O}$ factors), under the same streaming model and the same uniform input distribution. Unlike QBC, we do not assume a distribution over target hypotheses, and our algorithm does not need to store previously seen data points, only its current hypothesis. Moreover, in addition to requiring only one-at-a-time access to examples (as opposed to batch data access), neither our algorithm's memory usage, nor its computation time per example, scales with the number of seen examples.[2]

Our algorithm has the following structure.

```
Set initial hypothesis $v_0 \in \mathbb{R}^d$.
For $t = 0, 1, 2, \ldots$.
  Receive unlabeled point $x_t$.
  Make a prediction SGN($v_t \cdot x_t$).
  *Filtering step:* Decide whether to ask for $x_t$'s label.
  If label $y_t$ is requested:
    *Update step:* Set $v_{t+1}$ based on $v_t, x_t, y_t$.
    Adjust filtering rule.
  else:   $v_{t+1} = v_t$.
```

UPDATE STEP.

The regular perceptron update, whose convergence behavior was first analyzed by Rosenblatt (1958), consists of the following simple rule:

$$\text{if } (x_t, y_t) \text{ is misclassified, then } v_{t+1} = v_t + y_t x_t.$$

It turns out that this update cannot yield an error rate better than $\Omega(1/\sqrt{l_t})$, where l_t is the number of labels queried up to time t, no matter what filtering scheme is used.

Theorem 1 *Consider any sequence of data points $x_0, x_1, x_2, \ldots \in \mathbb{R}^d$ which is perfectly classified by some linear separator $u \in \mathbb{R}^d$. Suppose that perceptron updates are used, starting with an initial hypothesis v_0. Let k_t be the number of updates performed upto time t, let v_t be the hypothesis at time t, and let θ_t be the angle between u and v_t. Then for any $t \geq 0$, if $\theta_{t+1} \leq \theta_t$ then $\sin\theta_t \geq 1/(5\sqrt{k_t + \|v_0\|^2})$.*

This holds regardless of how the data is produced. When the points are distributed uniformly over the unit sphere, $\theta_t \geq \sin\theta_t$ (for $\theta_t \leq \frac{\pi}{2}$) is proportional to the error rate of v_t. In other words, the

1. In this paper, the $\tilde{O}$ notation is used to suppress multiplicative terms in $\log d, \log\log\frac{1}{\varepsilon}$ and $\log\frac{1}{\delta}$.
2. See Monteleoni (2006) for an extended discussion of learning with online constraints.

error rate is $\Omega(1/\sqrt{k_t})$, which in turn is $\Omega(1/\sqrt{l_t})$, since each update must be triggered by a label. As we will shortly see, the reason for this slow rate is that the magnitude of the perceptron update is too large for points near the decision boundary of the current hypothesis.

So instead we use a *variant* of the update rule, originally due to Motzkin and Schoenberg (1954):

$$\text{if } (x_t, y_t) \text{ is misclassified, then } v_{t+1} = v_t - 2(v_t \cdot x_t)x_t$$

(where x_t is assumed normalized to unit length). Note that the update can also be written as $v_{t+1} = v_t + 2y_t|v_t \cdot x_t|x_t$, since updates are only made on mistakes, in which case $y_t \neq \texttt{SGN}(v_t \cdot x_t)$, by definition. Thus we are scaling the standard perceptron's additive update by a factor of $2|v_t \cdot x_t|$ to avoid oscillations caused by points close to the half-space represented by the current hypothesis. Motzkin and Schoenberg (1954) introduced this rule, in the context of solving linear inequalities, and called it the "Reflexion" method, due to certain geometric properties it has, which we will discuss later. Hampson and Kibler (1999) subsequently applied it to learning linear separators, in an analysis framework that differs from ours. The same rule, but without the factor of two, has been used in previous work (Blum et al., 1996) on learning linear classifiers from noisy data, in a batch setting. We are able to show that our formulation has the following generalization performance in a supervised (non-active) setting.

Theorem 2 *Pick any* $\delta, \varepsilon > 0$. *Consider a stream of data points* x_t *drawn uniformly at random from the surface of the unit sphere in* $\mathbb{R}^d$, *and corresponding labels* y_t *that are consistent with some linear separator. When the modified Perceptron algorithm (Figure 2) is applied to this stream of data, then with probability* $1 - \delta$, *after* $O(d(\log \frac{1}{\varepsilon} + \log \frac{1}{\delta}))$ *mistakes, its generalization error is at most* ε.

This contrasts favorably with the $\tilde{O}(\frac{d}{\varepsilon^2})$ mistake bound of the Perceptron algorithm, and a more recent variant, on the same distribution (Baum, 1997; Servedio, 1999). Meanwhile, in terms of lower bounds, Theorem 1 also applies in the supervised case, and gives a lower bound on the number of mistakes (updates) made by the standard perceptron. Finally, there is the question of how many samples are needed in the supervised setting (as opposed to the number of mistakes). For data distributed uniformly over the unit sphere, this is known to be $\tilde{\Theta}(\frac{d}{\varepsilon})$ (lower bound, Long, 1995, and upper bound, Long, 2003).

FILTERING STEP.

Given the limited information the algorithm keeps, a natural filtering rule is to query points x_t when $|v_t \cdot x_t|$ is less than some threshold s_t. The choice of s_t is crucial. If it is too large, then only a miniscule fraction of the points queried will actually be misclassified (and thus trigger updates)—almost all labels will be wasted. On the other hand, if s_t is too small, then the waiting time for a query might be prohibitive, and when an update is actually made, the magnitude of this update might be tiny.

Therefore, we set the threshold adaptively: we start s_t high, and keep dividing it by two until we reach a level where there are enough misclassifications amongst the points queried. By wrapping this filtering strategy around the modified Perceptron update, we get an active learning algorithm (Figure 4) with the following label complexity guarantee.

Theorem 3 *Pick any* $\delta, \varepsilon > 0$. *Consider a stream of data points* x_t *drawn uniformly at random from the surface of the unit sphere in* $\mathbb{R}^d$, *and corresponding labels* y_t *that are consistent with some linear* *separator. With probability* $1-\delta$*, if the active modified Perceptron algorithm (Figure 4) is given a stream of* $\tilde{O}(\frac{d}{\varepsilon}\log\frac{1}{\varepsilon})$ *such unlabeled points, it will request* $\tilde{O}(d\log\frac{1}{\varepsilon})$ *labels, make* $\tilde{O}(d\log\frac{1}{\varepsilon})$ *errors (on all points, labeled or not), and have final error* $\leq \varepsilon$.

The proofs of Theorems 1 through 3 are in Sections 4 through 6, respectively.

2. Related Work

Much of the early theory work on active learning was in the *query learning* model, in which the learner has the ability to synthesize arbitrary data points and request their labels. See Angluin (2001) for an excellent survey of this area. In this paper, we consider a different setting, in which (1) there is an underlying joint distribution over data points and labels, (2) the learner has access to (unlabeled) data points drawn at random from this distribution, and (3) the learner is able to request labels only for points obtained in this way, not for arbitrary points. This framework for active learning was originally introduced by Cohn, Atlas, and Ladner (1994),[3] along with a simple and elegant querying algorithm. Unless we specify otherwise, we will use the term selective sampling to denote the framework. In this work, we focus on the *realizable* setting: the hypothesis class which we consider for learning contains a classifier with zero error on the data distribution.[4] Our contribution to active learning is for online learning of linear separators through the origin, under the uniform distribution.

Several methods for learning linear separators (or their probabilistic analogues) in the selective sampling framework, have been proposed in the literature. Some have been shown to work reasonably well in practice, for example Lewis and Gale's sequential algorithm for text classification (Lewis and Gale, 1994), which has batch access to the remaining unlabeled data points at each iteration. Several of these are similar in spirit to our approach, in that they query points with small margins, such as Tong and Koller's active learning algorithms that use a support vector machine (SVM) as the underlying classifier (Tong and Koller, 2001).

On the theoretical side, there have been some encouraging upper bounds on label complexity; however, some of the schemes achieving them have not yet been proven efficient. Dasgupta (2005) provided a result for learning general hypothesis classes, in a non-Bayesian, realizable setting. For homogeneous half-spaces with data distributed uniformly on the sphere, this result implies an upper bound on label complexity of $\tilde{O}(d\log^2(\frac{1}{\varepsilon}))$. Balcan, Beygelzimer, and Langford (2006) provided a technique for learning general hypothesis classes, in a non-Bayesian, agnostic setting, for which they showed a label complexity upper bound of $\tilde{O}(d^2\log\frac{1}{\varepsilon})$ for learning linear separators under the uniform input distribution.[5] Both of these results rely on schemes that are computationally prohibitive, requiring exponential storage and/or computation.

The literature contains several active learning algorithms that are both feasible to implement (at least in special cases) and have label complexity guarantees, although none of them is quite as simple as the algorithm we present in this paper. We have already discussed the label complexity upper bound attained by Freund et al. (1997) for the Query By Committee algorithm of Seung et al. (1992). More recently, it was shown how to efficiently implement this scheme for linear separators under certain prior distributions, and the empirical results were encouraging (Gilad-Bachrach et al.,

3. The conference version dates back to NIPS 1989, with a superset of the coauthors.
4. The *agnostic* setting removes this assumption.
5. This bound was further tightened to $\tilde{O}(d^{1.5}\log\frac{1}{\varepsilon})$, in Balcan et al. (2007), and this scheme has also been analyzed by Hanneke (2007), who introduced a new label complexity measure.

2005). Cesa-Bianchi et al. (2003) provided regret bounds on a selective sampling algorithm for learning linear thresholds from a stream of iid examples corrupted by random class noise whose rate scales with the examples' margins. For half-spaces under the uniform input distribution, in the realizable setting, the algorithm of Balcan et al. (2006) can be implementated efficiently, as shown by Balcan et al. (2007), which analyzed various margin-based techniques for active learning, matching our label complexity bound in the same setting. Dasgupta, Hsu, and Monteleoni (2007) recently gave an active learning algorithm for general concept classes in the non-Bayesian, agnostic setting (a generalization of the original selective sampling algorithm of Cohn et al. 1994) which, for half-spaces under the uniform input distribution, in the realizable case, has a label complexity upper bound of $\tilde{O}(d^{1.5} \log \frac{1}{\varepsilon})$.

Cesa-Bianchi et al. (2004) analyzed an algorithm which conforms to roughly the same template as ours but differs in both the update and filtering rule—it uses the regular perceptron update and it queries points x_t according to a fixed, randomized rule which favors small $|v_t \cdot x_t|$. The authors make no distributional assumptions on the input and they show that in terms of worst-case hinge-loss bounds, their algorithm does about as well as one which queries *all* labels. The actual fraction of points queried varies from data set to data set. In contrast, our objective is to achieve a target generalization error with minimum label complexity, although we also obtain a mistake bound (on both labeled and unlabeled points) under our distributional assumption.

It is known that active learning does not always give a large improvement in the sample complexity of learning linear separators. For instance, in our setting, in which data is distributed uniformly over the unit sphere, Dasgupta (2004) showed that if the target linear separator is allowed to be non-homogeneous, then the number of labels required to reach error ε is $\Omega(\frac{1}{\varepsilon})$, no matter what active learning scheme is used. This lower bound also applies to learning homogeneous linear separators with respect to an arbitrary distribution. In the fully agnostic setting, Kääriäinen (2006) provided a lower bound of $\Omega(\frac{\eta^2}{\varepsilon^2})$, where η is the error rate of the best hypothesis in the concept class.

3. Preliminaries

In our model, all data x_t lie on the surface of the unit ball in $\mathbb{R}^d$, which we denote by S:

$$S = \left\{ x \in \mathbb{R}^d \;\middle|\; \|x\| = 1 \right\}.$$

Their labels y_t are either -1 or $+1$, and the target function is a half-space $u \cdot x \geq 0$ represented by a unit vector $u \in \mathbb{R}^d$ which classifies all points perfectly, that is, $y_t(u \cdot x_t) > 0$ for all t, with probability one.

For any vector $v \in \mathbb{R}^d$, we define $\hat{v} = \frac{v}{\|v\|}$ to be the corresponding unit vector.

Our lower bound (Theorem 1) is distribution-free; thereafter we will assume that the data points x_t are drawn independently from the uniform distribution over S.

Under the uniform input distribution, any hypothesis $v \in \mathbb{R}^d$ has error

$$\varepsilon(v) = P_{x \in S}[\texttt{SGN}(v \cdot x) \neq \texttt{SGN}(u \cdot x)] = \frac{\arccos(u \cdot \hat{v})}{\pi}.$$

We will refer to the error rate of a hypothesis v as its *generalization error*, since in the realizable case the target itself has zero error.

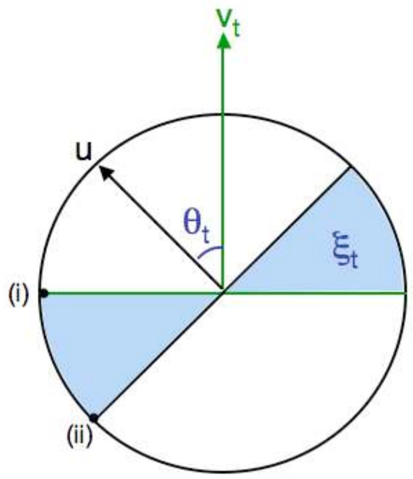


Figure 1: The projection of the error region ξ_t onto the plane defined by u and v_t.

For a hypothesis v_t, we will denote the angle between u and v_t by θ_t, and we will define the error region of v_t as $\xi_t = \{x \in S \,|\, \mathtt{SGN}(v_t \cdot x) \neq \mathtt{SGN}(u \cdot x)\}$. Figure 1 provides a schematic of the projection of the error region onto the plane defined by u and v_t.

We will use the term *margin*, in the context of learning half-spaces, to denote simply the distance from an example to the separator in question, as opposed to the standard use of this term (as the minimum over examples of this distance with respect to the target separator). For example, we will denote the margin of x with respect to v as $|x \cdot v|$.

We will use a few useful inequalities for θ on the interval $(0, \frac{\pi}{2}]$.

$$\frac{4}{\pi^2} \leq \frac{1-\cos\theta}{\theta^2} \leq \frac{1}{2}, \tag{1}$$

$$\frac{2}{\pi}\theta \leq \sin\theta \leq \theta. \tag{2}$$

Equation (1) can be verified by checking that for θ in this interval, $\frac{1-\cos\theta}{\theta^2}$ is a decreasing function, and evaluating it at the endpoints.

We will also make use of the following lemma.

Lemma 4 *For any fixed unit vector a and any $\gamma \leq 1$,*

$$\frac{\gamma}{4} \leq P_{x \in S}\left[|a \cdot x| \leq \frac{\gamma}{\sqrt{d}}\right] \leq \gamma.$$

The proof is deferred to the appendix.

4. A Lower Bound for the Perceptron Update

Consider an algorithm of the following form:

```
Pick some $v_0 \in \mathbb{R}^d$.
Repeat for $t = 0, 1, 2, \ldots$:
  Get some $(x, y)$ for which $y(v_t \cdot x) \le 0$.
  $v_{t+1} = v_t + yx$
```

On any update,

$$v_{t+1} \cdot u = v_t \cdot u + y(x \cdot u). \tag{3}$$

Thus, if we assume for simplicity that $v_0 \cdot u \geq 0$ (we can always just start count when this first occurs) then $v_t \cdot u \geq 0$ always, and θ_t, the angle between u and v_t is always acute. Since $\|u\| = 1$, the following holds:

$$\|v_t\| \cos\theta_t = v_t \cdot u.$$

The update rule also implies

$$\|v_{t+1}\|^2 = \|v_t\|^2 + 1 + 2y(v_t \cdot x). \tag{4}$$

Thus $\|v_t\|^2 \leq t + \|v_0\|^2$ for all t. In particular, this means that Theorem 1 is an immediate consequence of the following lemma.

Lemma 5 *Assume $v_0 \cdot u \geq 0$ (i.e., start count when this first occurs). Then*

$$\theta_{t+1} \leq \theta_t \;\;\Rightarrow\;\; \sin\theta_t \geq \min\left\{\frac{1}{3}, \frac{1}{5\|v_t\|}\right\}.$$

Proof Figure 1 shows the unit circle in the plane defined by u and v_t. The dot product of any point $x \in \mathbb{R}^d$ with either u or v_t depends only upon the projection of x into this plane. The point is misclassified when its projection lies in the shaded region. For such points, $y(u \cdot x)$ is at most $\sin\theta_t$ (point (i)) and $y(v_t \cdot x)$ is at least $-\|v_t\| \sin\theta_t$ (point (ii)).

Combining this with Equations (3) and (4), we get

$$\begin{array}{rcl} v_{t+1} \cdot u & \leq & v_t \cdot u + \sin\theta_t, \\ \|v_{t+1}\|^2 & \geq & \|v_t\|^2 + 1 - 2\|v_t\| \sin\theta_t. \end{array}$$

To establish the lemma, we first assume $\theta_{t+1} \leq \theta_t$ and $\sin\theta_t \leq \frac{1}{5\|v_t\|}$, and then conclude that $\sin\theta_t \geq \frac{1}{3}$.

$\theta_{t+1} \leq \theta_t$ implies

$$\cos^2\theta_t \;\leq\; \cos^2\theta_{t+1} \;=\; \frac{(u \cdot v_{t+1})^2}{\|v_{t+1}\|^2} \;\leq\; \frac{(u \cdot v_t + \sin\theta_t)^2}{\|v_t\|^2 + 1 - 2\|v_t\| \sin\theta_t}.$$

The final denominator is positive since $\sin\theta_t \leq \frac{1}{5\|v_t\|}$. Rearranging,

$$(\|v_t\|^2 + 1 - 2\|v_t\| \sin\theta_t) \cos^2\theta_t \;\leq\; (u \cdot v_t)^2 + \sin^2\theta_t + 2(u \cdot v_t) \sin\theta_t,$$

and using $\|v_t\| \cos\theta_t = (u \cdot v_t)$:

$$(1 - 2\|v_t\| \sin\theta_t) \cos^2\theta_t \;\leq\; \sin^2\theta_t + 2\|v_t\| \sin\theta_t \cos\theta_t.$$

```
Inputs: dimensionality $d$ and budget on number of updates
(mistakes) $M$.
  Let $v_1 = x_1 y_1$ for the first example $(x_1, y_1)$.
  For $t = 1$ to $M$:
    Let $(x_t, y_t)$ be the next example with $y(x \cdot v_t) < 0$.
    $v_{t+1} = v_t - 2(v_t \cdot x_t) x_t$
```

Figure 2: The (non-active) modified Perceptron algorithm. The standard Perceptron update, $v_{t+1} = v_t + y_t x_t$, is in the same direction (note $y_t = -\texttt{SGN}(v_t \cdot x_t)$) but different magnitude (scaled by a factor of $2|v_t \cdot x_t|$).

Again, since $\sin\theta_t \le \frac{1}{5\|v_t\|}$, it follows that $(1 - 2\|v_t\|\sin\theta_t) \ge \frac{3}{5}$ and that $2\|v_t\|\sin\theta_t\cos\theta_t \le \frac{2}{5}$. Using $\cos^2 = 1 - \sin^2$, we then get

$$\frac{3}{5}(1 - \sin^2\theta_t) \;\le\; \sin^2\theta_t + \frac{2}{5},$$

which works out to $\sin^2\theta_t \ge \frac{1}{8}$, implying $\sin\theta_t > \frac{1}{3}$. ■

The problem is that the perceptron update can be too large. In $\mathbb{R}^2$ (e.g., Figure 1), when θ_t is tiny, the update will cause v_{t+1} to overshoot the mark and swing too far to the other side of u, unless $\|v_t\|$ is very large: to be precise, we need $\|v_t\| = \Omega(\frac{1}{\sin\theta_t})$. But $\|v_t\|$ grows slowly, at best at a rate of $\sqrt{t}$. If $\sin\theta_t$ is proportional to the error of v_t, as in the case of data distributed uniformly over the unit sphere, this means that the perceptron update cannot stably maintain an error rate $\le \varepsilon$ until $t = \Omega(\frac{1}{\varepsilon^2})$.

5. The Modified Perceptron Update

We now describe a modified Perceptron algorithm. Unlike the standard Perceptron, it ensures that $v_t \cdot u$ is increasing, that is, the error of v_t is monotonically decreasing. Another difference from the standard update (and other versions) is that the magnitude of the current hypothesis, $\|v_t\|$, is always 1, which is convenient for the analysis.

The modified Perceptron algorithm is shown in Figure 2. We now show that the norm of v_t stays at one. Note that $\|v_1\| = 1$ and

$$\|v_{t+1}\|^2 = \|v_t\|^2 + 4(v_t \cdot x_t)^2\|x_t\|^2 - 4(v_t \cdot x_t)^2 = 1$$

by induction. In contrast, for the standard perceptron update, the magnitude of v_t increases steadily.

With the modified update, the error can only decrease, because $v_t \cdot u$ only increases:

$$v_{t+1} \cdot u = v_t \cdot u - 2(v_t \cdot x_t)(x_t \cdot u) = v_t \cdot u + 2|v_t \cdot x_t||x_t \cdot u|. \quad (5)$$

The second equality follows from the fact that v_t misclassified x_t. Thus $v_t \cdot u$ is increasing, and the increase can be bounded from below by showing that $|v_t \cdot x_t||x_t \cdot u|$ is large. This is a different approach from previous analyses.

Hampson and Kibler (1999) previously used this update for learning linear separators, calling it the "Reflection" method, based on the "Reflexion" method due to Motzkin and Schoenberg (1954). These names are likely due to the following geometric property of this update:

$$x_t \cdot v_{t+1} = x_t \cdot v_t - 2(x_t \cdot v_t)(x_t \cdot x_t) = -(x_t \cdot v_t).$$

In general, one can consider modified updates of the form $v_{t+1} = v_t - \alpha(v_t \cdot x_t)x_t$, which corresponds to the "Relaxation" method of solving linear inequalities (Agmon, 1954; Motzkin and Schoenberg, 1954). When $\alpha \neq 2$, the vectors v_t no longer remain of fixed length; however, one can verify that their corresponding unit vectors $\hat{v}_t$ satisfy

$$\hat{v}_{t+1} \cdot u = (\hat{v}_t \cdot u + \alpha|\hat{v}_t \cdot x_t||x_t \cdot u|)/\sqrt{1 - \alpha(2-\alpha)(\hat{v}_t \cdot x_t)^2},$$

and thus any choice of $\alpha \in [0,2]$ guarantees non-increasing error. Blum et al. (1996) used $\alpha = 1$ to guarantee progress in the denominator (their analysis did not rely on progress in the numerator) as long as $\hat{v}_t \cdot u$ and $(\hat{v}_t \cdot x_t)^2$ were bounded away from 0. Their approach was used in a batch setting as one piece of a more complex algorithm for noise-tolerant learning. In our sequential framework, we can bound $|\hat{v}_t \cdot x_t||x_t \cdot u|$ away from 0 in expectation, under the uniform distribution, and hence the choice of $\alpha = 2$ is most convenient, but $\alpha = 1$ would work as well. Although we do not further optimize our choice of the constant α, this choice itself may yield interesting future work, perhaps by allowing it to be a function of the dimension.

5.1 Analysis of (Non-Active) Modified Perceptron

How large do we expect $|v_t \cdot x_t|$ and $|u \cdot x_t|$ to be for an error (x_t, y_t)? As we shall see, in d dimensions, one expects each of these terms to be on the order of $d^{-1/2} \sin\theta_t$, where $\sin\theta_t = \sqrt{1-(v_t \cdot u)^2}$. Hence, we might expect their product to be about $(1-(v_t \cdot u)^2)/d$, which is how we prove the following lemma.

Note, we have made little effort to optimize constant factors.

Lemma 6 *For any v_t, with probability at least $\frac{1}{3}$,*

$$1 - v_{t+1} \cdot u \leq (1 - v_t \cdot u)\left(1 - \frac{1}{50d}\right).$$

There exists a constant $c > 0$, such that with probability at least $\frac{63}{64}$, for any v_t,

$$1 - v_{t+1} \cdot u \leq (1 - v_t \cdot u)\left(1 - \frac{c}{d}\right).$$

Proof We show only the first part of the lemma. The second part is quite similar. We will argue that each of $|v_t \cdot x_t|$,$|u \cdot x_t|$ is "small" with probability at most 1/3. This means, by the union bound, that with probability at least 1/3, they are both sufficiently large.

The error rate of v_t is θ_t/π, where $\cos\theta_t = v_t \cdot u$. Also define the error region $\xi_t = \{x \in S | \texttt{SGN}(v_t \cdot x) \neq \texttt{SGN}(u \cdot x)\}$. By Lemma 4, for an x drawn uniformly from the sphere,

$$P_{x \in S}\left[|v_t \cdot x| \leq \frac{\theta_t}{3\pi\sqrt{d}}\right] \leq \frac{\theta_t}{3\pi}.$$

Using $P[A|B] \le P[A]/P[B]$, we have,

$$P_{x\in S}\left[|v_t\cdot x| \le \frac{\theta_t}{3\pi\sqrt{d}} \;\middle|\; x\in\xi_t\right] \le \frac{P_{x\in S}[|v_t\cdot x|\le \frac{\theta_t}{3\pi\sqrt{d}}]}{P_{x\in S}[x\in\xi_t]} \le \frac{\theta_t/(3\pi)}{\theta_t/\pi} = \frac{1}{3}.$$

Similarly for $|u\cdot x|$, and by the union bound the probability that $x\in\xi_t$ is within margin $\frac{\theta}{3\pi\sqrt{d}}$ from either u or v is at most $\frac{2}{3}$. Since the updates only occur if x is in the error region, we now have a lower bound on the expected magnitude of $|v_t\cdot x||u\cdot x|$:

$$P_{x\in S}\left[|v_t\cdot x||u\cdot x| \ge \frac{\theta_t^2}{(3\pi\sqrt{d})^2}\middle| x\in\xi_t\right] \ge \frac{1}{3}.$$

Hence, we know that with probability at least 1/3, $|v_t\cdot x||u\cdot x| \ge \frac{1-(v_t\cdot u)^2}{100d}$, since $\theta_t^2 \ge \sin^2\theta_t = 1-(v_t\cdot u)^2$ and $(3\pi)^2 < 100$. In this case,

$$\begin{aligned} 1-v_{t+1}\cdot u &\le 1-v_t\cdot u - 2|v_t\cdot x_t||u\cdot x_t| \\ &\le 1-v_t\cdot u - \frac{1-(v_t\cdot u)^2}{50d} \\ &= (1-v_t\cdot u)\left(1-\frac{1+v_t\cdot u}{50d}\right), \end{aligned}$$

where the first inequality is by application of (5). ■

Finally, we give a high-probability bound, that is, Theorem 2, stated here with proof.

Theorem 7 *With probability* $1-\delta$ *with respect to the uniform distribution on the unit sphere, in the supervised, realizable setting, after* $M = O(d(\log\frac{1}{\varepsilon}+\log\frac{1}{\delta}))$ *mistakes, the generalization error of the modified Perceptron algorithm is at most* ε.

Proof By the above lemma, we can conclude that, for any vector v_t,

$$E[1-v_{t+1}\cdot u] \le (1-v_t\cdot u)\left(1-\frac{1}{3(50d)}\right).$$

This is because with $\ge 1/3$ probability it goes down by a factor of $1-\frac{1}{50d}$ and with the remaining $\le 2/3$ probability it does not increase. Hence, after M mistakes,

$$E[1-v_M\cdot u] \le (1-v_1\cdot u)\left(1-\frac{1}{150d}\right)^M \le \left(1-\frac{1}{150d}\right)^M,$$

since $v_1\cdot u \ge 0$. By Markov's inequality,

$$P\left[1-v_M\cdot u \ge \left(1-\frac{1}{150d}\right)^M \delta^{-1}\right] \le \delta.$$

Finally, using (1) and $\cos\theta_M = v_M\cdot u$, we see $P[\frac{4}{\pi^2}\theta_M^2 \ge (1-\frac{1}{150d})^M\delta^{-1}] \le \delta$. Using $M = 150d\log\frac{1}{\varepsilon\delta}$ gives $P[\frac{\theta_M}{\pi}\ge\varepsilon]\le\delta$, as required. ■

The additional factor of $\frac{1}{\varepsilon}$ in the bound on unlabeled samples ($\tilde{O}(\frac{d}{\varepsilon}\log\frac{1}{\varepsilon})$) follows by upper bounding the number of unlabeled samples until an update: when the hypothesis has error rate ε, the waiting time (in samples) until an update is $\frac{1}{\varepsilon}$, in expectation.

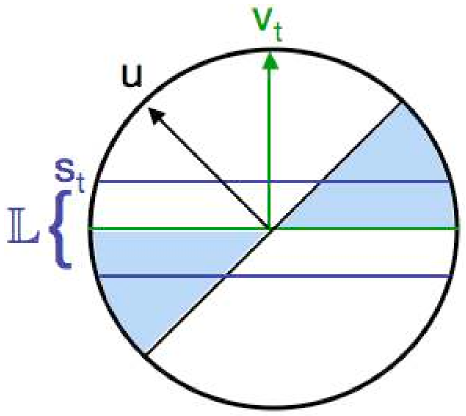


Figure 3: The active learning rule is to query for labels on points x in $\mathbb{L}$ which is defined by the threshold s_t on $|v_t \cdot x|$.

6. An Active Modified Perceptron

The ideal objective in designing an active learning rule that minimizes label complexity would be to query for labels only on points in the error region, ξ_t. However without knowledge of u, the algorithm is unaware of the location of ξ_t. The intuition behind our active learning rule is to approximate the error region, given the information the algorithm does have: v_t. As shown in Figure 3, the labeling region $\mathbb{L}$ is simply formed by thresholding the margin of a candidate example with respect to v_t.

The active version of the modified Perceptron algorithm is shown in Figure 4. The algorithm is similar to the algorithm of the previous section, in its update step. For its filtering rule, we maintain a threshold s_t and we only ask for labels of examples with $|v_t \cdot x| \leq s_t$. Approximating the error region is achieved by choosing the threshold, s_t, adaptively, so as to manage the tradeoff between $\mathbb{L}$ being too large, causing many labels to be wasted without hitting ξ_t (and thus yielding updates), and $\mathbb{L}$ only containing points with very small margins with respect to v_t, since our update step will make very small updates on such points. We decrease the threshold adaptively over time, starting at $s_1 = 1/\sqrt{d}$ and reducing it by a factor of two whenever we have a run of labeled examples on which we are correct.

For Theorem 3, we select values of R, L that yield ε error with probability at least $1-\delta$. The idea of the analysis is as follows:

Definition 7 *We say the tth update is "good" if,*

$$1 - v_{t+1} \cdot u \leq (1 - v_t \cdot u)\left(1 - \frac{c}{d}\right).$$

(The constant c is from Lemma 6.)

1. (Lemma 8) First, we argue that s_t is not too small (we do not decrease s_t too quickly). Assuming this is the case, then 2 and 3 hold.

2. (Lemma 10) We query for labels on at least an expected 1/32 of all *errors*. In other words, some errors may go undetected because we do not ask for their labels, but the number of

```
Inputs: Dimensionality $d$, maximum number of labels $L$,
and patience $R$.
  $v_1 = x_1 y_1$ for the first example $(x_1, y_1)$.
  $s_1 = 1/\sqrt{d}$
  **For** $t = 1$ **to** $L$:
    Wait for the next example $x$ : $|x \cdot v_t| \leq s_t$ and query its label.
    Call this labeled example $(x_t, y_t)$.
    **If** $(x_t \cdot v_t) y_t < 0$, **then**:
      $v_{t+1} = v_t - 2(v_t \cdot x_t) x_t$
      $s_{t+1} = s_t$
    **else**:
      $v_{t+1} = v_t$
      **If** predictions were correct on $R$ consecutive labeled
      examples (i.e., $(x_i \cdot v_i) y_i \geq 0 \ \forall i \in \{t-R+1, t-R+2, \ldots, t\}$),
      **then** set $s_{t+1} = s_t/2$, **else** $s_{t+1} = s_t$.
```

Figure 4: An active version of the modified Perceptron algorithm.

mistakes total should not be much more than 32 times the number of updates we actually perform.

3. (Lemma 11) Each update is *good* (Definition 7) with probability at least $1/2$.

4. (Theorem 3) Finally, we conclude that we cannot have too many label queries, updates, or total errors, because half of our updates are good, 1/32 of our errors are updates, and about $1/R$ of our labels are updates.

We first lower-bound s_t with respect to our error, showing that, with high probability, the threshold s_t is never too small.

Lemma 8 *With probability at least* $1 - L\left(\frac{3}{4}\right)^R$*, we have:*

$$s_t \geq \sqrt{\frac{1-(u \cdot v_t)^2}{16d}} \text{ for } t = 1, 2, \ldots, L, \text{ simultaneously.} \qquad (6)$$

Before proving this lemma, it will be helpful to show the following lemma. As before, let us define $\xi_t = \{x \in S | (x \cdot v_t)(x \cdot u) < 0\}$.

Lemma 9 *For any* $\gamma \in \left(0, \sqrt{\frac{1-(u \cdot v_t)^2}{4d}}\right]$,

$$P_{x_t \in S}\left[x_t \in \xi_t \;\middle|\; |x_t \cdot v_t| < \gamma\right] \geq \frac{1}{4}.$$

Proof Let x be a random example from S such that $|x \cdot v_t| < \gamma$ and, without loss of generality, suppose that $0 \leq x \cdot v_t \leq \gamma$. Then we want to calculate the probability we err, that is, $u \cdot x < 0$. We can decompose $x = x' + (x \cdot v_t)v_t$ where $x' = x - (x \cdot v_t)v_t$ is the component of x orthogonal to v_t, that is, $x' \cdot v_t = 0$. Similarly for $u' = u - (u \cdot v_t)v_t$. Hence,

$$u \cdot x = (u' + (u \cdot v_t)v_t) \cdot (x' + (x \cdot v_t)v_t) = u' \cdot x' + (u \cdot v_t)(x \cdot v_t).$$

In other words, we err iff $u' \cdot x' < -(u \cdot v_t)(x \cdot v_t)$. Using $u \cdot v_t \in [0,1]$ and since $x \cdot v_t \in [0, \sqrt{(1-(u \cdot v_t)^2)/(4d)}]$, we conclude that if

$$u' \cdot x' < -\sqrt{\frac{1-(u \cdot v_t)^2}{4d}}, \tag{7}$$

then we must err. Also, let $\hat{x}' = \frac{x'}{\|x'\|}$ be the unit vector in the direction of x'. It is straightforward to check that $\|x'\| = \sqrt{1-(x \cdot v_t)^2}$. Similarly, for u we define $\hat{u}' = \frac{u'}{\sqrt{1-(u \cdot v_t)^2}}$. Substituting these into (7), we must err if, $\hat{u}' \cdot \hat{x}' < -1/\sqrt{4d(1-(x \cdot v_t))^2}$, and since $\sqrt{1-(x \cdot v_t)^2} \geq \sqrt{1-1/(4d)}$, it suffices to show that,

$$P_{x \in S}\left[\hat{u}' \cdot \hat{x}' < \frac{-1}{\sqrt{4d(1-1/(4d))}} \;\middle|\; 0 \leq x \cdot v_t \leq \gamma\right] \geq \frac{1}{4}.$$

What is the probability that this happens? Well, one way to pick $x \in S$ would be to first pick $x \cdot v_t$ and then to pick $\hat{x}'$ uniformly at random from the set $S' = \{\hat{x}' \in S | \hat{x}' \cdot v_t = 0\}$, which is a unit sphere in one fewer dimensions. Hence the above probability does not depend on the conditioning. By Lemma 4, for any unit vector $a \in S'$, the probability that $|\hat{u}' \cdot a| \leq 1/\sqrt{4(d-1)}$ is at most $1/2$, so with probability at least $1/4$ (since the distribution is symmetric), the signed quantity $\hat{u}' \cdot \hat{x}' < -1/\sqrt{4(d-1)} < -1/\sqrt{4d(1-1/(4d))}$. ■

We are now ready to prove Lemma 8.

Proof [of Lemma 8] Suppose that condition (6) fails to hold for some t's. Let t be the smallest number such that (6) fails. By our choice of s_1, clearly $t > 1$. Moreover, since t is the smallest such number, and $u \cdot v_t$ is increasing, it must be the case that $s_t = s_{t-1}/2$, that is we just saw a run of R labeled examples (x_i, y_i), for $i = t-R, \ldots, t-1$, with no mistakes, $v_i = v_t$, and

$$s_i = 2s_t < \sqrt{\frac{1-(u \cdot v_t)^2}{4d}} = \sqrt{\frac{1-(u \cdot v_i)^2}{4d}}. \tag{8}$$

Such an event is highly unlikely, however, for any t. In particular, from Lemma 9, we know that the probability of (8) holding for any particular i and the algorithm not erring is at most $3/4$. Thus the chance of having any such run of length R is at most $L(3/4)^R$. ■

Lemma 9 also tells us something interesting about the fraction of errors that we are missing because we do not ask for labels. In particular,

Lemma 10 *Given that $s_t \geq \sqrt{(1-(u \cdot v_t)^2)/(16d)}$, upon the tth update, each erroneous example is queried with probability at least 1/32, that is,*

$$P_{x \in S}\left[|x \cdot v_t| \leq s_t \;\middle|\; x \in \xi_t\right] \geq \frac{1}{32}.$$

Proof Using Lemmas 9 and 4, we have

$$\begin{aligned}
P_{x\in S}\left[x\in\xi_t \wedge |x\cdot v_t|\le s_t\right] &\ge P_{x\in S}\left[x\in\xi_t\wedge|x\cdot v_t|\le\sqrt{\frac{1-(u\cdot v_t)^2}{16d}}\right]\\
&\ge \frac{1}{4}P_{x\in S}\left[|x\cdot v_t|\le\sqrt{\frac{1-(u\cdot v_t)^2}{16d}}\right]\\
&\ge \frac{1}{64}\sqrt{1-(u\cdot v_t)^2}=\frac{1}{64}\sin\theta_t\\
&\ge \frac{\theta_t}{32\pi}.
\end{aligned}$$

For the last inequality, we have used (2). However, $P_{x\in S}[x\in\xi_t]=\theta_t/\pi$, so we are querying an error $x\in\xi_t$ with probability at least $1/32$, that is, the above inequality implies,

$$P_{x\in S}\left[|x\cdot v_t|\le s_t \;\middle|\; x\in\xi_t\right]=\frac{P_{x\in S}[x\in\xi_t\wedge|x\cdot v_t|\le s_t]}{P_{x\in S}[x\in\xi_t]}\ge\frac{\theta_t/(32\pi)}{\theta_t/\pi}=\frac{1}{32}.$$

■

Next, we show that the updates are likely to make progress.

Lemma 11 *Assuming that* $s_t\ge\sqrt{(1-(u\cdot v_t)^2)/(16d)}$*, a random update is good with probability at least* $1/2$*, that is,*

$$P_{x_t\in S}\left[(1-v_{t+1}\cdot u)\le(1-v_t\cdot u)\left(1-\frac{c}{d}\right)\;\middle|\;|x\cdot v_t|\le s_t\wedge x_t\in\xi_t\right]\ge\frac{1}{2}.$$

Proof By Lemma 10, each error is queried with probability 1/32. On the other hand, by Lemma 6 of the previous section, 63/64 of all errors are good. Since we are querying at least 2/64 fraction of all errors, at least half of our queried errors must be good. ■

We now have the pieces to guarantee the convergence rate of the active algorithm, thereby proving Theorem 3. This involves bounding both the number of labels that we query as well as the number of total errors, which includes updates as well as errors that were never detected.

Theorem 3 *With probability* $1-\delta$ *with respect to the uniform distribution on the unit sphere, in the realizable setting, using* $L=O\big(d\log\big(\frac{1}{\epsilon\delta}\big)(\log\frac{d}{\delta}+\log\log\frac{1}{\epsilon})\big)$ *labels and making a total number of errors of* $O\big(d\log\big(\frac{1}{\epsilon\delta}\big)(\log\frac{d}{\delta}+\log\log\frac{1}{\epsilon})\big)$*, the final error of the active modified Perceptron algorithm will be* ϵ*, when run with the above* L *and* $R=O(\log\frac{d}{\delta}+\log\log\frac{1}{\epsilon})$*.*

Proof Let U be the number of updates performed. We know, by Lemma 8 that with probability $1-L(\frac{3}{4})^R$,

$$s_t\ge\frac{\sin\theta_t}{4\sqrt{d}}\ge\frac{\theta_t}{2\pi\sqrt{d}} \tag{9}$$

for all t. Again, we have used (2). By Lemma 11, we know that for each t which is an update, either (9) fails or

$$E[1-u\cdot v_{t+1}|v_t]\le(1-u\cdot v_t)\left(1-\frac{c}{2d}\right).$$

Hence, after U updates, using Markov's inequality,

$$P\left[1 - u \cdot v_L \geq \frac{4}{\delta}\left(1 - \frac{c}{2d}\right)^U\right] \leq \frac{\delta}{4} + L\left(\frac{3}{4}\right)^R.$$

In other words, with probability $1 - \frac{\delta}{4} - L(\frac{3}{4})^R$, we also have

$$U \leq \frac{2d}{c}\log\frac{4}{\delta(1 - u \cdot v_L)} \leq \frac{2d}{c}\log\frac{\pi^2}{\delta\theta_L^2} = O\left(d\log\frac{1}{\delta\varepsilon}\right),$$

where for the last inequality we used (1). In total, $L \leq R\left(U + \log_2\frac{1}{s_L}\right)$. This is because once every R labels we either have at least one update or we decrease s_L by a factor of 2. Equivalently, $s_L \leq 2^{U - L/R}$. Hence, with probability $1 - \frac{\delta}{4} - L(\frac{3}{4})^R$,

$$\frac{\theta_L}{2\pi\sqrt{d}} \leq s_L \leq 2^{O(d\log\frac{1}{\delta\varepsilon}) - L/R}.$$

Working backwards, we choose $L/R = \Theta(d\log\frac{1}{\varepsilon\delta})$ so that the above expression implies $\frac{\theta_L}{\pi} \leq \varepsilon$, as required. We choose

$$R = 10\log\frac{2L}{\delta R} = \Theta\left(\log\frac{d\log\frac{1}{\varepsilon\delta}}{\delta}\right) = O\left(\log\frac{d}{\delta} + \log\log\frac{1}{\varepsilon}\right).$$

The first equality ensures that $L(\frac{3}{4})^R \leq \frac{\delta}{4}$. Hence, for the L and R chosen in the theorem, with probability $1 - \frac{3}{4}\delta$, we have error $\frac{\theta_L}{\pi} < \varepsilon$. Finally, either condition (9) fails or each error is queried with probability at least $\frac{1}{32}$. By the multiplicative Chernoff bound, if there were a total of $E > 64U$ errors, with probability $\geq 1 - \frac{\delta}{4}$, at least $E/64 > U$ would have been caught and used as updates. Hence, with probability at most $1 - \delta$, we have achieved the target error using the specified number of labels and observing the specified number of errors. ■

7. Discussion and Conclusions

In the evolving theory of active learning, the most concrete, nontrivial scenario in which active learning has been shown to give an exponential improvement in sample complexity is that of learning a linear separator for data distributed uniformly over the unit sphere. In this paper, we have demonstrated that this particular case can be solved by a much simpler algorithm than was previously known. Table 1 summarizes our contributions in context. We report bounds for all the algorithms with respect to our setting: learning homogeneous half-spaces when the data distribution is uniform on the unit sphere and separable through the origin, although a few of the algorithms were designed for more general distributions. This paper gives the lower bounds stated for Perceptron, and provides an algorithm that attains the upper bounds in the bottom row. While we list a host of results for comparison, it is important to note that the algorithm of Dasgupta (2005) has not shown to be efficiently implementable, and that we state bounds for this realizable problem, even though the algorithm of Balcan et al. (2007) can handle certain types of noise, and A^2 (Balcan et al., 2006) and the algorithm of Dasgupta et al. (2007) were designed to handle the agnostic setting.

	Samples	Mistakes	Labels	Noise tolerance
PAC bounds	$\tilde{O}(\frac{d}{\varepsilon}), \Omega(\frac{d}{\varepsilon})$			
Perceptron	$\tilde{O}(\frac{d}{\varepsilon^3}), \Omega(\frac{1}{\varepsilon^2})$	$\tilde{O}(\frac{d}{\varepsilon^2}), \Omega(\frac{1}{\varepsilon^2})$	$\Omega(\frac{1}{\varepsilon^2})$	Unknown
D'05	$\tilde{O}(\frac{d}{\varepsilon}\log^2\frac{1}{\varepsilon})$	$\tilde{O}(d\log^2\frac{1}{\varepsilon})$	$\tilde{O}(d\log^2\frac{1}{\varepsilon})$	No
A^2	$\tilde{O}(\frac{d^{1.5}}{\varepsilon}\log\frac{1}{\varepsilon})$	$\tilde{O}(d^{1.5}\log\frac{1}{\varepsilon})$	$\tilde{O}(d^{1.5}\log\frac{1}{\varepsilon})$	Yes
DHM'07	$\tilde{O}(\frac{d^{1.5}}{\varepsilon}\log\frac{1}{\varepsilon})$	$\tilde{O}(d^{1.5}\log\frac{1}{\varepsilon})$	$\tilde{O}(d^{1.5}\log\frac{1}{\varepsilon})$	Yes
QBC	$\tilde{O}(\frac{d}{\varepsilon}\log\frac{1}{\varepsilon})$	$\tilde{O}(d\log\frac{1}{\varepsilon})$	$\tilde{O}(d\log\frac{1}{\varepsilon})$	No
BBZ'07	$\tilde{O}(\frac{d}{\varepsilon}\log\frac{1}{\varepsilon})$	$\tilde{O}(d\log\frac{1}{\varepsilon})$	$\tilde{O}(d\log\frac{1}{\varepsilon})$	Yes
Our algorithm	$\tilde{O}(\frac{d}{\varepsilon}\log\frac{1}{\varepsilon})$	$\tilde{O}(d\log\frac{1}{\varepsilon})$	$\tilde{O}(d\log\frac{1}{\varepsilon})$	Unknown

Table 1: Results in context, for learning half-spaces through the origin. The last column indicates whether each algorithm has been proved to exhibit some noise tolerance when used for active learning.

In all these papers, the uniform distribution of data has consistently proved amenable to analysis. This is an impressive distribution to learn against because it is difficult in some ways—most of the data is close to the decision boundary, for instance—but a more common assumption would be to make the two classes Gaussian, or to merely stipulate that they are separated by a margin. As a modest step towards relaxing this distributional assumption, we can show an (at most) polynomial dependence of the label complexity on λ (Monteleoni, 2006), when the input distribution is λ-similar to uniform, a setting studied in Freund et al. (1997).

Our algorithm is in some ways fine-tuned for linearly-separable data that are distributed uniformly; for instance, in the choice of the parameters R and s_1. An immediate open problem is therefore the following:

1. Design a version of the algorithm that is sensible for general data distributions which may not be linearly separable.

2. What types of noise can be tolerated by this scheme?

3. For what distributions can its label complexity be analyzed?

A step towards the practical realization of our algorithm is the work of Monteleoni and Kääriäinen (2007), which applies a version of it to an optical character recognition problem.

Acknowledgments

This work was done while ATK was at the Toyota Technological Institute at Chicago. Much of this work was done while CM was visiting the Toyota Technological Institute at Chicago. Some of this work was done while CM was at the Massachusetts Institute of Technology, Computer Science and Intelligence Laboratory, and at the University of California, San Diego, Department of Computer Science and Engineering. CM would like to thank Adam Klivans, Brendan McMahan, and Vikas Sindhwani, for various discussions at TTI, and David McAllester for the opportunity to visit. The authors thank the anonymous reviewers of COLT 2005 and JMLR for helpful comments used in revision.

Appendix A. Proof of Lemma 4

Proof [Lemma 4] Let $r = \gamma/\sqrt{d}$ and let A_d be the area of a d-dimensional unit sphere, that is, the surface of a $(d+1)$-dimensional unit ball. Then

$$P_x[|a \cdot x| \le r] = \frac{\int_{-r}^{r} A_{d-2}(1-z^2)^{\frac{d-2}{2}}(1-z^2)^{-1/2}dz}{A_{d-1}} = \frac{2A_{d-2}}{A_{d-1}} \int_0^r (1-z^2)^{(d-3)/2}dz.$$

First observe,

$$r(1-r^2)^{(d-3)/2} \le \int_0^r (1-z^2)^{(d-3)/2}dz \le r. \tag{10}$$

For $x \in [0,0.5]$, $1-x \ge 4^{-x}$. Hence, for $0 \le r \le 2^{-1/2}$,

$$(1-r^2)^{(d-3)/2} \ge 4^{-r^2((d-3)/2)} \ge 2^{-r^2 d}.$$

So we can conclude that the integral of (10) is in $[r/2, r]$ for $r \in [0, 1/\sqrt{d}]$. The ratio $2A_{d-2}/A_{d-1}$ can be shown to be in the range $[\sqrt{d/3}, \sqrt{d}]$ by straightforward induction on d, using the definition of the Γ function, and the fact that $A_{d-1} = 2\pi^{d/2}/\Gamma(d/2)$. ■

References

S. Agmon. The relaxation method for linear inequalities. *Canadian Journal of Math.*, 6(3):382–392, 1954.

D. Angluin. Queries revisited. *In Proc. 12th International Conference on Algorithmic Learning Theory*, LNAI,2225:12–31, 2001.

M.-F. Balcan, A. Beygelzimer, and J. Langford. Agnostic active learning. In *Proc. International Conference on Machine Learning*, 2006.

M.-F. Balcan, A. Broder, and T. Zhang. Margin based active learning. In *Proc. 20th Annual Conference on Learning Theory*, 2007.

E.B. Baum. The perceptron algorithm is fast for nonmalicious distributions. *Neural Computation*, 2:248–260, 1997.

A. Blum, A. Frieze, R. Kannan, and S. Vempala. A polynomial-time algorithm for learning noisy linear threshold functions. In *Proc. 37th Annual IEEE Symposium on the Foundations of Computer Science*, 1996.

N. Cesa-Bianchi, A. Conconi, and C. Gentile. Learning probabilistic linear-threshold classifiers via selective sampling. In *Proc. 16th Annual Conference on Learning Theory*, 2003.

N. Cesa-Bianchi, C. Gentile, and L. Zaniboni. Worst-case analysis of selective sampling for linear-threshold algorithms. In *Advances in Neural Information Processing Systems 17*, 2004.

D.A. Cohn, L. Atlas, and R.E. Ladner. Improving generalization with active learning. *Machine Learning*, 15(2):201–221, 1994.

S. Dasgupta. Analysis of a greedy active learning strategy. In *Advances in Neural Information Processing Systems 17*, 2004.

S. Dasgupta. Coarse sample complexity bounds for active learning. In *Advances in Neural Information Processing Systems 18*, 2005.

S. Dasgupta, D. Hsu, and C. Monteleoni. A general agnostic active learning algorithm. In *Advances in Neural Information Processing Systems*, 2007.

Y. Freund, H.S. Seung, E. Shamir, and N. Tishby. Selective sampling using the query by committee algorithm. *Machine Learning*, 28(2-3):133–168, 1997.

R. Gilad-Bachrach, A. Navot, and N. Tishby. Query by committee made real. In *Advances in Neural Information Processing Systems 18*, 2005.

S. Hampson and D. Kibler. Minimum generalization via reflection: A fast linear threshold learner. *Machine Learning*, 37(1):51–73, 1999.

S. Hanneke. A bound on the label complexity of agnostic active learning. In *Proc. International Conference on Machine Learning*, 2007.

M. Kääriäinen. Active learning in the non-realizable case. In *Proc. 17th International Conference on Algorithmic Learning Theory*, 2006.

D.D. Lewis and W.A. Gale. A sequential algorithm for training text classifiers. In *Proc. of SIGIR-94, 17th ACM International Conference on Research and Development in Information Retrieval*, 1994.

C. Monteleoni and M. Kääriäinen. Practical online active learning for classification. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition, Online Learning for Classification Workshop*, 2007.

C.E. Monteleoni. *Learning with Online Constraints: Shifting Concepts and Active Learning*. PhD Thesis, MIT Computer Science and Artificial Intelligence Laboratory, 2006.

T.S. Motzkin and I.J. Schoenberg. The relaxation method for linear inequalities. *Canadian Journal of Math.*, 6(3):393–404, 1954.

F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65:386–407, 1958.

R.A. Servedio. On PAC learning using winnow, perceptron, and a perceptron-like algorithm. In *Computational Learning Theory*, pages 296 – 307, 1999.

H.S. Seung, M. Opper, and H. Sompolinsky. Query by committee. In *Proc. Fifth Annual ACM Conference on Computational Learning Theory*, 1992.

S. Tong and D. Koller. Support vector machine active learning with applications to text classification. *Journal of Machine Learning Research*, 2:45–66, 2001.

Journal of Machine Learning Research 10 (2009) 301-340 Submitted 4/08; Revised 1/09; Published 2/09

Improving the Reliability of Causal Discovery from Small Data Sets Using Argumentation

Facundo Bromberg BROMBERG@CS.IASTATE.EDU
Dimitris Margaritis DMARG@CS.IASTATE.EDU
Dept. of Computer Science
Iowa State University
Ames, IA 50011

Editor: Constantin Aliferis

Abstract

We address the problem of improving the reliability of independence-based causal discovery algorithms that results from the execution of statistical independence tests on small data sets, which typically have low reliability. We model the problem as a knowledge base containing a set of independence facts that are related through Pearl's well-known axioms. Statistical tests on finite data sets may result in errors in these tests and inconsistencies in the knowledge base. We resolve these inconsistencies through the use of an instance of the class of defeasible logics called argumentation, augmented with a preference function, that is used to reason about and possibly correct errors in these tests. This results in a more robust conditional independence test, called an *argumentative independence test*. Our experimental evaluation shows clear positive improvements in the accuracy of argumentative over purely statistical tests. We also demonstrate significant improvements on the accuracy of causal structure discovery from the outcomes of independence tests both on sampled data from randomly generated causal models and on real-world data sets.

Keywords: independence-based causal discovery, causal Bayesian networks, structure learning, argumentation, reliability improvement

1. Introduction and Motivation

Directed graphical models, also called *Bayesian networks*, can be used to represent the probability distribution of a domain. This makes them a useful and important tool for machine learning where a common task is inference, that is, predicting the probability distribution of a variable of interest given some other knowledge, usually in the form of values of other variables in the domain. An additional use of Bayesian networks comes by augmenting them with causal semantics that represent cause and effect relationships in the domain. The resulting networks are called *causal*. An important problem is inferring the structure of these networks, a process that is sometimes called *causal discovery*, which can provide insights into the underlying data generation process.

Two major classes of algorithms exist for learning the structure of Bayesian networks. One class contains so-called *score-based* methods, which learn the structure by conducting a search in the space of all structures in an attempt to find the structure of maximum score. This score is usually penalized log-likelihood, for example, the Bayesian Information Criterion (BIC) or the (equivalent) Minimum Description Length (MDL). A second class of algorithms works by exploiting the fact that a causal Bayesian network implies the existence of a set of conditional independence statements between sets of domain variables. Algorithms in this class use the outcomes of a number of conditional independences to constrain the set of possible structures consistent with these to a singleton (if possible) and infer that structure as the only possible one. As such they are called *constraint-based* or *independence-based* algorithms. In this paper we address open problems related to the latter class of algorithms.

It is well-known that independence-based algorithms have several shortcomings. A major one has to do with the effect that unreliable independence information has on the their output. In general such independence information comes from two sources: (a) a domain expert that can provide his or her opinion on the validity of certain conditional independences among some of the variables, sometimes with a degree of confidence attached to them, and/or (b) statistical tests of independence, conducted on data gathered from the domain. As expert information is often costly and difficult to obtain, (b) is the most commonly used option in practice. A problem that occurs frequently however is that the data set available may be small. This may happen for various reasons: lack of subjects to observe (e.g., in medical domains), an expensive data-gathering process, privacy concerns and others. Unfortunately, the reliability of statistical tests significantly diminishes on small data sets. For example, Cochran (1954) recommends that Pearson's χ^2 independence test be deemed unreliable if more than 20% of the cells of the test's contingency table have an expected count of less than 5 data points. Unreliable tests, besides producing errors in the resulting causal model structure, may also produce cascading errors due to the way that independence-based algorithms work: their operation, including which test to evaluate next, typically depends on the outcomes of previous ones. Thus a single error in a statistical test can be propagated by the subsequent choices of tests to be performed by the algorithm, and finally when the edges are oriented. Therefore, an error in a previous test may have large (negative) consequences in the resulting structure, a property that is called *instability* in Spirtes et al. (2000). One possible method for addressing the effect of multiple errors in the construction of a causal model through multiple independence tests is the Bonferroni correction (Hochberg, 1988; Abdi, 2007), which works by dividing the type I error probability α of each test by the number of such tests evaluated during the entire execution of the causal learning algorithm. As a result, the collective type I error probability (of all tests evaluated) is α, that is, 0.05 typically. However, this may make the detection of true dependences harder, as now larger data sets would be required to reach the adjusted confidence threshold of each test. The types of adjustments that may be appropriate for each case to tests that may be dependent is an open problem and the subject of current research in statistics (Benjamini and Hochberg, 1995; Benjamini and Yekutieli, 2001; Storey, 2002).

In this paper we present and evaluate a number of methods for increasing the reliability of independence tests for small data sets. A result of this is the improvement in reliability of independence-based causal discovery algorithms that use these data sets, as we demonstrate in our experiments. We model this setting as a knowledge base whose contents are propositions representing conditional independences that may contain errors. Our main insight is to recognize that the outcomes of independence tests are not themselves independent but are constrained by the outcomes of other tests through Pearl's well-known properties of the conditional independence relation (Pearl, 1988; Dawid, 1979). These can therefore be seen as *integrity constraints* that can correct certain inconsistent test outcomes, choosing instead the outcome that can be inferred by tests that do not result in contradictions. We illustrate this by an example.

Example 1. *Consider an independence-based knowledge base that contains the following propositions, obtained through statistical tests on data.*

$$(\{0\} \perp\!\!\!\perp \{1\} \mid \{2\}) \tag{1}$$
$$(\{0\} \not\perp\!\!\!\perp \{3\} \mid \{2\}) \tag{2}$$
$$(\{0\} \perp\!\!\!\perp \{3\} \mid \{1,2\}) \tag{3}$$

where $(\mathbf{X} \perp\!\!\!\perp \mathbf{Y} \mid \mathbf{Z})$ *denotes conditional independence of the set of variables* $\mathbf{X}$ *with* $\mathbf{Y}$ *conditional on set* $\mathbf{Z}$*, and* $(\mathbf{X} \not\perp\!\!\!\perp \mathbf{Y} \mid \mathbf{Z})$ *denotes conditional dependence. Suppose that (3) is in fact wrong. Such an error can be avoided if there exists a constraint involving these independence propositions. For example, suppose that we also know that the following rule holds in the domain (this is an instance of an application of the Contraction and Decomposition axioms, described later in Section 2):*

$$(\{0\} \perp\!\!\!\perp \{1\} \mid \{2\}) \wedge (\{0\} \not\perp\!\!\!\perp \{3\} \mid \{2\}) \implies (\{0\} \not\perp\!\!\!\perp \{3\} \mid \{1,2\}). \tag{4}$$

Rule (4), together with independence proposition (1) and dependence proposition (2), contradict independence proposition (3), resulting in an inconsistent knowledge base. If Rule (4) and propositions (1) and (2) are accepted, then proposition (3) must be rejected (and its value reversed), correcting the error in this case. The framework presented in the rest of the paper provides a principled approach for resolving such inconsistencies.

The situation described in the previous example, while simple, illustrates the general idea that we will use in the rest of the paper: the set of independences and dependences used in a causal discovery algorithm form a potentially inconsistent knowledge base, and making use of general rules, derived from axioms and theorems that we know hold in the domain, helps us correct certain outcomes of statistical tests. In this way we will be able to improve the reliability of causal discovery algorithms that use them to derive causal models. To accomplish this we use the framework of *argumentation*, which provides a sound and elegant way of resolving inconsistencies in such knowledge bases, including ones that contain independences.

The rest of the paper is organized as follows. The next section introduces our notation and definitions. Section 3 presents the argumentation framework and its extension with preferences, and describes our approach for applying it to represent and reason in knowledge bases containing independence facts that may be inconsistent. Section 4 introduces the argumentative independence test, implemented by the top-down algorithm introduced in Section 5. We then present an approximation for the top-down algorithm in Section 6 that reduces its time complexity to polynomial. We experimentally evaluate our approach in Section 7, and conclude with a summary and possible directions of future research in Section 8. Most of the proofs are presented in detail in Appendices A and B, which contain proofs for the computability (termination) and the validity (no AIT test can return a dependence and an independence result at the same time) of AIT, respectively. Note that, as our main goal in this paper is to address the problem of robust causal learning and not necessarily to advance the theory of argumentation itself, our exposition in the rest of the paper is geared toward causality theorists and practitioners. As this community may be unfamiliar with the theory and methods of the argumentation framework, we have included a self-contained discussion that covers the basic definitions and theorems of argumentation theory in some detail.

2. Notation and Preliminaries

In this work we denote random variables with capitals (e.g., X, Y, Z) and sets of variables with bold capitals (e.g., $\mathbf{X}, \mathbf{Y}, \mathbf{Z}$). In particular, we denote by $\mathbf{V} = \{1, \ldots, n\}$ the set of all n variables in the domain, naming the variables by their indices in $\mathbf{V}$; for instance, we refer to the third variable in $\mathbf{V}$ simply by 3. We assume that all variables in the domain are discrete following a multinomial distribution or are continuous following a Gaussian distribution. We denote the data set by D and its size (number of data points) by N. We use the notation $(\mathbf{X} \perp\!\!\!\perp \mathbf{Y} \mid \mathbf{Z})$ to denote that the variables in set $\mathbf{X}$ are (jointly) independent of those in $\mathbf{Y}$ conditional on the values of the variables in $\mathbf{Z}$, for disjoint sets of variables $\mathbf{X}$, $\mathbf{Y}$, and $\mathbf{Z}$, while $(\mathbf{X} \not\!\perp\!\!\!\perp \mathbf{Y} \mid \mathbf{Z})$ denotes conditional dependence. For the sake of readability, we slightly abuse this notation and use $(X \perp\!\!\!\perp Y \mid Z)$ as shorthand for $(\{X\} \perp\!\!\!\perp \{Y\} \mid \{Z\})$.

A Bayesian network (**BN**) is a directed graphical model which represents the joint probability distribution over $\mathbf{V}$. Each node in the graph represents one of the random variables in the domain. The structure of the network implicitly represents a set of conditional independences on the domain variables. Given the structure of a BN, the set of independences implied by it can be identified by a process called *d-separation* (Pearl, 1988); the latter follows from the *local Markov property* that states that each node in the network is conditionally independent of all its non-descendants in the graph given its parents. All independences identified by d-separation are implied by the model structure. If, in addition, all remaining triplets $(\mathbf{X}, \mathbf{Y}, \mathbf{Z})$ correspond to dependencies, we say that the BN is *directed graph-isomorph* (abbreviated *DAG-isomorph*) or simply *causal* (as defined by Pearl, 1988). The concept of DAG-isomorphism is equivalent to a property called Faithfulness in Spirtes et al. (2000). A graph G is said to be *faithful* to some distribution if exactly those independences that exist in the distribution and no others are returned by the process of d-separation on G. In this paper we assume Faithfulness. For learning the structure of the Bayesian network of a domain we make use of the PC algorithm (Spirtes et al., 2000), which is only able to correctly identify the structure under the assumption of *causal sufficiency*. We therefore also assume causal sufficiency. A domain is causally sufficient if it does not contain any *hidden* or *latent* variables.

As mentioned above, independence-based algorithms operate by conducting a series of conditional independence queries. For these we assume that an *independence-query oracle* exists that is able to provide such information. This approach can be viewed as an instance of the statistical query oracle theory of Kearns and Vazirani (1994). In practice such an oracle does not exist, but can be implemented approximately by a statistical test evaluated on the data set (for example, this can be Pearson's conditional independence χ^2 (chi-square) test (Agresti, 2002), Wilk's G^2 test, a mutual information test etc.). In this work we used Wilk's G^2 test (Agresti, 2002). To determine conditional independence between two variables X and Y given a set $\mathbf{Z}$ from data, the statistical test G^2 (and many other independence tests based on hypothesis testing, for example, the χ^2 test) uses the values in the contingency table (a table containing the data point counts for each possible combination of the variables that participate in the test) to compute a *test statistic*. For a given value of the test statistic, the test then computes the likelihood of obtaining that or a more extreme value by chance under the *null hypothesis*, which in our case is that the two variables are conditionally independent. This likelihood, called the *p-value* of the test, is then returned. The p-value of a test equals the probability of falsely rejecting the null hypothesis (independence). Assuming that the p-value of a test is $p(X, Y \mid \mathbf{Z})$, the statistical test concludes independence if and only if $p(X, Y \mid \mathbf{Z})$ is greater than a threshold α, that is,

$$(X \perp\!\!\!\perp Y \mid \mathbf{Z}) \iff p(X, Y \mid \mathbf{Z}) > \alpha.$$

$$\boxed{\begin{array}{rl}
\textbf{(Symmetry)} & (\mathbf{X} \perp\!\!\!\perp \mathbf{Y} \mid \mathbf{Z}) \iff (\mathbf{Y} \perp\!\!\!\perp \mathbf{X} \mid \mathbf{Z}) \\
\textbf{(Decomposition)} & (\mathbf{X} \perp\!\!\!\perp \mathbf{Y} \cup \mathbf{W} \mid \mathbf{Z}) \implies (\mathbf{X} \perp\!\!\!\perp \mathbf{Y} \mid \mathbf{Z}) \wedge (\mathbf{X} \perp\!\!\!\perp \mathbf{W} \mid \mathbf{Z}) \\
\textbf{(Weak Union)} & (\mathbf{X} \perp\!\!\!\perp \mathbf{Y} \cup \mathbf{W} \mid \mathbf{Z}) \implies (\mathbf{X} \perp\!\!\!\perp \mathbf{Y} \mid \mathbf{Z} \cup \mathbf{W}) \\
\textbf{(Contraction)} & (\mathbf{X} \perp\!\!\!\perp \mathbf{Y} \mid \mathbf{Z}) \wedge (\mathbf{X} \perp\!\!\!\perp \mathbf{W} \mid \mathbf{Z} \cup \mathbf{Y}) \implies (\mathbf{X} \perp\!\!\!\perp \mathbf{Y} \cup \mathbf{W} \mid \mathbf{Z}) \\
\textbf{(Intersection)} & (\mathbf{X} \perp\!\!\!\perp \mathbf{Y} \mid \mathbf{Z} \cup \mathbf{W}) \wedge (\mathbf{X} \perp\!\!\!\perp \mathbf{W} \mid \mathbf{Z} \cup \mathbf{Y}) \implies (\mathbf{X} \perp\!\!\!\perp \mathbf{Y} \cup \mathbf{W} \mid \mathbf{Z})
\end{array}} \tag{5}$$

$$\boxed{\begin{array}{rl}
\textbf{(Symmetry)} & (\mathbf{X} \perp\!\!\!\perp \mathbf{Y} \mid \mathbf{Z}) \iff (\mathbf{Y} \perp\!\!\!\perp \mathbf{X} \mid \mathbf{Z}) \\
\textbf{(Composition)} & (\mathbf{X} \perp\!\!\!\perp \mathbf{Y} \mid \mathbf{Z}) \wedge (\mathbf{X} \perp\!\!\!\perp \mathbf{W} \mid \mathbf{Z}) \implies (\mathbf{X} \perp\!\!\!\perp \mathbf{Y} \cup \mathbf{W} \mid \mathbf{Z}) \\
\textbf{(Decomposition)} & (\mathbf{X} \perp\!\!\!\perp \mathbf{Y} \cup \mathbf{W} \mid \mathbf{Z}) \implies (\mathbf{X} \perp\!\!\!\perp \mathbf{Y} \mid \mathbf{Z}) \wedge (\mathbf{X} \perp\!\!\!\perp \mathbf{W} \mid \mathbf{Z}) \\
\textbf{(Intersection)} & (\mathbf{X} \perp\!\!\!\perp \mathbf{Y} \mid \mathbf{Z} \cup \mathbf{W}) \wedge (\mathbf{X} \perp\!\!\!\perp \mathbf{W} \mid \mathbf{Z} \cup \mathbf{Y}) \implies (\mathbf{X} \perp\!\!\!\perp \mathbf{Y} \cup \mathbf{W} \mid \mathbf{Z}) \\
\textbf{(Weak Union)} & (\mathbf{X} \perp\!\!\!\perp \mathbf{Y} \cup \mathbf{W} \mid \mathbf{Z}) \implies (\mathbf{X} \perp\!\!\!\perp \mathbf{Y} \mid \mathbf{Z} \cup \mathbf{W}) \\
\textbf{(Contraction)} & (\mathbf{X} \perp\!\!\!\perp \mathbf{Y} \mid \mathbf{Z}) \wedge (\mathbf{X} \perp\!\!\!\perp \mathbf{W} \mid \mathbf{Z} \cup \mathbf{Y}) \implies (\mathbf{X} \perp\!\!\!\perp \mathbf{Y} \cup \mathbf{W} \mid \mathbf{Z}) \\
\textbf{(Weak Transitivity)} & (\mathbf{X} \perp\!\!\!\perp \mathbf{Y} \mid \mathbf{Z}) \wedge (\mathbf{X} \perp\!\!\!\perp \mathbf{Y} \mid \mathbf{Z} \cup \gamma) \implies (\mathbf{X} \perp\!\!\!\perp \gamma \mid \mathbf{Z}) \vee (\gamma \perp\!\!\!\perp \mathbf{Y} \mid \mathbf{Z}) \\
\textbf{(Chordality)} & (\alpha \perp\!\!\!\perp \beta \mid \gamma \cup \delta) \wedge (\gamma \perp\!\!\!\perp \delta \mid \alpha \cup \beta) \implies (\alpha \perp\!\!\!\perp \beta \mid \gamma) \vee (\alpha \perp\!\!\!\perp \beta \mid \delta)
\end{array}} \tag{6}$$

Common values in statistics for α are 0.05 and 0.01, corresponding to *confidence thresholds* $(1-\alpha)$ of 0.95 and 0.99 respectively. The value 0.10 for α is also sometimes used, depending on the application, while values as low as 0.005 and 0.001 are sometimes used for structure learning.

The conditional independences and dependences of a domain are connected through a set of general rules, introduced in Pearl (1988) and shown boxed in Eq. (5). These can be seen as constraints in a meta-space representing all possible independences in the domain. More specifically, let us imagine a meta-space of binary variables, each corresponding to the truth value of the independence of a triplet $(\mathbf{X}, \mathbf{Y} \mid \mathbf{Z})$ (e.g., `true` for independence and `false` for dependence). Each point in this space corresponds to a conditional independence assignment to all possible triplets in the domain. In this conceptual space not all points are tenable; in particular the set of rules of Eq. (5) constrain the truth values of independences corresponding to triplets. For domains for which there exists a faithful Bayesian network a more relaxed set of properties hold, shown boxed in Eq. (6) where α, β, γ and δ correspond to single variables. In both sets of axioms, the property of Intersection holds if the probability distribution of the domain is positive, meaning that every assignment to all variables in the domain has a non-zero probability. Eq. (6) were first introduced by Dawid (1979) in a slightly different form and independently re-discovered by Pearl and Paz (1985).

Note that the axioms of Eq. (5) are necessarily incomplete; Studený (1991) showed that there is no finite axiomatization of the conditional independence relation in general. The implication of this is that there may be some inconsistencies involving some set of independences and dependences that no method can detect and resolve.

In the next section we describe the argumentation framework, which allows one to make beneficial use of these constraints. This is followed by its application to our problem of answering independence queries from knowledge bases that contain sets of potentially inconsistent independence propositions.

3. The Argumentation Framework

There exist two major approaches for reasoning with information contained in inconsistent knowledge bases such as those containing independence statements that were described in the previous section. These two distinct approaches correspond to two different attitudes: One is to resolve the inconsistencies by removing a subset of propositions such that the resulting KB becomes consistent; this is called *belief revision* in the literature (Gärdenfors, 1992; Gärdenfors and Rott, 1995; Shapiro, 1998; Martins, 1992). A potential shortcoming (Shapiro, 1998) of belief revision stems from the fact that it removes propositions, which discards potentially valuable information. In addition, an erroneous modification of the KB (such as the removal of a proposition) may have unintended negative consequences if later more propositions are inserted in the KB. A second approach to inconsistent KBs is to allow inconsistencies but to use rules that may be possibly contained in it to deduce which truth value of a proposition query is "preferred" in some way. One instance of this approach is *argumentation* (Dung, 1995; Loui, 1987; Prakken, 1997; Prakken and Vreeswijk, 2002), which is a sound approach that allows inconsistencies but uses a proof procedure that is able to deduce (if possible) that one of the truth values of certain propositions is preferred over its negation. Argumentation is a reasoning model that belongs to the broader class of defeasible logics (Pollock, 1992; Prakken, 1997). Our approach uses the argumentation framework of Amgoud and Cayrol (2002) that considers preferences over arguments, extending Dung's more fundamental framework (Dung, 1995). Preference relations give an extra level of specificity for comparing arguments, allowing a more refined form of selection between conflicting propositions. Preference-based argumentation is presented in more detail in Section 3.2.

We proceed now to describe the argumentation framework.

Definition 1. *An* argumentation framework *is a pair $\langle \mathcal{A}, \mathcal{R} \rangle$, where $\mathcal{A}$ is a set of arguments and $\mathcal{R}$ is a binary relation representing a defeasibility relationship between arguments, that is, $\mathcal{R} \subseteq \mathcal{A} \times \mathcal{A}$. $(a,b) \in \mathcal{R}$ or equivalently "a $\mathcal{R}$ b" reads that argument a defeats argument b. We also say that a and b are in conflict.*

An example of the defeat relation $\mathcal{R}$ is *logical defeat*, which occurs when an argument contradicts another logically.

The elements of the argumentation framework are not propositions but *arguments*. Given a potentially inconsistent knowledge base $\mathcal{K} = \langle \Sigma, \Psi \rangle$ with a set of propositions Σ and a set of inference rules Ψ, arguments are defined formally as follows.

Definition 2. *An* argument *over knowledge base $\langle \Sigma, \Psi \rangle$ is a pair (H,h) where $H \subseteq \Sigma$ such that:*

- *H is consistent,*
- *$H \vdash_\Psi h$,*
- *H is minimal (with respect to set inclusion).*

H is called the support *and h the* conclusion *or* head *of the argument.*

In the above definition $\vdash_\Psi$ stands for classical logical inference over the set of inference rules Ψ. Intuitively an argument (H,h) can be thought as an "if-then" rule, that is, "if H then h." In inconsistent knowledge bases two arguments may contradict or *defeat* each other. The defeat relation is defined through the *rebut* and *undercut* relations, defined as follows.

Definition 3. *Let (H_1,h_1), (H_2,h_2) be two arguments.*

- *(H_1,h_1)* rebuts *(H_2,h_2) iff $h_1 \equiv \neg h_2$.*

Algorithm 1 Recursive computation of acceptable arguments: $Acc_{\mathcal{R}} = \mathcal{F}(\mathcal{A}, \mathcal{R}, S)$

```
1: $S' \longleftarrow S \cup \{a \in \mathcal{A} \mid a \text{ is defended by } S\}$
2: **if** $S = S'$ **then**
3:     **return** $S'$
4: **else**
5:     **return** $\mathcal{F}(\mathcal{A}, \mathcal{R}, S')$
```

- (H_1, h_1) undercuts (H_2, h_2) *iff* $\exists h \in H_2$ *such that* $h \equiv \neg h_1$.

If (H_1, h_1) *rebuts or undercuts* (H_2, h_2) *we say that* (H_1, h_1) defeats (H_2, h_2).

(The symbol "$\equiv$" stands for logical equivalence.) In other words, $(H_1, h_1)\ \mathcal{R}\ (H_2, h_2)$ if and only if (H_1, h_1) *rebuts* or *undercuts* (H_2, h_2).

The objective of argumentation is to decide on the acceptability of a given argument. There are three possibilities: an argument can be accepted, rejected, or neither. This partitions the space of arguments $\mathcal{A}$ in three classes:

- The class $Acc_{\mathcal{R}}$ of *acceptable arguments*. Intuitively, these are the "good" arguments. In the case of an inconsistent knowledge base, these will be inferred from the knowledge base.

- The class $Rej_{\mathcal{R}}$ of *rejected arguments*. These are the arguments defeated by acceptable arguments. When applied to an inconsistent knowledge base, these will not be inferred from it.

- The class $Ab_{\mathcal{R}}$ of arguments *in abeyance*. These arguments are neither accepted nor rejected.

The semantics of acceptability proposed by Dung (1995) dictates that an argument should be accepted if it is not defeated, or if it is defended by acceptable arguments, that is, each of its defeaters is itself defeated by an acceptable argument. This is formalized in the following definitions.

Definition 4. *Let* $\langle \mathcal{A}, \mathcal{R} \rangle$ *be an argumentation framework, and* $S \subseteq \mathcal{A}$. *An argument* a *is* defended *by* S *if and only if* $\forall b$, *if* $(b\ \mathcal{R}\ a)$ *then* $\exists c \in S$ *such that* $(c\ \mathcal{R}\ b)$.

Dung characterizes the set of acceptable arguments by a monotonic function $\mathcal{F}$, that is, $\mathcal{F}(S) \subseteq \mathcal{F}(S \cup T)$ for some S and T. Given a set of arguments $S \subseteq \mathcal{A}$ as input, $\mathcal{F}$ returns the set of all arguments defended by S:

Definition 5. *Let* $S \subseteq \mathcal{A}$. *Then* $\mathcal{F}(S) = \{a \in \mathcal{A} \mid a \text{ is defended by } S\}$.

Slightly overloading our notation, we define $\mathcal{F}(\varnothing)$ to contain the set of arguments that are not defeated by any argument in the framework.

Definition 6. $\mathcal{F}(\varnothing) = \{a \in \mathcal{A} \mid a \text{ is not defeated by any argument in } \mathcal{A}\}$.

Dung proved that the set of acceptable arguments is the least fix-point of $\mathcal{F}$, that is, the smallest set S such that $\mathcal{F}(S) = S$.

Theorem 7 (Dung 1995)**.** *Let* $\langle \mathcal{A}, \mathcal{R} \rangle$ *be an argumentation framework. The set of acceptable arguments* $Acc_{\mathcal{R}}$ *is the least fix-point of the function* $\mathcal{F}$.

Dung also showed that if the argumentation framework $\langle \mathcal{A}, \mathcal{R} \rangle$ is finitary, that is, for each argument A there are finitely many arguments that defeat A, the least fix-point of function $\mathcal{F}$ can be obtained by iterative application of $\mathcal{F}$ to the empty set. We can understand this intuitively: From our semantics of acceptability it follows that all arguments in $\mathcal{F}(\varnothing)$ are accepted. Also, every argument in $\mathcal{F}(\mathcal{F}(\varnothing))$ must be acceptable as well since each of its arguments is defended by acceptable arguments. This reasoning can be applied recursively until a fix-point is reached. This happens when the arguments in S cannot be used to defend any other argument not in S, that is, no additional argument is accepted. This suggests a simple algorithm for computing the set of acceptable arguments. Algorithm 1 shows a recursive procedure for this, based on the above definition. The algorithm takes as input an argumentation framework $\langle \mathcal{A}, \mathcal{R} \rangle$ and the set S of arguments found acceptable so far, that is, $S = \varnothing$ initially.

Let us illustrate these ideas with an example.

Example 2. *Let $\langle \mathcal{A}, \mathcal{R} \rangle$ be an argumentation framework defined by $\mathcal{A} = \{a,b,c\}$ and $\mathcal{R} = \{(a,b), (b,c)\}$. The only argument that is not defeated is a, and therefore $\mathcal{F}(\varnothing) = \{a\}$. Argument b is defeated by the acceptable argument a, so b cannot be defended and is therefore rejected, that is, $b \in Rej_{\mathcal{R}}$. Argument c, though defeated by b, is defended by (acceptable argument) a which defeats b, so c is acceptable. The set of acceptable arguments is therefore $Acc_{\mathcal{R}} = \{a,c\}$ and the set of rejected arguments is $Rej_{\mathcal{R}} = \{b\}$.*

The **bottom-up** approach of Algorithm 1 has the disadvantage that it requires the computation of all acceptable arguments to answer the acceptability status of a single one. In practice, and in particular in the application of argumentation to independence tests, the entire set of acceptable arguments is rarely needed. An alternative is to take a top-down approach (Amgoud and Cayrol, 2002; Dung, 1995; Toni and Kakas, 1995; Kakas and Toni, 1999) that evaluate the acceptability of some input argument by evaluating (recursively) the acceptability of its attackers. Below we present an alternative algorithm, called the **top-down algorithm**, for deciding the acceptability of an input argument. This algorithm is a version of the *dialog tree* algorithm of Amgoud and Cayrol (2002), where details unnecessary for the current exposition are not shown. This algorithm is provably equivalent to Algorithm 1 (whenever it is given the same input it is guaranteed to produce the same output), but it is considerably more efficient (as shown later in Section 5.2). We sketch the algorithm here and show a concrete version using the preference-based argumentation framework in Section 3.2.

Given an input argument a, the top-down algorithm employs a goal-driven approach for answering whether a is accepted or not. Its operation is guided by the same acceptability semantics as those used for Algorithm 1. Let us denote the predicates $A(a) \equiv (a \in Acc_{\mathcal{R}})$, $R(a) \equiv (a \in Rej_{\mathcal{R}})$, and $Ab(a) \equiv (a \in Ab_{\mathcal{R}})$. Then, the acceptability semantics are as follows.

Algorithm 2 Top-down computation of acceptable arguments: *top-down*$(\mathcal{A}, \mathcal{R}, a)$

```
1: $defeaters \leftarrow$ set of arguments in $\mathcal{A}$ that defeat $a$ according to $\mathcal{R}$.
2: for $d \in defeaters$ do
3:     if $top\text{-}down(\mathcal{A}, \mathcal{R}, d)$ = accepted then
4:         return rejected
5: return accepted
```

(Acceptance) A node is accepted iff it has no defeaters or all its defeaters are rejected:

$$A(a) \iff \forall b \in \mathit{defeaters}(a), R(b).$$

(Rejection) A node is rejected iff at least one of its defeaters is accepted:

$$R(a) \iff \exists b \in \mathit{defeaters}(a), A(b). \tag{7}$$

(Abeyance) A node is in abeyance iff its not accepted nor rejected:

$$Ab(a) \iff \neg A(a) \wedge \neg R(a).$$

The logic of these equations can be easily implemented with a recursive algorithm, shown in Algorithm 2. The algorithm, given some input argument a, loops over all defeaters of a and responds `rejected` if any of its defeaters is accepted (line 4). If execution reaches the end of the loop at line 5 then that means that none of its defeaters was accepted, and thus the algorithm accepts the input argument a. We can represent the execution of the top-down algorithm graphically by a tree that contains a at the root node, and all the defeaters of a node as its children. A leaf is reached when a node has no defeaters. In that case the loop contains no iterations and line 5 is reached trivially.

Unfortunately, the top-down algorithm, as shown in Algorithm 2, will fail to terminate when a node is in abeyance. This is clear from the following lemma (proved formally in Appendix A but reproduced here to aid our intuition).

Lemma 8. *For every argument a,*

$$Ab(a) \implies \exists b \in \mathit{attackers}(a), Ab(b).$$

(An attacker is a type of defeater; it is explained in detail in the next section. For the following discussion the reader can substitute "attacker" with "defeater" in the lemma above.) From this lemma we can see that, if an argument is in abeyance, its set of defeaters must contain an argument in abeyance and thus the recursive call of the top-down algorithm will never terminate, as there will always be another defeater in abeyance during each call. While there are ways to overcome this difficulty in the general case, we can prove that using the preference-based argumentation framework (presented later in the paper) and for the particular preference relation introduced for deciding on independence tests (c.f. Section 3.3), no argument can be in abeyance and thus the top-down algorithm always terminates. A formal proof of this is presented later in Section 5.

We conclude the section by proving that the top-down algorithm is equivalent to the bottom-up algorithm of Algorithm 1 that is, given the same input as Algorithm 1 it is guaranteed to produce the same output. The proof assumes no argument is in abeyance. This assumption is satisfied for argumentation in independence knowledge bases (c.f. Theorem 20, Section 5).

Theorem 9. *Let a be an argument in the argumentation framework $\langle \mathcal{A}, \mathcal{R} \rangle$, and let $\mathcal{F}$ be the set of acceptable arguments output by Algorithm 1. Assuming a is not in abeyance,*

$$\mathit{top\text{-}down}(\mathcal{A}, \mathcal{R}, a) = \texttt{accepted} \iff a \in \mathcal{F} \tag{8}$$

$$\mathit{top\text{-}down}(\mathcal{A}, \mathcal{R}, a) = \texttt{rejected} \iff a \notin \mathcal{F}. \tag{9}$$

Proof According to Theorem 7, the fix point of function $\mathcal{F}$ returned by Algorithm 1 contains the set of arguments considered acceptable by the acceptability semantics of Dung. As the top-down algorithm is a straightforward implementation of Dung's acceptability semantics expressed by Eq. (7), the double implication of Eq. (8) must follow. To prove Eq. (9) we can prove the equivalent expression with both sides negated, that is,

$$\textit{top-down}(\mathcal{A},\mathcal{R},a) \neq \texttt{rejected} \iff a \in \mathcal{F}.$$

Since a is not in abeyance, if the top-down algorithm does not return `rejected` it must return `accepted`. The double implication is thus equivalent to Eq. (8), which was proved true. ■

3.1 Argumentation in Independence Knowledge Bases

We can now apply the argumentation framework to our problem of answering queries from knowledge bases that contain a number of potentially inconsistent independences and dependencies and a set of rules that express relations among them.

Definition 10. *An* independence knowledge base *(**IKB**) is a knowledge base $\langle \Sigma, \Psi \rangle$ such that its set of propositions Σ contains independence propositions of the form $(\mathbf{X} \perp\!\!\!\perp \mathbf{Y} \mid \mathbf{Z})$ or $(\mathbf{X} \not\perp\!\!\!\perp \mathbf{Y} \mid \mathbf{Z})$ for $\mathbf{X}$, $\mathbf{Y}$ and $\mathbf{Z}$ disjoint subsets of $\mathbf{V}$, and its set of inference rules Ψ is either the general set of axioms shown in Eq. (5) or the specific set of axioms shown in Eq. (6).*

For IKBs, the set of arguments $\mathcal{A}$ is obtained in two steps. First, for each proposition $\sigma \in \Sigma$ (independence or dependence) we add to $\mathcal{A}$ the argument $(\{\sigma\},\sigma)$. This is a valid argument according to Definition 2 since its support $\{\sigma\}$ is (trivially) consistent, it (trivially) implies the head σ, and it is minimal (the pair $(\varnothing,\sigma)$ is not a valid argument since $\varnothing$ is equivalent to the proposition `true` which does not entail σ in general). We call arguments of the form $(\{\sigma\},\sigma)$ *propositional arguments* since they correspond to single propositions. The second step in the construction of the set of arguments $\mathcal{A}$ concerns rules. Based on the chosen set of axioms (general or directed) we construct an alternative, logically equivalent set of rules Ψ', each member of which is *single-headed*, that is, contains a single proposition as the consequent, and *decomposed*, that is, each of its propositions is an independence statement over single variables (the last step is justified by the fact that typical algorithms for causal learning never produce nor require the evaluation of independence between sets).

To construct the set of single-headed rules we consider, for each axiom, all possible contrapositive versions of it that have a single head. To illustrate, consider the Weak Transitivity axiom

$$(\mathbf{X} \perp\!\!\!\perp \mathbf{Y} \mid \mathbf{Z}) \wedge (\mathbf{X} \perp\!\!\!\perp \mathbf{Y} \mid \mathbf{Z} \cup \gamma) \implies (\mathbf{X} \perp\!\!\!\perp \gamma \mid \mathbf{Z}) \vee (\gamma \perp\!\!\!\perp \mathbf{Y} \mid \mathbf{Z})$$

from which we obtain the following set of single-headed rules:

$$\begin{aligned}
(\mathbf{X} \perp\!\!\!\perp \mathbf{Y} \mid \mathbf{Z}) \wedge (\mathbf{X} \perp\!\!\!\perp \mathbf{Y} \mid \mathbf{Z} \cup \gamma) \wedge (\mathbf{X} \not\perp\!\!\!\perp \gamma \mid \mathbf{Z}) &\implies (\gamma \perp\!\!\!\perp \mathbf{Y} \mid \mathbf{Z})\\
(\mathbf{X} \perp\!\!\!\perp \mathbf{Y} \mid \mathbf{Z}) \wedge (\mathbf{X} \perp\!\!\!\perp \mathbf{Y} \mid \mathbf{Z} \cup \gamma) \wedge (\gamma \not\perp\!\!\!\perp \mathbf{Y} \mid \mathbf{Z}) &\implies (\mathbf{X} \perp\!\!\!\perp \gamma \mid \mathbf{Z})\\
(\mathbf{X} \perp\!\!\!\perp \mathbf{Y} \mid \mathbf{Z} \cup \gamma) \wedge (\gamma \not\perp\!\!\!\perp \mathbf{Y} \mid \mathbf{Z}) \wedge (\mathbf{X} \not\perp\!\!\!\perp \gamma \mid \mathbf{Z}) &\implies (\mathbf{X} \not\perp\!\!\!\perp \mathbf{Y} \mid \mathbf{Z})\\
(\mathbf{X} \perp\!\!\!\perp \mathbf{Y} \mid \mathbf{Z}) \wedge (\gamma \not\perp\!\!\!\perp \mathbf{Y} \mid \mathbf{Z}) \wedge (\mathbf{X} \not\perp\!\!\!\perp \gamma \mid \mathbf{Z}) &\implies (\mathbf{X} \not\perp\!\!\!\perp \mathbf{Y} \mid \mathbf{Z} \cup \gamma).
\end{aligned}$$

To obtain decomposed rules we apply the Decomposition axiom to every single-headed rule to produce only propositions over singletons. To illustrate, consider the Intersection axiom:

$$(\mathbf{X} \perp\!\!\!\perp \mathbf{Y} \mid \mathbf{Z} \cup \mathbf{W}) \wedge (\mathbf{X} \perp\!\!\!\perp \mathbf{W} \mid \mathbf{Z} \cup \mathbf{Y}) \implies (\mathbf{X} \perp\!\!\!\perp \mathbf{Y} \cup \mathbf{W} \mid \mathbf{Z}).$$

In the above the consequent coincides with the antecedent of the Decomposition axiom, and we thus replace the Intersection axiom with a decomposed version:

$$(\mathbf{X} \perp\!\!\!\perp \mathbf{Y} \mid \mathbf{Z} \cup \mathbf{W}) \wedge (\mathbf{X} \perp\!\!\!\perp \mathbf{W} \mid \mathbf{Z} \cup \mathbf{Y}) \implies (\mathbf{X} \perp\!\!\!\perp \mathbf{Y} \mid \mathbf{Z}) \wedge (\mathbf{X} \perp\!\!\!\perp \mathbf{W} \mid \mathbf{Z}).$$

Finally, note that it is easy to show that this rule is equivalent to two single-headed rules, one implying $(\mathbf{X} \perp\!\!\!\perp \mathbf{Y} \mid \mathbf{Z})$ and the other implying $(\mathbf{X} \perp\!\!\!\perp \mathbf{W} \mid \mathbf{Z})$.

The result of the application of the above procedures is a set of single-headed, decomposed rules Ψ'. We construct, for each such rule $(\Phi_1 \wedge \Phi_2 \ldots \wedge \Phi_n \implies \varphi) \in \Psi'$ and for each subset of Σ that matches exactly the set of antecedents, that is, each subset $\{\varphi_1, \varphi_2 \ldots, \varphi_n\}$ of Σ such that $\Phi_1 \equiv \varphi_1, \Phi_2 \equiv \varphi_2 \ldots \Phi_n \equiv \varphi_n$, the argument $(\{\varphi_1 \wedge \varphi_2 \wedge \ldots \wedge \varphi_n\}, \varphi)$, and add it to $\mathcal{A}$.[1]

IKBs can be augmented with a set of preferences that allow one to take into account the reliability of each test when deciding on the truth value of independence queries. This is described in the next section.

3.2 Preference-based Argumentation Framework

Following Amgoud and Cayrol (2002), we now refine the argumentation framework of Dung (1995) for cases where it is possible to define a preference order Π over arguments.

Definition 11. *A* preference-based argumentation framework *(**PAF**) is a triplet $\langle \mathcal{A}, \mathcal{R}, \Pi \rangle$ where $\mathcal{A}$ is a set of arguments, $\mathcal{R} \subseteq \mathcal{A} \times \mathcal{A}$ is a binary relation representing a defeat relationship between pairs of arguments, and Π is a (partial or total) order over $\mathcal{A}$.*

For the case of inconsistent knowledge bases, preference Π over arguments follows the preference π over their support, that is, stronger support implies a stronger argument, which is given as a partial or total order over sets of propositions. Formally:

Definition 12. *Let $\mathcal{K} = \langle \Sigma, \Psi \rangle$ be a knowledge base, π be a (partial or total) order on subsets of Σ and (H, h), (H', h') two arguments over $\mathcal{K}$. Argument (H, h) is* π-preferred *to (H', h') (denoted $(H, h) \gg_\pi (H', h')$) if and only if H is preferred to H' with respect to π.*

In what follows we overload our notation by using π to denote either the ordering over arguments or over their supports.

An important sub-class of preference relations is the *strict* and *transitive* preference relation, defined as follows.

Definition 13. *We say that preference relation π over arguments is* strict *if the order of arguments induced by it is strict and total, that is, for every pair of arguments a and b,*

$$\left(a \gg_\pi b\right) \iff \neg\left(b \gg_\pi a\right).$$

1. This is equivalent to propositionalizing the set of rules, which are first-order (the rules of Eqs. (5) and (6) are universally quantified over all sets of variables, and thus are the rules in Ψ'). As this may be expensive (exponential in the number of propositions), in practice it is not implemented in this way; instead, appropriate rules are matched on the fly during the argumentation inference process.

Definition 14. *We say that preference relation* π *over arguments is* transitive *if, for every three arguments a, b and c,*

$$\big(a \gg_\pi b\big) \wedge \big(b \gg_\pi c\big) \implies \big(a \gg_\pi c\big).$$

The importance of the properties of strictness and transitivity will become clear later when we talk about the correctness of the argumentative independence test (defined later in Section 4).

We now introduce the concept of *attack* relation, a combination of the concepts of defeat and preference relation.

Definition 15. *Let* $\langle \mathcal{A}, \mathcal{R}, \pi \rangle$ *be a PAF, and* $a, b \in \mathcal{A}$ *be two arguments. We say* b attacks a *if and only if* $b \ \mathcal{R}\ a$ *and* $\neg(a \gg_\pi b)$.

We can see that the attack relation is a special case of the defeat relation and therefore the same conclusions apply; in particular Theorem 7, which allows us to compute the set of acceptable arguments of a PAF using Algorithm 1 or Algorithm 2.

In Sections 3.3 and 4 below, we apply these ideas to construct an approximation to the independence-query oracle that is more reliable than a statistical independence test.

3.3 Preference-based Argumentation in Independence Knowledge Bases

We now describe how to apply the preference-based argumentation framework of Section 3.2 to improve the reliability of conditional independence tests conducted on a (possibly small) data set. A preference-based argumentation framework has three components. The first two, namely $\mathcal{A}$ and $\mathcal{R}$, are identical to the general argumentation framework. We now describe how to construct the third component, namely the preference order π over subsets H of Σ, in IKBs. We define it using a belief estimate $\nu(H)$ that all propositions in H are correct,

$$H \gg_\pi H' \iff \nu(H) > \nu(H') \vee \big[\nu(H) = \nu(H') \wedge f(H, H')\big]. \tag{10}$$

That is, H is preferred over H' if and only if its belief of correctness is higher than that of H' or, in the case that these beliefs are equal, we break the tie using predicate f. For that we require that

$$\forall H, H' \subseteq \mathcal{A}, \text{ such that } H \neq H', \ f(H, H') = \neg f(H', H). \tag{11}$$

In addition, we require that f be transitive, that is, $f(H,H') \wedge f(H',H'') \Longrightarrow f(H,H'')$. This implies that the preference relation π is transitive, which is a necessary condition for proving a number of important theorems in Appendix A. In our implementation we resolved ties by assuming an arbitrary order of the variables in the domain, determined at the beginning of the algorithm and maintained fixed during its entire execution. Based on this ordering, $f(H,H')$ resolved ties by the lexicographic order of the variables in H and H'. By this definition, our f is both non-commutative and transitive.

Before we define $\nu(H)$ we first show that π, as defined by Eqs. (10) and (11) and for any definition of $\nu(H)$, satisfies two important properties, namely strictness (Definition 13) and transitivity (Definition 14). We do this in the following two lemmas.

Lemma 16. *The preference relation for independence knowledge bases defined by Equations (10) and (11) is strict.*

Proof

$$\begin{aligned}
& H \gg_\pi H' \\
\iff\; & \nu(H) > \nu(H') \vee \big[\nu(H) = \nu(H') \wedge f(H,H')\big] && \textbf{[By Eq. (10)]} \\
\iff\; & \nu(H) \geq \nu(H') \wedge \big[\nu(H) > \nu(H') \vee f(H,H')\big] && \textbf{[Distributivity of } \vee \textbf{ over } \wedge\textbf{]} \\
\iff\; & \neg\big\{\nu(H') > \nu(H) \vee \big[\nu(H') \geq \nu(H) \wedge f(H',H)\big]\big\} && \textbf{[Double negation and Eq. (11)]} \\
\iff\; & \neg\big\{\big[\nu(H') > \nu(H) \vee \nu(H') \geq \nu(H)\big] \wedge \big[\nu(H') > \nu(H) \vee f(H',H)\big]\big\} \\
\iff\; & \neg\big\{\nu(H') \geq \nu(H) \wedge \big[\nu(H') > \nu(H) \vee f(H',H)\big]\big\} \\
\iff\; & \neg\big\{\big[\nu(H') > \nu(H) \vee \nu(H') = \nu(H)\big] \wedge \big[\nu(H') > \nu(H) \vee f(H',H)\big]\big\} \\
\iff\; & \neg\big\{\nu(H') > \nu(H) \vee \big[\nu(H') = \nu(H) \wedge f(H',H)\big]\big\} && \textbf{[Common factor } \nu(H') > \nu(H)\textbf{]} \\
\iff\; & \neg(H' \gg_\pi H) && \textbf{[Again by Eq. (10)]}
\end{aligned}$$

■

Lemma 17. *The preference relation defined by Equations (10) and (11) is transitive.*

Proof

$$\begin{aligned}
& H \gg_\pi J \;\wedge\; J \gg_\pi K \\
\iff\; & \Big\{\nu(H) > \nu(J) \;\vee\; \big[\nu(H) = \nu(J) \;\wedge\; f(H,J)\big]\Big\} \\
& \wedge \Big\{\nu(J) > \nu(K) \;\vee\; \big[\nu(J) = \nu(K) \;\wedge\; f(J,K)\big]\Big\} && \textbf{[By Eq. (10)]} \\
\iff\; & \big[\nu(H) > \nu(J) \;\wedge\; \nu(J) > \nu(K)\big] && \textbf{[Case A]} \\
& \vee \big[\nu(H) > \nu(J) \;\wedge\; \nu(J) = \nu(K) \;\wedge\; f(J,K)\big] && \textbf{[Case B]} \\
& \vee \big[\nu(H) = \nu(J) \;\wedge\; f(H,J) \;\wedge\; \nu(J) > \nu(K)\big] && \textbf{[Case C]} \\
& \vee \big[\nu(H) = \nu(J) \;\wedge\; f(H,J) \;\wedge\; \nu(J) = \nu(K) \;\wedge\; f(J,K)\big] && \textbf{[Case D]}
\end{aligned}$$

To complete the proof we show that each of the cases A, B, C and D implies $H \gg_\pi K$.

(Case A) $\nu(H) > \nu(J) \;\wedge\; \nu(J) > \nu(K) \Longrightarrow \nu(H) > \nu(K) \Longrightarrow H \gg_\pi K$.

(Case B) $\nu(H) > \nu(J) \;\wedge\; \nu(J) = \nu(K) \;\wedge\; f(J,K) \Longrightarrow \nu(H) > \nu(K) \Longrightarrow H \gg_\pi K$.

(Case C) $\nu(H) = \nu(J) \;\wedge\; f(H,J) \;\wedge\; \nu(J) > \nu(K) \Longrightarrow \nu(H) > \nu(K) \Longrightarrow H \gg_\pi K$.

(Case D)

$$\begin{aligned}
\nu(H) = \nu(J) \;\wedge\; f(H,J) \;\wedge\; \nu(J) = \nu(K) \;\wedge\; f(J,K) \quad &\Longrightarrow \quad \nu(H) = \nu(K) \;\wedge\; f(H,K) \\
&\Longrightarrow \quad H \gg_\pi K,
\end{aligned}$$

due to the transitivity of predicate f.

■

We now return to the computation of $\nu(H)$. We estimate the belief $\nu(H)$ that a set of propositions H is correct by assuming independence among these propositions.[2] Overloading notation and denoting by $\nu(h)$ the probability of an individual proposition h being correct, the probability of all elements in H being correct under this assumption of independence is

$$\nu(H) = \prod_{h \in H} \nu(h). \tag{12}$$

The belief that a proposition stating independence is correct can be computed in different ways, depending on the particular choice of independence oracle chosen. In this paper we use Wilk's G^2 test, but the resulting belief can be easily adapted to any other independence oracle that produces p-values. We hope that the following discussion serves as a starting point for others to adapt it to other types of independence oracles.

As discussed in Section 2, the p-value $p(X,Y \mid \mathbf{Z})$ computed by this test is the probability of error in rejecting the null hypothesis (conditional independence in our case) and assuming that X and Y are dependent. Therefore, the probability of a test returning dependence of being correct is

$$\nu_D(X \not\perp\!\!\!\perp Y \mid \mathbf{Z}) = 1 - p(X,Y \mid \mathbf{Z})$$

where the subscript D indicates that this expression is valid only for dependencies. Formally, the error of falsely rejecting the null hypothesis is called a *type I error*. To determine the preference of a test returning independence we can, in principle, use this procedure symmetrically: use the probability of error in falsely accepting the null hypothesis (again, this is conditional independence), called a *type II error*, which we denote by $\beta(X,Y \mid \mathbf{Z})$. In this case we can define the preference of independence $(X \perp\!\!\!\perp Y \mid \mathbf{Z})$ as the probability of correctly assuming independence by

$$\nu_I(X \perp\!\!\!\perp Y \mid \mathbf{Z}) = 1 - \beta(X,Y \mid \mathbf{Z})$$

where again the subscript I indicates that it is valid only for independences. Unfortunately value of β cannot be obtained without assumptions, because it requires the computation of the probability of the test statistic under the hypothesis of dependence, and there are typically an infinite number of dependent models. In statistical applications, the β value is commonly approximated by assuming one particular dependence model if prior knowledge about that is available. In the absence of such information however in this paper we estimate it using a heuristic function of the p-value, assuming the following heuristic constraints on β:

$$\beta(X,Y \mid \mathbf{Z}) = \begin{cases} 1 & \text{if } p(X,Y \mid \mathbf{Z}) = 0 \\ \alpha - \frac{\alpha}{2+|\mathbf{Z}|} & \text{if } p(X,Y \mid \mathbf{Z}) = 1 \\ \alpha & \text{if } p(X,Y \mid \mathbf{Z}) = \alpha. \end{cases}$$

The first constraint (for $p(X,Y \mid \mathbf{Z}) = 0$) corresponds to the intuition that when the p-value of the test is close to 0, the test statistic is very far from its value under the model that assumes independence, and thus we would give more preference to the "dependence" decision. The intuition for

2. The assumption of independence is a heuristic, and is made mainly due to the difficulty of determining the dependence between two or more statistical tests evaluated on the same data set. Other possible ways of defining the preference of a set of propositions are possible. The problem of dealing with multiple tests is an open problem and an area of active research in statistics; see Section 1 for a discussion.

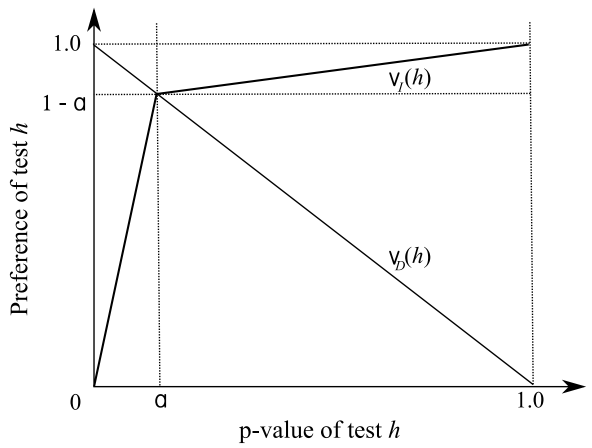


Figure 1: Preference functions $\nu_I(h)$ and $\nu_D(h)$ for statements of independence and dependence respectively, as functions of the p-value of test h.

the second case ($p(X,Y \mid \mathbf{Z}) = 1$) is reversed—when the value of the statistic is very close to the expected one under independence then independence is preferred. The value of the second case is tempered by the number of variables in the conditioning set. This reflects the practical consideration that, as the number $2+|\mathbf{Z}|$ of variables involved in the test increases, given a fixed data set, the discriminatory power of the test diminishes as $|\mathbf{Z}| \to \infty$. The third case causes the two functions ν_I and ν_D to intersect at p-value α. This is due to fairness: in the absence of non-propositional arguments (i.e., in the absence of inference rules in our knowledge base), the independence decisions of the argumentation framework should match those of the purely statistical tests, that is, "dependence" if and only if (p-value $\leq \alpha$). If instead we chose a different intersection point, then the resulting change in the outcome of tests may have been simply due to bias in the independence decision that favors dependence or independence, that is, equivalent to an arbitrary change of the threshold of the statistical test, and the comparison of the statistical and the new test based on argumentation would not be a fair one. The remaining values of β are approximated by linear interpolation among the above points. The result is summarized in Fig. 1, which depicts preference functions ν_D and ν_I with respect to the p-value of the corresponding statistical test.

Let us illustrate how the preference-based argumentation can be used to resolve the inconsistencies of Example 1.

Example 3. *In example 1 we considered an IKB with the following propositions*

$$(0 \perp\!\!\!\perp 1 \mid 2) \tag{13}$$
$$(0 \not\perp\!\!\!\perp 3 \mid 2) \tag{14}$$
$$(0 \perp\!\!\!\perp 3 \mid \{1,2\}) \tag{15}$$
$$(0 \perp\!\!\!\perp 1 \mid 2) \wedge (0 \not\perp\!\!\!\perp 3 \mid 2) \implies (0 \not\perp\!\!\!\perp 3 \mid \{1,2\}). \tag{16}$$

Following the IKB construction procedure described in the previous section, propositions (13), (14) and (15) correspond to the following arguments, respectively:

$$\begin{gathered}\left(\left\{(0 \perp\!\!\!\perp 1 \mid 2)\right\}, (0 \perp\!\!\!\perp 1 \mid 2)\right)\\ \left(\left\{(0 \not\perp\!\!\!\perp 3 \mid 2)\right\}, (0 \not\perp\!\!\!\perp 3 \mid 2)\right)\\ \left(\left\{(0 \perp\!\!\!\perp 3 \mid \{1,2\})\right\}, (0 \perp\!\!\!\perp 3 \mid \{1,2\})\right) \qquad (17)\end{gathered}$$

while rule (16) corresponds to the argument

$$\left(\left\{(0 \perp\!\!\!\perp 1 \mid 2), (0 \not\perp\!\!\!\perp 3 \mid 2)\right\}, (0 \not\perp\!\!\!\perp 3 \mid \{1,2\})\right). \qquad (18)$$

Let us extend this IKB with the following preference values for its propositions and rule.

$$\begin{aligned} Pref[(0 \perp\!\!\!\perp 1 \mid 2)] &= 0.8\\ Pref[(0 \not\perp\!\!\!\perp 3 \mid 2)] &= 0.7\\ Pref[(0 \perp\!\!\!\perp 3 \mid \{1,2\})] &= 0.5.\end{aligned}$$

According to Definition (12), the preference of each argument $(\{\sigma\}, \sigma)$ *is equal to the preference value of* $\{\sigma\}$ *which is equal to the preference of* σ*, as it contains only a single proposition. Thus,*

$$\begin{aligned} Pref\left[\left(\left\{(0 \perp\!\!\!\perp 1 \mid 2)\right\}, (0 \perp\!\!\!\perp 1 \mid 2)\right)\right] &= 0.8\\ Pref\left[\left(\left\{(0 \not\perp\!\!\!\perp 3 \mid 2)\right\}, (0 \not\perp\!\!\!\perp 3 \mid 2)\right)\right] &= 0.7\\ Pref\left[\left(\left\{(0 \perp\!\!\!\perp 3 \mid \{1,2\})\right\}, (0 \perp\!\!\!\perp 3 \mid \{1,2\})\right)\right] &= 0.5.\end{aligned}$$

The preference of argument (18) equals the preference of the set of its antecedents, which, according to Eq. (12), is equal to the product of their individual preferences, that is,

$$Pref\left[\left(\left\{(0 \perp\!\!\!\perp 1 \mid 2), (0 \not\perp\!\!\!\perp 3 \mid 2)\right\}, (0 \not\perp\!\!\!\perp 3 \mid \{1,2\})\right)\right] = 0.8 \times 0.7 = 0.56.$$

Proposition (15) and rule (16) contradict each other logically, that is, their corresponding arguments (17) and (18) defeat each other. However, argument (18) is not attacked as its preference is 0.56 *which is larger than* 0.5*, the preference of its defeater argument (17). Since no other argument defeats (18), it is acceptable, and (17), being attacked by an acceptable argument, must be rejected. We therefore see that using preferences the inconsistency of Example 1 has been resolved in favor of rule (16).*

Let us now illustrate the defend relation, that is, how an argument can be defended by some other argument. The example also illustrates an alternative resolution for the inconsistency of Example 1, this time in favor of the independence proposition (15).

Example 4. *Let us extend the IKB of Example 3 with two additional independence propositions and an additional rule. The new propositions and their preference are:*

$$\begin{aligned} Pref[(0 \perp\!\!\!\perp 4 \mid \{2,3\})] &= 0.9\\ Pref[(0 \perp\!\!\!\perp 3 \mid \{2,4\})] &= 0.8\end{aligned}$$

and the new rule is:

$$(0 \perp\!\!\!\perp 4 \mid \{2,3\}) \wedge (0 \perp\!\!\!\perp 3 \mid \{2,4\}) \quad \Longrightarrow \quad (0 \perp\!\!\!\perp 3 \mid 2).$$

This rule is an instance of the Intersection axiom followed by Decomposition.

The corresponding arguments and preferences are:

$$\begin{aligned} Pref\Big[\Big(\Big\{(0 \perp\!\!\!\perp 4 \mid \{2,3\})\Big\}, (0 \perp\!\!\!\perp 4 \mid \{2,3\})\Big)\Big] &= 0.9 \\ Pref\Big[\Big(\Big\{(0 \perp\!\!\!\perp 3 \mid \{2,4\})\Big\}, (0 \perp\!\!\!\perp 3 \mid \{2,4\})\Big)\Big] &= 0.8 \end{aligned}$$

corresponding to the two propositions, and

$$Pref\Big[\Big(\Big\{(0 \perp\!\!\!\perp 4 \mid \{2,3\}), (0 \perp\!\!\!\perp 3 \mid \{2,4\})\Big\}, (0 \perp\!\!\!\perp 3 \mid 2)\Big)\Big] = 0.9 \times 0.8 = 0.72 \tag{19}$$

corresponding to the rule.

As in Example 3, argument (17) is attacked by argument (18). Let us represent this graphically using an arrow from argument a to argument b to denote that a attacks b, that is,

$$Argument\ (18) \longrightarrow Argument\ (17).$$

If the IKB was as in Example 3, (18) would had been accepted and (17) would have been rejected. However, the additional argument (19) now defeats (undercuts) (18) by logically contradicting its antecedent $(0 \not\perp\!\!\!\perp 3 \mid 2)$. *Since the preference of (19), namely* 0.72, *is larger than that of (18), namely* 0.56, *(19) attacks (18). Therefore, (19) defends all arguments that are attacked by argument (18), and in particular (17). Graphically,*

$$Argument\ (19) \longrightarrow Argument\ (18) \longrightarrow Argument\ (17).$$

Note this is not sufficient for accepting (17) as it has not been proved that its defender (19) is itself acceptable. We leave the proof of this as an exercise for the reader.

4. The Argumentative Independence Test (AIT)

The independence-based preference argumentation framework described in the previous section provides a semantics for the acceptance of arguments consisting of independence propositions. However, what we need is a procedure for a test of independence that, given as input a triplet $\sigma = (X, Y \mid \mathbf{Z})$ responds whether X is independent or dependent of Y given $\mathbf{Z}$. In other words, we need a semantics for the acceptance of propositions, not arguments. Let us consider the two propositions related to the input triplet $\sigma = (X, Y \mid \mathbf{Z})$, proposition $(\sigma = \texttt{true})$, abbreviated $\sigma_{\texttt{t}}$, and proposition $(\sigma = \texttt{false})$, abbreviated $\sigma_{\texttt{f}}$, that correspond to independence $(X \perp\!\!\!\perp Y \mid \mathbf{Z})$ and dependence $(X \not\perp\!\!\!\perp Y \mid \mathbf{Z})$ of σ, respectively. The basic idea for deciding on the independence or dependence of input triplet σ is to define a semantics for the acceptance or rejection of propositions $\sigma_{\texttt{t}}$ and $\sigma_{\texttt{f}}$ based on the acceptance or rejection of their respective propositional arguments $(\{\sigma_{\texttt{t}}\}, \sigma_{\texttt{t}})$ and $(\{\sigma_{\texttt{f}}\}, \sigma_{\texttt{f}})$. Formally,

$$\begin{aligned} (X \not\perp\!\!\!\perp Y \mid \mathbf{Z}) \text{ is accepted} \quad &\text{iff} \quad (\{(X \not\perp\!\!\!\perp Y \mid \mathbf{Z})\}, (X \not\perp\!\!\!\perp Y \mid \mathbf{Z})) \text{ is accepted, and} \\ (X \perp\!\!\!\perp Y \mid \mathbf{Z}) \text{ is accepted} \quad &\text{iff} \quad (\{(X \perp\!\!\!\perp Y \mid \mathbf{Z})\}, (X \perp\!\!\!\perp Y \mid \mathbf{Z})) \text{ is accepted.} \end{aligned} \tag{20}$$

Based on this semantics over propositions, we decide on the dependence or independence of triplet σ as follows:

$$\begin{aligned} \sigma_{\mathtt{t}} = (X \perp\!\!\!\perp Y \mid \mathbf{Z}) \text{ is accepted} &\implies (X \perp\!\!\!\perp Y \mid \mathbf{Z}) \\ \sigma_{\mathtt{f}} = (X \not\perp\!\!\!\perp Y \mid \mathbf{Z}) \text{ is accepted} &\implies (X \not\perp\!\!\!\perp Y \mid \mathbf{Z}). \end{aligned} \tag{21}$$

We call the test that determines independence in this manner the **Argumentative Independence Test** or **AIT**. For the above semantics to be well-defined, a triplet σ must be either independent or dependent, that is, not both or neither. For that, exactly one of the antecedents of the above implications of Eq. (21) must be true. Formally,

Theorem 18. *For any input triplet $\sigma = (X, Y \mid \mathbf{Z})$, the argumentative independence test (**AIT**) defined by Eqs. (20) and (21) produces a non-ambiguous decision, that is, it decides σ evaluates to either independence or dependence, but nor both or neither.*

For that to happen, one and only one of its corresponding propositions $\sigma_{\mathtt{t}}$ or $\sigma_{\mathtt{f}}$ must be accepted. A necessary condition for this is given by the following theorem.

Theorem 19. *Given a PAF $\langle \mathcal{A}, \mathcal{R}, \pi \rangle$ with a strict and transitive preference relation π, every propositional argument $(\{\sigma_{\mathtt{t}}\}, \sigma_{\mathtt{t}}) \in \mathcal{A}$ and its negation $(\{\sigma_{\mathtt{f}}\}, \sigma_{\mathtt{f}})$ satisfy*

$$(\{\sigma_{\mathtt{t}}\}, \sigma_{\mathtt{t}}) \textit{ is accepted iff } (\{\sigma_{\mathtt{f}}\}, \sigma_{\mathtt{f}}) \textit{ is rejected}.$$

The above theorem is not sufficient because the propositions may still be in abeyance, but this possibility is ruled out for strict preference relations by Theorem 20, presented in the next section.

The formal proofs of Theorems 18, 19 and 20 are presented in Appendix B. We now illustrate the use of AIT with an example.

Example 5. *We consider an extension of Example 3 to illustrate the use of the AIT to decide on the independence or dependence of input triplet $(0, 3 \mid \{1, 2\})$. According to Eq. (20) the decision depends on the status of the two propositional arguments:*

$$(\{(0 \not\perp\!\!\!\perp 3 \mid \{1,2\})\}, (0 \not\perp\!\!\!\perp 3 \mid \{1,2\})), \textit{ and} \tag{22}$$

$$(\{(0 \perp\!\!\!\perp 3 \mid \{1,2\})\}, (0 \perp\!\!\!\perp 3 \mid \{1,2\})). \tag{23}$$

Argument (23) is equal to argument (17) of Example 3 that was proved to be rejected in that example. Therefore, according to Theorem 19, its negated propositional argument Eq. (22) must be accepted, and we can conclude that triplet $(0, 3 \mid \{1, 2\})$ corresponds to a dependence, that is, we conclude that $(0 \not\perp\!\!\!\perp 3 \mid \{1, 2\})$.

5. The Top-down AIT Algorithm

We now discuss in more detail the top-down algorithm which is used to implement the argumentative independence test, introduced in Section 3. We start by simplifying the recursion of Eq. (7) that determines the state (accepted, rejected, or in abeyance) of an argument a. We then explain the algorithm and analyze its computability (i.e., prove that its recursive execution is always finite) and its time complexity.

To simplify the recursion Eq. (7) we use the following theorem (proved in Appendix B).

Theorem 20. *Let $\langle \mathcal{A}, \mathcal{R}, \pi \rangle$ be a PAF with a strict preference relation π. Then no argument $a \in \mathcal{A}$ is in abeyance.*

This theorem reduces the number of states of each argument to two, that is, an argument can be either accepted or not accepted (rejected). We will use the name of the argument a to denote the predicate "a is accepted" and its negation $\neg a$ to denote the predicate "a is rejected." With this notation, the above theorem, and the fact that we have extended the semantics of acceptability from the defeat to the attack relation (using preferences), the recursion of Eq. (7) can be expressed as follows

$$\begin{aligned} a &\iff \forall b \in attackers(a),\ \neg b \\ \neg a &\iff \exists b \in attackers(a),\ b \end{aligned}$$

or, equivalently,

$$\begin{aligned} a &\iff \bigwedge_{b \in attackers(a)} \neg b \\ \neg a &\iff \bigvee_{b \in attackers(a)} b. \end{aligned}$$

Finally, we notice that the second formula is logically equivalent to the first (simply negating both sides of the double implication recovers the first). Therefore, the Boolean value of the dialog tree for a can be computed by the simple expression

$$a \iff \bigwedge_{b \in attackers(a)} \neg b. \tag{24}$$

To illustrate, consider an attacker b of a. If b is rejected, that is, $\neg b$, the conjunction on the right cannot be determined without examining the other attackers of a. Only when all attackers of a are known to be rejected can the value of a be determined, that is, accepted. Instead, if b is accepted, that is, b, the state of $\neg b$ is `false` and the conjunction can be immediately evaluated to `false`, that is, a is rejected regardless of the acceptability of any other attackers.

An iterative version of the top-down algorithm is shown in Algorithm 3. We assume that the algorithm can access a global PAF $\langle \mathcal{A}, \mathcal{R}, \pi \rangle$, with arguments in $\mathcal{A}$ defined over a knowledge base $\mathcal{K} = \langle \Sigma, \Psi \rangle$. Given as input a triplet $t = (X, Y \mid \mathbf{Z})$, if the algorithm returns `true` (`false`) then we conclude that t is independent (dependent). It starts by creating a root node u for the propositional argument U of proposition $t = \texttt{true}$ (lines 1–6). According to Eqs. (20) and (21), the algorithm then decides `true` if U is accepted (line 22). Otherwise, the algorithm returns `false` (line 23). This is because in this case, according to Theorem 19, the negation of propositional argument U must be accepted.

Algorithm 3 is an iterative version of a tree traversal algorithm. It maintains a queue of the nodes that have not been expanded yet. A node is expanded when its children are added to the tree. In the algorithm, this is done in the loop of lines 17 to 21, which uses subroutine *getAttackers* of Algorithm 5 to obtain all attackers of an argument. This subroutine finds all attackers of the input argument a in a backward-chaining fashion, that is, given an argument $a = (H, h)$, it searches for all rules in the knowledge base $\mathcal{K}$ whose consequent matches the negation of some proposition in the

Algorithm 3 *independent*$(triplet\ t)$.

```
1: $f_{\texttt{true}} \leftarrow$ proposition $(t = \texttt{true})$    /* Creates independence proposition $(t = \texttt{true})$. */
2: $U_{\texttt{true}} \leftarrow (\{f_{\texttt{true}}\}, f_{\texttt{true}})$
3: $u_{\texttt{true}} \leftarrow$ node for argument $U_{\texttt{true}}$
4: $u_{\texttt{true}}.parent \leftarrow \texttt{nil}$
5: $u.STATE \leftarrow \texttt{nil}$
6: $fringe \leftarrow [u]$        /* Initialize with u (root). */
7: /* Create global rejected node, denoted by $\rho$. */
8: $\rho \leftarrow$ node with no argument and state rejected
9: while $fringe \neq \varnothing$ do
10:    $u \leftarrow dequeue(fringe)$
11:    $attackers \leftarrow getAttackers(u.argument)$
12:    if $(attackers = \varnothing)$ then
13:        $u.STATE \leftarrow \texttt{accepted}$
14:        if $sendMsg(\rho, u) = terminate$ then break
15:    $attackers \leftarrow$ sort attackers in decreasing order of preference.
16:    /* Enqueue attackers after decomposing them. */
17:    for each $A \in attackers$ do
18:        $a \leftarrow$ node for argument $A$
19:        $a.parent \leftarrow u$
20:        $a.STATE \leftarrow \texttt{nil}$
21:        enqueue $a$ in $fringe$        /* See details in text. */
22: if $(u.STATE = \texttt{accepted})$ then return true
23: if $(u.STATE = \texttt{rejected})$ then return false
```

Algorithm 4 *sendMsg*$(Node\ c, Node\ p)$.

```
1: /* Try to evaluate node p given new information in c.STATE */
2: if $p \neq \texttt{nil}$ then
3:    if $c.STATE = \texttt{accepted}$ then $p.STATE \leftarrow \texttt{rejected}$
4:    else if ($\forall$ children $q$ of $p$, $q.STATE \neq \texttt{rejected}$) then $p.STATE \leftarrow \texttt{accepted}$
5:    /* If p was successfully evaluated, try to evaluate its parent by sending message upward. */
6:    if $p.STATE \neq \texttt{nil}$ then
7:        return $sendMsg(p, p.parent)$
8:    else
9:        return $continue$
10: else
11:    return $terminate$ /* The root node has been evaluated. */
```

support H (undercutters), or the negation of its head h (rebutters). Every node maintains a three-valued state variable $STATE \in \{\texttt{nil}, \texttt{accepted}, \texttt{rejected}\}$. The nil state denotes that the value of the node is not yet known, and a node is initialized to this state when it is added to the tree.

The algorithm recurses down the dialog tree until a node is found that has no attackers (line 12). Such a node is accepted in line 13, that is, the conjunction of Eq. (24) is trivially true, and its *STATE* is propagated upwards toward the root to the parent using subroutine *sendMsg* (Algorithm 4). Every

Algorithm 5 Finds all attackers of input argument a in knowledge base $\mathcal{K} = \langle \Sigma, \Psi \rangle$: *getAttackers*$(a = (H, h))$

```
1:  $attackers \leftarrow \varnothing$
2:  /* Get all undercutters or rebutters of $a$. */
3:  for all propositions $\varphi \in H \cup \{h\}$ do
4:      /* Get all defeaters of proposition $\varphi$. */
5:      for all rules $(\Phi_1 \wedge \Phi_2 \ldots \wedge \Phi_n \implies \neg\varphi) \in \Psi$ do
6:          /* Find all propositionalizations of the rule whose consequent matches $\neg\varphi$. */
7:          for all subsets $\{\varphi_1, \varphi_2 \ldots, \varphi_n\}$ of $\Sigma$ s.t. $\Phi_1 \equiv \varphi_1, \Phi_2 \equiv \varphi_2 \ldots \Phi_n \equiv \varphi_n$ do
8:              $d \leftarrow (\{\varphi_1 \wedge \varphi_2 \ldots \varphi_n\}, \neg\varphi)$ /* Create defeater. */
9:              /* Is the defeater an attacker? */
10:             if $\neg(a \gg_\pi d)$ then
11:                 $attackers \leftarrow attackers \cup \{d\}$
12: return $attackers$
```

time a node receives a message from a child, if the message is `accepted`, the node is rejected (line 3 of Algorithm 4), otherwise the node is accepted if all its children has been evaluated to `rejected` (line 4 of Algorithm 4). The subroutine *sendMsg* then proceeds recursively by forwarding a message to the parent whenever a node has been evaluated (line 7). If the root is reached and evaluated, the message is sent to its parent, which is `nil`. In this case, the subroutine returns the special keyword *terminate* back to the caller, indicating that the root has been evaluated and thus the main algorithm (Algorithm 3) can terminate. The caller can be either the subroutine *sendMsg*, in which case it pushes the returned message up the method-calling stack, or the top-down algorithm in line 14, in which case its "while" loop is terminated.

An important part of the algorithm is yet underspecified, namely the order in which the attackers of a node are explored in the tree (i.e., the priority with which nodes are enqueued in line 21). This affects the order of expansion of the nodes in the dialog tree. Possible orderings are depth-first, breadth-first, iterative deepening, as well as informed searches such as best-first when a heuristic is available. In our experiments we used iterative deepening because it combines the benefits of depth-first and breadth-first search, that is, small memory requirements on the same order as depth-first search (i.e., on the order of the maximum number of children a node can have) but also the advantage of finding the shallowest solution like breadth-first search. We also used a heuristic for enqueuing the children of a node. According to iterative deepening, the position in the queue of the children of a node is specified relative to other nodes, but not relative to each other. We therefore specified the relative order of the children according to the value of the preference function: children with higher preference are enqueued first (line 15 of the top-down algorithm), and thus, according to iterative deepening, would be dequeued first.

5.1 Computability of the Top-Down Algorithm

An open question is whether the top-down algorithm is computable, that is, whether it always terminates. In this section we prove that it is. To prove this we need to show that under certain general conditions the acceptability of an argument a can always be determined, that is, that the algorithm always terminates. This is proved by the following theorem.

Theorem 21. *Given an arbitrary triplet $t = (X, Y \mid \mathbf{Z})$, and a PAF $\langle \mathcal{A}, \mathcal{R}, \pi \rangle$ with a strict preference relation π, Algorithm 3 with input t over $\langle \mathcal{A}, \mathcal{R}, \pi \rangle$ terminates.*

The proof consists on showing that the path from the root a to any leaf is always finite. For that, the concept of an attack sequence is needed.

Definition 22. *An* attack sequence *is a sequence $\langle a_1, a_2, \dots, a_n \rangle$ of n arguments such that for every $2 \leq i \leq n$, a_i attacks a_{i-1}.*

By the manner in which the top-down algorithm constructs the dialog tree it is clear that any path from the root to a leaf is an attack sequence. It therefore suffices to show that any such sequence is finite. This is done by the following theorem.

Theorem 23. *Every attack sequence $\langle a_1, a_2, \dots, a_n \rangle$ in a PAF $\langle \mathcal{A}, \mathcal{R}, \pi \rangle$ with strict π and finite $\mathcal{A}$ is finite.*

Intuitively, if the preference relation is strict then an element can attack its predecessor in the sequence but not vice versa. Since the set of arguments $\mathcal{A}$ is finite, the only way for an attack sequence to be infinite is to contain a cycle. In that case, an argument would be attacking at least one of its predecessors, which cannot happen in a PAF with a strict preference relation. We present formal proofs of Theorems 21 and 23 in Appendix A.

We thus arrived at the important conclusion that, under a strict preference function and a finite argument set, the state of any argument is computable. As we showed in Section 3.3, the preference function for independence knowledge bases is strict, and thus the computability of the top-down algorithm is guaranteed.

5.2 Computational Complexity of the Top-Down Algorithm

Since Algorithm 3 is a tree traversal algorithm, its time complexity can be obtained by techniques contained in standard algorithmic texts, for example, Cormen et al. (2001). The actual performance depends on the tree exploration procedure. In our case we used iterative deepening due to its linear memory requirements in d, where d is the smallest depth at which the algorithm terminates. Iterative deepening has a worst-time time complexity of $O(b^d)$, where b is an upper bound on the *dialog tree branching factor*. Therefore, for a constant $b > 1$ the execution time is exponential in d in the worst case. Furthermore, for the case of independence tests, b itself may also be exponential in n (the number of variables in the domain). This is because the inference rules of Eqs. (5) and (6) are universally quantified, and therefore their propositionalization (lines 7–11 of Algorithm 5), may result in an exponential number of rules with the same consequent (attackers). A tighter bound may be possible but, lacking such a bound, we introduce in the next section an approximate top-down algorithm, which reduces the running time to polynomial. As we show in our experiments, the use of this approximation does not appreciably affect the accuracy improvement due to argumentation.

6. The Approximate Top-Down AIT Algorithm

As the top-down algorithm has a theoretically exponential running time in the worst case, we hereby present a practical polynomial-time approximation of the top-down algorithm. We make use of two approximations: (a) To address the exponential behavior due to the depth of the dialog tree we apply a *cutoff depth d* for the process of iterative deepening. (b) To address the potentially exponential size of the branching factor b (which equals the maximum number of defeaters of any argument appearing in the dialog tree) we limit the number of defeaters of each node—thus bounding the number of its attackers/children—to a polynomial function of n (the domain size) during the propositionalization process of Algorithm 5 (lines 7–11). Let (H,h) be an argument and let $\varphi \in H \cup \{h\}$ be one of its propositions, as in line 3 of Algorithm 5. The set of attackers Σ_φ of (H,h) consists of all rules $\{\varphi_1 \wedge \varphi_2 \ldots \wedge \varphi_k \implies \neg\varphi\}$ of Σ, for some constant upper bound k on the size of their support. If $\varphi = (\mathbf{X},\mathbf{Y} \mid \mathbf{Z})$ and $\varphi_i = (\mathbf{X}_i,\mathbf{Y}_i \mid \mathbf{Z}_i)$ for all $1 \le i \le k$, then our approximation generates and uses a subset of Σ_φ in the dialog tree such that

$$\begin{array}{rcccl} |\mathbf{X}|-c & \le & |\mathbf{X}_i| & \le & |\mathbf{X}|+c \\ |\mathbf{Y}|-c & \le & |\mathbf{Y}_i| & \le & |\mathbf{Y}|+c \\ |\mathbf{Z}|-c & \le & |\mathbf{Z}_i| & \le & |\mathbf{Z}|+c \end{array} \tag{25}$$

where $|\cdot|$ denotes set cardinality, and c is a user-specified integer parameter that defines the approximation. We call this the **approximate top-down algorithm**. The computational complexity of the approximate top-down algorithm is polynomial in n, as shown in the next section.

6.1 Test Complexity of the Approximate Top-Down Algorithm

In this section we prove that the number of statistical tests required by the Approximate Top-Down algorithm is polynomial in n. As described in the previous section, the approximate algorithm generates a bounded number of attackers for each proposition in the argument corresponding to some node in the dialog tree. A bound on the number of the possible attackers can be defined by the approximation of Eq. (25). These equations dictate that the size of each possible set $\mathbf{X}_i$ in some proposition $(\mathbf{X}_i,\mathbf{Y}_i \mid \mathbf{Z}_i)$ of some attacker of proposition $(\mathbf{X},\mathbf{Y} \mid \mathbf{Z})$ is between $|\mathbf{X}|+c$ and $|\mathbf{X}|-c$ (inclusively). As the number of elements that can be members of $\mathbf{X}_i$ is bounded by n (the domain size), this produces at most n^{2c+1} possible instantiations for set $\mathbf{X}_i$. Similarly, the number of possible instantiations for $\mathbf{Y}_i$ and $\mathbf{Z}_i$ is also n^{2c+1}. Therefore, an upper bound for the number of matches to some proposition in the antecedent of an attacking rule is $O(n^{6c+3})$ for some constant c. As there are r rules in the rule set and up to k propositions in each rule for some constants r and k (for example, $r = 5$ and $k = 3$ for Eq. (5) and $r = 8$ and $k = 4$ for Eq. (6)), an upper bound on the number of children of a node in the dialog tree is $O(rkn^{6c+3})$, and thus an upper bound on the number of nodes in the dialog tree of depth d is $O((rk)^d n^{d(6c+3)})$. As we demonstrate in our experiments, this is a rather loose upper bound and the performance of the approximate top-down algorithm is reasonable in practice, but it does serve to show that the theoretical worst-case performance is polynomial in n. In the experiments shown in the next section we used $c = 1$ and $d = 3$.

7. Experimental Results

We conducted experiments on sampled and real-world data sets for the purpose of (a) evaluating the accuracy improvement of the argumentative test (both the exact and approximate versions) over its statistical counterpart; (b) demonstrating the performance improvements that can be achieved by the approximate version compared to the exact counterpart, without significant reduction in accuracy improvement; and (c) evaluating the improvements that result by the use of the argumentative framework for causal discovery. We address these issues below.

7.1 Comparative Evaluation of Bottom-Up, Exact Top-Down, and Approximate Top-Down Argumentative Tests

In this section we demonstrate that the argumentation approach, implemented either by the (exact) bottom-up or the exact top-down algorithm (Algorithm 3), improves the accuracy of independence tests on small data sets. We also show that the approximate top-down algorithm (see Section 6) has accuracy performance improvements similar to its exact counterpart but significantly better execution times (orders of magnitude), that make it more practical and usable for larger domains. As the output of the bottom-up algorithm is guaranteed to be equal to the exact top-down algorithm as Theorem 9 of Section 3, we omit accuracy results for the bottom-up algorithm here.

As the exact algorithm is impractical for large domains, for the present comparison we sampled data sets from two randomly generated Bayesian networks with $n = 8$ nodes. The networks were generated using *BNGenerator* (Ide et al., 2002), a publicly available Java package, with maximum degree per node τ equal to 3 and 7 to evaluate the performance in sparsely as well as densely connected domains. A large data set D was sampled from each network and our experiments were conducted on subsets of it containing an increasing number of data points N. This was done in order to assess the accuracy on varying conditions of reliability, as the reliability of a test varies (typically increases) with the amount of data available. To reduce variance, each experiment was repeated for ten data subsets of equal size, obtained by permuting the data points of D randomly and using the first N of them as input to our algorithms.

We first compare the accuracy of argumentative tests versus their purely statistical counterparts (i.e., the G^2 test) on several data sets sampled from randomly generated Bayesian networks. Sampled data experiments have the advantage of a more precise estimation of the accuracy since the underlying model is known. We present experiments for two versions of the exact top-down argumentative test, one using Pearl's general axioms of of Eq. (5), denoted $\mathbf{AIT_t\text{-}G}$, and another that uses Pearl's "directed" axioms of Eq. (6), denoted $\mathbf{AIT_t\text{-}D}$, as well as two versions of the approximate top-down argumentative test, denoted $\widehat{\mathbf{AIT}}_\mathbf{t}\mathbf{\text{-}G}$ and $\widehat{\mathbf{AIT}}_\mathbf{t}\mathbf{\text{-}D}$ respectively. We abbreviate purely statistical independence tests as **SIT**.

We report the estimated accuracy, which, for each data set, is calculated by comparing the result of a number of conditional independence tests (SITs or AITs) on data with the true value of independence, computed by querying the underlying model for the conditional independence of the same test using d-separation. Since the number of possible tests is exponential, we estimated the independence accuracy by randomly sampling a set $\mathcal{T}$ of 1,000 triplets $(X,Y,\mathbf{Z})$, evenly distributed among all possible conditioning set sizes $m \in \{0,\ldots,n-2\}$, that is, $1000/(n-1)$ tests for each m. The independence or dependence value of the triplets (in the true, underlying model) were not controlled, but left to be decided randomly. This resulted in a non-uniform distribution of dependencies and independences. For instance, in the experiments shown next ($n = 8, \tau = 3, 7$), the average proportion of independences vs. dependencies was 36.6% to 63.4% respectively for $\tau = 3$, and 11.4% to 88.6% respectively for $\tau = 7$. Denoting a triplet in $\mathcal{T}$ by t, by $I_{\text{true}}(t)$ the result of a test on t performed on the underlying model, and by $I_{\text{data-}\mathcal{Y}}(t)$ the results of performing a test on t of type $\mathcal{Y}$ on data, for $\mathcal{Y}$ equal to SIT, AIT_t-G, AIT_t-D, $\widehat{\text{AIT}}_\text{t}$-G, or $\widehat{\text{AIT}}_\text{t}$-D, the estimated accuracy of test type $\mathcal{Y}$ is defined as

$$\widehat{acc}_{\mathcal{Y}}^{\text{data}} = \frac{1}{|\mathcal{T}|}\left|\left\{t \in \mathcal{T} \mid I_{\text{data-}\mathcal{Y}}(t) = I_{\text{true}}(t)\right\}\right|.$$

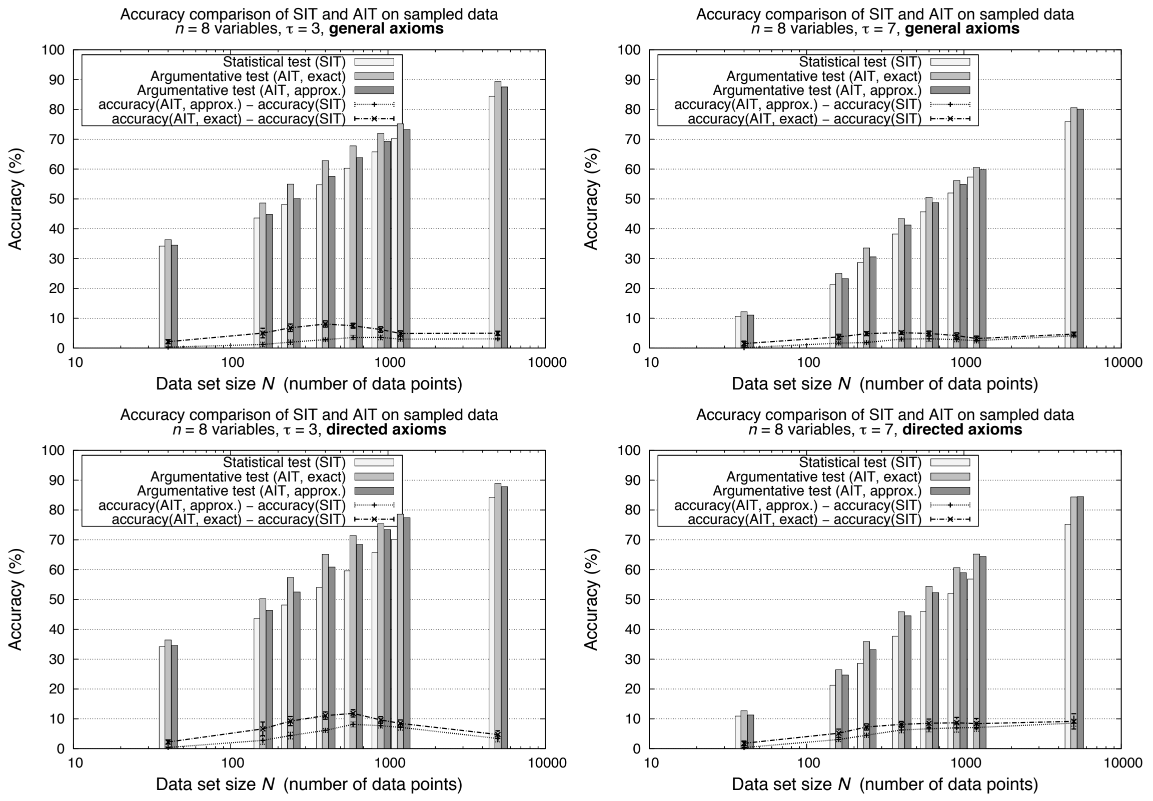


Figure 2: Accuracy comparison of statistical tests (SIT) vs. exact and approximate argumentative tests for domain size $n = 8$ and maximum degree per node $\tau = 3, 7$. The histograms show the absolute value of the accuracy while the line curves show the difference between SIT and the argumentative tests. 95% confidence intervals are also shown for the line graphs. **Top row:** General axioms. **Bottom row:** Directed axioms.

Figure 2 (top row) shows a comparison of the SIT with the exact and approximate top-down argumentative test over the general axioms for data set with increasing number of data points. The figure shows two plots for $\tau = 3, 7$ of the mean values (over runs for ten different data sets) of $\widehat{acc}^{\text{data}}_{\text{SIT}}$, $\widehat{acc}^{\text{data}}_{\text{AIT}_t\text{-G}}$, and $\widehat{acc}^{\text{data}}_{\widehat{\text{AIT}_t}\text{-G}}$ (histograms) and the difference between the accuracies of the AIT tests and the statistical one (line graphs) for various data set sizes N. A positive value of the difference corresponds to an improvement of the argumentative test over SIT. The plots also show the statistical significance of this difference with 95% confidence intervals (error bars), that is, the interval around the mean value that has a 0.95 probability of containing the true difference. The figure demonstrates that there exist modest but consistent and statistically significant improvements in the accuracy of both the exact and approximate argumentative tests over the statistical test. We can observe improvements over the entire range of data set sizes in both cases with maximum improvements of up to 9% and 6% for the exact and approximate cases respectively (at $\tau = 3$ and $N = 600$).

In certain situations where the experimenter knows that the underlying distribution belongs to the class of Bayesian networks, it is appropriate to use the specific axioms of Eq. (6) instead of the general axioms of Eq. (5). The bottom row of Figure 2 presents the same comparison as the top row but for the exact and approximate argumentative tests AIT_t-D and $\widehat{\mathrm{AIT}}_t$-D that use the directed axioms instead of the general ones. As in the case for AIT using the general axioms, we can observe statistically significant improvements over the entire range of data set sizes in both cases. In this case however, the improvements are larger, with maximum increases in the accuracy of the exact and approximate test of up to 13% and 9% respectively (again for $\tau = 3$ and $N = 600$).

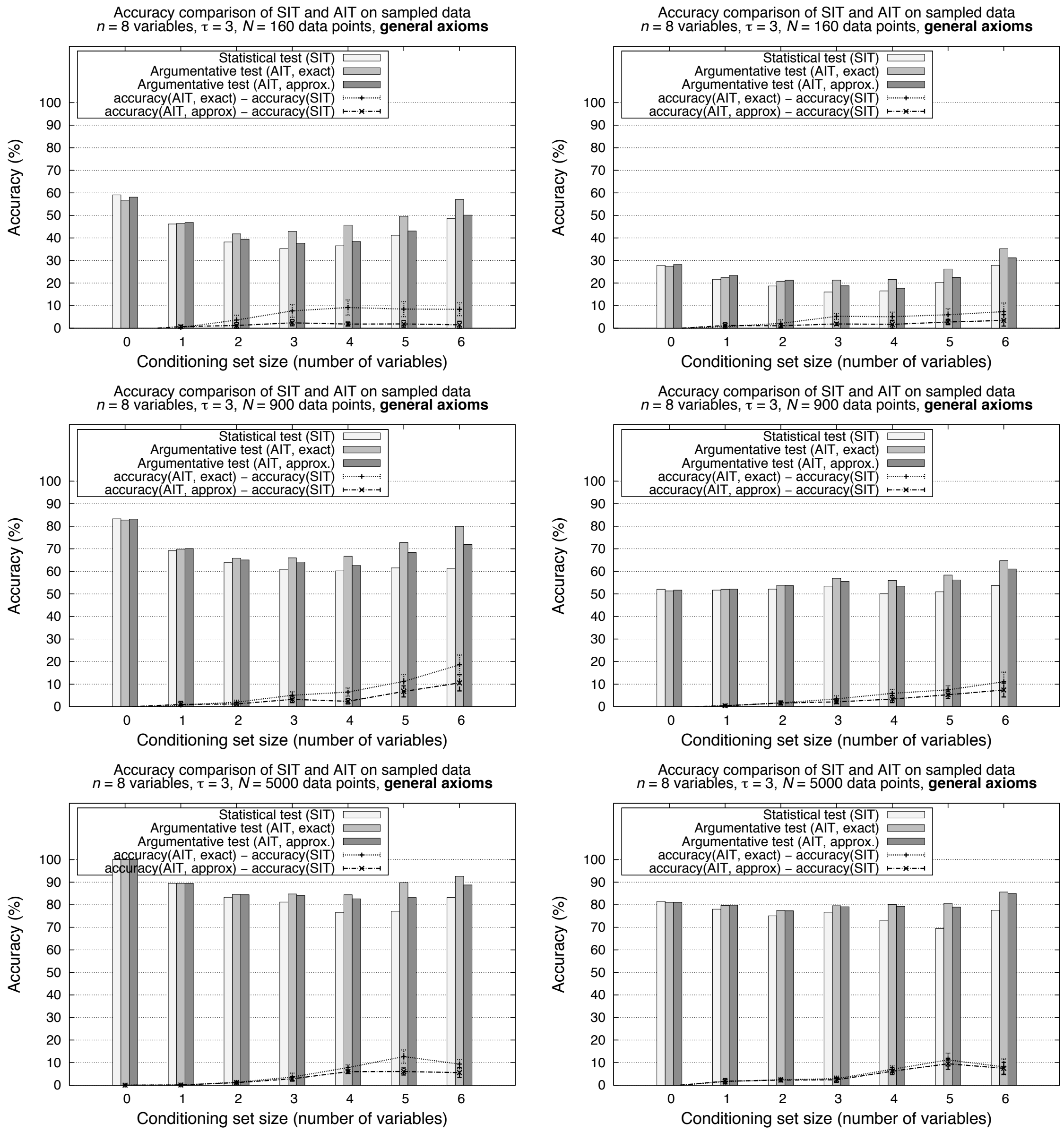


Figure 3: Accuracy comparison of SIT vs. exact (AIT_t-G) and approximate ($\widehat{\mathrm{AIT}}_t$-G) argumentative tests over the general axioms for increasing conditioning set sizes. The six plots correspond to maximum degrees per node $\tau = 3, 7$, and data set sizes $N = 160, 900$ and 5000.

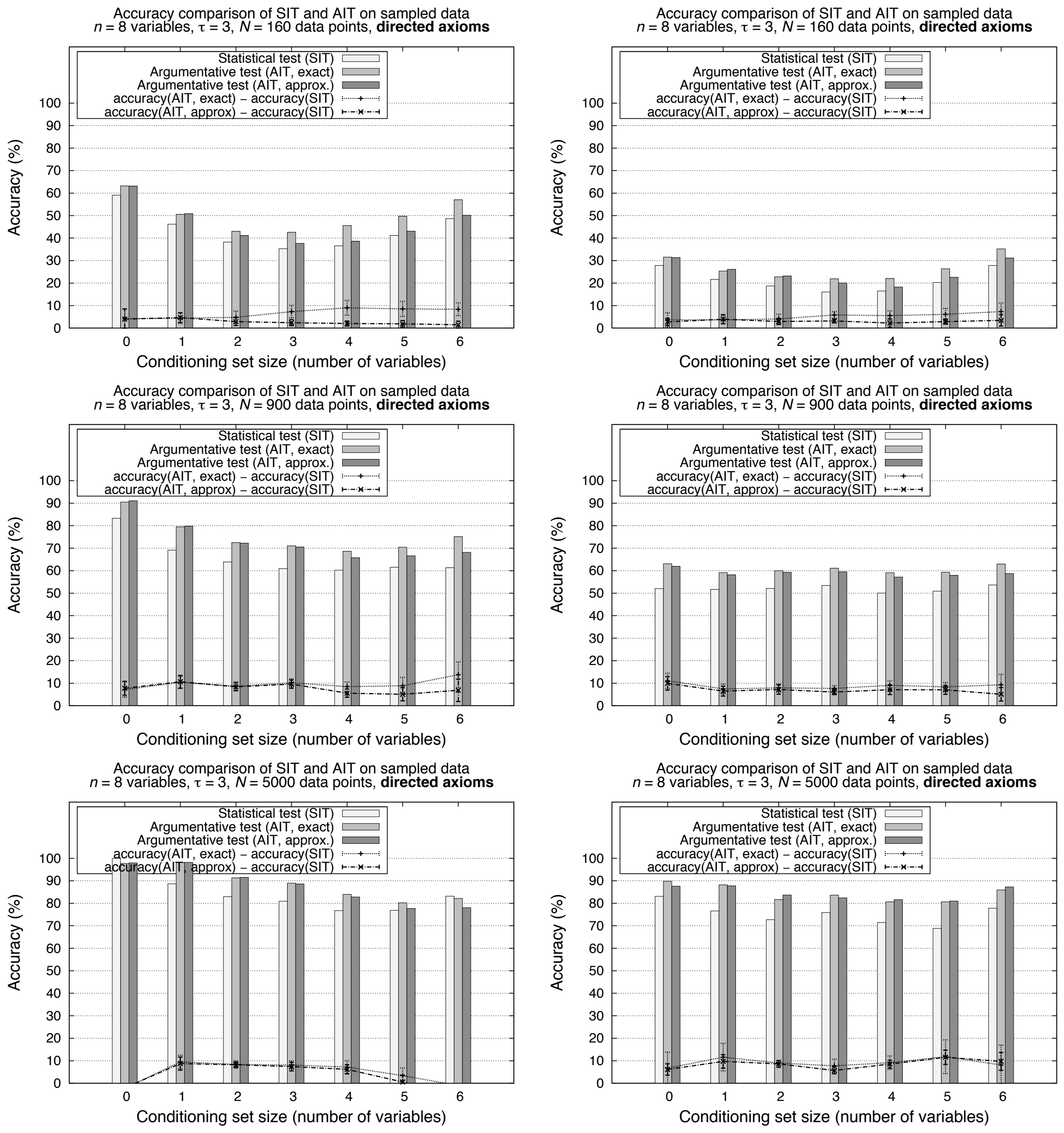


Figure 4: Same as Figure 3 but for AIT using the directed axioms instead of the general ones.

We also evaluated the accuracy of these tests for increasing conditioning set sizes. Figures 3 and 4 show a comparison of the SIT with the exact and approximate top-down argumentative test using the general and directed axioms respectively, for accuracies over increasing conditioning set size for data sizes $N = 160, 900$, and 5000 (different rows). We can observe statistically significant improvements over the entire range of conditioning set sizes in all twelve graphs. Except in one case (directed axioms, $N = 5000, \tau = 3$), all graphs show an upward trend in accuracy for increasing conditioning set size, representing a positive impact of the argumentative approach that increases with a decrease in test reliability, that is, increasing conditioning set size.

We also compared the execution times of the bottom-up, exact top-down and approximate top-down algorithms on the same data sets. To run the bottom-up algorithm we generated the set of all propositional arguments possible, that is, arguments of the form $(\{\sigma\},\sigma)$, by iterating over all possible triplets $(\mathbf{X},\mathbf{Y} \mid \mathbf{Z})$, and inserted them in the knowledge base together with their preference, as described in Section 3.1. Similarly, for the set of axioms that we used in each case, that is, either the general (Eq. (5)) or the specific ones (Eq. (6)), we iterated over all possible matches of each rule, inserting the corresponding (single-headed and decomposed) instantiated rule in the knowledge base again together with its preference. The reason for including all propositional and rule-based arguments in our IKB is to allow the argumentation framework to consider all possible arguments in favor of or against an independence query. We compared the bottom-up algorithm AIT_b, the exact top-down algorithms AIT_t, and the approximate top-down algorithm $\widehat{\mathrm{AIT}}_\mathrm{t}$. For this, we measured the time it takes to discover the structure of a Bayesian networks using three versions of the PC algorithm (Spirtes et al., 2000), each using one of the three argumentative tests AIT_b, AIT_t, or $\widehat{\mathrm{AIT}}_\mathrm{t}$ to conduct the independence tests. As usual, we consider two versions of each test AIT_b, AIT_t, and $\widehat{\mathrm{AIT}}_\mathrm{t}$, one that uses the general axioms of Eq. (5), that is, AIT_b-G, AIT_t-G, and $\widehat{\mathrm{AIT}}_\mathrm{t}$-G, respectively, and one that uses the specific axioms of Eq. (6) (applicable to Bayesian networks), that is, AIT_b-D, AIT_t-D, and $\widehat{\mathrm{AIT}}_\mathrm{t}$-D, respectively. The data sets used are the same as the ones used in the accuracy comparisons above.

Figure 5 plots the execution time of argumentative tests AIT_b-G vs. AIT_t-G vs. $\widehat{\mathrm{AIT}}_\mathrm{t}$-G (top row) and AIT_b-D vs. AIT_t-D vs. $\widehat{\mathrm{AIT}}_\mathrm{t}$-D (bottom row) for tests that were conducted by the PC algorithm while learning the structure. Note that both the x and y-axes are plotted in log-scale. We can observe improvements in the execution time of the exact top-down algorithm over that of the bottom-up algorithm of an order of magnitude over the entire range of data set sizes in all four plots. We can also see improvement of a similar order between the exact and approximate top-down argumentative algorithms. For instance, for the general axioms and $\tau = 3$ (top-left plot), the execution time for $N = 5000$ is 2749 seconds for the bottom-up against 107 seconds for the exact top-down and 15 seconds for the approximate top-down algorithm. We see even more pronounced execution time improvements when using the directed axioms (bottom row of Fig. 5).

The execution-time results demonstrate that the exact top-down algorithm performs significantly better than the bottom-up algorithm, while producing the exact same output (according to Theorem 9 of Section 3). This implies a clear advantage of using the top-down over the bottom-up algorithm. Furthermore, we also saw that the approximate top-down algorithm performs similarly in terms of accuracy improvement while having polynomial worst-case execution time and in practice being several orders of magnitude faster than the exact top-down algorithm, which is exponential in the worst-case. As in the next two sections we continue our evaluation on domains significantly larger than the $n = 8$ variables that we examined here, it would be difficult or impractical for the exact algorithms to be employed. For these reasons in the following experiments we use the more practical approximate algorithm, which can be applied to larger domains.

7.2 Causal Discovery in Larger Domains

We also conducted experiments that demonstrate the performance of the approximate top-down algorithm by (a) showing its applicability to large domains, and (b) demonstrating positive improvements in accuracy of argumentative tests on the learning of the structure of Bayesian networks, the main problem faced by causal discovery algorithms. In the following experiments we used the PC

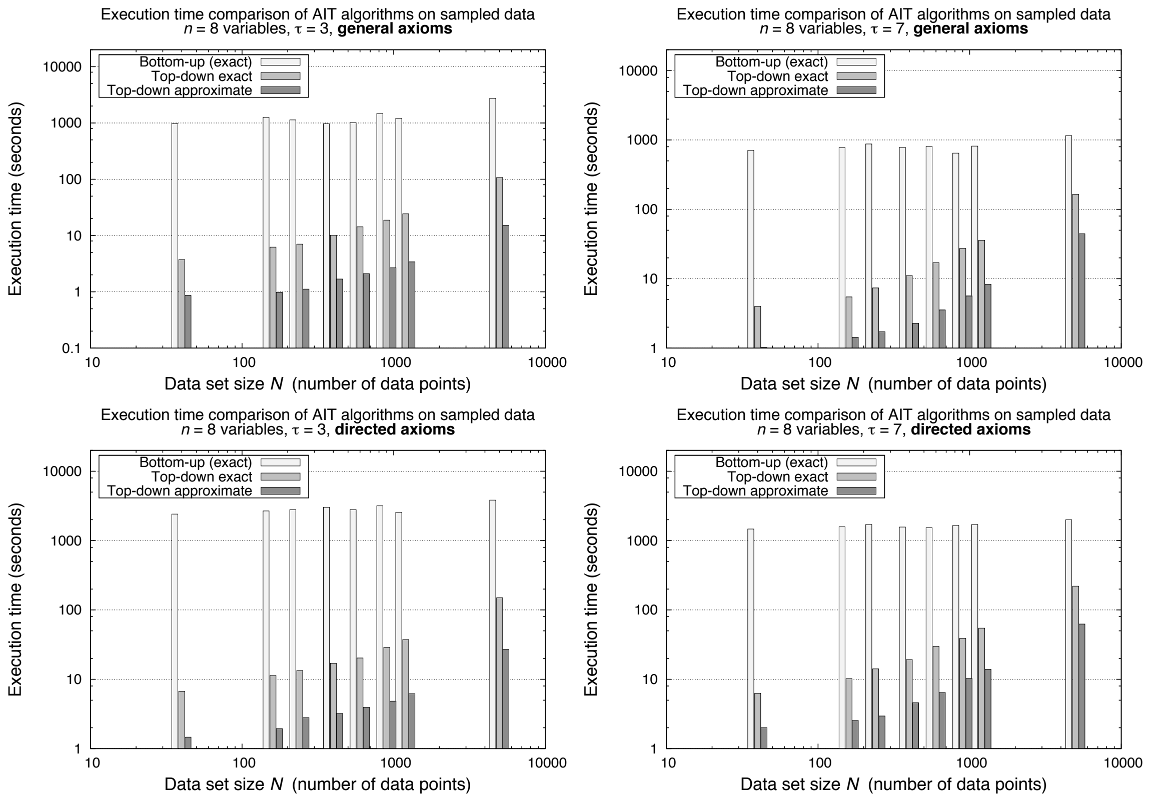


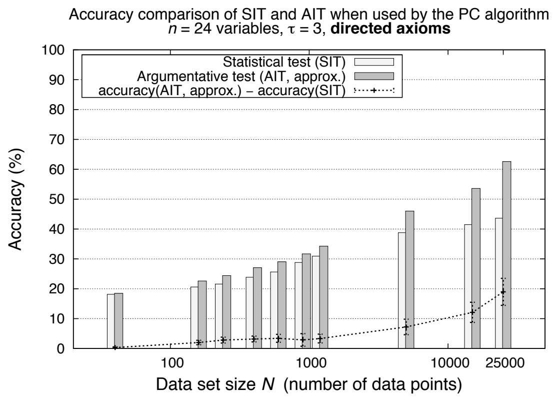
Figure 5: Execution time comparison for the PC algorithm when it uses the bottom-up and exact top-down and approximate top-down argumentative tests to learn the structure of a Bayesian network from data sampled from Bayesian models with domain size $n = 8$, maximum degrees $\tau = 3, 7$. The bars show the absolute value of the running time using a logarithmic scale. **Top row:** general axioms. **Bottom row:** directed axioms.

algorithm. We compared the true structure of the underlying model to the resulting structure of the PC algorithm when it uses SITs as independence tests, denoted PC-SIT, and its output when it uses argumentative independence tests, denoted PC-$\widehat{\mathrm{AIT}}_{\mathrm{t}}$-D, when using the directed axioms.

We evaluated the resulting networks by their ability to accurately represent the true independences in the domain, calculated by comparing the results (`true` or `false`) of a number of conditional tests conducted using d-separation on the output networks (PC-SIT or PC-$\widehat{\mathrm{AIT}}_{\mathrm{t}}$-D). Denoting by $\mathcal{T}$ this set of 2,000 triplets, by $t \in \mathcal{T}$ a triplet, by $I_{\mathrm{true}}(t)$ the result of a test performed on the underlying model, and by $I_{\text{PC-}\mathcal{Y}}(t)$ the result of performing a d-separation test on the network output by the PC algorithm using the $\mathcal{Y}$ test, $\mathcal{Y}$ equal to SIT or $\widehat{\mathrm{AIT}}_{\mathrm{t}}$-D, the estimated accuracy is defined as

$$\widehat{acc}^{\mathrm{PC}}_{\mathcal{Y}} = \frac{1}{|\mathcal{T}|}\left|\left\{t \in \mathcal{T} \mid I_{\text{PC-}\mathcal{Y}}(t) = I_{\mathrm{true}}(t)\right\}\right|. \tag{26}$$

We considered data sampled from randomly generated Bayesian networks of sizes $n = 24$, and maximum degrees $\tau = 3, 7$. For each network we sampled ten data sets, and, for each data set, we



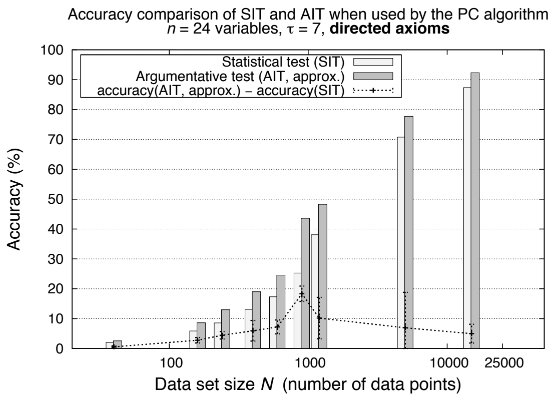


Figure 6: Comparison of statistical tests (SIT) vs. approximate argumentative tests on the directed axioms ($\widehat{\text{AIT}_t}$-D) for data sets sampled from Bayesian models for domain size $n = 24$ and maximum degrees $\tau = 3,7$.

conducted experiments on subsets of D containing an increasing number of data points. We report the average over the ten data sets of the estimated accuracy calculated using Eq. (26), for $\mathcal{Y}$ = SIT or $\widehat{\text{AIT}_t}$-D, as well as the difference between the average accuracies including the 95% confidence interval for the difference.

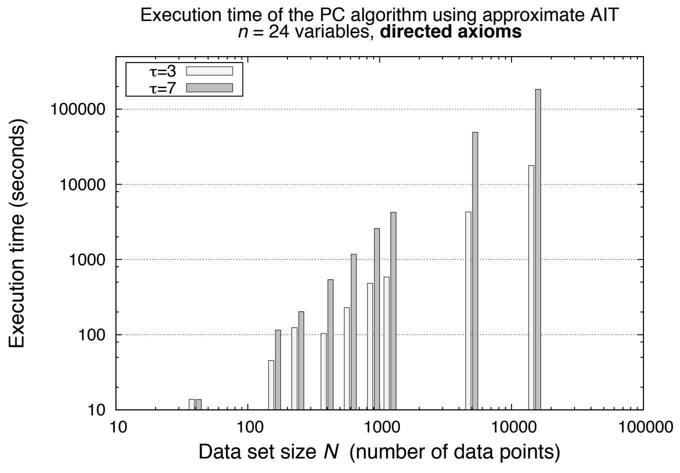


Figure 7: Execution times for the PC algorithm using the approximate argumentative test on the directed axioms ($\widehat{\text{AIT}_t}$-D) on data sets sampled from Bayesian models for domain size $n = 24$ and maximum degrees $\tau = 3,7$. For the approximate AIT test we limited the depth of the dialog tree to 3 and its the branching factor as described in Section 6.

Figure 6 shows a comparison of the argumentative tests $\widehat{\text{AIT}_t}$-D using the directed axioms with the corresponding SIT. The figure shows two plots for different values of τ of the mean values (over runs for ten different data sets) of $\widehat{acc}^{\text{PC}}_{\text{SIT}}$ and $\widehat{acc}^{\text{PC}}_{\widehat{\text{AIT}_t}\text{-D}}$ (histograms), the difference between these averages (line graph), and the 95% confidence intervals for the difference (error bars), for different data set sizes N. As usual, a positive value of the difference corresponds to an improvement of $\widehat{\text{AIT}}_\text{t}$-D over SIT. As in practically all experiments so far, we have statistically significant improvements over the entire range of data set sizes, with maximum improvements of up to 20% for $\tau = 3$, $N = 25000$, and $\tau = 7, N = 900$. The corresponding execution times for the entire PC algorithm are shown in Fig. 7. We can make two observations from this graph. One, the cost is significantly lower for sparse domains, which benefits real-world application domains that are sparse. The second observation is that the execution time scales linearly with the number of data points; this exhibits the same behavior as the use of a SIT test in PC, as each test needs to scan the data set once to compute the contingency table and relevant test statistics.

In summary, these results demonstrate that the approximate argumentative test is practical for larger domains and can result in positive, statistically significant accuracy improvements when used for causal discovery. However, the cost of AIT for large data sets, although not prohibitive, can be non-negligible. Therefore the accuracy benefits of AIT vs. a SIT must be carefully weighed off the ability of the user to expend the extra computation. Note that the practicality of the approximate algorithm also depends on the parameters used (the cutoff depth of iterative deepening and the branching factor limit—see Section 6); different parameter values or alternative ways of limiting the size of the dialog tree may be needed for even larger domains.

7.3 Real-world and Benchmark Data Experiments

While the sampled data set studies of the previous section have the advantage of a more controlled and systematic study of the performance of the algorithms, experiments on real-world data are necessary for a more realistic assessment. In this section we present experiments on a number of real-world and benchmark data sets obtained from the UCI machine learning repository (D. J. Newman and Merz, 1998) and the Knowledge Discovery Data repository (Hettich and Bay, 1999). As in the sampled data case of the previous section, for each data set D, we conducted experiments on subsets of D containing an increasing number of data points N to assess the performance of the independence tests on varying conditions of reliability. Again, to reduce variance we repeated each experiment ten times, each time choosing a different randomly selected data subset of equal size.

Because for real-world data sets the underlying model is unknown, we could only be sure the general axioms of Eq. (5) apply. We therefore only used these axioms in this section. Also, as mentioned in the previous section, because some of the data sets have much larger domains (e.g., the alarm data set contains 37 variables), and given the exponential nature of the exact algorithms we could only perform experiments for the approximate version of the argumentative test. For these reasons, in the following experiments we only report the accuracy of $\widehat{\text{AIT}}_\text{t}$-G, the approximate argumentative independence test defined over the general axioms. Unfortunately, for real-world data the underlying model is typically unknown and therefore it is impossible to know the true value of any independence. We therefore approximate it by a statistical test on the entire data set, and limit the size of the data set subsets that we use up to a third of the size of the entire data set. This corresponds to the hypothetical scenario that a much smaller data set is available to the researcher, allowing us to evaluate the improvement of argumentation under these more challenging situations. Again, as in the previous two sections, for comparison we sampled 2,000 triplets and calculated the accuracy as a fraction of tests correct, where for the true value of independences and dependences we used the method just described.

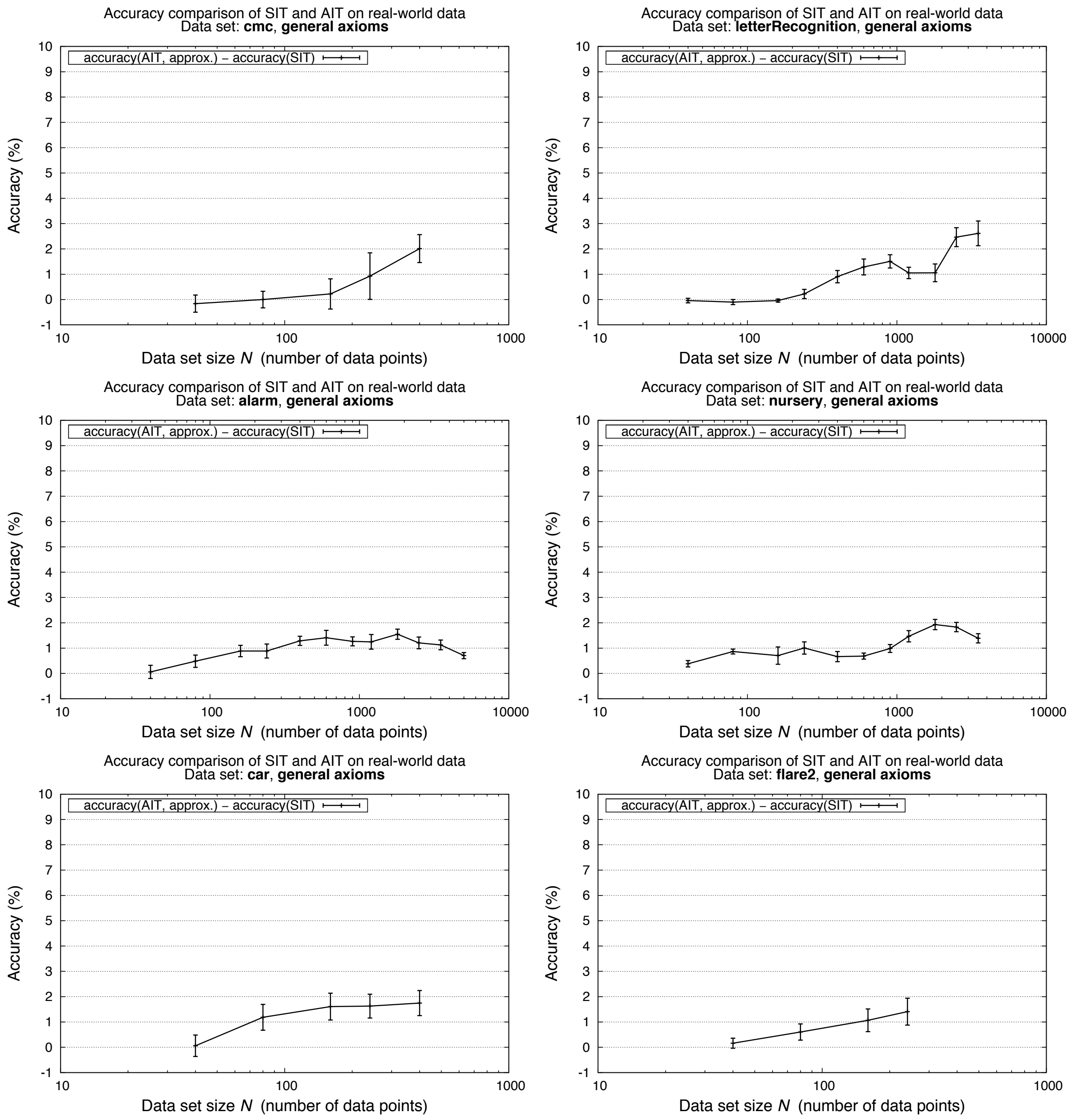


Figure 8: Difference in the mean value of the accuracy $\widehat{\mathrm{AIT}}_{\mathrm{t}}$-G with the mean value of the accuracy of SIT for a number of real-world data sets. The error bars denote the 95% confidence interval of the difference.

Figure 8 and Table 1 show the result of our comparison between the argumentative test $\widehat{\mathrm{AIT}}_{\mathrm{t}}$-G and statistical test SIT for real-world data sets. In the table, the best-performing method is shown in bold. The figure contains 6 plots, one for each data set, depicting the difference between the mean value of the accuracy of $\widehat{\mathrm{AIT}}_{\mathrm{t}}$-G and that of SIT, where as usual a positive value denotes an improvement of $\widehat{\mathrm{AIT}}_{\mathrm{t}}$-G over SIT. While in a few cases the average difference is negative (e.g., data set `cmc`, $N = 40$), in each case the negative value is not statistically significant as the confidence in-

	Data set	`car`	`cmc`	`flare2`	`letterRecognition`	`nursery`	`alarm`
	Domain size	7	10	13	17	9	37
	Data set size	1730	1475	1067	20002	12962	20003
	SIT	80.1	**77.8**	77.0	**47.9**	83.3	76.7
$N = 40$	$\widehat{\text{AIT}}_{\text{t}}$-G	80.1	77.5	**77.1**	47.8	**83.8**	76.7
	$\widehat{\text{AIT}}_{\text{t}}$-G − SIT	0.0 ± 0.7	-0.3 ± 0.6	0.1 ± 0.3	-0.1 ± 0.2	0.4 ± 0.1	0.0 ± 0.4
	Runtime of $\widehat{\text{AIT}}_{\text{t}}$-G (ms)	0.56	1.07	2.61	4.19	0.88	52.06
	SIT	86.7	84.1	85.5	50.7	86.1	84.3
$N = 240$	$\widehat{\text{AIT}}_{\text{t}}$-G	**88.6**	**84.7**	**86.9**	**51.0**	**87.2**	**85.1**
	$\widehat{\text{AIT}}_{\text{t}}$-G − SIT	1.9 ± 0.6	0.5 ± 1.2	1.3 ± 0.8	0.2 ± 0.4	1.1 ± 0.4	0.8 ± 0.3
	Runtime of $\widehat{\text{AIT}}_{\text{t}}$-G (ms)	1.37	5.19	8.73	90.50	1.84	202.05
	SIT				55.8	88.5	88.6
$N = 600$	$\widehat{\text{AIT}}_{\text{t}}$-G				**57.3**	**89.3**	**89.8**
	$\widehat{\text{AIT}}_{\text{t}}$-G − SIT				1.5 ± 0.5	0.8 ± 0.1	1.2 ± 0.4
	Runtime of $\widehat{\text{AIT}}_{\text{t}}$-G (ms)				575.53	4.37	547.77
	SIT				63.3	89.7	90.8
$N = 1200$	$\widehat{\text{AIT}}_{\text{t}}$-G				**64.3**	**91.2**	**92.0**
	$\widehat{\text{AIT}}_{\text{t}}$-G − SIT				1.0 ± 0.3	1.5 ± 0.3	1.2 ± 0.4
	Runtime of $\widehat{\text{AIT}}_{\text{t}}$-G (ms)				2008.76	14.05	1151.05
	SIT				73.8	94.1	95.2
$N = 3500$	$\widehat{\text{AIT}}_{\text{t}}$-G				**76.5**	**95.4**	**96.3**
	$\widehat{\text{AIT}}_{\text{t}}$-G − SIT				2.6 ± 0.7	1.3 ± 0.3	1.1 ± 0.3
	Runtime of $\widehat{\text{AIT}}_{\text{t}}$-G (ms)				24540.51	76.48	3895.2

Table 1: Average accuracies (in percentage) of SIT and $\widehat{\text{AIT}}_{\text{t}}$-G, their differences (denoted $\widehat{\text{AIT}}_{\text{t}}$-G − SIT in the table), the 95% confidence interval for the difference, and the average runtime per test (in ms) for $\widehat{\text{AIT}}_{\text{t}}$-G for several real-world and benchmark data sets. For each data set the table shows these quantities for number of data points $N = 40, 240, 600, 1200, 3500$. The best performing algorithm ($\widehat{\text{AIT}}_{\text{t}}$-G or SIT, with respect to accuracy) is indicated in bold. Empty cells correspond to cases where one third of the data set was smaller than the value of N in that column.

terval contains a portion of the positive half-plane. The figure demonstrates a clear advantage of the argumentative approach, with all data sets reaching statistically significant positive improvements in accuracy of up to 3% and all confidence intervals covering positive values either partially or completely. The table also shows the average execution time (in ms) for the $\widehat{\text{AIT}}_{\text{t}}$-G tests evaluated.

8. Conclusion

We presented a framework for addressing one of the most important problems of independence-based structure discovery algorithms, namely the problem of unreliability of statistical independence tests. Our main idea was to recognize the existence of interdependences among the outcomes of conditional independence tests—in the form of Pearl's axiomatic characterization of the conditional independence relation—that can be seen as integrity constraints and exploited to correct unreliable statistical tests. We modeled this setting as a knowledge base containing conditional independences that are potentially inconsistent, and used the preference-based argumentation framework to reason with and resolve these inconsistencies. We presented in detail how to apply the argumentation framework to independence knowledge bases and how to compute the preference among the independence propositions. We also presented a number of algorithms, both exact and approximate, for implementing statistical testing using this framework. We analyzed the approximate algorithm and proved that is has polynomial worst-case execution time. We also experimentally verified that its accuracy improvement is close to the exact one while providing orders of magnitude faster execution, making possible its use for causal discovery in large domains. Overall, our experimental evaluation demonstrated statistically significant improvements in the accuracy of causal discovery for the overwhelming majority of sampled, benchmark and real-world data sets.

Appendix A. Computability of the Argumentative Independence Test

In this appendix we prove that the argumentative test terminates, a property that we call its *computability*. Some of the theorems and lemmas presented are not original work but adaptations of well known properties of relations. We include them to allow a complete exposition of the proof of computability, given by Theorem 21. We first introduce some notation. We denote independence propositions (e.g., $(X \perp\!\!\!\perp Y \mid \mathbf{Z})$) by σ and their negation (e.g., $(X \not\!\perp\!\!\!\perp Y \mid \mathbf{Z})$) by $\neg\sigma$. We abbreviate their corresponding propositional arguments $(\{\sigma\}, \sigma)$ and $(\{\neg\sigma\}, \neg\sigma)$ by a_σ and $a_{\neg\sigma}$, respectively, and we will refer to $a_{\neg\sigma}$ as the *negation* of a_σ (and vice versa). Also, we use the predicates $A(a)$, $R(a)$, $Ab(a)$ to denote the fact the argument a is accepted, rejected, or in abeyance, respectively.

For completeness we repeat here the definition of strict and transitive preference relation.

Definition 13. *We say preference relation π over arguments is* strict *if the ordering of arguments induced by it is strict and total, that is, for every pair of arguments a and b,*

$$a \gg_\pi b \iff \neg\big(b \gg_\pi a\big). \tag{27}$$

Definition 14. *We say that preference relation π over arguments is* transitive *if, for every three arguments a, b and c,*

$$\big(a \gg_\pi b\big) \wedge \big(b \gg_\pi c\big) \implies \big(a \gg_\pi c\big).$$

Lemma 24. *A strict preference relation π satisfies the condition that for every pair of arguments such that a defeats b and b defeats a, it is the case that a attacks b or b attacks a, that is, at least one of a and b attacks the other.*

Proof We prove by contradiction: Let us assume that a defeats b and b defeats a but neither a attacks b nor b attacks a. By definition of the attack relation (Definition 15),

$$\neg\big(a \textit{ attacks } b\big) \implies \neg\big(\neg(b \gg_\pi a)\big) \implies b \gg_\pi a$$

and

$$\neg\big(b \textit{ attacks } a\big) \implies \neg\big(\neg(a \gg_\pi b)\big) \implies a \gg_\pi b.$$

However, this is a contradiction since, by assumption, the preference ordering is strict, and therefore it cannot be true that both $a \gg_\pi b$ and $b \gg_\pi a$ are true at the same time. ■

Lemma 25. *A strict preference π satisfies the condition that for every pair a and b of arguments, it is not the case that both a attacks b and b attacks a, that is, there can be no mutual attack.*

Proof We prove by contradiction. Let us consider two mutually attacking arguments a and b. By the definition of the attack relation, and because π is a total order, we have that

$$a \textit{ attacks } b \implies \neg(b \gg_\pi a) \implies \big(a \gg_\pi b \vee a \equiv_\pi b\big)$$

and

$$b \text{ attacks } b \implies \neg(a \gg_\pi b) \implies \big(b \gg_\pi a \vee b \equiv_\pi a\big)$$

where $a \equiv_\pi b$ means a is equally preferable to b. However, equality of preference is not possible in a strict preference relation. Therefore it must be the case that $a \gg_\pi b$ and $b \gg_\pi a$, which is a contradiction of Eq. (27), again due to strictness. ■

We next prove that no argument is in abeyance if the preference relation over arguments is strict. For that, we first prove that an argument in abeyance is always attacked by at least another argument in abeyance.

Lemma 8. *For every argument a,*

$$Ab(a) \implies \exists b \in attackers(a), Ab(b).$$

Proof By definition, an argument a is in abeyance if it is neither accepted nor rejected. Applying the definitions of acceptance and rejection and manipulating the Boolean formulae we obtain,

$$\begin{aligned}
Ab(a) &\iff \neg A(a) \wedge \neg R(a) \\
&\iff \neg\big(\forall b \in attackers(a), R(b)\big) \wedge \neg\big(\exists b \in attackers(a), A(b))\big) \\
&\iff \big(\exists b \in attackers(a), \neg R(b)\big) \wedge \big(\forall b \in attackers(a), \neg A(b))\big) \\
&\iff \big(\exists b \in attackers(a), (A(b) \vee Ab(b))\big) \wedge \big(\forall b \in attackers(a), \neg A(b))\big) \\
&\iff \big(\exists b \in attackers(a), Ab(b)\big) \wedge \big(\forall b \in attackers(a), \neg A(b))\big) \\
&\implies \exists b \in attackers(a), Ab(b).
\end{aligned}$$

■

Definition 22. *An* attack sequence *is a sequence $\langle a_1, a_2, \ldots, a_n\rangle$ of n arguments such that for every $2 \le i \le n$, a_i attacks a_{i-1}.*

Lemma 26. *Let $\langle \mathcal{A}, \mathcal{R}, \pi\rangle$ be a PAF with a strict and transitive preference relation π. Then, no argument can appear more than once in any attack sequence, that is, for every attack sequence $\langle a_1, a_2, \ldots, a_n\rangle$ and every pair of integers $i, j \in [1, n]$ such that $i \neq j$, $a_i \neq a_j$.*

Proof

We first note that by definition of the attack relation, it must be the case that for any two consecutive arguments a_i, a_{i+1}, it is true that $\neg(a_i \gg_\pi a_{i+1})$. Since π is strict, this is equivalent to $a_{i+1} \gg_\pi a_i$ (c.f. Eq. (27)). That is,

$$a_n \gg_\pi a_{n-1} \gg_\pi \ldots \gg_\pi a_2 \gg_\pi a_1 \tag{28}$$

We now assume, for contradiction, there exists an argument $a^\star$ that appears twice in the attack sequence at indexes $i^\star$ and $j^\star$, that is,

$$\exists\ i^\star, j^\star \in [1, n], i^\star \neq j^\star, \text{ such that } a_{i^\star} = a_{j^\star} = a^\star.$$

Since no argument defeats itself, it cannot attack itself, and thus the smallest possible attack sequence with a repeated argument must have at least length 3. From this fact, Eq. (28), and transitivity, there must exist an argument $b \neq a^\star$ such that $a^\star \gg_\pi b \gg_\pi a^\star$. This last fact implies that $a^\star \gg_\pi b$ and $b \gg_\pi a^\star$ must hold, which contradicts strictness (Eq. (27)). ■

A corollary of this lemma is the following theorem.

Theorem 23. *Every attack sequence $\langle a_1, a_2, \ldots, a_n \rangle$ in a PAF $\langle \mathcal{A}, \mathcal{R}, \pi \rangle$ with strict and transitive π, and finite $\mathcal{A}$ is finite.*

Proof Follows directly from Lemma 26 and the fact that $\mathcal{A}$ is finite. ■

We can now prove the main result of this section in the following theorem.

Theorem 21. *Given an arbitrary triplet $t = (X, Y \mid \mathbf{Z})$, and a PAF $\langle \mathcal{A}, \mathcal{R}, \pi \rangle$ with a strict and transitive preference relation π, and finite arguments set $\mathcal{A}$, the top-down algorithm of Algorithm 3 run for input t over $\langle \mathcal{A}, \mathcal{R}, \pi \rangle$ terminates.*

Proof In the tree traversed by the top-down algorithm, any path from the root to a leaf is an attack sequence. Since for strict and transitive π, and finite $\mathcal{A}$ each such sequence is finite, the algorithm always terminates. ■

Appendix B. Validity of the Argumentative Independence Test

In this section we prove the property of the argumentative independence test of deciding that an input triplet $(X, Y \mid \mathbf{Z})$ evaluates to either independence or dependence, but not both or neither. We call this property the *validity* of the test.

We start we proving that under the assumption of a strict and transitive preference relation, no argument is in abeyance.

Theorem 20. *Let $\langle \mathcal{A}, \mathcal{R}, \pi \rangle$ be a PAF with a strict and transitive preference relation π. Then no argument $a \in \mathcal{A}$ is in abeyance.*

Proof Let us assume, for contradiction, that there is an argument a in abeyance. From Lemma 8, not only a has an attacker in abeyance, say argument b, but b also has an attacker in abeyance, and so on. That is, we can construct an attack sequence starting at a that contains only arguments in abeyance. Moreover, this sequence must be infinite, since the lemma assures as we always have at least one attacker in abeyance. This is in direct contradiction with Theorem 23. ■

Corollary 27. *For every argument a in a PAF $\langle \mathcal{A}, \mathcal{R}, \pi \rangle$ with strict and transitive π,*

$$A(a) \iff \neg R(a).$$

We now prove a number of lemmas that hold only for the sub-class of propositional arguments (arguments whose support contains only one proposition, equal to the head of that argument). We start with a lemma that demonstrates that it cannot be the case that an attacker of a propositional argument a_σ and an attacker of its negation $a_{\neg\sigma}$ do not attack each other. The former must attack the latter or vice versa.

Lemma 28. *Let $\langle \mathcal{A}, \mathcal{R}, \pi \rangle$ be a PAF with a strict preference relation π, $a_\sigma \in \mathcal{A}$ be a propositional argument, and $a_{\neg\sigma}$ its negation. For every pair of arguments b and c that attacks a_σ and $a_{\neg\sigma}$ respectively,*

$$(b \textit{ attacks } c) \vee (c \textit{ attacks } b).$$

Proof Since a_σ and $a_{\neg\sigma}$ are propositional arguments, their support contains the head and only the head, and thus any defeater (i.e., rebutter or undercutter) must have as head $\neg\sigma$ and σ, respectively, that is, the head of b must be $\neg\sigma$ and the head of c must be σ. Thus, b rebuts (and thus defeats) c and vice versa. The lemma then follows directly from Lemma 24. ■

Lemma 29. *Let $\langle \mathcal{A}, \mathcal{R}, \pi \rangle$ be a PAF with a strict preference relation π, and a_σ and $a_{\neg\sigma}$ be a propositional argument and its negation. Then,*

$$R(a_\sigma) \implies \neg R(a_{\neg\sigma}).$$

Proof By assumption, $R(a_\sigma)$. We assume, for contradiction, that $R(a_{\neg\sigma})$. Therefore, by the definition of rejection, $\exists b \in attackers(a_\sigma)$ such that $A(b)$, and $\exists c \in attackers(a_{\neg\sigma})$ such that $A(c)$. By Lemma 28 b *attacks* c or c *attacks* b. In either case, an accepted argument is attacking an accepted argument, which contradicts the definition of acceptance. ■

Lemma 30. *Given a PAF $\langle \mathcal{A}, \mathcal{R}, \pi \rangle$ with a strict preference relation π, every propositional argument $a_\sigma \in \mathcal{A}$ satisfies*

$$A(a_\sigma) \implies \neg A(a_{\neg\sigma})$$

Proof We prove by contradiction. Let us assume that both a_σ and $a_{\neg\sigma}$ are accepted. Since a_σ and $a_{\neg\sigma}$ are propositional arguments, they defeat each other. Then, by Lemma 24 a_σ attacks $a_{\neg\sigma}$ or vice versa. In either case an accepted argument has an accepted attacker, which is a contradiction. ■

We now prove Theorem 19 that was introduced in Section 4, reproduced here for convenience.

Theorem 19. *Given a PAF $\langle \mathcal{A}, \mathcal{R}, \pi \rangle$ with a strict and transitive preference relation π, every propositional argument $a_\sigma \in \mathcal{A}$ and its negation $a_{\neg\sigma}$ satisfy*

$$A(a_\sigma) \iff R(a_{\neg\sigma}).$$

Proof The ($\implies$) direction follows from Lemma 30 and Theorem 20. The ($\impliedby$) direction follows from Lemma 29 and Theorem 20. ■

In Section 4 we defined the following semantics for deciding on the dependence or independence of an input triplet $(X, Y \mid \mathbf{Z})$:

$$\begin{aligned} (\{(X \not\perp\!\!\!\perp Y \mid \mathbf{Z})\}, (X \not\perp\!\!\!\perp Y \mid \mathbf{Z})) \text{ is accepted} &\iff (X \not\perp\!\!\!\perp Y \mid \mathbf{Z}) \text{ is accepted} \implies (X \not\perp\!\!\!\perp Y \mid \mathbf{Z}) \\ (\{(X \perp\!\!\!\perp Y \mid \mathbf{Z})\}, (X \perp\!\!\!\perp Y \mid \mathbf{Z})) \text{ is accepted} &\iff (X \perp\!\!\!\perp Y \mid \mathbf{Z}) \text{ is accepted} \implies (X \perp\!\!\!\perp Y \mid \mathbf{Z}) \end{aligned} \quad (29)$$

where acceptance is defined over an independence-based PAF as defined in Section 3.3. For this argumentative test of independence to be valid, its decision must be non-ambiguous, that is, it must decide either independence or dependence, but not both or neither. For that, exactly one of the antecedents of the above implications must be true. Formally:

Theorem 18. *For any input triplet $\sigma = (X, Y \mid \mathbf{Z})$, the argumentative independence test defined by Eq. (29) produces a non-ambiguous decision, that is, it decides that σ evaluates to either independence or dependence, but not both or neither.*

Proof Let us denote $(X \perp\!\!\!\perp Y \mid \mathbf{Z})$ by $\sigma_{\mathtt{t}}$ and $(X \not\perp\!\!\!\perp Y \mid \mathbf{Z})$ by $\sigma_{\mathtt{f}}$. Since strictness and transitivity of the independence preference relation hold (proved in Section 3.3, lemmas 16 and 17 respectively), Theorems 19 and 20 hold as well. From Theorem 20 we know that neither of the propositional arguments is in abeyance. Thus, since $a_{\sigma_{\mathtt{t}}}$ corresponds to the negation of $a_{\sigma_{\mathtt{f}}}$ it follows from Theorem 19 that exactly one of them is accepted. ■

References

H. Abdi. The Bonferonni and Šidák corrections for multiple comparisons. In Neil Salkind, editor, *Encyclopedia of Measurement and Statistics*. Thousand Oaks (CA): Sage, 2007.

A. Agresti. *Categorical Data Analysis*. Wiley, 2nd edition, 2002.

L. Amgoud and C. Cayrol. A reasoning model based on the production of acceptable arguments. *Annals of Mathematics and Artificial Intelligence*, 34:197–215, 2002.

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

Y. Benjamini and D. Yekutieli. The control of the false discovery rate in multiple testing under dependency. *Annals of Statistics*, 29(4):1165–1188, 2001.

W. G. Cochran. Some methods of strengthening the common χ^2 tests. *Biometrics*, 10:417–451, 1954.

T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. MIT Press, 2nd edition, 2001.

C. L. Blake D. J. Newman, S. Hettich and C. J. Merz. UCI repository of machine learning databases. *Irvine, CA: University of California, Department of Information and Computer Science*, 1998.

A. P. Dawid. Conditional independence in statistical theory. *Journal of the Royal Statistical Society*, 41:1–31, 1979.

P. M. Dung. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and *n*-person games. *Artificial Intelligence*, 77:321–357, 1995.

P. Gärdenfors. *Belief Revision*. Cambridge Computer Tracts. Cambridge University Press, Cambridge, 1992.

P. Gärdenfors and H. Rott. Belief revision. In Gabbay, D. M., Hogger, C. J. and Robinson, J. A., editors, *Handbook of Logic in Artificial Intelligence and Logic Programming*, volume 4. Clarendon Press, Oxford, 1995.

S. Hettich and S. D. Bay. The UCI KDD archive. *Irvine, CA: University of California, Department of Information and Computer Science*, 1999.

Y. Hochberg. A sharper Bonferroni procedure for multiple tests of significance. *Biometrika*, 75(4): 800–802, December 1988.

J. S. Ide, F. G. Cozman, and F. T. Ramos. Generating random Bayesian networks with constraints on induced width. *Brazilian Symposium on Artificial Intelligence, Recife, Pernambuco, Brazil*, 2002.

A. C. Kakas and F. Toni. Computing argumentation in logic programming. *Journal of Logic and Computation*, 9(4):515–562, 1999.

M. J. Kearns and U. V. Vazirani. *An Introduction to Computational Learning Theory*. MIT Press, Cambridge, MA, 1994.

R. P. Loui. Defeat among arguments: a system of defeasible inference. *Computational Intelligence*, 2:100–106, 1987.

J. P. Martins. Belief revision. In Shapiro, S. C., editor, *Encyclopedia of Artificial Intelligence*, pages 110–116. John Wiley & Sons, New York, second edition, 1992.

J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, San Francisco, CA, 2nd edition, 1988.

J. Pearl and A. Paz. GRAPHOIDS: A graph-based logic for reasoning about relevance relations. Technical Report 850038 (R-53-L), Cognitive Systems Laboratory, University of California, 1985.

J. L. Pollock. How to reason defeasibly. *Artificial Intelligence*, 57:1–42, 1992.

H. Prakken. *Logical Tools for Modelling Legal Argument. A Study of Defeasible Reasoning in Law.* Kluwer Law and Philosophy Library, Dordrecht, 1997.

H. Prakken and G. Vreeswijk. *Logics for Defeasible Argumentation*, volume 4 of *Handbook of Philosophical Logic*. Kluwer Academic Publishers, Dordrecht, 2 edition, 2002.

S. C. Shapiro. Belief revision and truth maintenance systems: An overview and a proposal. Technical Report CSE 98-10, Dept of Computer Science and Engineering, State University of New York at Buffalo, 1998.

P. Spirtes, C. Glymour, and R. Scheines. *Causation, Prediction, and Search*. Adaptive Computation and Machine Learning Series. MIT Press, 2nd edition, January 2000.

J. D. Storey. A direct approach to false discovery rates. *Journal of the Royal Statistical Society, Series B (Methodological)*, 64(3):479–498, 2002.

M. Studený. Conditional independence relations have no finite complete characterization. In *Transactions of the 11th Prague Conference on Information Theory, Statistical Decision Functions and Random Processes*, volume B, pages 377–396, 1991.

F. Toni and A. C. Kakas. In A. Nerode, editor, *3rd International Conference on Logic Programming and Non-monotonic Reasoning*, volume 928 of *Lecture Notes in Artificial Intelligence*, pages 401–415. Springer Verlag, 1995.

Journal of Machine Learning Research 10 (2009) 341-376 Submitted 8/07; Revised 10/08; Published 2/09

Low-Rank Kernel Learning with Bregman Matrix Divergences

Brian Kulis KULIS@CS.UTEXAS.EDU
Mátyás A. Sustik SUSTIK@CS.UTEXAS.EDU
Inderjit S. Dhillon INDERJIT@CS.UTEXAS.EDU
Department of Computer Sciences
University of Texas at Austin
Austin, TX 78712, USA

Editor: Michael I. Jordan

Abstract

In this paper, we study low-rank matrix nearness problems, with a focus on learning low-rank positive semidefinite (kernel) matrices for machine learning applications. We propose efficient algorithms that scale linearly in the number of data points and quadratically in the rank of the input matrix. Existing algorithms for learning kernel matrices often scale poorly, with running times that are cubic in the number of data points. We employ Bregman matrix divergences as the measures of nearness—these divergences are natural for learning low-rank kernels since they preserve rank as well as positive semidefiniteness. Special cases of our framework yield faster algorithms for various existing learning problems, and experimental results demonstrate that our algorithms can effectively learn both low-rank and full-rank kernel matrices.

Keywords: kernel methods, Bregman divergences, convex optimization, kernel learning, matrix nearness

1. Introduction

Underlying many machine learning algorithms is a measure of distance, or divergence, between data objects. A number of factors affect the choice of the particular divergence measure used: the data may be intrinsically suited for a specific divergence measure, an algorithm may be faster or easier to implement for some measure, or analysis may be simpler for a particular measure. For example, the KL-divergence is popular when the data is represented as discrete probability vectors (i.e., non-negative vectors whose sum is 1), and the ℓ_1-norm distance is often used when sparsity is desired.

When measuring the divergence between matrices, similar issues must be considered. Typically, matrix norms such as the Frobenius norm are used, but these measures are not appropriate for all problems. Analogous to the KL-divergence for vectors, when the matrices under consideration are positive semidefinite (i.e., they have non-negative eigenvalues), then one may want to choose a divergence measure that is well-suited to such matrices; positive semidefinite (PSD) matrices arise frequently in machine learning, in particular with kernel methods. Existing learning techniques involving PSD matrices often employ matrix norms, but their use requires an additional constraint that the matrix stay positive semidefinite, which ultimately leads to algorithms involving expensive eigenvector computation. Kernel alignment, a measure of similarity between PSD matrices used in some kernel learning algorithms (Cristianini et al., 2002), also requires explicit enforcement of positive definiteness. On the other hand, the two main divergences used in this paper are defined only over the cone of positive semidefinite matrices, and our algorithms lead to automatic enforcement of positive semidefiniteness.

This paper focuses on kernel learning using two divergence measures between PSD matrices—the LogDet divergence and the von Neumann divergence. Our kernel learning goal is to find a PSD matrix which is as close as possible (under the LogDet or von Neumann divergence) to some input PSD matrix, but which additionally satisfies linear equality or inequality constraints. We argue that these two divergences are natural for problems involving positive definite matrices. First, they have several properties which make optimization computationally efficient and useful for a variety of learning tasks. For example, the LogDet divergence has a scale-invariance property which makes it particularly well-suited to machine learning applications. Second, these divergences arise naturally in several problems; for example, the LogDet divergence arises in problems involving the differential relative entropy between Gaussian distributions while the von Neumann divergence arises in quantum computing and problems such as online PCA.

One of the key properties that we demonstrate and exploit is that, for low-rank matrices, the divergences enjoy a *range-space preserving* property. That is, the LogDet divergence between two matrices is finite if and only if their range spaces are identical, and a similar property holds for the von Neumann divergence. This property leads to simple projection-based algorithms for learning PSD (or kernel) matrices that preserve the rank of the input matrix—if the input matrix is low-rank then the resulting learned matrix will also be low-rank. These algorithms are efficient; they scale linearly with the number of data points n and quadratically with the rank of the input matrix. The efficiency of the algorithms arises from new results developed in this paper which demonstrate that Bregman projections for these matrix divergences can be computed in time that scales quadratically with the rank of the input matrix. Our algorithms stand in contrast to previous work on learning kernel matrices, which scale as $O(n^3)$ or worse, relying on semidefinite programming or repeated eigenvector computation. We emphasize that our methods *preserve* rank, so they can learn low-rank kernel matrices when the input kernel matrix has low rank; however our methods do not decrease rank so they are not applicable to the non-convex problem of finding a low-rank solution given a full (or higher) rank input kernel matrix.

One special case of our method is the DefiniteBoost optimization problem from Tsuda et al. (2005); our analysis shows how to improve the running time of their algorithm by a factor of n, from $O(n^3)$ time per projection to $O(n^2)$. This projection is also used in online-PCA (Warmuth and Kuzmin, 2006), and we obtain a factor of n speedup for that problem as well. In terms of experimental results, a direct application of our techniques is in learning low-rank kernel matrices in the setting where we have background information for a subset of the data; we discuss and experiment with learning low-rank kernel matrices for classification and clustering in this setting, demonstrating that we can scale to large data sets. We also discuss the use of our divergences for learning Mahalanobis distance functions, which allows us to move beyond the transductive setting and generalize to new points. In this vein, we empirically compare our methods to existing metric learning algorithms; we are particularly interested in large-scale applications, and discuss some recent applications of the algorithms developed in this paper to computer vision tasks.

2. Background and Related Work

In this section, we briefly review relevant background material and related work.

2.1 Kernel Methods

Given a set of training points $\boldsymbol{a}_1, ..., \boldsymbol{a}_n$, a common step in kernel algorithms is to transform the data using a nonlinear function ψ. This mapping, typically, represents a transformation of the data to a higher-dimensional feature space. A kernel function κ gives the inner product between two vectors in the feature space:

$$\kappa(\boldsymbol{a}_i, \boldsymbol{a}_j) = \psi(\boldsymbol{a}_i) \cdot \psi(\boldsymbol{a}_j).$$

It is often possible to compute this inner product without explicitly computing the expensive mapping of the input points to the higher-dimensional feature space. Generally, given n points $\boldsymbol{a}_i$, we form an $n \times n$ matrix K, called the kernel matrix, whose (i, j) entry corresponds to $\kappa(\boldsymbol{a}_i, \boldsymbol{a}_j)$. In kernel-based algorithms, the only information needed about the input data points is the inner products; hence, the kernel matrix provides all relevant information for learning in the feature space. A kernel matrix formed from any set of input data points is always positive semidefinite. See Shawe-Taylor and Cristianini (2004) for more details.

2.2 Low-Rank Kernel Representations and Kernel Learning

Despite the popularity of kernel methods in machine learning, many kernel-based algorithms scale poorly; low-rank kernel representations address this issue. Given an $n \times n$ kernel matrix K, if the matrix is of low rank, say $r < n$, we can represent the kernel matrix in terms of a factorization $K = GG^T$, with G an $n \times r$ matrix.

In addition to easing the burden of memory overhead from $O(n^2)$ storage to $O(nr)$, this low-rank decomposition can lead to improved efficiency. For example, Fine and Scheinberg (2001) show that SVM training reduces from $O(n^3)$ to $O(nr^2)$ when using a low-rank decomposition. Empirically, the algorithm in Fine and Scheinberg (2001) outperforms other SVM training algorithms in terms of training time by several factors. In clustering, the kernel k-means algorithm (Dhillon et al., 2004) has a running time of $O(n^2)$ per iteration, which can be improved to $O(nrc)$ time per iteration with a low-rank kernel representation, where c is the number of desired clusters. Low-rank kernel representations are often obtained using incomplete Cholesky decompositions (Fine and Scheinberg, 2001). Recently, work has been done on using labeled data to improve the quality of low-rank decompositions (Bach and Jordan, 2005).

Low-rank decompositions have also been employed for solving a number of other machine learning problems. For example, in Kulis et al. (2007b), low-rank decompositions were employed for clustering and embedding problems. In contrast to work in this paper, their focus was on using low-rank decompositions to develop algorithms for such problems as k-means and maximum variance unfolding. Other examples of using low-rank decompositions to speed up machine learning algorithms include Weinberger et al. (2006) and Torresani and Lee (2006).

In this paper, our focus is on using distance and similarity constraints to learn a low-rank kernel matrix. In related work, Lanckriet et al. (2004) have studied transductive learning of the kernel matrix and multiple kernel learning using semidefinite programming. In Kwok and Tsang (2003), a formulation based on idealized kernels is presented to learn a kernel matrix when some labels are given. Another recent paper (Weinberger et al., 2004) considers learning a kernel matrix for nonlinear dimensionality reduction; like much of the research on learning a kernel matrix, they use semidefinite programming and the running time is at least cubic in the number of data points. Our work is closest to that of Tsuda et al. (2005), who learn a (full-rank) kernel matrix using von Neumann divergence under linear constraints. However, our framework is more general and our emphasis is on low-rank kernel learning. Our algorithms are more efficient than those of Tsuda et al.; we use exact instead of approximate projections to speed up convergence, and we consider algorithms for the LogDet divergence in addition to the von Neumann divergence. For the case of the von Neumann divergence, our algorithm also corrects a mistake in Tsuda et al. (2005); see Appendix B for details.

An earlier version of our work appeared in Kulis et al. (2006). In this paper, we substantially expand on the analysis of Bregman matrix divergences, giving a formal treatment of low-rank Bregman matrix divergences and proving several new properties. We also present further algorithmic analysis for the LogDet and von Neumann algorithms, provide connections to semidefinite programming and metric learning, and present additional experiments, including ones on much larger data sets.

3. Optimization Framework

We begin with an overview of the optimization framework applied to the problem studied in this paper. We introduce Bregman matrix divergences—the matrix divergence measures considered in this paper—and discuss their properties. Then we overview Bregman's method, the optimization algorithm that is used to learn low-rank kernel matrices.

3.1 Bregman Matrix Divergences

To measure the nearness between two matrices, we will use Bregman matrix divergences, which are generalizations of Bregman vector divergences. Let φ be a real-valued strictly convex function defined over a convex set $S = \text{dom}(\varphi) \subseteq \mathbb{R}^m$ such that φ is differentiable on the relative interior of S. The *Bregman vector divergence* (Bregman, 1967) with respect to φ is defined as

$$D_\varphi(\boldsymbol{x}, \boldsymbol{y}) = \varphi(\boldsymbol{x}) - \varphi(\boldsymbol{y}) - (\boldsymbol{x} - \boldsymbol{y})^T \nabla\varphi(\boldsymbol{y}).$$

For example, if $\varphi(\boldsymbol{x}) = \boldsymbol{x}^T\boldsymbol{x}$, then the resulting Bregman divergence is $D_\varphi(\boldsymbol{x}, \boldsymbol{y}) = \|\boldsymbol{x} - \boldsymbol{y}\|_2^2$. Another example is $\varphi(\boldsymbol{x}) = \sum_i (x_i \log x_i - x_i)$, where the resulting Bregman divergence is the (unnormalized) relative entropy $D_\varphi(\boldsymbol{x}, \boldsymbol{y}) = KL(\boldsymbol{x}, \boldsymbol{y}) = \sum_i (x_i \log \frac{x_i}{y_i} - x_i + y_i)$. Bregman divergences generalize many properties of squared loss and relative entropy. See Censor and Zenios (1997) for more details.

We can naturally extend this definition to real, symmetric $n \times n$ matrices, denoted by S^n. Given a strictly convex, differentiable function $\phi : S^n \to \mathbb{R}$, the Bregman matrix divergence is defined to be

$$D_\phi(X,Y) = \phi(X) - \phi(Y) - \text{tr}((\nabla\phi(Y))^T (X - Y)),$$

where $\text{tr}(A)$ denotes the trace of matrix A. Examples include $\phi(X) = \|X\|_F^2$, which leads to the well-known squared Frobenius norm $\|X - Y\|_F^2$. In this paper, we will extensively study two less well-known divergences. Let ϕ be the entropy of the eigenvalues of a positive definite matrix. Specifically, if X has eigenvalues $\lambda_1, ..., \lambda_n$, let $\phi(X) = \sum_i (\lambda_i \log \lambda_i - \lambda_i)$, which may be expressed as $\phi(X) = \text{tr}(X \log X - X)$, where $\log X$ is the matrix logarithm.[1] The resulting Bregman divergence is

$$D_{vN}(X,Y) = \text{tr}(X \log X - X \log Y - X + Y), \tag{1}$$

1. If $X = V\Lambda V^T$ is the eigendecomposition of the positive definite matrix X, the matrix logarithm can be written as $V \log\Lambda V^T$, where $\log\Lambda$ is the diagonal matrix whose entries contain the logarithm of the eigenvalues. The matrix exponential can be defined analogously.

and we call it the von Neumann divergence. This divergence is also called quantum relative entropy, and is used in quantum information theory (Nielsen and Chuang, 2000). Another important matrix divergence arises by taking the Burg entropy of the eigenvalues, that is, $\phi(X) = -\sum_i \log \lambda_i$, or equivalently as $\phi(X) = -\log \det X$. The resulting Bregman divergence over positive definite matrices is

$$D_{\ell d}(X,Y) = \mathrm{tr}(XY^{-1}) - \log\det(XY^{-1}) - n, \tag{2}$$

and is commonly called the LogDet divergence (we called it the Burg matrix divergence in Kulis et al. (2006)). For now, we assume that X is positive definite for both divergences; later we will discuss extensions when X is positive semidefinite, that is, low-rank.

3.2 Properties

It is important to justify the use of the divergences introduced above for kernel learning, so we now discuss some important properties of the divergences.

The most obvious computational benefit of using the divergences for kernel learning arises from the fact that they are defined only over positive definite matrices. Because of this, our algorithms will not need to explicitly constrain our learned matrices to be positive definite, which in turn leads to efficient algorithms. Beyond automatic enforcement of positive definiteness, we will see later that the divergences have a range-space preserving property, which leads to further computational benefits.

Appendix A covers some important properties of the divergences. Examples include the scale-invariance of the LogDet divergence ($D_{\ell d}(X,Y) = D_{\ell d}(\alpha X, \alpha Y)$) and, more generally, transformation-invariance ($D_{\ell d}(X,Y) = D_{\ell d}(M^T X M, M^T Y M)$ for any square, non-singular M), which are useful properties for learning algorithms. For instance, if we have applied a linear kernel over a set of data vectors, scale-invariance implies that we can scale all the features of our data vectors, and our learned kernel will simply be scaled by the same factor. Similarly, if we scale each of the features in our data vectors differently (for example, if some features are measured in feet and others in meters as opposed to all features measured in feet), then the learned kernel will be scaled by an appropriate diagonal transformation. Note that this natural property does not hold for loss functions such as the Frobenius norm distance.

Another critical property concerns generalization to new points. Typically kernel learning algorithms work in the transductive setting, meaning that all of the data is given up-front, with some of it labeled and some unlabeled, and one learns a kernel matrix over all the data points. If some new data point is given, there is no way to compare it to existing data points, so a new kernel matrix must be learned. However, as has recently been shown in Davis et al. (2007), we can move beyond the transductive setting when using the LogDet divergence, and can evaluate the kernel function over new data points. A similar result can be shown for the von Neumann divergence. We overview this recent work in Section 6.1, though the main focus in this paper remains in the transductive setting.

Furthermore, both the LogDet divergence and the von Neumann divergence have precedence in a number of areas. The von Neumann divergence is used in quantum information theory (Nielsen and Chuang, 2000), and has been employed in machine learning for online principal component analysis (Warmuth and Kuzmin, 2006). The LogDet divergence is called Stein's loss in the statistics literature, where it has been used as a measure of distance between covariance matrices (James and Stein, 1961). It has also been employed in the optimization community; the updates for the BFGS and DFP algorithms (Fletcher, 1991), both quasi-Newton algorithms, can be viewed as LogDet optimization programs. In particular, the update to the approximation of the Hessian given in these algorithms is the result of a LogDet minimization problem with linear constraints (given by the secant equation).

For all three divergences introduced above, the generating convex function of the Bregman matrix divergence can be viewed as a composition $\phi(X) = (\varphi \circ \lambda)(X)$, where $\lambda(X)$ is the function that lists the eigenvalues in algebraically decreasing order, and φ is a strictly convex function defined over vectors (Dhillon and Tropp, 2007). In general, every such φ defines a Bregman matrix divergence over real, symmetric matrices via the eigenvalue mapping. For example, if $\varphi(\boldsymbol{x}) = \boldsymbol{x}^T\boldsymbol{x}$, then the resulting composition $(\varphi \circ \lambda)(X)$ is the squared Frobenius norm. We call such divergences *spectral Bregman matrix divergences*.

Consider a spectral Bregman matrix divergence $D_\phi(X,Y)$, where $\phi = \varphi \circ \lambda$. We now show an alternate expression for $D_\phi(X,Y)$ based on the eigenvalues and eigenvectors of X and Y, which will prove to be useful when motivating extensions to the low-rank case.

Lemma 1 *Let the eigendecompositions of X and Y be $V\Lambda V^T$ and $U\Theta U^T$ respectively, and assume that φ is* separable*, that is, $\phi(X) = (\varphi \circ \lambda)(X) = \sum_i \varphi_i(\lambda_i)$. Then*

$$D_\phi(X,Y) = \sum_{i,j} (\boldsymbol{v}_i^T \boldsymbol{u}_j)^2 (\varphi_i(\lambda_i) - \varphi_j(\theta_j) - (\lambda_i - \theta_j)\nabla\varphi_j(\theta_j)).$$

Proof We have

$$\begin{aligned} D_\phi(X,Y) &= \phi(X) - \phi(Y) - \mathrm{tr}((X-Y)^T(\nabla\phi(Y))) \\ &= \sum_i \varphi_i(\lambda_i) - \sum_j \varphi_j(\theta_j) - \mathrm{tr}((V\Lambda V^T - U\Theta U^T)^T(\nabla\phi(Y))) \\ &= \sum_{i,j} (\boldsymbol{v}_i^T \boldsymbol{u}_j)^2 \varphi_i(\lambda_i) - \sum_{i,j} (\boldsymbol{v}_i^T \boldsymbol{u}_j)^2 \varphi_j(\theta_j) - \mathrm{tr}((V\Lambda V^T - U\Theta U^T)^T(\nabla\phi(Y))). \end{aligned}$$

The second line above uses the separability of φ, while the third line uses the fact that $\sum_i (\boldsymbol{v}_i^T \boldsymbol{u}_j)^2 = \sum_j (\boldsymbol{v}_i^T \boldsymbol{u}_j)^2 = 1$. We can express $\nabla\phi(Y)$ as:

$$\nabla\phi(Y) = U \begin{pmatrix} \nabla\varphi_1(\theta_1) & 0 & \dots \\ 0 & \nabla\varphi_2(\theta_2) & \dots \\ \vdots & \vdots & \ddots \end{pmatrix} U^T,$$

and so we have $\mathrm{tr}(U\Theta U^T \nabla\phi(Y)) = \sum_j \theta_j \nabla\varphi_j(\theta_j) = \sum_{i,j} (\boldsymbol{v}_i^T \boldsymbol{u}_j)^2 \theta_j \nabla\varphi_j(\theta_j)$. Finally, the term $\mathrm{tr}(V\Lambda V^T \nabla\phi(Y))$ can be expanded as $\sum_{i,j} (\boldsymbol{v}_i^T \boldsymbol{u}_j)^2 \lambda_i \nabla\varphi_j(\theta_j)$. Putting this all together, we have:

$$D_\phi(X,Y) = \sum_{i,j} (\boldsymbol{v}_i^T \boldsymbol{u}_j)^2 (\varphi_i(\lambda_i) - \varphi_j(\theta_j) - \nabla\varphi_j(\theta_j) \cdot (\lambda_i - \theta_j)).$$

■

Note that each of the divergences discussed earlier—the squared Frobenius divergence, the LogDet divergence, and the von Neumann divergence—arise from separable convex functions. Furthermore, in these three cases, the functions φ_i do not depend on i (so we denote $\varphi = \varphi_1 = \ldots = \varphi_n$) and the corollary below follows:

Corollary 2 *Given $X = V\Lambda V^T$ and $Y = U\Theta U^T$, the squared Frobenius, von Neumann and LogDet divergences satisfy:*

$$D_\phi(X,Y) = \sum_{i,j} (\boldsymbol{v}_i^T \boldsymbol{u}_j)^2 D_\varphi(\lambda_i, \theta_j). \tag{3}$$

This formula highlights the connection between a separable spectral matrix divergence and the corresponding vector divergence. It also illustrates how the divergence relates to the geometrical properties of the argument matrices as represented by the eigenvectors.

3.3 Problem Description

We now give a formal statement of the problem that we study in this paper. Given an input kernel matrix X_0, we attempt to solve the following for X:

$$\begin{aligned} \text{minimize} \quad & D_\phi(X, X_0) \\ \text{subject to} \quad & \operatorname{tr}(XA_i) \le b_i,\ 1 \le i \le c, \\ & \operatorname{rank}(X) \le r, \\ & X \succeq 0. \end{aligned} \tag{4}$$

Any of the linear inequality constraints $\operatorname{tr}(XA_i) \le b_i$ may be replaced with equalities. The above problem is clearly non-convex in general, due to the rank constraint. However, when the rank of X_0 does not exceed r, then this problem turns out to be convex when we use the von Neumann and LogDet matrix divergences. This is because these divergences restrict the search for the optimal X to the linear subspace of matrices that have the same range space as X_0. The details will emerge in Section 4. Another advantage of using the von Neumann and LogDet divergences is that the algorithms used to solve the minimization problem implicitly maintain the positive semidefiniteness constraint, so no extra work needs to be done to enforce positive semidefiniteness. This is in contrast to the squared Frobenius divergence, where the positive semidefiniteness constraint has to be explicitly enforced.

Though it is possible to handle general linear constraints of the form $\operatorname{tr}(XA_i) \le b_i$, we will focus on two specific types of constraints, which will be useful for our kernel learning applications. The first is a distance constraint. The squared Euclidean distance in feature space between the jth and the kth data points is given by $X_{jj} + X_{kk} - 2X_{jk}$. The constraint $X_{jj} + X_{kk} - 2X_{jk} \le b_i$ can be represented as $\operatorname{tr}(XA) \le b_i$, where $A_i = \boldsymbol{z}_i \boldsymbol{z}_i^T$, and $\boldsymbol{z}_i(j) = 1$, $\boldsymbol{z}_i(k) = -1$, and all other entries 0 (equivalently, $\boldsymbol{z}_i = \boldsymbol{e}_j - \boldsymbol{e}_k$). The second type of constraint has the form $X_{jk} \le b_i$, which can be written as $\operatorname{tr}(XA_i) \le b_i$ using $A_i = \frac{1}{2}(\boldsymbol{e}_j \boldsymbol{e}_k^T + \boldsymbol{e}_k \boldsymbol{e}_j^T)$ (we maintain symmetry of A_i).

For the algorithms developed in Section 5, we will assume that the matrices A_i in (4) are rank one (so $A_i = \boldsymbol{z}_i \boldsymbol{z}_i^T$) and that $r = \operatorname{rank}(X_0)$ (we briefly discuss extensions to higher-rank constraints in Section 6.2.2). In this case, we can write the optimization problem as:

$$\begin{aligned} \text{minimize} \quad & D_\phi(X, X_0) \\ \text{subject to} \quad & \boldsymbol{z}_i^T X \boldsymbol{z}_i \le b_i,\ 1 \le i \le c, \\ & \operatorname{rank}(X) \le r, \\ & X \succeq 0. \end{aligned} \tag{5}$$

Furthermore, we assume that there exists a feasible solution to the above problem; we discuss an extension to the infeasible case involving slack variables in Section 6.2.1. Note that in this case, we assume $b_i \geq 0$, as otherwise there cannot be a feasible solution.

The key application of the above optimization problem is in learning a kernel matrix in the setting where we have side information about some of the data points (e.g., labels or constraints), and we want to learn a kernel matrix over all the data points in order to perform classification, regression, or semi-supervised clustering. We will also discuss other applications throughout the paper.

3.4 Bregman Projections

Consider the convex optimization problem (4) presented above, but without the rank constraint (we will see how to handle the rank constraint later). To solve this problem, we use the method of Bregman projections (Bregman, 1967; Censor and Zenios, 1997). Suppose we wish to minimize $D_\phi(X,X_0)$ subject to linear equality and inequality constraints. The idea behind Bregman projections is to choose one constraint per iteration, and perform a projection so that the current solution satisfies the chosen constraint. Note that the projection is not an orthogonal projection, but rather a *Bregman projection*, which is tailored to the particular function that is being minimized. In the case of inequality constraints, an appropriate correction is also enforced. This process is then repeated by cycling through the constraints (or employing a more sophisticated control mechanism). This method may also be viewed as a dual coordinate ascent procedure that optimizes the dual with respect to a single dual variable per iteration (with all other dual variables remaining fixed). Under mild conditions, it can be shown that the method of cyclic Bregman projections (or a control mechanism that visits each constraint infinitely often) converges to the globally optimal solution; see Censor and Zenios (1997) and Dhillon and Tropp (2007) for more details.

Now for the details of each iteration. For an equality constraint of the form $\mathrm{tr}(XA_i) = b_i$, the Bregman projection of the current iterate X_t may be computed by solving:

$$\begin{aligned} \text{minimize}_X \quad & D_\phi(X,X_t) \\ \text{subject to} \quad & \mathrm{tr}(XA_i) = b_i. \end{aligned} \tag{6}$$

Introduce the dual variable α_i, and form the Lagrangian $\mathcal{L}(X,\alpha_i) = D_\phi(X,X_t) + \alpha_i(b_i - \mathrm{tr}(XA_i))$. By setting the gradient of the Lagrangian (with respect to X and α_i) to zero, we can obtain the Bregman projection by solving the resulting system of equations simultaneously for α_i and X_{t+1}:

$$\begin{aligned} \nabla\phi(X_{t+1}) &= \nabla\phi(X_t) + \alpha_i A_i \qquad (7) \\ \mathrm{tr}(X_{t+1}A_i) &= b_i. \end{aligned}$$

For an inequality constraint of the form $\mathrm{tr}(XA_i) \leq b_i$, let $\nu_i \geq 0$ be the corresponding dual variable. To maintain non-negativity of the dual variable (which is necessary for satisfying the KKT conditions), we can solve (7) for α_i and perform the following update:

$$\alpha_i' = \min(\nu_i, \alpha_i), \;\; \nu_i \leftarrow \nu_i - \alpha_i'. \tag{8}$$

See Appendix B for a discussion on why the dual variable corrections are needed. Clearly the update guarantees that $\nu_i \geq 0$. Finally, update X_{t+1} via

$$\nabla\phi(X_{t+1}) = \nabla\phi(X_t) + \alpha_i' A_i. \tag{9}$$

Note that (8) may be viewed as a correction step that follows the projection given by (7). Both of our algorithms in Section 5 are based on this method, which iteratively chooses a constraint and performs a Bregman projection until convergence. The main difficulty lies in efficiently solving the nonlinear system of equations given in (7).

In the case where $A_i = \boldsymbol{z}_i\boldsymbol{z}_i^T$, by calculating the gradient for the LogDet and von Neumann matrix divergences, respectively, (7) simplifies to:

$$\begin{aligned} X_{t+1} &= \left(X_t^{-1} - \alpha_i \boldsymbol{z}_i\boldsymbol{z}_i^T\right)^{-1} \\ X_{t+1} &= \exp(\log(X_t) + \alpha_i \boldsymbol{z}_i\boldsymbol{z}_i^T), \end{aligned} \tag{10}$$

subject to $\boldsymbol{z}_i^T X_{t+1}\boldsymbol{z}_i = b_i$. In (10), exp and log denote the matrix exponential and matrix logarithm, respectively. As we will see, for the von Neumann and LogDet divergences, these projections can be computed very efficiently (and are thus more desirable than other methods that involve multiple constraints at a time).

4. Bregman Divergences for Rank-Deficient Matrices

As given in (1) and (2), the von Neumann and LogDet divergences are undefined for low-rank matrices. For the LogDet divergence, the convex generating function $\phi(X) = -\log\det X$ is infinite when X is singular, that is, its effective domain is the set of positive definite matrices. For the von Neumann divergence the situation is somewhat better, since one can define $\phi(X) = \mathrm{tr}(X\log X - X)$ via continuity for rank-deficient matrices.

The key to using these divergences in the low-rank setting comes from restricting the convex function ϕ to the range spaces of the matrices. We motivate our approach using Corollary 2, before formalizing it in Section 4.2. Subsequently, we discuss the computation of Bregman projections for low-rank matrices in Section 4.3.

4.1 Motivation

If X and Y have eigendecompositions $X = V\Lambda V^T$ and $Y = U\Theta U^T$, respectively, then whenever X or Y is of low-rank, some eigenvalues λ_i of X or θ_j of Y are equal to zero. Consequently, if we could apply Corollary 2, the $D_\varphi(\lambda_i, \theta_j)$ terms that involve zero eigenvalues need careful treatment. More specifically, for the von Neumann divergence we have:

$$D_\varphi(\lambda_i, \theta_j) = \lambda_i \log\lambda_i - \lambda_i \log\theta_j - \lambda_i + \theta_j. \tag{11}$$

Using the convention that $0\log 0 = 0$, $D_\varphi(\lambda_i, \theta_j)$ equals 0 when both λ_i and θ_j are 0, but is infinite when θ_j is 0 but λ_i is non-zero. Similarly, with the LogDet divergence, we have

$$D_\varphi(\lambda_i, \theta_j) = \frac{\lambda_i}{\theta_j} - \log\frac{\lambda_i}{\theta_j} - 1. \tag{12}$$

In cases where $\lambda_i = 0$ and $\theta_j \neq 0$, or $\lambda_i \neq 0$ and $\theta_j = 0$, $D_\varphi(\lambda_i, \theta_j)$ is infinite.

For finiteness of the corresponding matrix divergence we require that $\boldsymbol{v}_i^T\boldsymbol{u}_j = 0$ whenever $D_\varphi(\lambda_i, \theta_j)$ is infinite so a cancellation will occur (via appropriate continuity arguments) and the divergence will be finite. This leads to properties of rank-deficient X and Y that are required for the matrix divergence to be finite. The following observations are discussed more formally in Section 4.2.

Observation 1 *The von Neumann divergence $D_{vN}(X,Y)$ is finite iff* $\text{range}(X) \subseteq \text{range}(Y)$.

The term $\lambda_i \log \theta_j$ from (11) is $-\infty$ if $\theta_j = 0$ but $\lambda_i \neq 0$. By using Corollary 2 and distributing the $(\boldsymbol{v}_i^T \boldsymbol{u}_j)^2$ term into the scalar divergence, we obtain: $\lambda_i(\boldsymbol{v}_i^T \boldsymbol{u}_j)^2 \log \theta_j$. Thus, if $\boldsymbol{v}_i^T \boldsymbol{u}_j = 0$ when $\theta_j = 0$, then $\lambda_i(\boldsymbol{v}_i^T \boldsymbol{u}_j)^2 \log \theta_j = 0$ (using $0 \log 0 = 0$), and the divergence is finite. When $\theta_j = 0$, the corresponding eigenvector $\boldsymbol{u}_j$ is in the null space of Y; therefore, for finiteness of the divergence, every vector $\boldsymbol{u}_j$ in the null space of Y is orthogonal to any vector $\boldsymbol{v}_i$ in the range space of X. This implies that the null space of Y contains the null space of X or, equivalently, $\text{range}(X) \subseteq \text{range}(Y)$.

Observation 2 *The LogDet divergence $D_{\ell d}(X,Y)$ is finite iff* $\text{range}(X) = \text{range}(Y)$.

To show the observation, we adopt the conventions that $\log \frac{0}{0} = 0$ and $\frac{0}{0} = 1$ in (12), which follow by continuity assuming that the rate at which the numerator and denominator approach zero is the same. Then, distributing the $(\boldsymbol{v}_i^T \boldsymbol{u}_j)^2$ term into D_φ, we have that $\boldsymbol{v}_i$ and $\boldsymbol{u}_j$ must be orthogonal whenever $\lambda_i = 0, \theta_j \neq 0$ or $\lambda_i \neq 0, \theta_j = 0$. This in turn says that: a) every vector $\boldsymbol{u}_j$ in the null space of Y must be orthogonal to every vector $\boldsymbol{v}_i$ in the range space of X and, b) every vector $\boldsymbol{v}_i$ in the null space of X must be orthogonal to every vector $\boldsymbol{u}_j$ in the range space of Y. It follows that the range spaces of X and Y are equal.

Assuming the eigenvalues of X and Y are listed in non-increasing order, we can write the low-rank equivalent of (3) simply as:

$$D_\phi(X,Y) = \sum_{i,j \leq r} (\boldsymbol{v}_i^T \boldsymbol{u}_j)^2 D_\varphi(\lambda_i, \theta_j),$$

where $r = \text{rank}(X)$.

If we now revisit our optimization problem formulated in (4), where we aim to minimize $D_\phi(X, X_0)$, we see that the LogDet and von Neumann divergences naturally maintain rank constraints if the problem is feasible. For the LogDet divergence, the equality of the range spaces of X and X_0 implies that when minimizing $D_\phi(X, X_0)$, we maintain $\text{rank}(X) = \text{rank}(X_0)$, assuming that there is a feasible X with a finite $D_\phi(X, X_0)$. Similarly, for the von Neumann divergence, the property that $\text{range}(X) \subseteq \text{range}(X_0)$ implies $\text{rank}(X) \leq \text{rank}(X_0)$.

4.2 Formalization Via Restrictions on the Range Space

The above section demonstrated informally that for $D_\phi(X,Y)$ to be finite the range spaces of X and Y must be equal for the LogDet divergence, and the range space of X must be contained in the range space of Y for the von Neumann divergence.

We now formalize the generalization to low-rank matrices. Let W be an orthogonal $n \times r$ matrix, such that its columns span the range space of Y. We will use the following simple and well known lemma later on:

Lemma 3 *Let Y be a symmetric $n \times n$ matrix with* $\text{rank}(Y) \leq r$, *and let W be an $n \times r$ column orthogonal matrix ($W^T W = I$) with* $\text{range}(Y) \subseteq \text{range}(W)$. *Then $Y = WW^T YWW^T$.*

Proof Extend W to a full $n \times n$ orthogonal matrix and denote it by W_f. Notice that the last $(n-r)$ rows of $W_f^T Y$ and the last $(n-r)$ columns of YW_f consist of only zeros. It follows that the matrix $Y_1 = W_f^T Y W_f$ contains only zeros except for the $r \times r$ sub-matrix in the top left corner, which coincides with $\hat{Y} = W^T Y W$. Observe that $Y = W_f Y_1 W_f^T = W \hat{Y} W^T = WW^T YWW^T$ to finish the proof. ■

We are now ready to extend the domain of the von Neumann and LogDet divergences to low-rank matrices.

Definition 4 *Consider the positive semidefinite $n \times n$ matrices X and Y that satisfy* $\mathrm{range}(X) \subseteq \mathrm{range}(Y)$ *when considering the von Neumann divergence, and* $\mathrm{range}(X) = \mathrm{range}(Y)$ *when considering the LogDet divergence. Let W be an $n \times r$ column orthogonal matrix such that range$(Y) \subseteq$ range(W) and define:*

$$D_\phi(X,Y) = D_{\phi,W}(X,Y) = D_\phi(W^T XW, W^T YW), \tag{13}$$

where D_ϕ is either the von Neumann or the LogDet divergence.

The first equality in (13) implicitly assumes that the right hand side is not dependent on the choice of W.

Lemma 5 *In Definition 4, $D_{\phi,W}(X,Y)$ is independent of the choice of W.*

Proof All the $n \times r$ orthogonal matrices with the same range space as W can be expressed as a product WQ, where Q is an $r \times r$ orthogonal matrix. Introduce $\hat{X} = W^T XW$ and $\hat{Y} = W^T YW$ and substitute WQ in place of W in (13):

$$\begin{aligned} D_{\phi,WQ}(X,Y) &= D_\phi((WQ)^T X(WQ), (WQ)^T Y(WQ)) \\ &= D_\phi(Q^T \hat{X} Q, Q^T \hat{Y} Q) \\ &= D_\phi(\hat{X}, \hat{Y}) = D_{\phi,W}(X,Y). \end{aligned}$$

The first equality is by Definition 4, and the third follows from the fact that the von Neumann and the LogDet divergences are invariant under any orthogonal similarity transformation[2] (see Proposition 11 in Appendix A). ■

We now show that Definition 4 is consistent with Corollary 2, demonstrating that our low-rank formulation agrees with the informal discussion given earlier.

Theorem 6 *Let the positive semidefinite matrices X and Y have eigendecompositions $X = V\Lambda V^T$, $Y = U\Theta U^T$ and let* $\mathrm{range}(X) \subseteq \mathrm{range}(Y)$. *Let the rank of Y equal r. Assuming that the eigenvalues of X and Y are sorted in non-increasing order, that is, $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_r$ and $\theta_1 \geq \theta_2 \geq \ldots \geq \theta_r$, then*

$$D_{vN}(X,Y) = \sum_{i,j \leq r} (\boldsymbol{v}_i^T \boldsymbol{u}_j)^2 (\lambda_i \log \lambda_i - \lambda_i \log \theta_j - \lambda_i + \theta_j).$$

Proof Denote the upper left $r \times r$ submatrices of Λ and Θ by Λ_r and Θ_r respectively, and the corresponding reduced eigenvector matrices for X and Y by V_r and U_r. Picking W in (13) to equal U_r, we get:

$$D_{vN}(X,Y) = D_{vN}(U_r^T XU_r, U_r^T YU_r) = D_{vN}((U_r^T V_r)\Lambda_r(V_r^T U_r), \Theta_r).$$

The arguments on the right hand side are $r \times r$ matrices and Θ_r is not rank-deficient. We can now apply Corollary 2 to get the result. ■

2. In fact, in the case of the LogDet divergence we have invariance under any invertible congruence transformation, see Proposition 12 in Appendix A.

Theorem 7 *Let the positive semidefinite matrices X and Y have eigendecompositions $X = V\Lambda V^T$, $Y = U\Theta U^T$, and let* range$(X) =$ range(Y) *and assume that the eigenvalues of X and Y are sorted in decreasing order. Then:*

$$D_{\ell d}(X,Y) = \sum_{i,j \le r} (\boldsymbol{v}_i^T \boldsymbol{u}_j)^2 \left(\frac{\lambda_i}{\theta_j} - \log \frac{\lambda_i}{\theta_j} - 1 \right).$$

Proof Similar to the proof of Theorem 6. Note that the range spaces must coincide in this case, because the determinant of XY^{-1} should not vanish for the restricted transformations, which agrees with the range$(X) =$ range(Y) restriction. ■

We next show that the optimization problem and Bregman's projection algorithm for low-rank matrices can be cast as a full rank problem in a lower dimensional space, namely the range space. This equivalence implies that we do not have to rederive the convergence proofs and other properties of Bregman's algorithm in the low-rank setting.

Consider the optimization problem (4) for low-rank X_0, and denote a suitable orthogonal matrix as required in Definition 4 by W. The matrix of the eigenvectors of the reduced eigendecomposition of X_0 is a suitable choice. Consider the following matrix mapping:

$$M \longrightarrow \hat{M} = W^T M W.$$

By Lemma 3, the mapping is one-to-one on the set of symmetric matrices with range space contained in range(W). We now apply the mapping to all matrices in the optimization problem (4) to obtain:

$$\begin{array}{ll} \text{minimize} & D_\phi(\hat{X}, \hat{X}_0) \\ \text{subject to} & \operatorname{tr}(\hat{X}\hat{A}_i) \le b_i, \; 1 \le i \le c \\ & \operatorname{rank}(\hat{X}) \le r \\ & \hat{X} \succeq 0. \end{array} \tag{14}$$

The rank constraint is automatically satisfied when rank$(X_0) = r$ and the problem is feasible. Clearly, $\hat{X} \succeq 0$ if and only if $X \succeq 0$. By Definition 4, $D_\phi(\hat{X}, \hat{X}_0) = D_\phi(X, X_0)$. Finally, the next lemma verifies that the constraints are equivalent as well.

Lemma 8 *Given a column orthogonal $n \times r$ matrix W satisfying* range$(X) \subseteq$ range(W), *it follows that* tr$(\hat{X}\hat{A}_i) =$ tr(XA_i), *where $\hat{X} = W^T X W$, $\hat{A}_i = W^T A_i W$.*

Proof Choose a reduced rank-r eigendecomposition of X to be $V\Lambda V^T$ such that the columns of V form an orthogonal basis of range(W). Note that Λ will be singular when rank(X) is less than r. There exists an $r \times r$ orthogonal Q that satisfies $W = VQ$, and so:

$$\begin{aligned} \operatorname{tr}(\hat{X}\hat{A}_i) &= \operatorname{tr}((W^T X W)(W^T A_i W)) = \operatorname{tr}(Q^T V^T V \Lambda V^T V Q Q^T V^T A_i V Q) \\ &= \operatorname{tr}(V Q Q^T \Lambda Q Q^T V^T A_i) = \operatorname{tr}(V \Lambda V^T A_i) = \operatorname{tr}(X A_i). \end{aligned}$$

■

If we assume that the optimization problem (4) has a (rank-deficient) solution with finite divergence measure, then the corresponding full-rank optimization problem (14) also has a solution. Conversely, by Lemma 3, for a solution $\hat{X}$ of (14), there is a unique solution of (4), namely $X = W\hat{X}W^T$ (with finite $D_\phi(X, X_0)$) that satisfies the range space restriction.

4.3 Bregman Projections for Rank-deficient Matrices

Lastly, we derive the explicit updates for Bregman's algorithm in the low-rank setting. From now on we omit the constraint index for simplicity. Recall (7), which we used to calculate the projection update for Bregman's algorithm and apply it to the mapped problem (14):

$$\begin{aligned}\nabla\phi(\hat{X}_{t+1}) &= \nabla\phi(\hat{X}_t)+\alpha\hat{A}\\ \mathrm{tr}(\hat{X}_{t+1}\hat{A}) &= b.\end{aligned}$$

In case of the von Neumann divergence this leads to the update $\hat{X}_{t+1}=\exp(\log(\hat{X}_t)+\alpha\hat{A})$. The discussion in Section 4.2 and induction on t implies that $X_{t+1}=W\hat{X}_{t+1}W^T$ (with W as in Definition 4), or explicitly:

$$X_{t+1}=W\exp(\log(W^TX_tW)+\alpha(W^TAW))W^T.$$

If we choose $W=V_t$ from the reduced eigendecomposition $X_t=V_t\Lambda_tV_t^T$, then the update is written as:

$$X_{t+1}=V_t\exp(\log(\Lambda_t)+\alpha V_t^TAV_t)V_t^T, \tag{15}$$

which we will use in the von Neumann kernel learning algorithm. Note that the limit of $\exp(\log(X_t+\varepsilon I)+\alpha A)$ as ε approaches zero yields the same formula, which becomes clear if we apply a basis transformation that puts X_t in diagonal form.

The same argument applies to the Bregman projection for the LogDet divergence. In this case we arrive at the update:

$$X_{t+1}=V_t((V_t^TX_tV_t)^{-1}-\alpha(V_t^TAV_t))^{-1}V_t^T, \tag{16}$$

using Lemma 3 and the fact that $\mathrm{range}(X_{t+1})=\mathrm{range}(X_t)$. The right hand side of (16) can be calculated without any eigendecomposition as we will show in Section 5.1.1.

5. Algorithms

In this section, we derive efficient algorithms for solving the optimization problem (5) for low-rank matrices using the LogDet and von Neumann divergences.

5.1 LogDet Divergence

We first develop a cyclic projection algorithm to solve (5) when the matrix divergence is the LogDet divergence.

5.1.1 MATRIX UPDATES

Consider minimizing $D_{\ell d}(X,X_0)$, the LogDet divergence between X and X_0. As given in (16), the projection update rule for low-rank X_0 and a rank-one constraint matrix $A=\boldsymbol{z}\boldsymbol{z}^T$ is:

$$X_{t+1}=V_t((V_t^TX_tV_t)^{-1}-\alpha(V_t^T\boldsymbol{z}\boldsymbol{z}^TV_t))^{-1}V_t^T,$$

where the eigendecomposition of X_t is $V_t\Lambda_tV_t^T$. Recall the Sherman-Morrison inverse formula (Sherman and Morrison, 1949; Golub and Van Loan, 1996):

$$(A+\boldsymbol{u}\boldsymbol{v}^T)^{-1}=A^{-1}-\frac{A^{-1}\boldsymbol{u}\boldsymbol{v}^TA^{-1}}{1+\boldsymbol{v}^TA^{-1}\boldsymbol{u}}.$$

Applying this formula to the middle term of the projection update, we arrive at a simplified expression for X_{t+1}:

$$\begin{aligned} X_{t+1} &= V_t\left(V_t^T X_t V_t + \frac{\alpha V_t^T X_t V_t V_t^T \boldsymbol{z}\boldsymbol{z}^T V_t V_t^T X_t V_t}{1-\alpha \boldsymbol{z}^T V_t V_t^T X_t V_t V_t^T \boldsymbol{z}}\right)V_t^T \\ &= V_t\left(\Lambda_t + \frac{\alpha \Lambda_t V_t^T \boldsymbol{z}\boldsymbol{z}^T V_t \Lambda_t}{1-\alpha \boldsymbol{z}^T V_t \Lambda_t V_t^T \boldsymbol{z}}\right)V_t^T \\ &= V_t \Lambda_t V_t^T + \frac{\alpha V_t \Lambda_t V_t^T \boldsymbol{z}\boldsymbol{z}^T V_t \Lambda_t V_t^T}{1-\alpha \boldsymbol{z}^T X_t \boldsymbol{z}} \\ &= X_t + \frac{\alpha X_t \boldsymbol{z}\boldsymbol{z}^T X_t}{1-\alpha \boldsymbol{z}^T X_t \boldsymbol{z}}. \end{aligned} \tag{17}$$

Since X_{t+1} must satisfy the constraint, i. e., $\mathrm{tr}(X_{t+1}\boldsymbol{z}\boldsymbol{z}^T) = \boldsymbol{z}^T X_{t+1}\boldsymbol{z} = b$, we can solve the following equation for α:

$$\boldsymbol{z}^T\left(X_t + \frac{\alpha X_t \boldsymbol{z}\boldsymbol{z}^T X_t}{1-\alpha \boldsymbol{z}^T X_t \boldsymbol{z}}\right)\boldsymbol{z} = b. \tag{18}$$

Let $p = \boldsymbol{z}^T X_t \boldsymbol{z}$. Note that in the case of distance constraints, p is the distance between the two data points. When $p \neq 0$, elementary arguments reveal that there is exactly one solution for α provided that $b \neq 0$. The unique solution, in this case, can be expressed as:

$$\alpha = \frac{1}{p} - \frac{1}{b}. \tag{19}$$

If we let

$$\beta = \alpha/(1-\alpha p), \tag{20}$$

then our matrix update is given by

$$X_{t+1} = X_t + \beta X_t \boldsymbol{z}\boldsymbol{z}^T X_t. \tag{21}$$

This update is pleasant and surprising; usually the projection parameter for Bregman's algorithm does not admit a closed form solution (see Section 5.2.2 for the case of the von Neumann divergence). In the case where $p = 0$, (18) rewrites to $b = p/(1-\alpha p)$, which has a solution if and only if $b = 0$ (while α is arbitrary). It follows that $\boldsymbol{z}$ is in the null space of X_t, and by (17), $X_{t+1} = X_t$.

The following lemma confirms the expectation that we remain in the positive semidefinite cone and that the range space is unchanged.

Lemma 9 *Given a positive semidefinite matrix X_t, the matrix X_{t+1} from the update in (21) is positive semidefinite with* $\mathrm{range}(X_{t+1}) = \mathrm{range}(X_t)$, *assuming that (6) is feasible.*

Proof Factor the positive semidefinite matrix X_t as GG^T, where G is an $n \times r$ matrix and $r = \mathrm{rank}(X_t)$. The LogDet update produces:

$$X_{t+1} = GG^T + \beta GG^T \boldsymbol{z}\boldsymbol{z}^T GG^T = G(I + \beta G^T \boldsymbol{z}\boldsymbol{z}^T G)G^T.$$

We deduce immediately that $\mathrm{range}(X_{t+1}) \subseteq \mathrm{range}(G) = \mathrm{range}(X_t)$.

In order to prove that X_{t+1} is positive semidefinite and that the range space does not shrink, it suffices to show that all eigenvalues of $\beta G^T \boldsymbol{z}\boldsymbol{z}^T G$ are strictly larger than -1, implying that $I + \beta G^T \boldsymbol{z}\boldsymbol{z}^T G$ remains positive definite. The only non-zero eigenvalue of the rank-one $\beta G^T \boldsymbol{z}\boldsymbol{z}^T G$ equals $\mathrm{tr}(\beta G^T \boldsymbol{z}\boldsymbol{z}^T G) = \beta \boldsymbol{z}^T G G^T \boldsymbol{z} = \beta p$. According to (19) and (20) we calculate $\beta p = \frac{b}{p} - 1$, and noting that $b, p > 0$ completes the proof. ■

The update for X_{t+1} can alternatively be written using the pseudoinverse (Golub and Van Loan, 1996):

$$X_{t+1} = (X_t^\dagger - \alpha \tilde{A}_i)^\dagger,$$

where $\tilde{A}_i = WW^T A_i WW^T$ for an orthogonal matrix W satisfying $\mathrm{range}(W) = \mathrm{range}(X_t)$.[3]

Lemma 10 *Let $\tilde{A}_i = WW^T A_i WW^T$ and W be an orthogonal $n \times r$ matrix such that* $\mathrm{range}(W) = \mathrm{range}(X_t)$. *Then the following updates are equivalent:*

$$\begin{aligned} a)\ X_{t+1} &= W((W^T X_t W)^{-1} - \alpha(W^T A_i W))^{-1} W^T \\ b)\ X_{t+1} &= (X_t^\dagger - \alpha \tilde{A}_i)^\dagger \end{aligned}$$

Proof Since $\mathrm{range}(W) = \mathrm{range}(X_t)$, by Lemma 3 we have $\hat{X}_t = W^T X_t W$ and $X_t = W\hat{X}_t W^T$, where $\hat{X}_t$ is an invertible $r \times r$ matrix. Note that $W\hat{X}_t^{-1}W^T$ satisfies the properties required for the Moore-Penrose pseudoinverse, for example:

$$X_t(W\hat{X}_t^{-1}W^T)X_t = W\hat{X}_t W^T W \hat{X}_t^{-1} W^T W \hat{X}_t W^T = W\hat{X}_t W^T = X_t.$$

Thus $X_t^\dagger = W\hat{X}_t^{-1}W^T$ and we finish the proof by rewriting:

$$\begin{aligned} (X_t^\dagger - \alpha \tilde{A}_i)^\dagger &= (W\hat{X}_t^{-1}W^T - \alpha WW^T A_i WW^T)^\dagger = \left(W(\hat{X}_t^{-1} - \alpha W^T A_i W)W^T\right)^\dagger \\ &= W((W^T X_t W)^{-1} - \alpha(W^T A_i W))^{-1} W^T. \end{aligned}$$

■

5.1.2 UPDATE FOR THE FACTORED MATRIX

A naive implementation of the update given in (21) costs $O(n^2)$ operations per iteration. However, we can achieve a more efficient update for low-rank matrices by working on a suitable factored form of the matrix X_t resulting in an $O(r^2)$ algorithm. Both the reduced eigendecomposition and the Cholesky factorization are possible candidates for the factorization; we prefer the latter because the resulting algorithm does not have to rely on iterative methods.

The positive semidefinite rank-r matrix X_t can be factored as GG^T, where G is an $n \times r$ matrix, and thus the update can be written as:

$$X_{t+1} = G(I + \beta G^T \boldsymbol{z}\boldsymbol{z}^T G)G^T.$$

The matrix $I + \beta \tilde{\boldsymbol{z}}_i \tilde{\boldsymbol{z}}_i^T$, where $\tilde{\boldsymbol{z}}_i = G^T \boldsymbol{z}$, is an $r \times r$ matrix. To update G for the next iteration, we factor this matrix as LL^T; then our new G is updated to GL. Since $I + \beta \tilde{\boldsymbol{z}}_i \tilde{\boldsymbol{z}}_i^T$ is a rank-one

3. In Kulis et al. (2006), we had inadvertently used A_i instead of $\tilde{A}_i$.

Algorithm 1 CHOLUPDATEMULT(α, x, B). Right multiplication of a lower triangular $r \times r$ matrix B with the Cholesky factor of $I + \alpha \boldsymbol{x}\boldsymbol{x}^T$ in $O(r^2)$ time.

```
Input: $\alpha, \boldsymbol{x}, B$, with $I + \alpha \boldsymbol{x}\boldsymbol{x}^T \succeq 0$, $B$ is lower triangular
Output: $B \leftarrow BL$, with $LL^T = I + \alpha \boldsymbol{x}\boldsymbol{x}^T$
 1: $\alpha_1 = \alpha$
 2: for $i = 1$ to $r$ do
 3:     $t = 1 + \alpha_i x_i^2$
 4:     $h_i = \sqrt{t}$
 5:     $\alpha_{i+1} = \alpha_i / t$
 6:     $t = B_{ii}$
 7:     $s = 0$
 8:     $B_{ii} = B_{ii} h_i$
 9:     for $j = i - 1$ downto $1$ do
10:         $s = s + t x_{j+1}$
11:         $t = B_{ij}$
12:         $B_{ij} = (B_{ij} + \alpha_{j+1} x_j s) h_j$
13:     end for
14: end for
```

perturbation of the identity, this update can be done in $O(r^2)$ time using a standard Cholesky rank-one update routine.

To increase computational efficiency, we note that $G = G_0 B$, where B is the product of all the L matrices from every iteration and $X_0 = G_0 G_0^T$. Instead of updating G explicitly at each iteration, we simply update B to BL. The matrix $I + \beta G^T \boldsymbol{z}\boldsymbol{z}^T G$ is then $I + \beta B^T G_0^T \boldsymbol{z}\boldsymbol{z}^T G_0 B$. In the case of distance constraints, we can compute $G_0^T \boldsymbol{z}$ in $O(r)$ time as the difference of two rows of G_0. The multiplication update BL appears to have $O(r^3)$ complexity, dominating the run time. In the next section we derive an algorithm that combines the Cholesky rank-one update with the matrix multiplication into a single $O(r^2)$ routine.

5.1.3 FAST MULTIPLICATION WITH A CHOLESKY UPDATE FACTOR

We efficiently combine the Cholesky rank-one update with the matrix multiplication in CHOLUPDATEMULT given by Algorithm 1. A simple analysis of this algorithm reveals that it requires $3r^2 + 2r$ floating point operations (flops). This is opposed to the usual $O(r^3)$ time needed by matrix multiplication. We devote this section to the development of this fast multiplication algorithm.

Recall the algorithm used for the Cholesky factorization of an $r \times r$ matrix A (see Demmel, 1997, page 78):

```
for $j = 1$ to $r$ do
    $l_{jj} = (a_{jj} - \sum_{k=1}^{j-1} l_{jk}^2)^{1/2}$
    for $i = j + 1$ to $r$ do
        $l_{ij} = (a_{ij} - \sum_{k=1}^{j-1} l_{ik} l_{jk}) / l_{jj}$
    end for
end for
```

We will derive a corresponding algorithm in a manner similar to Demmel (1997), while exploiting the special structure present in our problem. Let us denote $I_r + \alpha_1 \boldsymbol{x}\boldsymbol{x}^T$ by A ($\alpha_1 = \alpha$) and write it as a product of three block matrices:

$$A = \begin{bmatrix} \sqrt{1+\alpha_1 x_1^2} & 0 \\ \frac{\alpha_1 x_1}{\sqrt{1+\alpha_1 x_1^2}}\boldsymbol{x}_{2:r} & I_{r-1} \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & \tilde{A}_{22} \end{bmatrix} \begin{bmatrix} \sqrt{1+\alpha_1 x_1^2} & \frac{\alpha_1 x_1}{\sqrt{1+\alpha_1 x_1^2}}\boldsymbol{x}_{2:r}^T \\ 0 & I_{r-1} \end{bmatrix}.$$

It follows that $\tilde{A}_{22} + \frac{\alpha_1^2 x_1^2}{1+\alpha_1 x_1^2}\boldsymbol{x}_{2:r}\boldsymbol{x}_{2:r}^T = A_{22} = I_{r-1} + \alpha_1 \boldsymbol{x}_{2:r}\boldsymbol{x}_{2:r}^T$, leading to:

$$\tilde{A}_{22} = I_{r-1} + \frac{\alpha_1}{1+\alpha_1 x_1^2}\boldsymbol{x}_{2:r}\boldsymbol{x}_{2:r}^T.$$

Introduce $\alpha_2 = \frac{\alpha_1}{1+\alpha_1 x_1^2}$ and proceed by induction. We extract the following algorithm which calculates L satisfying $LL^T = A$:

```
for $j = 1$ to $r$ do
    $t = 1 + \alpha_j x_j^2$
    $l_{jj} = \sqrt{t}$
    $\alpha_{j+1} = \alpha_j / t$
    $t = \alpha_j x_j / l_{jj}$
    for $i = j+1$ to $r$ do
        $l_{ij} = t x_i$
    end for
end for
```

The above algorithm uses $\frac{1}{2}r^2 + \frac{13}{2}r$ flops to calculate L, while $\frac{1}{2}r^3 + O(r^2)$ are needed for the general algorithm. However, we do not necessarily have to calculate L explicitly, since the parameters α_i together with $\boldsymbol{x}$ implicitly determine L. Notice that the cost of calculating $\alpha_1, \alpha_2, \ldots, \alpha_r$ is linear in r.

Next we show how to calculate $\boldsymbol{u}^T L$ for a given vector $\boldsymbol{u}$ without explicitly calculating L and arrive at an $O(r)$ algorithm for this vector-matrix multiplication. The elements of $\boldsymbol{v}^T = \boldsymbol{u}^T L$ are equal to:

$$\begin{aligned} v_1 &= u_1\sqrt{1+\alpha_1 x_1^2} + \frac{\alpha_2}{\sqrt{1+\alpha_1 x_1^2}} x_1(u_2x_2 + u_3x_3 + \ldots u_r x_r) \\ v_2 &= u_2\sqrt{1+\alpha_2 x_2^2} + \frac{\alpha_3}{\sqrt{1+\alpha_2 x_2^2}} x_2(u_3x_3 + \ldots u_r x_r) \\ &\vdots \\ v_{r-1} &= u_{r-1}\sqrt{1+\alpha_{r-1} x_{r-1}^2} + \frac{\alpha_r}{\sqrt{1+\alpha_{r-1} x_{r-1}^2}} x_{r-1} u_r x_r \\ v_r &= u_r\sqrt{1+\alpha_r x_r^2}. \end{aligned}$$

We can avoid the recalculation of some intermediate results if we evaluate v_r first, followed by v_{r-1} up to v_1. This strategy leads to the following algorithm:

Algorithm 2 Learning a low-rank kernel matrix in LogDet divergence under distance constraints.

```
Input: $G_0$: input kernel factor matrix, that is, $X_0 = G_0G_0^T$, $r$: desired rank, $\{A_i\}_{i=1}^c$: distance
    constraints, where $A_i = (e_{i_1} - e_{i_2})(e_{i_1} - e_{i_2})^T$
Output: $G$: output low-rank kernel factor matrix, that is, $X = GG^T$
 1: Set $B = I_r$, $i = 1$, and $\nu_j = 0$ $\forall$ constraints $j$.
 2: repeat
 3:     $\boldsymbol{v}^T = G_0(i_1,:) - G_0(i_2,:)$
 4:     $\boldsymbol{w} = B^T\boldsymbol{v}$
 5:     $\alpha = \min\left(\nu_i, \frac{1}{\|\boldsymbol{w}\|_2^2} - \frac{1}{b_i}\right)$
 6:     $\nu_i \leftarrow \nu_i - \alpha$
 7:     $\beta = \alpha/(1 - \alpha\|\boldsymbol{w}\|_2^2)$
 8:     Call CHOLUPDATEMULT$(\beta, \boldsymbol{w}, B)$ to factor $I + \beta\boldsymbol{w}\boldsymbol{w}^T = LL^T$ and update $B \leftarrow BL$.
 9:     Set $i \leftarrow \mathrm{mod}(i+1, c)$.
10: until convergence of $\nu$
11: return $G = G_0B$
```

```
$v_r = u_r\sqrt{1 + \alpha_r x_r^2}$
s = 0;
for $j = r - 1$ downto 1 do
    $s = s + u_{j+1}x_{j+1}$
    $t = \sqrt{1 + \alpha_j x_j^2}$
    $v_j = u_j t + \alpha_{j+1}x_j s/t$
end for
```

Exactly $11r - 6$ flops are required by the above algorithm, and therefore we can readily multiply an $r \times r$ matrix by L using $11r^2 - 6r$ flops. Even fewer flops are sufficient to implement the matrix multiplication if we observe that the square root expression above is repeatedly calculated for each row, since it depends only on x_j and α_j. Additionally, when multiplying with a lower triangular matrix, the presence of zeros allows further simplifications. Taking these considerations into account we arrive at the previously presented Algorithm 1, which uses exactly $3r^2 + 2r$ flops.

5.1.4 KERNEL LEARNING ALGORITHM

We are now ready to present the overall algorithm for distance inequality constraints using the LogDet divergence, see Algorithm 2. As discussed in the previous section, every projection can be done in $O(r^2)$ time. Thus, cycling through all c constraints requires $O(cr^2)$ time, but the total number of Bregman iterations may be large. The only dependence on the number of data points n occurs in steps 3 and 11. The last step, multiplying $G = G_0B$, takes $O(nr^2)$ time.

As discussed earlier, convergence is guaranteed since we have mapped the original low-rank problem into a full-rank problem in a lower-dimensional space. Convergence can be checked by using the dual variables ν. The cyclic projection algorithm can be viewed as a dual coordinate ascent algorithm, thus convergence can be measured as follows: after cycling through all constraints, we check to see how much ν has changed after a full cycle. At convergence, this change (as measured with a vector norm) should be small.

5.2 Von Neumann Divergence

In this section we develop a cyclic projection algorithm to solve (5) when the matrix divergence is the von Neumann divergence.

5.2.1 MATRIX UPDATES

Consider minimizing $D_{vN}(X,X_0)$, the von Neumann divergence between X and X_0. Recall the projection update (15) for constraint i:

$$X_{t+1} = V_t \exp(\log(\Lambda_t) + \alpha V_t^T \boldsymbol{z}\boldsymbol{z}^T V_t) V_t^T.$$

If the eigendecomposition of the exponent $\log(\Lambda_t) + \alpha V_t^T \boldsymbol{z}\boldsymbol{z}^T V_t$ is $U_t \Theta_{t+1} U_t^T$, then the eigendecomposition of X_{t+1} is given by $V_{t+1} = V_t U_t$ and $\Lambda_{t+1} = \exp(\Theta_{t+1})$. This special eigenvalue problem (diagonal plus rank one update) can be solved in $O(r^2)$ time; see Golub (1973), Gu and Eisenstat (1994) and Demmel (1997). This means that the matrix multiplication $V_{t+1} = V_t U_t$ becomes the most expensive step in the computation, yielding $O(nr^2)$ complexity.

We reduce this cost by modifying the decomposition slightly. Let $X_t = V_t W_t \Lambda_t W_t^T V_t^T$ be the factorization of X_t, where W_t is an $r \times r$ orthogonal matrix, initially $W_0 = I_r$, while V_t and Λ_t are defined as before. The matrix update becomes

$$X_{t+1} = V_t W_t \exp(\log \Lambda_t + \alpha W_t^T V_t^T \boldsymbol{z}\boldsymbol{z}^T V_t W_t) W_t^T V_t^T,$$

yielding the following formulae:

$$V_{t+1} = V_t,\ W_{t+1} = W_t U_t,\ \Lambda_{t+1} = \exp(\Theta_{t+1}),$$

where $\log \Lambda_t + \alpha W_t^T V_t^T \boldsymbol{z}\boldsymbol{z}^T V_t W_t = U_t \Theta_{t+1} U_t^T$. For a general rank-one constraint the product $V_t^T \boldsymbol{z}$ is calculated in $O(nr)$ time, but for distance constraints $O(r)$ operations are sufficient. The calculation of $W_t^T V_t^T \boldsymbol{z}$ and computing the eigendecomposition $U_t \Theta_{t+1} U_t^T$ will both take $O(r^2)$ time. The matrix product $W_t U_t$ appears to cost $O(r^3)$ time, but in fact a right multiplication by U_t^T can be approximated very accurately in $O(r^2 \log r)$ and even in $O(r^2)$ time using the fast multipole method—see Barnes and Hut (1986) and Greengard and Rokhlin (1987).

Since we repeat the above update calculations until convergence, we can avoid calculating the logarithm of Λ_t at every step by maintaining $\Theta_t = \log \Lambda_t$ throughout the algorithm.

5.2.2 COMPUTING THE PROJECTION PARAMETER

In the previous section, we assumed that the projection parameter α has already been calculated. In contrast to the LogDet divergence, this parameter does not have a closed form solution. Instead, we must compute α by solving the nonlinear system of equations given by (7).

In the von Neumann case and in the presence of distance constraints, the problem amounts to finding the unique root of the function

$$f(\alpha) = \boldsymbol{z}^T V_t W_t \exp(\Theta_t + \alpha W_t^T V_t^T \boldsymbol{z}\boldsymbol{z}^T V_t W_t) W_t^T V_t^T \boldsymbol{z} - b.$$

It can be shown that $f(\alpha)$ is monotone in α, see Sustik and Dhillon (2008).

Using the approach from the previous section, $f(\alpha)$ can be computed in $O(r^2)$ time. One natural choice to find the root of $f(\alpha)$ is to apply Brent's general root finding method (Brent, 1973),

Algorithm 3 Learning a low-rank kernel matrix in von Neumann matrix divergence under distance constraints.

```
Input: $V_0, \Lambda_0$: input kernel factors, i. e., $X_0 = V_0\Lambda_0V_0^T$, $r$: rank of desired kernel matrix, $\{A_i\}_{i=1}^c$:
    distance constraints, where $A_i = (e_{i_1} - e_{i_2})(e_{i_1} - e_{i_2})^T$
Output: $V, \Lambda$: output low-rank kernel factors, i. e., $X = V\Lambda V^T$
1: Set $W = I_r$, $\Theta = \log\Lambda_0$, $i = 1$, and $\nu_j = 0$ $\forall$ constraints $j$.
2: repeat
3:     $\boldsymbol{v}^T = V_0(i_1,:) - V_0(i_2,:)$
4:     $\boldsymbol{w} = W^T\boldsymbol{v}$
5:     Call ZEROFINDER$(\Theta, \boldsymbol{w}, b_i)$ to determine $\alpha$.
6:     $\beta = \min(\nu_i, \alpha)$
7:     $\nu_i \leftarrow \nu_i - \beta$.
8:     Compute eigendecomposition of $\Theta + \beta\boldsymbol{w}\boldsymbol{w}^T = U\tilde{\Theta}U^T$, $\Theta \leftarrow \tilde{\Theta}$.
9:     Update $W \leftarrow WU$.
10:    $i = \mod(i+1, c)$
11: until convergence of $\nu$
12: return $V = V_0W$, $\Lambda = \exp(\Theta)$
```

which does not need the calculation of the derivative of $f(\alpha)$. We have built an even more efficient custom root finder that is optimized for this problem. We rarely need more than six evaluations per projection to accurately compute α. A more detailed description of our root finder is beyond the scope of this paper, see Sustik and Dhillon (2008) for details.

5.2.3 KERNEL LEARNING ALGORITHM

The algorithm for distance inequality constraints using the von Neumann divergence is given as Algorithm 3. By using the fast multipole method every projection can be done in $O(r^2)$ time. Note that the asymptotic running time of this algorithm is the same as the LogDet divergence algorithm, although the root finding step makes Algorithm 3 slightly slower than Algorithm 2.

5.3 Limitations of our Approach

We conclude this section by briefly mentioning some limitations of our approach. First, though we are able to learn low-rank kernel matrices in our framework, the initial kernel matrix must be low-rank. As a result, we cannot use our methods to reduce the dimensionality of our data. Secondly, the method of Bregman projections may require iterating many cycles through all constraints to reach convergence. Although we have heavily optimized each iteration in this paper, it may be beneficial to search for an algorithm with faster convergence.

6. Discussion

We now further develop several aspects of the algorithms presented in the previous section. In particular, we discuss ways to move beyond the transductive setting in our framework for learning kernels, we briefly overview some generalizations of the problem formulation, we highlight how special cases of our method are related to existing techniques, and we briefly analyze connections between our approach and semidefinite programming.

6.1 Handling New Data Points

The kernel learning algorithms of Section 5 are restricted to learning in the transductive setting; that is, we assume that we have all the data points up front, but labels or other forms of supervision are only available for some of the data points. This approach suffers when new, unseen data points are added, since this would require re-learning the entire kernel matrix.

Though we do not consider this situation in depth in the current paper, we can circumvent it. When learning a kernel matrix minimizing either the LogDet divergence or the von Neumann divergence, the range space restriction implies that the learned kernel K has the form $K = G_0 B B^T G_0^T$, where B is $r \times r$ and the input kernel is $K_0 = G_0 G_0^T$. We can view the matrix B as a linear transformation applied to our input data vectors G_0 and therefore we can apply this linear transformation to new points. In particular, recent work (Davis et al., 2007) has shown that the learning algorithms considered in this paper can equivalently be viewed as learning a *Mahalanobis distance function* given constraints on the data, which is simply the Euclidean distance after applying a linear transformation over the input data.

Since our algorithms can be viewed as Mahalanobis distance learning techniques, it is natural to compare against other existing metric learning algorithms. Such methods include metric learning by collapsing classes (MCML) (Globerson and Roweis, 2005), large-margin nearest neighbor metric learning (LMNN) (Weinberger et al., 2005), and many others. In the experimental results section, we provide some results comparing our algorithms with these existing methods.

6.2 Generalizations

There are several simple ways to extend the algorithms developed in Section 5. In this section, we introduce slack variables and discuss more general constraints than the distance constraints discussed earlier.

6.2.1 SLACK VARIABLES

In many cases, especially if the number of constraints is large, no feasible solution will exist to the Bregman divergence optimization problem given in (4). When no feasible solution exists, a common approach is to incorporate *slack variables*, which allows constraints to be violated but penalizes such violations.

There are many ways to introduce slack variables into the optimization problem (4). We add a new vector variable $\boldsymbol{b}$ with coordinates representing perturbed right hand sides of the linear constraints, and use the corresponding vector divergence to measure the deviation from the original constraints described by the input vector $\boldsymbol{b}_0$. The resulting optimization problem is as follows:

$$\begin{aligned} \text{minimize}_{X,\boldsymbol{b}} \quad & D_\phi(X, X_0) + \gamma D_\varphi(\boldsymbol{b}, \boldsymbol{b}_0) \\ \text{subject to} \quad & \operatorname{tr}(XA_i) \leq \boldsymbol{e}_i^T \boldsymbol{b},\ 1 \leq i \leq c \\ & X \succeq 0. \end{aligned} \tag{22}$$

Note that we use $D_\varphi(\boldsymbol{b}, \boldsymbol{b}_0)$ as the penalty for constraints as it is computationally simple; other choices of constraint penalty are possible, as long as the resulting objective function is convex.

The $\gamma > 0$ parameter governs the tradeoff between satisfying the constraints and minimizing the divergence between X and X_0. Note that we have also removed the implicit rank constraint for simplicity.

We form the Lagrangian to solve the above problem; recall the similar system of equations (7) in Section 3.4. Define $\mathcal{L}(X_{t+1}, \boldsymbol{b}_{t+1}, \alpha_i) = D_\phi(X_{t+1}, X_t) + \gamma D_\varphi(\boldsymbol{b}_{t+1}, \boldsymbol{b}_t) + \alpha_i(\boldsymbol{e}_i^T \boldsymbol{b}_{t+1} - \operatorname{tr}(X_{t+1}A_i))$, and set the gradients to zero with respect to X_{t+1}, $\boldsymbol{b}_{t+1}$ and α_i to get the following update equations:

$$\begin{aligned} \nabla\phi(X_{t+1}) &= \nabla\phi(X_t) + \alpha_i A_i, \\ \nabla\varphi(\boldsymbol{b}_{t+1}) &= \nabla\varphi(\boldsymbol{b}_t) - \frac{\alpha_i}{\gamma}\boldsymbol{e}_i, \\ \operatorname{tr}(X_{t+1}A_i) &= \boldsymbol{e}_i^T \boldsymbol{b}_{t+1}. \end{aligned} \tag{23}$$

In particular, for the LogDet divergence we arrive at the following updates:

$$X_{t+1}^{-1} = X_t^{-1} - \alpha_i A_i, \quad \boldsymbol{e}_i^T \boldsymbol{b}_{t+1} = \frac{\gamma \boldsymbol{e}_i^T \boldsymbol{b}_t}{\gamma + \alpha_i \boldsymbol{e}_i^T \boldsymbol{b}_t}, \quad \operatorname{tr}(X_{t+1}A_i) - \boldsymbol{e}_i^T \boldsymbol{b}_{t+1} = 0.$$

Assuming $A_i = \boldsymbol{z}_i \boldsymbol{z}_i^T$, we can still compute α_i in closed form as

$$\alpha_i = \frac{\gamma}{\gamma+1}\left(\frac{1}{p} - \frac{1}{b_i}\right), \text{ where } p = \boldsymbol{z}_i^T X_t \boldsymbol{z}_i, b_i = \boldsymbol{e}_i^T \boldsymbol{b}_t.$$

The ith element of $\boldsymbol{b}_t$ is updated to $\gamma b_i/(\gamma + \alpha_i b_i)$ and the matrix update is calculated as in the non-slack case.

In case of the von Neumann divergence the update rules turn out to be:

$$\log X_{t+1} = \log X_t + \alpha_i A_i, \quad \boldsymbol{e}_i^T \boldsymbol{b}_{t+1} = \boldsymbol{e}_i^T \boldsymbol{b}_t e^{-\frac{\alpha_i}{\gamma}}, \quad \operatorname{tr}(X_{t+1}A_i) - \boldsymbol{e}_i^T \boldsymbol{b}_{t+1} = 0.$$

The projection parameter α_i is not available in closed form, instead we calculate it as the root of the non-linear equation:

$$\log \operatorname{tr}(\exp(\log X_t + \alpha_i A_i)A_i) + \frac{\alpha_i}{\gamma} - \log \boldsymbol{e}_i^T \boldsymbol{b}_t = 0.$$

The scale invariance of the LogDet divergence implies that $D_{\ell d}(\boldsymbol{b}, \boldsymbol{b_0}) = D_{\ell d}(\boldsymbol{b}/\boldsymbol{b_0}, \mathbf{1})$ where the division is elementwise and therefore we implicitly measure "relative" error, that is the magnitude of the coordinates of vector $\boldsymbol{b}_0$ are naturally taken into account. For the von Neumann divergence scale invariance does not hold, but one may alternatively use $D_{vN}(\boldsymbol{b}/\boldsymbol{b_0}, \mathbf{1})$ as the penalty function.

6.2.2 OTHER CONSTRAINTS

In earlier sections, we focused on the case of distance constraints. We now briefly discuss generalizing to other constraints.

When the constraints are similarity constraints (i.e., $K_{jl} \leq b_i$), the updates can be easily modified. As discussed earlier, we must retain symmetry of the A_i constraint matrices, so for similarity constraints, we require $A_i = \frac{1}{2}(\boldsymbol{e}_j \boldsymbol{e}_l^T + \boldsymbol{e}_l \boldsymbol{e}_j^T)$. Notice that the constraint matrix now has rank two. This complicates the algorithms slightly; for example, with the LogDet divergence, the Sherman-Morrison formula must be applied twice, leading to a more complicated update rule (albeit one that still has a closed-form solution and can be computed in $O(r^2)$ time).

Other constraints are possible as well; for example, one could incorporate distance constraints such as $\|\boldsymbol{a}_j - \boldsymbol{a}_k\|_2^2 \leq \|\boldsymbol{a}_l - \boldsymbol{a}_m\|_2^2$, or further similarity constraints such as $K_{jk} \leq K_{lm}$. These are sometimes referred to as relative comparison constraints, and are useful in ranking algorithms (Schultz and Joachims, 2003); for these also, the cost per projection remains $O(r^2)$. Finally, arbitrary linear constraints can also be applied, the cost per projection update will then be $O(nr)$.

6.3 Special Cases

If we use the von Neumann divergence, let $r = n$, and set $b_i = 0$ for all constraints, we exactly obtain the DefiniteBoost optimization problem from Tsuda et al. (2005). In this case, our algorithm computes the projection update in $O(n^2)$ time. In contrast, the algorithm from Tsuda et al. (2005) computes the projection in a more expensive manner in $O(n^3)$ time. Another difference with our approach is that we compute the exact projection, whereas Tsuda et al. (2005) compute an approximate projection. Though computing an approximate projection may lead to a faster per-iteration cost, it takes many more iterations to converge to the optimal solution. We illustrate this further in the experimental results section.

The online-PCA problem discussed in Warmuth and Kuzmin (2006) employs a similar update based on the von Neumann divergence. As with the DefiniteBoost algorithm, the projection is not an exact Bregman projection; however, the projection can be computed in the same way as in our von Neumann kernel learning projection. As a result, the cost of an iteration of online-PCA can be improved from $O(n^3)$ to $O(n^2)$ with our approach.

Another special case is the *nearest correlation matrix* problem (Higham, 2002) that arises in financial applications. A correlation matrix is a positive semidefinite matrix with unit diagonal. For this case, we set the constraints to be $K_{ii} = 1$ for all i. Our algorithms from this paper give new divergence measures and methods for finding low-rank correlation matrices. Previous algorithms scale cubically in n, whereas our method scales linearly with n and quadratically with r for low-rank correlation matrices.

Our formulation can also be employed to solve a problem similar to that of Weinberger et al. (2004). The enforced constraints on the kernel matrix (centering and isometry) are linear, and thus can be encoded into our framework. The only difference is that Weinberger et al. maximize the trace of K, whereas we minimize a matrix divergence. Comparisons between these approaches is a potential area of future research.

6.4 Connections to Semidefinite Programming

In this section, we present a connection between minimizing the LogDet divergence and the solution to a semidefinite programming problem (SDP). As an example, we consider the min-balanced-cut problem.

Suppose we are given an n-vertex graph, whose adjacency matrix is A. Let $L = \text{diag}(A\boldsymbol{e}) - A$ be the Laplacian of A, where $\boldsymbol{e}$ is the vector of all ones. The semidefinite relaxation to the minimum balanced cut problem (Lang, 2005) results in the following SDP:

$$\begin{array}{rl}\min_X & \mathrm{tr}(LX) \\ \text{subject to} & \mathrm{diag}(X) = \boldsymbol{e} \\ & \mathrm{tr}(X\boldsymbol{e}\boldsymbol{e}^T) = 0 \\ & X \succeq 0.\end{array}$$

Let $L^\dagger$ denote the pseudoinverse of the Laplacian, and let V be an orthonormal basis for the range space of L. Let r denote the rank of L; it is well known that $n-r$ equals the number of connected components of the graph. Consider the LogDet divergence between X and $\varepsilon L^\dagger$:

$$\begin{aligned}D_{\ell d}(X, \varepsilon L^\dagger) &= D_{\ell d}(V^TXV, V^T(\varepsilon L^\dagger)V) \\ &= \mathrm{tr}(V^TXV(V^T(\varepsilon L^\dagger)V)^{-1}) - \log\det(V^TXV(V^T(\varepsilon L^\dagger)V)^{-1}) - r \\ &= \mathrm{tr}\left(V^TXV\left(\frac{1}{\varepsilon}V^TLV\right)\right) - \log\det\left(V^TXV\left(\frac{1}{\varepsilon}V^TLV\right)\right) - r \\ &= \frac{1}{\varepsilon}\mathrm{tr}(LX) - \log\det\left(\frac{1}{\varepsilon}V^TXVV^TLV\right) - r \\ &= \frac{1}{\varepsilon}\mathrm{tr}(LX) - \log\det(V^TXVV^TLV) - r - \log(\varepsilon^{-r}).\end{aligned}$$

The fourth line uses Lemma 8 to replace $\mathrm{tr}(V^TXV(\frac{1}{\varepsilon}V^TLV))$ with $\frac{1}{\varepsilon}\mathrm{tr}(LX)$. If we aim to minimize this divergence, we can drop the last two terms, as they are constants. In this case, we have:

$$\begin{aligned}\mathrm{argmin}_X D_{\ell d}(X, \varepsilon L^\dagger) &= \mathrm{argmin}_X \frac{1}{\varepsilon}\mathrm{tr}(LX) - \log\det(V^TXVV^TLV) \\ &= \mathrm{argmin}_X\, \mathrm{tr}(LX) - \varepsilon\log\det(V^TXVV^TLV).\end{aligned}$$

As ε becomes small, the $\mathrm{tr}(LX)$ term dominates the objective function.

Now consider the constraints from the min balanced cut SDP. The $\mathrm{diag}(X) = \boldsymbol{e}$ constraint is exactly the constraint from the nearest correlation matrix problem (that is, $X_{ii} = 1$ for all i). The $X \succeq 0$ constraint is implicitly satisfied when LogDet is used, leaving the balancing constraint $\boldsymbol{e}^TX\boldsymbol{e} = 0$. Recall that $\mathrm{null}(X) = \mathrm{null}(L^\dagger)$, and from standard spectral graph theory, that $L\boldsymbol{e} = 0$. Therefore $L^\dagger\boldsymbol{e} = 0$, which further implies

$$\boldsymbol{e}^TL^\dagger\boldsymbol{e} = 0 \Rightarrow \boldsymbol{e}^TX\boldsymbol{e} = 0$$

by the fact that the LogDet divergence preserves the range space of $L^\dagger$. Thus, the null space restriction in LogDet divergence naturally yields the constraint $\boldsymbol{e}^TX\boldsymbol{e} = 0$ and so the min balanced cut SDP problem on A is equivalent (for sufficiently small ε) to finding the nearest correlation matrix to $\varepsilon L^\dagger$ under the LogDet divergence.

Many other semidefinite programming problems can be solved in a similar manner to the one given above for the min balanced cut problem. It is beyond the scope of this paper to determine the practicality of such an optimization method; our aim here is simply to demonstrate an intriguing relationship between semidefinite programming and minimizing the LogDet divergence. For further information on this relationship, see Kulis et al. (2007a).

6.5 Changing the Range Space

The key property of the LogDet and von Neumann divergences which allow the algorithms to learn low-rank kernel matrices is their range space preservation property, discussed in Section 4. While this property leads to efficient algorithms for learning low-rank kernel matrices, it also forces the learned matrices to maintain the range space of the input matrix, which is a limitation of our method.

Allowing the range space to change is potentially a very useful tool, but is still an open research question. One possibility worth exploring is to augment the input matrix X_0 with a matrix capturing the complementary space, that is, the input would be $X_0 + \varepsilon N$, where N captures the null space of X_0. Even if there does not exist a solution to the optimization problem of minimizing $D_\phi(X, X_0)$, there is always a solution to minimizing $D_\phi(X, X_0 + \varepsilon N)$. Details of this approach are to be pursued as future work.

A related approach to circumvent this problem is to apply an appropriate kernel function over the input data, the result of which is that the algorithm learns a non-linear transformation of the data over the input space. Recently, Davis et al. (2007) discussed how one can generalize to new data points with the LogDet divergence even after applying such a kernel function, making this approach practical in many situations.

7. Experiments

In this section, we present a number of results using our algorithms, and provide references to recent work which has applied our algorithms to large-scale learning tasks. We first begin with the basic algorithm, and present results on transductive learning—the scenario where all data is provided upfront but labels are available for only a subset of the data—and clustering. We then discuss some results comparing our methods to existing metric learning algorithms.

We run the kernel learning algorithms in MATLAB, with some routines written in C and compiled with the MEX compiler. An efficient implementation of the special matrix multiplication appearing in step 9 of Algorithm 3 could achieve further run-time improvements. Unfortunately, an efficient and accurate implementation—based on the fast multipole method—is not readily available.

7.1 Transductive Learning and Clustering Results

To show the effectiveness of our algorithms, we present results from clustering as well as classification. We consider several data sets from real-life applications:

1. `Digits:` a subset of the *Pendigits* data from the UCI repository that contains handwritten samples of the digits 3, 8, and 9. The raw data for each digit is 16-dimensional; this subset contains 317 digits and is a standard data set for semi-supervised clustering (e.g., see, Bilenko et al., 2004).

2. `GyrB:` a protein data set: a 52×52 kernel matrix among bacteria proteins, containing three bacteria species. This matrix is identical to the one used to test the DefiniteBoost algorithm in Tsuda et al. (2005).

3. `Spambase:` a data set from the UCI repository containing 4601 email messages, classified as spam or not spam—1813 of the emails are spam (39.4 percent). This data has 57 attributes.

4. `Nursery:` data set developed to rank applications for nursery schools. This set contains 12960 instances with 8 attributes, and 5 class labels.

For classification, we compute accuracy using a k-nearest neighbor classifier ($k = 5$) that computes distance in the feature space, with a 50/50 training/test split and two-fold cross validation. Our results are averaged over 20 runs. For clustering, we use the kernel k-means algorithm and compute accuracy using the Normalized Mutual Information (NMI) measure, a standard technique for determining quality of clusters, which measures the amount of statistical information shared by the random variables representing the cluster and class distributions (Strehl et al., 2000). If C is the random variable denoting the cluster assignments of the points, and K is the random variable denoting the underlying class labels on the points then the NMI measure is defined as:

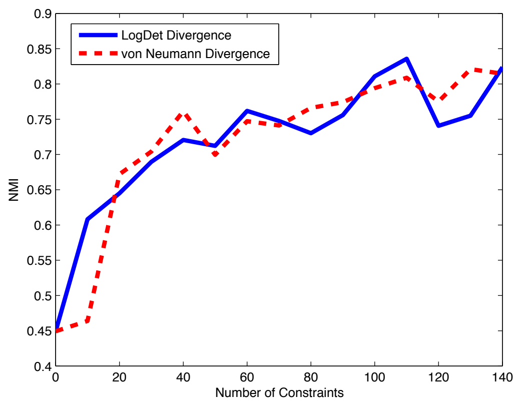
$$NMI = \frac{I(C;K)}{(H(C)+H(K))/2}$$

where $I(X;Y) = H(X) - H(X|Y)$ is the mutual information between the random variables X and Y, $H(X)$ is the Shannon entropy of X, and $H(X|Y)$ is the conditional entropy of X given Y (Cover and Thomas, 1991). The normalization by the average entropy of C and K makes the value of NMI stay between 0 and 1.

We consider two different experiments, each with their own constraint selection procedure. One experiment is to learn a kernel matrix *only using constraints*. In this case, we generate constraints from some target kernel matrix, and judge performance on the learned kernel matrix as we provide an increasing number of constraints. This setup was employed for the `GyrB` data set allowing us to compare to the methods and results of Tsuda et al. (2005). Constraints were generated randomly as follows: two data points are chosen at random and a distance constraint is constructed as follows. If the data points are in the same class, the constraint is of the form $d(i, j) \leq b$, where b is the corresponding distance between points i and j from the original kernel matrix, and if the data points are from different classes, then the constraint is of the form $d(i, j) \geq b$.

For the remaining data sets, we were interested in adding supervised class information to *improve* an existing low-rank kernel matrix. In this case, we took our initial low-rank kernel to be a linear kernel over the original data matrix and we added constraints of the form $d(i, j) \leq (1-\varepsilon)b$ for same class pairs and $d(i, j) \geq (1+\varepsilon)b$ for different class pairs (b is the original distance and $\varepsilon = .25$). This is very similar to "idealizing" the kernel, as in Kwok and Tsang (2003). Our convergence tolerance was set to 10^{-3}, and we incorporated slack variables for the larger data sets `Spambase` and `Nursery` (the smaller data sets did not require slack variables to converge). Note that there are other methods for choosing constraints; for example, instead of using $(1-\varepsilon)b$ for the right-hand side for same class pairs, one could instead choose a global value u for all same class constraints (and analogously for different class pairs).

We first show that the low-rank kernels learned by our algorithms attain good clustering and classification accuracies. We ran our algorithms on the `Digits` data set to learn a rank-16 kernel matrix using randomly generated constraints. The Gram matrix of the original `Digits` data was used as our initial (rank-16) kernel matrix. The left plot of Figure 1 shows clustering NMI values with increasing constraints. Adding just a few constraints improves the results significantly, and both of the kernel learning algorithms perform comparably. Classification accuracy using the k-nearest neighbor method was also computed for this data set: marginal classification accuracy gains were observed with the addition of constraints (an increase from 94 to 97 percent accuracy for



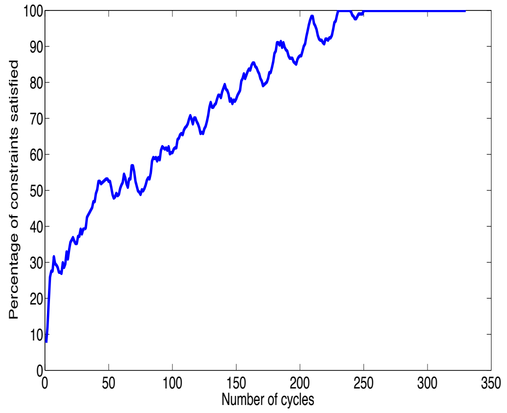
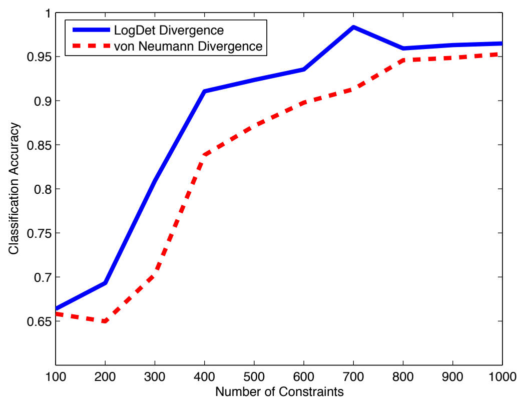


Figure 1: (Left) Normalized Mutual Information results of clustering the `Digits` data set—both algorithms improve with a small number of constraints (Right) Percentage of constraints satisfied (to a tolerance of 10^{-3}) in Bregman's algorithm as a function of the number of cycles, using the LogDet divergence



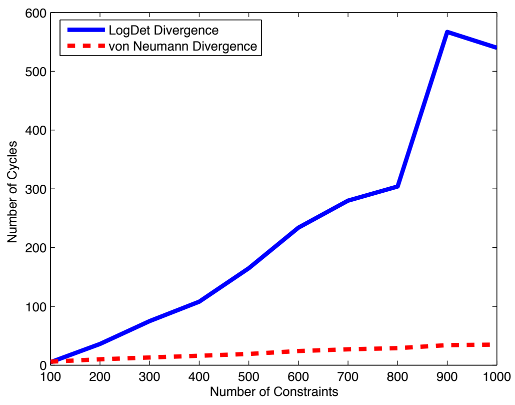
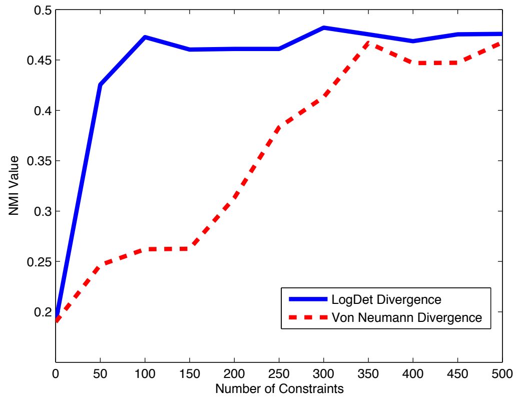


Figure 2: Classification accuracy (left) and convergence (right) on the `GyrB` data set

both divergences). We also recorded convergence data on `Digits` in terms of the number of cycles through all constraints; for the von Neumann divergence, convergence was attained in 11 cycles for 30 constraints and in 105 cycles for 420 constraints, while for the LogDet divergence, between 17 and 354 cycles were needed for convergence. This experiment highlights how our algorithm can use constraints to learn a low-rank kernel matrix. It is noteworthy that the learned kernel performs better than the original kernel for clustering as well as classification. Furthermore, in the right plot of Figure 1, we plot the number of constraints satisfied within a tolerance of 10^{-3} (300 total constraints) as a function of the number of cycles through all constraints, when using the LogDet divergence. Similar results are obtained with the von Neumann divergence. The number of cycles required for convergence to high accuracy can potentially be large, but for typical machine learning tasks, high accuracy solutions are not necessary and low to medium accuracy solutions can be achieved much faster.



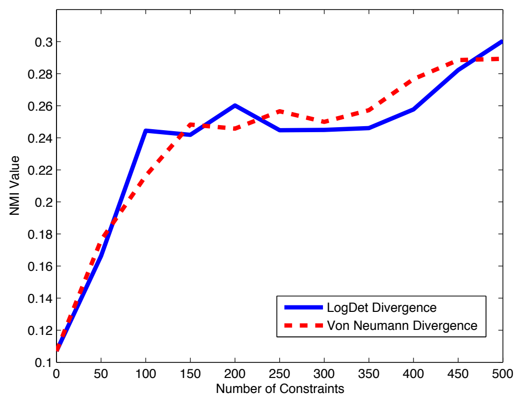


Figure 3: Clustering accuracy on `Spambase` (left) and `Nursery` (right)

As a second experiment, we performed experiments on `GyrB` and comparisons to the DefiniteBoost algorithm of Tsuda et al. (2005) (modified to correctly handle inequality constraints). Using only constraints, we attempt to learn a kernel matrix that achieves high classification accuracy. In order to compare to DefiniteBoost, we learned a full-rank kernel matrix starting from the scaled identity matrix as in Tsuda et al. (2005). In our experiments, we observed that using approximate projections—as done in DefiniteBoost—considerably increases the number of cycles needed for convergence. For example, starting with the scaled identity as the initial kernel matrix and 100 constraints, it took our von Neumann algorithm only 11 cycles to converge, whereas it took 3220 cycles for the DefiniteBoost algorithm to converge. Since the optimal solutions are the same for approximate versus exact projections, we converge to the same kernel matrix as DefiniteBoost but in far fewer iterations. Furthermore, our algorithm has the ability to learn low-rank kernels (as in the case of the `Digits` example). Figure 2 depicts the classification accuracy and convergence of our LogDet and von Neumann algorithms on this data set. The slow convergence of the DefiniteBoost algorithm did not allow us to run it with a larger set of constraints. For the LogDet and the von Neumann exact projection algorithms, the number of cycles required for convergence never exceeded 600 on runs of up to 1000 constraints on `GyrB`. The classification accuracy on the original matrix is .948, and so our learned kernels achieve even higher accuracy than the target kernel with a sufficient number of constraints. These results highlight that excellent classification accuracy can be obtained using a kernel that is learned using only distance constraints. Note that the starting kernel was the identity matrix, and so did not encode any domain information.

On the larger data sets `Spambase` and `Nursery`, we found similar improvements in clustering accuracy while learning rank-57 and rank-8 kernel matrices, respectively, using the same setup as in the `Digits` experiment. Figures 3 and 4 show the clustering accuracy and convergence on these data sets. Note that both data sets are too large for DefiniteBoost to handle, and furthermore, storing the full kernel matrix would require significant memory overhead (for example, MATLAB runs out of memory while attempting to store the full kernel matrix for `Nursery`); this scenario highlights the memory overhead advantage of using low-rank kernels. We see that on the `Spambase` data set, the LogDet divergence algorithm produces clustering accuracy improvements with very little supervision, but the number of cycles to converge is much larger than the von Neumann algorithm (for 500 constraints, the von Neumann algorithm requires only 29 cycles to converge). The slower

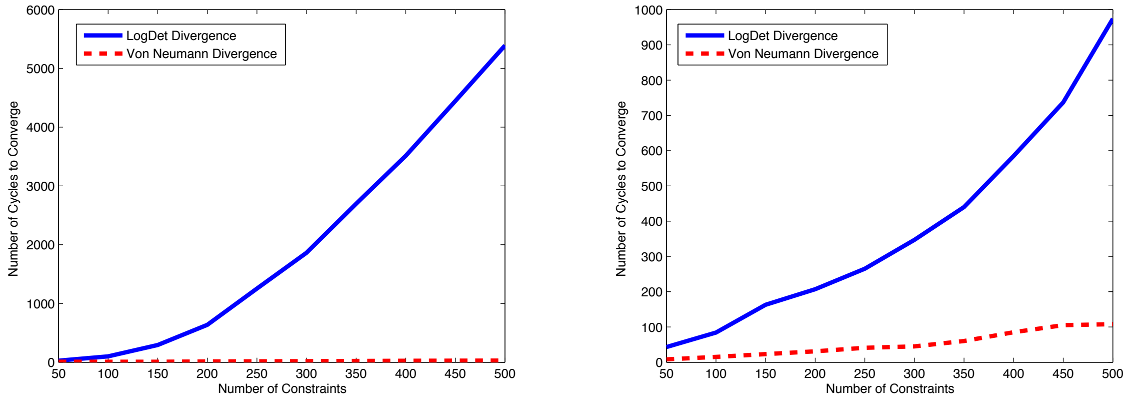


Figure 4: Convergence on `Spambase` (left) and `Nursery` (right)

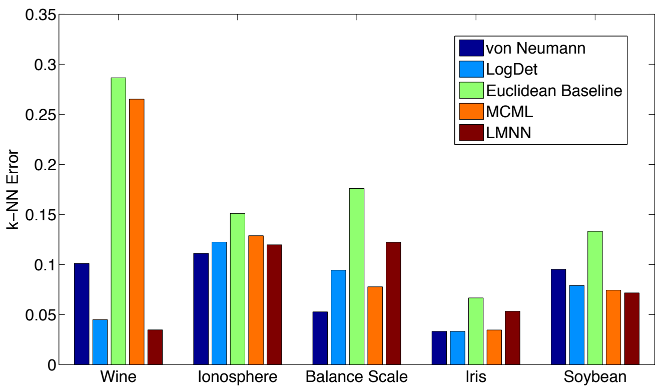


Figure 5: Classification error rates for *k*-nearest neighbor classification via different learned metrics over UCI data sets. The von Neumann and LogDet algorithms are competitive with existing state of the art metric learning methods, and significantly outperform the baseline.

convergence of the LogDet divergence algorithm is a topic of future work—note, however, that even though the number of cycles for LogDet to converge is higher than for von Neumann, the overall running time is often lower due to the efficiency of each LogDet iteration and the lack of a suitable implementation of the fast multiple method. On the `Nursery` data set, both algorithms perform similarly in terms of clustering accuracy, and the discrepancy in terms of cycles to converge between the algorithms is less drastic. We stress that other methods for learning kernel matrices, such as using the squared Frobenius norm in place of the LogDet or von Neumann divergences, would lead to learning full-rank matrices and would require significantly more memory and computational resources.

7.2 Metric Learning and Large-Scale Experiments

We now move beyond the transductive setting in order to compare with existing metric learning algorithms. As briefly discussed in Section 6.1, learning a low-rank kernel with the same range space as the input kernel is equivalent to learning a linear transformation of the input data, so we now compare against some existing methods for learning linear transformations. In particular, we compare against two popular Mahalanobis metric learning algorithms: metric learning by collapsing classes (MCML) (Globerson and Roweis, 2005) and large-margin nearest neighbor metric learning (LMNN) (Weinberger et al., 2005). As a baseline, we consider the squared Euclidean distance.

For each data set, we use two-fold cross validation to learn a linear transformation of our training data. This transformation is used in a k-nearest neighbor classifier on our test data, and the resulting error is reported. For the LogDet and von Neumann algorithms, we cross-validate the γ parameter and use the constraint selection procedure described in Davis et al. (2007). In particular, we generate constraints of the form $d(i,j) \leq u$ for same-class pairs and $d(i,j) \geq \ell$ for different-class pairs, where u and ℓ are chosen based on the 5th and 95th percentile of all distances in the training data. We randomly choose $40c^2$ constraints, where c is the number of classes in the data.

We ran on a number of standard UCI data sets to compare with existing methods. In Figure 5, we display k-NN accuracy ($k = 4$) over five data sets. Overall, we see that both the LogDet and von Neumann algorithms compare favorably with existing methods.

In terms of large-scale experiments, we also ran on the MNIST data set, a handwritten digits data set. This data has 60,000 training points and 10,000 test points. After deskewing the images and performing dimensionality reduction to the first 100 principal components of MNIST, we ran our methods successfully with 10,000 and 100,000 constraints, chosen as in the metric learning experiments above. The baseline error for a 1-nearest neighbor search over the top 100 principal components after deskewing is 2.35%. For 10,000 constraints, the LogDet algorithm ran in 4.8 minutes and achieved a test error of 2.29%, while the von Neumann algorithm ran in 7.7 minutes and achieved 2.30% error. For 100,000 constraints, LogDet ran in 11.3 minutes and achieved an error of 2.18% while von Neumann ran in 71.4 minutes and achieved an error of 2.17%.

Finally, we note that our methods have recently been applied to very large problems in the computer vision domain. We refer the reader to Jain et al. (2008), which adapts the algorithms discussed in this paper to three large-scale vision experiments: human body pose estimation, feature indexing for 3-d reconstruction, and object classification. For pose estimation, the size of the data was 500,000 images and for feature indexing, there were 300,000 image patches. These experiments validate the use of our methods on large-scale data, and also demonstrate that they can be used to outperform state of the art methods in computer vision.

8. Conclusions

In this paper, we have developed algorithms for using Bregman matrix divergences for low-rank matrix nearness problems. In particular, we have developed a framework for using the LogDet divergence and the von Neumann divergence when the initial matrices are low-rank; this is achieved via a restriction of the range space of the matrices. Unlike previous kernel learning algorithms, which have running times that are cubic in the number of data points, our resulting algorithms are efficient—both algorithms have running times linear in the number of data points and quadratic in the rank of the kernel. Furthermore, our algorithms can be used in conjunction with a number of kernel-based learning algorithms that are optimized for low-rank kernel representations. The experimental results demonstrate that our algorithms are effective in learning low-rank and full-rank kernels for classification and clustering problems on large-scale data sets that arise in diverse applications.

There is still much to be gained from studying these divergences. Algorithmically, we have considered only projections onto a single constraint, but it is worth pursuing other approaches to solve the optimization problems such as methods that optimize with respect to multiple constraints simultaneously. We discussed how the LogDet divergence exhibits connections to Mahalanobis metric learning and semidefinite programming, and there is significant ongoing work in these areas. Finally, we hope that our framework will lead to new insights and algorithms for other machine learning problems where matrix nearness problems need to be solved.

Acknowledgments

This research was supported by NSF grant CCF-0431257, NSF Career Award ACI-0093404, and NSF-ITR award IIS-0325116. We thank Koji Tsuda for the protein data set and Kilian Weinberger for the image deskewing code.

Appendix A. Properties of Bregman Matrix Divergences

In this section, we highlight some important properties of Bregman matrix divergences.

Proposition 11 *Let Q be a square, orthogonal matrix, that is, $Q^TQ = QQ^T = I$. Then for all spectral Bregman matrix divergences, $D_\phi(Q^TXQ, Q^TYQ) = D_\phi(X,Y)$.*

Proof This follows from Lemma 1 by observing that if $X = V\Lambda V^T$ and $Y = U\Theta U^T$, the ith eigenvector of Q^TXQ is $Q^T\boldsymbol{v}_i$ and the jth eigenvector of Q^TYQ is $Q^T\boldsymbol{u}_j$. The eigenvalues remain unchanged by the orthogonal similarity transformation. Thus, the term $((Q^T\boldsymbol{v}_i)^T(Q^T\boldsymbol{u}_j))^2$ in Lemma 1 simplifies to $(\boldsymbol{v}_i^T\boldsymbol{u}_j)^2$ using the fact that $QQ^T = I$. ■

Proposition 12 *Let M be a square, non-singular matrix, and let X and Y be $n\times n$ positive definite matrices. Then $D_{\ell d}(M^TXM, M^TYM) = D_{\ell d}(X,Y)$.*

Proof First observe that if X is positive definite, then M^TXM is positive definite. Then:

$$\begin{aligned}
D_{\ell d}(M^TXM, M^TYM) &= \operatorname{tr}(M^TXM(M^TYM)^{-1}) - \log\det(M^TXM(M^TYM)^{-1}) - n \\
&= \operatorname{tr}(M^TXMM^{-1}Y^{-1}M^{-T}) - \log\det(M^TXMM^{-1}Y^{-1}M^{-T}) - n \\
&= \operatorname{tr}(XY^{-1}) - \log\det(XY^{-1}) - n \\
&= D_{\ell d}(X,Y)
\end{aligned}$$

■

Note that, as a corollary, we have that the LogDet divergence is *scale-invariant*, that is, $D_{\ell d}(X,Y) = D_{\ell d}(cX, cY)$, for any positive scalar c. Furthermore, this proposition may be extended to the case when X and Y are positive semidefinite, using the definition of the LogDet divergence for rank-deficient matrices.

Proposition 13 *Let Y be a positive definite matrix, and let X be any positive semidefinite matrix. Then $D_{\ell d}(X,Y) = D_{vN}(I, Y^{-\frac{1}{2}}XY^{-\frac{1}{2}}) = D_{\ell d}(Y^{-\frac{1}{2}}XY^{-\frac{1}{2}}, I)$*

Proof The first equality by expanding the definition of the von Neumann divergence:

$$\begin{aligned} D_{vN}(I, Y^{-\frac{1}{2}}XY^{-\frac{1}{2}}) &= \mathrm{tr}(I \log I - I \log(Y^{-\frac{1}{2}}XY^{-\frac{1}{2}}) - I + Y^{-\frac{1}{2}}XY^{-\frac{1}{2}}) \\ &= \mathrm{tr}(Y^{-\frac{1}{2}}XY^{-\frac{1}{2}}) - \mathrm{tr}(\log(Y^{-\frac{1}{2}}XY^{-\frac{1}{2}})) - \mathrm{tr}(I) \\ &= \mathrm{tr}(XY^{-1}) - \log\det(Y^{-\frac{1}{2}}XY^{-\frac{1}{2}}) - n \\ &= \mathrm{tr}(XY^{-1}) - \log\det(XY^{-1}) - n \\ &= D_{\ell d}(X,Y), \end{aligned}$$

where we have used the fact that $\mathrm{tr}(\log A) = \log\det(A)$. The second equality follows by applying Proposition 12 to $D_{\ell d}(X,Y)$ with $M = Y^{-\frac{1}{2}}$. ∎

Appendix B. Necessity of Dual Variable Corrections

In Section 3.4, we presented the method of Bregman projections and showed that, in the presence of inequality constraints, a dual variable correction is needed to guarantee convergence to the globally optimal solution. In some recent machine learning papers such as Tsuda et al. (2005), this correction has been omitted from the algorithm. We now briefly demonstrate why such corrections are needed.

A very simple example illustrating the failure of Bregman projections without corrections is in finding of the nearest 2×2 (positive definite) matrix X to the identity matrix that satisfies a single linear constraint:

$$\begin{aligned} \text{minimize}_X \quad & D_\phi(X, I) \\ \text{subject to} \quad & X_{11} + X_{22} - 2X_{12} \geq 1 \\ & X \succeq 0. \end{aligned}$$

The starting (identity) matrix satisfies the linear constraint, but a single projection step (to the corresponding equality constraint) without corrections will produce a suboptimal matrix, regardless of the divergence used. On the other hand, employing corrections leads to $\alpha_i' = 0$, and the input matrix remains unchanged.

It is also not sufficient to repeatedly perform Bregman projections only on violated constraints (without any corrections) until all constraints are satisfied—this approach is used by Tsuda et al. (2005). To demonstrate this more involved case, we consider an example over vectors, where the goal is to minimize the relative entropy to a given vector $\boldsymbol{x}_0$ under linear constraints. Note that the argument carries over to the matrix case, but is more difficult to visualize.

$$\begin{aligned} \text{minimize}_{\boldsymbol{x}} \quad & KL(\boldsymbol{x}, \boldsymbol{x}_0) \\ \text{subject to} \quad & \boldsymbol{x}^T \begin{bmatrix} 0.0912 & 0.9385 & -0.4377 \end{bmatrix} \geq 0.0238 \\ & \boldsymbol{x}^T \begin{bmatrix} 0.6020 & 0.6020 & -0.4377 \end{bmatrix} \geq 0.2554 \\ & \boldsymbol{x}^T \begin{bmatrix} 1 & 1 & 1 \end{bmatrix} = 1. \end{aligned} \tag{24}$$

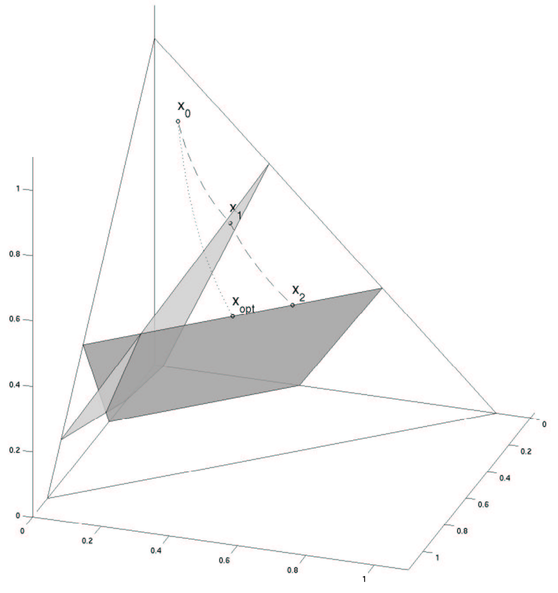


Figure 6: Termination points for the relative entropy example described by (24). The shaded regions illustrate the hyperplanes associated with the two inequality constraints of the optimization problem. The optimal solution is the maximum entropy vector $x_{opt} = \left[\frac{1}{3}\ \frac{1}{3}\ \frac{1}{3}\right]$, but without corrections we arrive at $x_2 = \left[\frac{1}{5}\ \frac{7}{15}\ \frac{1}{3}\right]$ after two projection steps starting from $x_0 = [0.1\ 0.1\ 0.8]$.

In the above problem, $KL(\boldsymbol{x}, \boldsymbol{x}_0)$ refers to the relative entropy, defined earlier. The first two constraints are linear inequality constraints while the third constraint forces $\boldsymbol{x}$ to remain on the unit simplex so that it is a probability vector. Let $\boldsymbol{x}_0$ equal $[0.1\ 0.1\ 0.8]$. If we process the constraints in the above order without corrections, after two projection steps (one projection onto each of the first two constraints), we arrive at the vector

$$\begin{bmatrix} \frac{1}{5} & \frac{7}{15} & \frac{1}{3} \end{bmatrix},$$

which satisfies all three constraints. Therefore this vector is returned as the optimal solution if no corrections are applied. In comparison, a proper execution of Bregman's algorithm employing corrections arrives at

$$\begin{bmatrix} \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \end{bmatrix},$$

which is the maximum entropy vector. It is also worth noting that if we project to the second constraint first, then we arrive at the optimal solution immediately. By appropriately modifying the constraints and the starting point in this example, the incorrect algorithm can be made to converge arbitrarily close to $[0\ 0\ 1]$, a minimum entropy vector, thus drastically demonstrating the necessity of dual variable corrections.

References

F. Bach and M. Jordan. Predictive low-rank decomposition for kernel methods. In *Proc. 22nd International Conference on Machine Learning (ICML)*, 2005.

J. Barnes and P. Hut. A hierarchical $O(n \log n)$ force calculation algorithm. *Nature*, 324:446–449, 1986.

M. Bilenko, S. Basu, and R. Mooney. Integrating constraints and metric learning in semi-supervised clustering. In *Proceedings of the 21st International Conference on Machine Learning*, 2004.

L. Bregman. The relaxation method of finding the common point of convex sets and its application to the solution of problems in convex programming. *USSR Comp. Mathematics and Mathematical Physics*, 7:200–217, 1967.

R.P. Brent. *Algorithms for Minimization without Derivatives*. Prentice-Hall, Englewood Cliffs, NJ., 1973. ISBN 0-13-022335-2.

Y. Censor and S. Zenios. *Parallel Optimization*. Oxford University Press, 1997.

T. M. Cover and J. A. Thomas. *Elements of Information Theory*. Wiley-Interscience, 1991.

N. Cristianini, J. Shawe-Taylor, A. Elisseeff, and J. Kandola. On kernel-target alignment. In *Advances in Neural Information Processing Systems (NIPS) 14*, 2002.

J. Davis, B. Kulis, S. Sra, and I. S. Dhillon. Information-theoretic metric learning. In *Proc. 24th International Conference on Machine Learning (ICML)*, 2007.

J. D. Demmel. *Applied Numerical Linear Algebra*. Society for Industrial and Applied Mathematics, 1997.

I. S. Dhillon and J. A. Tropp. Matrix nearness problems with Bregman divergences. *SIAM Journal on Matrix Analysis and Applications*, 29(4):1120–1146, 2007.

I. S. Dhillon, Y. Guan, and B. Kulis. Kernel k-means, spectral clustering and normalized cuts. In *Proc. 10th ACM SIGKDD Conference*, 2004.

S. Fine and K. Scheinberg. Efficient SVM training using low-rank kernel representations. *Journal of Machine Learning Research*, 2:243–264, 2001.

R. Fletcher. A new variational result for quasi-Newton formulae. *SIAM Journal on Optimziation*, 1 (1), February 1991.

A. Globerson and S. Roweis. Metric learning by collapsing classes. In *Advances in Neural Information Processing Systems (NIPS) 18*, 2005.

G. Golub. Some modified matrix eigenvalue problems. *SIAM Review*, 15:318–334, 1973.

G. H. Golub and C. F. Van Loan. *Matrix Computations*. Johns Hopkins University Press, 1996.

L. Greengard and V. Rokhlin. A fast algorithm for particle simulations. *J. Comput. Phys.*, 73: 325–348, 1987.

M. Gu and S. Eisenstat. A stable and efficient algorithm for the rank-one modification of the symmetric eigenproblem. *SIAM J. Matrix Anal. Appl.*, 15:1266–1276, 1994.

N. Higham. Computing the nearest correlation matrix—a problem from finance. *IMA J. Numerical Analysis*, 22(3):329–343, 2002.

P. Jain, B. Kulis, and K. Grauman. Fast image search for learned metrics. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2008.

W. James and C. Stein. Estimation with quadratic loss. In *Proc. Fourth Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, pages 361–379. Univ. of California Press, 1961.

B. Kulis, M. A. Sustik, and I. S. Dhillon. Learning low-rank kernel matrices. In *Proc. 23rd International Conference on Machine Learning (ICML)*, 2006.

B. Kulis, S. Sra, S. Jegelka, and I. S. Dhillon. Scalable semidefinite programming using convex perturbations. Technical Report TR-07-47, University of Texas at Austin, September 2007a.

B. Kulis, A. Surendran, and J. Platt. Fast low-rank semidefinite programming for embedding and clustering. In *Proc. 11th International Conference on AI and Statistics (AISTATS)*, 2007b.

J. Kwok and I. Tsang. Learning with idealized kernels. In *Proc. 20th International Conference on Machine Learning (ICML)*, 2003.

G. Lanckriet, N. Cristianini, P. Bartlett, L. E. Ghaoui, and M. Jordan. Learning the kernel matrix with semidefinite programming. *Journal of Machine Learning Research*, 5:27–72, 2004.

K. Lang. Fixing two weaknesses of the spectral method. In *Advances in Neural Information Processing Systems (NIPS) 18*, 2005.

M.A. Nielsen and I.L. Chuang. *Quantum Computation and Quantum Information*. Cambridge University Press, 2000.

M. Schultz and T. Joachims. Learning a distance metric from relative comparisons. In *Advances in Neural Information Processing Systems (NIPS) 16*, 2003.

J. Shawe-Taylor and N. Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, 2004.

J. Sherman and W. J. Morrison. Adjustment of an inverse matrix corresponding to changes in the elements of a given column or a given row of the original matrix. *Annals of Mathematical Statistics*, 20, 1949.

A. Strehl, J. Ghosh, and R. Mooney. Impact of similarity measures on web-page clustering. In *Workshop on Artificial Intelligence for Web Search (AAAI)*, 2000.

M. A. Sustik and I. S. Dhillon. On some modified root-finding problems. Working manuscript, 2008.

L. Torresani and K. Lee. Large margin component analysis. In *Advances in Neural Information Processing Systems (NIPS) 19*, 2006.

K. Tsuda, G. Rátsch, and M. Warmuth. Matrix exponentiated gradient updates for online learning and Bregman projection. *Journal of Machine Learning Research*, 6:995–1018, 2005.

M. K. Warmuth and D. Kuzmin. Randomized PCA algorithms with regret bounds that are logarithmic in the dimension. In *Advances in Neural Information Processing Systems (NIPS) 20*, 2006.

K. Weinberger, F. Sha, and L. Saul. Learning a kernel matrix for nonlinear dimensionality reduction. In *Proc. 21st International Conference on Machine Learning (ICML)*, 2004.

K. Weinberger, J. Blitzer, and L. Saul. Distance metric learning for large margin nearest neighbor classification. In *Advances in Neural Information Processing Systems (NIPS) 18*, 2005.

K. Weinberger, F. Sha, Q. Zhu, and L. Saul. Graph Laplacian methods for large-scale semidefinite programming, with an application to sensor localization. In *Advances in Neural Information Processing Systems (NIPS) 19*, 2006.

Journal of Machine Learning Research 10 (2009) 377-403 Submitted 1/08; Revised 8/08; Published 2/09

Supervised Descriptive Rule Discovery: A Unifying Survey of Contrast Set, Emerging Pattern and Subgroup Mining

Petra Kralj Novak PETRA.KRALJ.NOVAK@IJS.SI
Nada Lavrač[*] NADA.LAVRAC@IJS.SI
Department of Knowledge Technologies
Jožef Stefan Institute
Jamova 39, 1000 Ljubljana, Slovenia

Geoffrey I. Webb GEOFF.WEBB@INFOTECH.MONASH.EDU.AU
Faculty of Information Technology
Monash University
Building 63, Clayton Campus, Wellington Road, Clayton
VIC 3800, Australia

Editor: Stefan Wrobel

Abstract

This paper gives a survey of contrast set mining (CSM), emerging pattern mining (EPM), and subgroup discovery (SD) in a unifying framework named *supervised descriptive rule discovery*. While all these research areas aim at discovering patterns in the form of rules induced from labeled data, they use different terminology and task definitions, claim to have different goals, claim to use different rule learning heuristics, and use different means for selecting subsets of induced patterns. This paper contributes a novel understanding of these subareas of data mining by presenting a unified terminology, by explaining the apparent differences between the learning tasks as variants of a unique supervised descriptive rule discovery task and by exploring the apparent differences between the approaches. It also shows that various rule learning heuristics used in CSM, EPM and SD algorithms all aim at optimizing a trade off between rule coverage and precision. The commonalities (and differences) between the approaches are showcased on a selection of best known variants of CSM, EPM and SD algorithms. The paper also provides a critical survey of existing supervised descriptive rule discovery visualization methods.

Keywords: descriptive rules, rule learning, contrast set mining, emerging patterns, subgroup discovery

1. Introduction

Symbolic data analysis techniques aim at discovering comprehensible patterns or models in data. They can be divided into techniques for *predictive induction*, where models, typically induced from class labeled data, are used to predict the class value of previously unseen examples, and *descriptive induction*, where the aim is to find comprehensible patterns, typically induced from unlabeled data. Until recently, these techniques have been investigated by two different research communities: predictive induction mainly by the machine learning community, and descriptive induction mainly by the data mining community.

*. Also at University of Nova Gorica, Vipavska 13, 5000 Nova Gorica, Slovenia.

Data mining tasks where the goal is to find humanly interpretable differences between groups have been addressed by both communities independently. The groups can be interpreted as class labels, so the data mining community, using the association rule learning perspective, adapted association rule learners like Apriori by Agrawal et al. (1996) to perform a task named *contrast set mining* (Bay and Pazzani, 2001) and *emerging pattern mining* (Dong and Li, 1999). On the other hand, the machine learning community, which usually deals with class labeled data, was challenged by, instead of building sets of classification/prediction rules (e.g., Clark and Niblett, 1989; Cohen, 1995), to build individual rules for exploratory data analysis and interpretation, which is the goal of the task named *subgroup discovery* (Wrobel, 1997).

This paper gives a survey of contrast set mining (CSM), emerging pattern mining (EPM), and subgroup discovery (SD) in a unifying framework, named *supervised descriptive rule discovery*. Typical applications of supervised descriptive rule discovery include patient risk group detection in medicine, bioinformatics applications like finding sets of overexpressed genes for specific treatments in microarray data analysis, and identifying distinguishing features of different customer segments in customer relationship management. The main aim of these applications is to understand the underlying phenomena and not to classify new instances. Take another illustrative example, where a manufacturer wants to know in what circumstances his machines may break down; his intention is not to predict breakdowns, but to understand the factors that lead to them and how to avoid them.

The main contributions of this paper are as follows. It provides a survey of supervised descriptive rule discovery approaches addressed in different communities, and proposes a unifying supervised descriptive rule discovery framework, including a critical survey of visualization methods. The paper is organized as follows: Section 2 gives a survey of past research done in the main supervised descriptive rule discovery areas: contrast set mining, emerging pattern mining, subgroup discovery and other related approaches. Section 3 is dedicated to unifying the terminology, definitions and the heuristics. Section 4 addresses visualization as an important open issue in supervised descriptive rule discovery. Section 5 provides a short summary.

2. A Survey of Supervised Descriptive Rule Discovery Approaches

Research on finding interesting rules from class labeled data evolved independently in three distinct areas—contrast set mining, mining of emerging patterns and subgroup discovery—each area using different frameworks and terminology. In this section, we provide a survey of these three research areas. We also discuss other related approaches.

2.1 An Illustrative Example

Let us illustrate contrast set mining, emerging pattern mining and subgroup discovery using data from Table 1, a very small, artificial sample data set,[1] adapted from Quinlan (1986). The data set contains the results of a survey on 14 individuals, concerning the approval or disapproval of an issue analyzed in the survey. Each individual is characterized by four attributes—`Education` (with values `primary` school, `secondary` school, or `university`), `MaritalStatus` (`single`, `married`, or `divorced`), `Sex` (`male` or `female`), and `HasChildren` (`yes` or `no`)—that encode rudimentary information about the sociodemographic background. The last column `Approved` is the designated

1. Thanks to Johannes Fürnkranz for providing this data set.

Education	Marital Status	Sex	Has Children	Approved
primary	single	male	no	no
primary	single	male	yes	no
primary	married	male	no	yes
university	divorced	female	no	yes
university	married	female	yes	yes
secondary	single	male	no	no
university	single	female	no	yes
secondary	divorced	female	no	yes
secondary	single	female	yes	yes
secondary	married	male	yes	yes
primary	married	female	no	yes
secondary	divorced	male	yes	no
university	divorced	female	yes	no
secondary	divorced	male	no	yes

Table 1: A sample database.

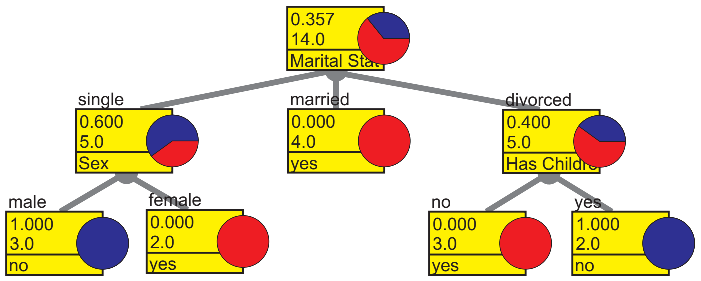


Figure 1: A decision tree, modeling the data set shown in Table 1.

class attribute, encoding whether the individual approved or disapproved the issue. Since there is no need for expert knowledge to interpret the results, this data set is appropriate for illustrating the results of supervised descriptive rule discovery algorithms, whose task is to find interesting patterns describing individuals that are likely to approve or disapprove the issue, based on the four demographic characteristics.

The task of *predictive induction* is to induce, from a given set of *training examples*, a domain model aimed at predictive or classification purposes, such as the *decision tree* shown in Figure 1, or a *rule set* shown in Figure 2, as learned by C4.5 and C4.5rules (Quinlan, 1993), respectively, from the sample data in Table 1.

```
Sex = female  →  Approved = yes
MaritalStatus = single  AND  Sex = male  →  Approved = no
MaritalStatus = married  →  Approved = yes
MaritalStatus = divorced  AND  HasChildren = yes  →  Approved = no
MaritalStatus = divorced  AND  HasChildren = no  →  Approved = yes
```

Figure 2: A set of predictive rules, modeling the data set shown in Table 1.

```
MaritalStatus = single AND Sex = male  →  Approved = no
Sex = male  →  Approved = no
Sex = female  →  Approved = yes
MaritalStatus = married  →  Approved = yes
MaritalStatus = divorced AND HasChildren = yes  →  Approved = no
MaritalStatus = single  →  Approved = no
```

Figure 3: Selected descriptive rules, describing individual patterns in the data of Table 1.

In contrast to predictive induction algorithms, *descriptive induction* algorithms typically result in rules induced from unlabeled examples. E.g., given the examples listed in Table 1, these algorithms would typically treat the class Approved no differently from any other attribute. Note, however, that in the learning framework discussed in this paper, that is, in the framework of *supervised descriptive rule discovery*, the discovered rules of the form $X \rightarrow Y$ are induced from class labeled data: the class labels are taken into account in learning of patterns of interest, constraining Y at the right hand side of the rule to assign a value to the class attribute.

Figure 3 shows six descriptive rules, found for the sample data using the Magnum Opus (Webb, 1995) software. Note that these rules were found using the default settings except that the critical value for the statistical test was relaxed to 0.25. These descriptive rules differ from the predictive rules in several ways. The first rule is redundant with respect to the second. The first is included as a strong pattern (*all* 3 single males do not approve) whereas the second is weaker but more general (4 out of 7 males do not approve, which is not highly predictive, but accounts for 4 out of all 5 respondents who do not approve). Most predictive systems will include only one of these rules, but either may be of interest to someone trying to understand the data, depending upon the specific application. This particular approach to descriptive pattern discovery does not attempt to second guess which of the more specific or more general patterns will be the more useful.

Another difference between the predictive and the descriptive rule sets is that the descriptive rule set does not include the pattern that divorcees without children approve. This is because, while the pattern is highly predictive in the sample data, there are insufficient examples to pass the statistical test which assesses the probability that, given the frequency of respondents approving, the apparent correlation occurs by chance. The predictive approach often includes such rules for the sake of completeness, while some descriptive approaches make no attempt at such completeness, assessing each pattern on its individual merits.

Exactly which rules will be induced by a supervised descriptive rule discovery algorithm depends on the task definition, the selected algorithm, as well as the user-defined constraints concerning minimal rule support, precision, etc. In the following section, the example set of Table 1 is used to illustrate the outputs of emerging pattern and subgroup discovery algorithms (see Figures 4 and 5, respectively), while a sample output for contrast set mining is shown in Figure 3 above.

2.2 Contrast Set Mining

The problem of mining contrast sets was first defined by Bay and Pazzani (2001) as finding contrast sets as "conjunctions of attributes and values that differ meaningfully in their distributions across groups." The example rules in Figure 3 illustrate this approach, including all conjunctions of attributes and values that pass a statistical test for productivity (explained below) with respect to attribute Approved that defines the 'groups.'

2.2.1 CONTRAST SET MINING ALGORITHMS

The STUCCO algorithm (Search and Testing for Understandable Consistent Contrasts) by Bay and Pazzani (2001) is based on the Max-Miner rule discovery algorithm (Bayardo, 1998). STUCCO discovers a set of contrast sets along with their supports[2] on groups. STUCCO employs a number of pruning mechanisms. A potential contrast set X is discarded if it fails a statistical test for independence with respect to the group variable Y. It is also subjected to what Webb (2007) calls a test for *productivity*. Rule $X \rightarrow Y$ is productive iff

$$\forall Z \subset X : \textit{confidence}(Z \rightarrow Y) < \textit{confidence}(X \rightarrow Y)$$

where $\textit{confidence}(X \rightarrow Y)$ is a maximum likelihood estimate of conditional probability $P(Y|X)$, estimated by the ratio $\frac{\textit{count}(X,Y)}{\textit{count}(X)}$, where $\textit{count}(X,Y)$ represents the number of examples for which both X and Y are true, and $\textit{count}(X)$ represents the number of examples for which X is true. Therefore a more specific contrast set must have higher confidence than any of its generalizations. Further tests for minimum counts and effect sizes may also be imposed.

STUCCO introduced a novel variant of the Bonferroni correction for multiple tests which applies ever more stringent critical values to the statistical tests employed as the number of conditions in a contrast set is increased. In comparison, the other techniques discussed below do not, by default, employ any form of correction for multiple comparisons, as result of which they have high risk of making *false discoveries* (Webb, 2007).

It was shown by Webb et al. (2003) that contrast set mining is a special case of the more general rule learning task. A contrast set can be interpreted as the antecedent of rule $X \rightarrow Y$, and group G_i for which it is characteristic—in contrast with group G_j—as the rule consequent, leading to rules of the form $\textit{ContrastSet} \rightarrow G_i$. A standard descriptive rule discovery algorithm, such as an association-rule discovery system (Agrawal et al., 1996), can be used for the task if the consequent is restricted to a variable whose values denote group membership.

In particular, Webb et al. (2003) showed that when STUCCO and the general-purpose descriptive rule learning system Magnum Opus were each run with their default settings, but the consequent restricted to the contrast variable in the case of Magnum Opus, the contrasts found differed mainly as a consequence only of differences in the statistical tests employed to screen the rules.

Hilderman and Peckham (2005) proposed a different approach to contrast set mining called CIGAR (ContrastIng Grouped Association Rules). CIGAR uses different statistical tests to STUCCO or Magnum Opus for both independence and productivity and introduces a test for *minimum support*.

Wong and Tseng (2005) have developed techniques for discovering contrasts that can include negations of terms in the contrast set.

In general, contrast set mining approaches require discrete data, which is in real world applications frequently not the case. A data discretization method developed specifically for set mining purposes is described by Bay (2000). This approach does not appear to have been further used by the contrast set mining community, except for Lin and Keogh (2006), who extended contrast set mining to time series and multimedia data analysis. They introduced a formal notion of a time series contrast set along with a fast algorithm to find time series contrast sets. An approach to quantitative contrast set mining without discretization in the preprocessing phase is proposed by Simeon

2. The support of a contrast set *ContrastSet* with respect to a group G_i, $\textit{support}(\textit{ContrastSet}, G_i)$, is the percentage of examples in G_i for which the contrast set is true.

and Hilderman (2007) with the algorithm Gen_QCSets. In this approach, a slightly modified equal width binning interval method is used.

Common to most contrast set mining approaches is that they generate all candidate contrast sets from discrete (or discretized) data and later use statistical tests to identify the interesting ones. Open questions identified by Webb et al. (2003) are yet unsolved: selection of appropriate heuristics for identifying interesting contrast sets, appropriate measures of quality for sets of contrast sets, and appropriate methods for presenting contrast sets to the end users.

2.2.2 SELECTED APPLICATIONS OF CONTRAST SET MINING

The contrast mining paradigm does not appear to have been pursued in many published applications. Webb et al. (2003) investigated its use with retail sales data. Wong and Tseng (2005) applied contrast set mining for designing customized insurance programs. Siu et al. (2005) have used contrast set mining to identify patterns in synchrotron x-ray data that distinguish tissue samples of different forms of cancerous tumor. Kralj et al. (2007b) have addressed a contrast set mining problem of distinguishing between two groups of brain ischaemia patients by transforming the contrast set mining task to a subgroup discovery task.

2.3 Emerging Pattern Mining

Emerging patterns were defined by Dong and Li (1999) as itemsets whose support increases significantly from one data set to another. Emerging patterns are said to capture emerging trends in time-stamped databases, or to capture differentiating characteristics between classes of data.

2.3.1 EMERGING PATTERN MINING ALGORITHMS

Efficient algorithms for mining emerging patterns were proposed by Dong and Li (1999) and Fan and Ramamohanarao (2003). When first defined by Dong and Li (1999), the purpose of emerging patterns was "to capture emerging trends in time-stamped data, or useful contrasts between data classes". Subsequent emerging pattern research has largely focused on the use of the discovered patterns for classification purposes, for example, classification by emerging patterns (Dong et al., 1999; Li et al., 2000) and classification by jumping emerging patterns[3] (Li et al., 2001). An advanced Bayesian approach (Fan and Ramamohanara, 2003) and bagging (Fan et al., 2006) were also proposed.

From a semantic point of view, emerging patterns are association rules with an itemset in rule antecedent, and a fixed consequent: $ItemSet \rightarrow D_1$, for given data set D_1 being compared to another data set D_2.

The measure of quality of emerging patterns is the *growth rate* (the ratio of the two supports). It determines, for example, that a pattern with a 10% support in one data set and 1% in the other is better than a pattern with support 70% in one data set and 10% in the other (as $\frac{10}{1} > \frac{70}{10}$). From the association rule perspective, $GrowthRate(ItemSet, D_1, D_2) = \frac{confidence(ItemSet \rightarrow D_1)}{1 - confidence(ItemSet \rightarrow D_1)}$. Thus it can be seen that growth rate provides an identical ordering to confidence, except that growth rate is undefined when confidence = 1.0.

3. Jumping emerging patterns are emerging patterns with support zero in one data set and greater then zero in the other data set.

```
MaritalStatus = single AND Sex = male  →  Approved = no
MaritalStatus = married  →  Approved = yes
MaritalStatus = divorced AND HasChildren = yes  →  Approved = no
```

Figure 4: Jumping emerging patterns in the data of Table 1.

Some researchers have argued that finding all the emerging patterns above a minimum growth rate constraint generates too many patterns to be analyzed by a domain expert. Fan and Ramamohanarao (2003) have worked on selecting the interesting emerging patterns, while Soulet et al. (2004) have proposed condensed representations of emerging patterns.

Boulesteix et al. (2003) introduced a CART-based approach to discover emerging patterns in microarray data. The method is based on growing decision trees from which the emerging patterns are extracted. It combines pattern search with a statistical procedure based on Fisher's exact test to assess the significance of each emerging pattern. Subsequently, sample classification based on the inferred emerging patterns is performed using maximum-likelihood linear discriminant analysis.

Figure 4 shows all jumping emerging patterns found for the data in Table 1 when using a minimum support of 15%. These were discovered using the Magnum Opus software, limiting the consequent to the variable *approved*, setting minimum confidence to 1.0 and setting minimum support to 2.

2.3.2 SELECTED APPLICATIONS OF EMERGING PATTERNS

Emerging patterns have been mainly applied to the field of bioinformatics, more specifically to microarray data analysis. Li et al. (2003) present an interpretable classifier based on simple rules that is competitive to the state of the art black-box classifiers on the acute lymphoblastic leukemia (ALL) microarray data set. Li and Wong (2002) have focused on finding groups of genes by emerging patterns and applied it to the ALL/AML data set and the colon tumor data set. Song et al. (2001) used emerging patterns together with unexpected change and the added/perished rule to mine customer behavior.

2.4 Subgroup Discovery

The task of subgroup discovery was defined by Klösgen (1996) and Wrobel (1997) as follows: "Given a population of individuals and a property of those individuals that we are interested in, find population subgroups that are statistically 'most interesting', for example, are as large as possible and have the most unusual statistical (distributional) characteristics with respect to the property of interest".

2.4.1 SUBGROUP DISCOVERY ALGORITHMS

Subgroup descriptions are conjunctions of features that are characteristic for a selected class of individuals (property of interest). A subgroup description can be seen as the condition part of a rule *SubgroupDescription* → *Class*. Therefore, subgroup discovery can be seen as a special case of a more general rule learning task.

Subgroup discovery research has evolved in several directions. On the one hand, exhaustive approaches guarantee the optimal solution given the optimization criterion. One system that can use both exhaustive and heuristic discovery algorithms is Explora by Klösgen (1996). Other algo-

```
Sex = female  →  Approved = yes
MaritalStatus = married  →  Approved = yes
MaritalStatus = divorced AND HasChildren = no  →  Approved = yes
Education = university  →  Approved = yes
MaritalStatus = single AND Sex = male  →  Approved = no
```

Figure 5: Subgroup descriptions induced by Apriori-SD from the data of Table 1.

rithms for exhaustive subgroup discovery are the SD-Map method by Atzmüller and Puppe (2006) and Apriori-SD by Kavšek and Lavrač (2006). On the other hand, adaptations of classification rule learners to perform subgroup discovery, including algorithm SD by Gamberger and Lavrač (2002) and algorithm CN2-SD by Lavrač et al. (2004b), use heuristic search techniques drawn from classification rule learning coupled with constraints appropriate for descriptive rules.

Relational subgroup discovery approaches have been proposed by Wrobel (1997, 2001) with algorithm Midos, by Klösgen and May (2002) with algorithm SubgroupMiner, which is designed for spatial data mining in relational space databases, and by Železný and Lavrač (2006) with the algorithm RSD (Relational Subgroup Discovery). RSD uses a propositionalization approach to relational subgroup discovery, achieved through appropriately adapting rule learning and first-order feature construction. Other non-relational subgroup discovery algorithms were developed, including an algorithm for exploiting background knowledge in subgroup discovery (Atzmüller et al., 2005a), and an iterative genetic algorithm SDIGA by del Jesus et al. (2007) implementing a fuzzy system for solving subgroup discovery tasks.

Different heuristics have been used for subgroup discovery. By definition, the interestingness of a subgroup depends on its unusualness and size, therefore the rule quality evaluation heuristics needs to combine both factors. Weighted relative accuracy (*WRAcc*, see Equation 2 in Section 3.3) is used by algorithms CN2-SD, Apriori-SD and RSD and, in a different formulation and in different variants, also by MIDOS and EXPLORA. Generalization quotient (q_g, see Equation 3 in Section 3.3) is used by the SD algorithm. SubgroupMiner uses the classical binominal test to verify if the target share is significantly different in a subgroup.

Different approaches have been used for eliminating redundant subgroups. Algorithms CN2-SD, Apriori-SD, SD and RSD use weighted covering (Lavrač et al., 2004b) to achieve rule diversity. Algorithms Explora and SubgroupMiner use an approach called subgroup suppression (Klösgen, 1996). A sample set of subgroup describing rules, induced by Apriori-SD with parameters *support* set to 15% (requiring at least 2 covered training examples per rule) and *confidence* set to 65%, is shown in Figure 5.

2.4.2 SELECTED APPLICATIONS OF SUBGROUP DISCOVERY

Subgroup discovery was used in numerous real-life applications. The applications in medical domains include the analysis of coronary heart disease (Gamberger and Lavrač, 2002) and brain ischaemia data analysis (Kralj et al., 2007b,a; Lavrač et al., 2007), as well as profiling examiners for sonographic examinations (Atzmüller et al., 2005b). Spatial subgroup mining applications include mining of census data (Klösgen et al., 2003) and mining of vegetation data (May and Ragia, 2002). There are also applications in other areas like marketing (del Jesus et al., 2007; Lavrač et al., 2004a) and analysis of manufacturing shop floor data (Jenkole et al., 2007).

2.5 Related Approaches

Research in some closely related areas of rule learning, performed independently from the above described approaches, is outlined below.

2.5.1 CHANGE MINING

The paper by Liu et al. (2001) on *fundamental rule changes* proposes a technique to identify the set of fundamental changes in two given data sets collected from two time periods. The proposed approach first generates rules and in the second phase it identifies changes (rules) that can not be explained by the presence of other changes (rules). This is achieved by applying statistical χ^2 test for homogeneity of support and confidence. This differs from contrast set discovery through its consideration of rules for each group, rather than itemsets. A change in the frequency of just one itemset between groups may affect many association rules, potentially all rules that have the itemset as either an antecedent or consequent.

Liu et al. (2000) and Wang et al. (2003) present techniques that identify differences in the decision trees and classification rules, respectively, found on two different data sets.

2.5.2 MINING CLOSED SETS FROM LABELED DATA

Closed sets have been proven successful in the context of compacted data representation for association rule learning. However, their use is mainly descriptive, dealing only with unlabeled data. It was recently shown that when considering labeled data, closed sets can be adapted for classification and discrimination purposes by conveniently contrasting covering properties on positive and negative examples (Garriga et al., 2006). The approach was successfully applied in potato microarray data analysis to a real-life problem of distinguishing between virus sensitive and resistant transgenic potato lines (Kralj et al., 2006).

2.5.3 EXCEPTION RULE MINING

Exception rule mining considers a problem of finding a set of rule pairs, each of which consists of an exception rule (which describes a regularity for fewer objects) associated with a strong rule (description of a regularity for numerous objects with few counterexamples). An example of such a rule pair is “using a seat belt is safe” (strong rule) and “using a seat belt is risky for a child” (exception rule). While the goal of exception rule mining is also to find descriptive rules from labeled data, in contrast with other rule discovery approaches described in this paper, the goal of exception rule mining is to find “weak” rules—surprising rules that are an exception to the general belief of background knowledge.

Suzuki (2006) and Daly and Taniar (2005), summarizing the research in exception rule mining, reveal that the key concerns addressed by this body of research include interestingness measures, reliability evaluation, practical application, parameter reduction and knowledge representation, as well as providing fast algorithms for solving the problem.

2.5.4 IMPACT RULES, BUMP HUNTING, QUANTITATIVE ASSOCIATION RULES

Supervised descriptive rule discovery seeks to discover sets of conditions that are related to deviations in the class distribution, where the class is a qualitative variable. A related body of research seeks to discover sets of conditions that are related to deviations in a target quantitative variable.

Contrast Set Mining	Emerging Pattern Mining	Subgroup Discovery	Rule Learning
contrast set	itemset	subgroup description	rule condition
groups $G_1, \ldots G_n$	data sets D_1 and D_2	class/property C	class/concept C_i
attribute-value pair	item	logical (binary) feature	condition
examples in groups $G_1, \ldots G_n$	transactions in data sets D_1 and D_2	examples of C and $\overline{C}$	examples of $C_1 \ldots C_n$
examples for which the contrast set is true	transactions containing the itemset	subgroup of instances	covered examples
support of contrast set on G_i support of contrast set on G_j	support of EP in data set D_1 support of EP in data set D_2	true positive rate false positive rate	true positive rate false positive rate

Table 2: Table of synonyms from different communities, showing the compatibility of terms.

Such techniques include Bump Hunting (Friedman and Fisher, 1999), Quantitative Association Rules (Aumann and Lindell, 1999) and Impact Rules (Webb, 2001).

3. A Unifying Framework for Supervised Descriptive Rule Induction

This section presents a unifying framework for contrast set mining, emerging pattern mining and subgroup discovery, as the main representatives of supervised descriptive rule discovery approaches. This is achieved by unifying the terminology, the task definitions and the rule learning heuristics.

3.1 Unifying the Terminology

Contrast set mining (CSM), emerging pattern mining (EPM) and subgroup discovery (SD) were developed in different communities, each developing their own terminology that needs to be clarified before proceeding. Below we show that terms used in different communities are compatible, according to the following definition of compatibility.

Definition 1: Compatibility of terms. *Terms used in different communities are compatible if they can be translated into equivalent logical expressions and if they bare the same meaning, that is, if terms from one community can replace terms used in another community.*

Lemma 1: *Terms used in CSM, EPM and SD are compatible.*
Proof The compatibility of terms is proven through a term dictionary, whose aim is to translate all the terms used in CSM, EPM and SD into the terms used in the rule learning community. The term dictionary is proposed in Table 2. More specifically, this table provides a dictionary of equivalent terms from contrast set mining, emerging pattern mining and subgroup discovery, in a unifying terminology of classification rule learning, and in particular of concept learning (considering class C_i as the concept to be learned from the positive examples of this concept, and the negative examples formed of examples of all other classes). ■

3.2 Unifying the Task Definitions

Having established a unifying view on the terminology, the next step is to provide a unifying view on the different task definitions.

CSM A contrast set mining task is defined as follows (Bay and Pazzani, 2001). Let A_1, A_2, ..., A_k be a set of k variables called attributes. Each A_i can take values from the set $\{v_{i1}, v_{i2}, \ldots, v_{im}\}$. Given a set of user defined groups G_1, G_2, ..., G_n of data instances, a contrast set is a conjunction of attribute-value pairs, defining a pattern that best discriminates the instances of different user-defined groups. A special case of contrast set mining considers only two contrasting groups (G_1 and G_2). In such cases, we wish to find characteristics of one group discriminating it from the other and vice versa.

EPM An emerging patterns mining task is defined as follows (Dong and Li, 1999). Let $I = \{i_1, i_2, \ldots, i_N\}$ be a set of items (note that an item is equivalent to a binary feature in SD, and an individual attribute-value pair in CSM). A transaction is a subset T of I. A *dataset* is a set D of transactions. A subset X of I is called an *itemset*. Transaction T contains an itemset X in a data set D, if $X \subseteq T$. For two data sets D_1 and D_2, emerging pattern mining aims at discovering itemsets whose support increases significantly from one data set to another.

SD In subgroup discovery, subgroups are described as conjunctions of features, where features are of the form $A_i = v_{ij}$ for nominal attributes, and $A_i > value$ or $A_i \leq value$ for continuous attributes. Given the property of interest C, and the population of examples of C and $\overline{C}$, the subgroup discovery task aims at finding population subgroups that are as large as possible and have the most unusual statistical (distributional) characteristics with respect to the property of interest C (Wrobel, 1997).

The definitions of contrast set mining, emerging pattern mining and subgroup discovery appear different: contrast set mining searches for discriminating characteristics of groups called contrast sets, emerging pattern mining aims at discovering itemsets whose support increases significantly from one data set to another, while subgroup discovery searches for subgroup descriptions. By using the dictionary from Table 2 we can see that the goals of these three mining tasks are very similar, it is primarily the terminology that differs.

Definition 2: Compatibility of task definitions. *Definitions of different learning tasks are compatible if one learning task can be translated into another learning task without substantially changing the learning goal.*

Lemma 2: *Definitions of CSM, EPM and SD tasks are compatible.*

Proof To show the compatibility of task definitions, we propose a unifying table (Table 3) of task definitions, allowing us to see that emerging pattern mining task $EPM(D_1,D_2)$ is equivalent to $CSM(G_i,G_j)$. It is also easy to show that a two-group contrast set mining task $CSM(G_i,G_j)$ can be directly translated into the following two subgroup discovery tasks: $SD(G_i)$ for $C = G_i$ and $\overline{C} = G_j$, and $SD(G_j)$ for $C = G_j$ and $\overline{C} = G_i$.

Contrast Set Mining	Emerging Pattern Mining	Subgroup Discovery	Rule Learning
Given	**Given**	**Given**	**Given**
examples in G_1 vs. G_j	transactions in D_1 and D_2	in examples C	examples in C_i
from $G_1, \dots G_i$	from D_1 and D_2	from C and $\overline{C}$	from $C_1 \dots C_n$
Find	**Find**	**Find**	**Find**
$ContrastSet_{i_k} \rightarrow G_i$	$ItemSet_{1_k} \rightarrow D_1$	$SubgrDescr_k \rightarrow C$	$\{RuleCond_{i_k} \rightarrow C_i\}$
$ContrastSet_{j_l} \rightarrow G_j$	$ItemSet_{2_l} \rightarrow D_2$		

Table 3: Table of task definitions from different communities, showing the compatibility of task definitions in terms of output rules.

Having proved that the subgroup discovery task is compatible with a two-group contrast set mining task, it is by induction compatible with a general contrast set mining task, as shown below.

```
$CSM(G_1, \dots G_n)$
  for i=2 to n do
    for j=1, j≠ i to n-1 do
      $SD(C = G_i \ vs. \ \overline{C} = G_j)$
```

Note that in Table 3 of task definitions column 'Rule Learning' again corresponds to a concept learning task instead of the general classification rule learning task. In the concept learning setting, which is better suited for the comparisons with supervised descriptive rule discovery approaches, a distinguished class C_i is learned from examples of this class, and examples of all other classes $C_1, \dots, C_{i-1}, C_{i+1}, C_N$ are merged to form the set of examples of class $\overline{C_i}$. In this case, induced rule set $\{RuleCond_{i_k} \rightarrow C_i\}$ consists only of rules for distinguished class C_i. On the other hand, in a general classification rule learning setting, from examples of N different classes a set of rules would be learned $\{\dots, RuleCond_{i_k} \rightarrow C_i, RuleCond_{i_{k+1}} \rightarrow C_i, \dots, RuleCond_{j_l} \rightarrow C_j, \dots, Default\}$, consisting of sets of rules of the form $RuleCond_{i_k} \rightarrow C_i$ for each individual class C_i, supplemented by the default rule. ■

While the primary tasks are very closely related, each of the three communities has concentrated on different sets of issues around this task. The contrast set discovery community has paid greatest attention to the statistical issues of multiple comparisons that, if not addressed, can result in high risks of false discoveries. The emerging patterns community has investigated how supervised descriptive rules can be used for classification. The contrast set and emerging pattern communities have primarily addressed only categorical data whereas the subgroup discovery community has also considered numeric and relational data. The subgroup discovery community has also explored techniques for discovering small numbers of supervised descriptive rules with high coverage of the data.

3.3 Unifying the Rule Learning Heuristics

The aim of this section is to provide a unifying view on rule learning heuristics used in different communities. To this end, we first investigate the rule quality measures.

Most rule quality measures are derived by analyzing the covering properties of the rule and the class in the rule consequent considered as positive. This relationship can be depicted by a confusion

	predicted		
actual	# of positives	# of negatives	
# of positives	$p = \lvert TP(X,Y)\rvert$	$\overline{p} = \lvert FN(X,Y)\rvert$	P
# of negatives	$n = \lvert FP(X,Y)\rvert$	$\overline{n} = \lvert TN(X,Y)\rvert$	N
	$p+n$	$\overline{p}+\overline{n}$	$P+N$

Table 4: Confusion matrix: $TP(X,Y)$ stands for true positives, $FP(X,Y)$ for false positives, $FN(X,Y)$ for false negatives and $TN(X,Y)$ for true negatives, as predicted by rule $X \rightarrow Y$.

matrix (Table 4, see, e.g., Kohavi and Provost, 1998), which considers that rule $R = X \rightarrow Y$ is represented as (X,Y), and defines p as the number of true positives (positive examples correctly classified as positive by rule (X,Y)), n as the number of false positives, etc., from which other covering characteristics of a rule can be derived: true positive rate $TPr(X,Y) = \frac{p}{P}$ and false positive rate $FPr(X,Y) = \frac{n}{N}$.

CSM Contrast set mining aims at discovering contrast sets that best discriminate the instances of different user-defined groups. The support of contrast set X with respect to group G_i, $support(X,G_i)$, is the percentage of examples in G_i for which the contrast set is true. Note that *support of a contrast set with respect to group G* is the same as *true positive rate* in the classification rule and subgroup discovery terminology, that is, $support(X,G_i) = \frac{count(X,G_i)}{|G_i|} = TPr(X,G_i)$. A derived goal of contrast set mining, proposed by Bay and Pazzani (2001), is to find contrast sets whose support differs meaningfully across groups, for δ being a user-defined parameter.

$$SuppDiff(X,G_i,G_j) = |support(X,G_i) - support(X,G_j)| \geq \delta.$$

EPM Emerging pattern mining aims at discovering itemsets whose support increases significantly from one data set to another Dong and Li (1999), where *support* of itemset X in data set D is computed as $support(X,D) = \frac{count(X,D)}{|D|}$, for $count(X,D)$ being the number of transactions in D containing X. Suppose we are given an ordered pair of data sets D_1 and D_2. The *GrowthRate* of an itemset X from D_1 to D_2, denoted as $GrowthRate(X,D_1,D_2)$, is defined as follows:

$$GrowthRate(X,D_1,D_2) = \frac{support(X,D_1)}{support(X,D_2)}. \quad (1)$$

Definitions of special cases of $GrowthRate(X,D_1,D_2)$ are as follows, if $support(X,D_1) = 0$ then $GrowthRate(X,D_1,D_2) = 0$, if $support(X,D_2) = 0$ then $GrowthRate(X,D_1,D_2) = \infty$.

SD Subgroup discovery aims at finding population subgroups that are as large as possible and have the most unusual statistical (distributional) characteristics with respect to the property of interest (Wrobel, 1997). There were several heuristics developed and used in the subgroup discovery community. Since they follow from the task definition, they try to maximize subgroup size and the distribution difference at the same time. Examples of such heuristics are the *weighted relative accuracy* (Equation 2, see Lavrač et al., 2004b) and the *generalization*

Contrast Set Mining	Emerging Pattern Mining	Subgroup Discovery	Rule Learning
$SuppDiff(X,G_i,G_j)$		$WRAcc(X,C)$	Piatetski-Shapiro heuristic leverage
	$GrowthRate(X,D_1,D_2)$	$q_g(X,C)$	odds ratio for $g=0$ accuracy/precision, for $g=p$

Table 5: Table of relationships between the pairs of heuristics, and their equivalents in classification rule learning.

quotient (Equation 3, see Gamberger and Lavrač, 2002) , for g being a user-defined parameter.

$$WRAcc(X,C) = \frac{p+n}{P+N} \cdot \left(\frac{p}{p+n} - \frac{P}{P+N} \right), \tag{2}$$

$$q_g(X,C) = \frac{p}{n+g}. \tag{3}$$

Let us now investigate whether the heuristics used in CSM, EPM and SD are compatible, using the following definition of compatibility.

Definition 3: Compatibility of heuristics.
Heuristic function h_1 is compatible *with h_2 if h_2 can be derived from h_1 and if for any two rules R and R', $h_1(R) > h_1(R') \Leftrightarrow h_2(R) > h_2(R')$.*

Lemma 3: *Definitions of CSM, EPM and SD heuristics are pairwise compatible.*
Proof The proof of Lemma 3 is established by proving two sub-lemmas, Lemma 3a and Lemma 3b, which prove the compatibility of two pairs of heuristics, whereas the relationships between these pairs is established through Table 5, and illustrated in Figures 6 and 7. ■

Lemma 3a: *The support difference heuristic used in CSM and the weighted relative accuracy heuristic used in SD are compatible.*
Proof Note that, as shown below, weighted relative accuracy (Equation 2) can be interpreted in terms of probabilities of rule antecedent X and consequent Y (class C representing the property of interest), and the conditional probability of class Y given X, estimated by relative frequencies.

$$WRAcc(X,Y) = P(X) \cdot (P(Y|X) - P(Y)).$$

From this equation we see that, indeed, when optimizing weighted relative accuracy of rule $X \rightarrow Y$, we optimize two contrasting factors: rule coverage $P(X)$ (proportional to the size of the subgroup), and distributional unusualness $P(Y|X) - P(Y)$ (proportional to the difference of the number of positive examples correctly covered by the rule and the number of positives in the original training set). It is straightforward to show that this measure is equivalent to the Piatetski-Shapiro measure, which evaluates the conditional (in)dependence of rule consequent and rule antecedent as follows:

$$PS(X,Y) = P(X \cdot Y) - P(X) \cdot P(Y).$$

Weighted relative accuracy, known from subgroup discovery, and support difference between groups, used in contrast set mining, are related as follows:[4]

$$\begin{aligned}
&WRAcc(X,Y) = \\
&\quad = P(X)\cdot[P(Y|X)-P(Y)] = P(Y\cdot X)-P(Y)\cdot P(X)\\
&\quad = P(Y\cdot X)-P(Y)\cdot[P(Y\cdot X)+P(\overline{Y}\cdot X)]\\
&\quad = (1-P(Y))\cdot P(Y\cdot X)-P(Y)\cdot P(\overline{Y}\cdot X)\\
&\quad = P(\overline{Y})\cdot P(Y)\cdot P(X|Y)-P(Y)\cdot P(\overline{Y})\cdot P(X|\overline{Y})\\
&\quad = P(\overline{Y})\cdot P(Y)\cdot[P(X|Y)-P(X|\overline{Y})]\\
&\quad = P(Y)\cdot P(\overline{Y})\cdot[TPr(X,Y)-FPr(X,Y)].
\end{aligned}$$

Since the distribution of examples among classes is constant for any data set, the first two factors $P(Y)$ *and* $P(\overline{Y})$ are constant within a data set. Therefore, when maximizing the weighted relative accuracy, one is maximizing the second factor $TPr(X,Y)-FPr(X,Y)$, which actually is support difference when we have a two group contrast set mining problem. Consequently, for $C=G_1$, and $\overline{C}=G_2$ the following holds:

$$WRAcc(X,C) = WRAcc(X,G_1) = P(G_1)\cdot P(G_2)\cdot[support(X,G_1)-support(X,G_2)].$$

■

Lemma 3b: *The growth rate heuristic used in EPM and the generalization quotient heuristic used in SD are compatible.*
Proof Equation 1 can be rewritten as follows:

$$GrowthRate(X,D_1,D_2) = \frac{support(X,D_1)}{support(C,D_2)} =$$
$$= \frac{count(X,D_1)}{count(X,D_2)}\cdot\frac{|D_2|}{|D_1|} = \frac{p}{n}\cdot\frac{N}{P}.$$

Since the distribution of examples among classes is constant for any data set, the quotient $\frac{N}{P}$ is constant. Consequently, the growth rate is the generalization quotient with $g=0$, multiplied by a constant. Therefore, the growth rate is compatible with the generalization quotient.

$$GrowthRate(X,C,\overline{C}) = q_0(X,C)\cdot\frac{N}{P}.$$

■

The lemmas prove that heuristics used in CSM and EPM can be translated into heuristics used in SD and vice versa. In this way, we have shown the compatibility of CSM and SD heuristics, as well as the compatibility of EPM and SD heuristics. While the lemmas do not prove direct compatibility of CSM and EPM heuristics, they prove that heuristics used in CSM and EPM can be translated into two heuristics used in SD, both aiming at trading-off between coverage and distributional difference.

4. Peter A. Flach is acknowledged for having derived these equations.

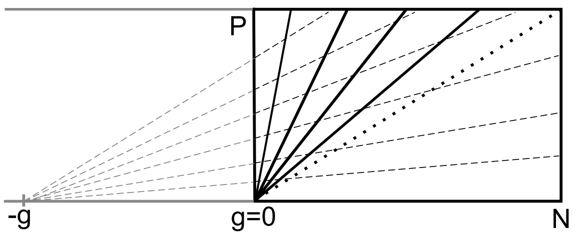


Figure 6: Isometrics for q_g. The dotted lines show the isometrics for a selected $g > 0$, while the full lines show the special case when $g = 0$, compatible to the EPM *growth rate* heuristic.

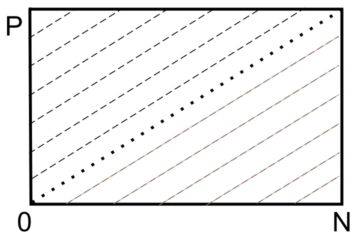


Figure 7: Isometrics for *WRAcc*, compatible to the CSM *support difference* heuristic.

Table 5 provides also the equivalents of these heuristics in terms of heuristics known from the classification rule learning community, details of which are beyond the scope of this paper (an interested reader can find more details on selected heuristics and their ROC representations in Fürnkranz and Flach, 2003).

Note that the growth rate heuristic from EPM, as a special case of the generalization quotient heuristic with $g = 0$, does not consider rule coverage. On the other hand, its compatible counterpart, the generalization quotient q_g heuristic used in SD, can be tailored to favor more general rules by setting the g parameter value, as for a general g value, the q_g heuristic provides a trade-off between rule accuracy and coverage. Figure 6[5] illustrates the q_g isometrics, for a general g value, as well as for value $g = 0$.

Note also that standard rule learners (such as CN2 by Clark and Niblett, 1989) tend to generate very specific rules, due to using accuracy heuristic $Acc(X,Y) = \frac{p+\bar{n}}{P+N}$ or its variants: the Laplace and the *m*-estimate. On the other hand, the CSM support difference heuristic and its SD counterpart *WRAcc* both optimize a trade-off between rule accuracy and coverage. The *WRAcc* isometrics are plotted in Figure 7.[6]

3.4 Comparison of Rule Selection Mechanisms

Having established a unifying view on the terminology, definitions and rule learning heuristics, the last step is to analyze rule selection mechanisms used by different algorithms. The motivation for rule selection can be either to find only significant rules or to avoid overlapping rules (too many too similar rules), or to avoid showing redundant rules to the end users. Note that rule selection is not always necessary and that depending on the goal, redundant rules can be valuable (e.g., clas-

5. This figure is due to Gamberger and Lavrač (2002).
6. This figure is due to Fürnkranz and Flach (2003).

sification by aggregating emerging patterns by Dong et al., 1999). Two approaches are commonly used: statistic tests and the (weighted) covering approach. In this section, we compare these two approaches.

Webb et al. (2003) show that contrast set mining is a special case of the more general rule discovery task. However, an experimental comparison of STUCCO, OPUS_AR and C4.5 has shown that standard rule learners return a larger set of rules compared to STUCCO, and that some of them are also not interesting to end users. STUCCO (see Bay and Pazzani 2001 for more details) uses several mechanisms for rule pruning. Statistical significance pruning removes contrast sets that, while significant and large, derive these properties only due to being specializations of more general contrast sets: any specialization is pruned that has a similar support to its parent or that fails a χ^2 test of independence with respect to its parent.

In the context of OPUS_AR, the emphasis has been on developing statistical tests that are robust in the context of the large search spaces explored in many rule discovery applications Webb (2007). These include tests for independence between the antecedent and consequent, and tests to assess whether specializations have significantly higher confidence than their generalizations.

In subgroup discovery, the *weighted covering approach* (Lavrač et al., 2004b) is used with the aim of ensuring the diversity of rules induced in different iterations of the algorithm. In each iteration, after selecting the best rule, the weights of positive examples are decreased according to the number of rules covering each positive example $rule\_count(e)$; they are set to $w(e) = \frac{1}{rule\_count(e)}$. For selecting the best rule in consequent iterations, the SD algorithm (Gamberger and Lavrač, 2002) uses—instead of the unweighted q_g measure (Equation 3)—the weighted variant of q_g defined in Equation 4, while the CN2-SD (Lavrač et al., 2004b) and APRIORI-SD (Kavšek and Lavrač, 2006) algorithms use the weighted relative accuracy (Equation 2) modified with example weights, as defined in Equation 5, where $p' = \sum_{TP(X,Y)} w(e)$ is the sum of the weights of all covered positive examples, and P' is the sum of the weights of all positive examples.

$$q'_g(X,Y) = \frac{p'}{n+g}, \tag{4}$$

$$WRAcc'(X,Y) = \frac{p'+n}{P'+N} \cdot \left(\frac{p'}{p'+n} - \frac{P}{P+N} \right). \tag{5}$$

Unlike in the sections on the terminology, task definitions and rule learning heuristics, the comparison of rule pruning mechanisms described in this section does not result in a unified view; although the goals of rule pruning may be the same, the pruning mechanisms used in different subareas of supervised descriptive rule discovery are—as shown above—very different.

4. Visualization

Webb et al. (2003) identify a need to develop appropriate methods for presenting contrast sets to end users, possibly through contrast set visualization. This open issue, concerning the visualization of contrast sets and emerging patterns, can be resolved by importing some of the solutions proposed in the subgroup discovery community. Several methods for subgroup visualization were developed by Wettschereck (2002), Wrobel (2001), Gamberger et al. (2002), Kralj et al. (2005) and Atzmüller and Puppe (2005). They are here illustrated using the coronary heart disease data set, originally analyzed by Gamberger and Lavrač (2002). The visualizations are evaluated by considering their

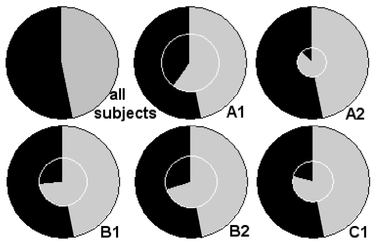


Figure 8: Subgroup visualization by pie charts.

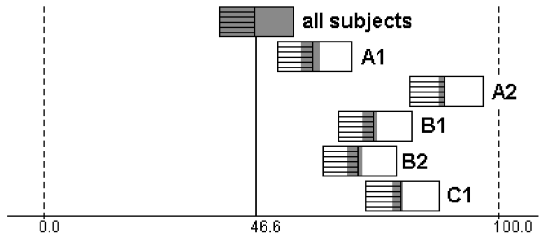


Figure 9: Subgroup visualization by box plots.

intuitiveness, correctness of displayed data, usefulness, ability to display contents besides the numerical properties of subgroups, (e.g., plot subgroup probability densities against the values of an attribute), and their extensibility to multi-class problems.

4.1 Visualization by Pie Charts

Slices of pie charts are the most common way of visualizing parts of a whole. They are widely used and understood. Subgroup visualization by pie chart, proposed by Wettschereck (2002), consists of a two-level pie for each subgroup. The base pie represents the distribution of individuals in terms of the property of interest of the entire example set. The inner pie represents the size and the distribution of individuals in terms of the property of interest in a specific subgroup. An example of five subgroups (subgroups A1, A2, B1, B2, C1), as well as the base pie "all subjects" are visualized by pie charts in Figure 8.

The main weakness of this visualization is the misleading representation of the relative size of subgroups. The size of a subgroup is represented by the radius of the circle. The faultiness arises from the surface of the circle which increases with the square of its radius. For example, a subgroup that covers 20% of examples is represented by a circle that covers only 4% of the whole surface, while a subgroup that covers 50% of examples is represented by a circle that covers 25% of the whole surface. In terms of usefulness, this visualization is not very handy since—in order to compare subgroups—one would need to compare sizes of circles, which is difficult. The comparison of distributions in subgroups is also not straightforward. This visualization also does not show the contents of subgroups. It would be possible to extend this visualization to multi-class problems.

4.2 Visualization by Box Plots

In subgroup visualization by box plots, introduced by Wrobel (2001), each subgroup is represented by one box plot (all examples are also considered as one subgroup and are displayed in the top box). Each box shows the entire population; the horizontally stripped area on the left represents the positive examples and the white area on the right-hand side of the box represents the negative examples. The grey area within each box indicates the respective subgroup. The overlap of the grey area with the hatched area shows the overlap of the group with the positive examples. Hence, the more to the left the grey area extends the better. The less the grey area extends to the right of the hatched area, the more specific a subgroup is (less overlap with the subjects of the negative class). Finally, the location of the box along the X-axis indicates the relative share of the target class within each subgroup: the more to the right a box is placed, the higher is the share of the target value within this subgroup. The vertical line (in Figure 9 at value 46.6%) indicates the default accuracy, that is, the number of positive examples in the entire population. An example box plot visualization of five subgroups is presented in Figure 9.

On the negative side, the intuitiveness of this visualization is relatively poor since an extensive explanation is necessary for understanding it. It is also somewhat illogical since the boxes that are placed more to the right and have more grey color on the left-hand side represent the best subgroups. This visualization is not very attractive since most of the image is white; the grey area (the part of the image that really represents the subgroups) is a relatively tiny part of the entire image. On the positive side, all the visualized data are correct and the visualization is useful since the subgroups are arranged by their confidence. It is also easier to contrast the sizes of subgroups compared to their pie chart visualization. However, this visualization does not display the contents of the data. It would also be difficult to extend this visualization to multi-class problems.

4.3 Visualizing Subgroup Distribution w.r.t. a Continuous Attribute

The distribution of examples w.r.t. a continuous attribute, introduced by Gamberger and Lavrač (2002) and Gamberger et al. (2002), was used in the analysis of several medical domains. It is the only subgroup visualization method that offers an insight of the visualized subgroups. The approach assumes the existence of at least one numeric (or ordered discrete) attribute of expert's interest for subgroup analysis. The selected attribute is plotted on the X-axis of the diagram. The Y-axis represents the target variable, or more precisely, the number of instances belonging to target property C (shown on the $Y+$ axis) or not belonging to C (shown on the $Y-$ axis) for the values of the attribute on the X-axis. It must be noted that both directions of the Y-axis are used to indicate the number of instances. The entire data set and two subgroups A1 and B2 are visualized by their distribution over a continuous attribute in Figure 10.

This visualization method is not completely automatic, since the automatic approach does not provide consistent results. The automatic approach calculates the number of examples for each value of the attribute on the X-axis by moving a sliding window and counting the number of examples in that window. The outcome is a smooth line. The difficulty arises when the attribute from the X-axis appears in the subgroup description. In such a case, a manual correction is needed for this method to be realistic.

This visualization method is very intuitive since it practically does not need much explanation. It is attractive and very useful to the end user since it offers an insight in the contents of displayed

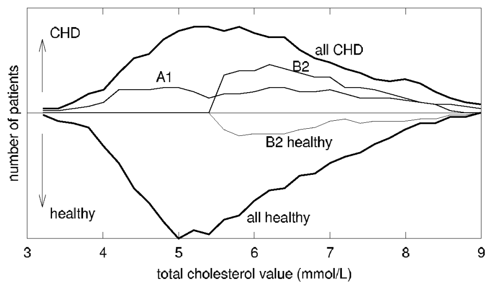


Figure 10: Subgroup visualization w.r.t. a continuous attribute. For clarity of the picture, only the positive (Y+) side of subgroup A1 is depicted.

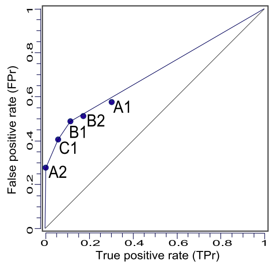


Figure 11: Representation of subgroups in the ROC space.

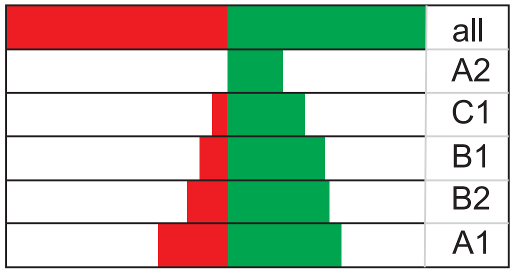


Figure 12: Subgroup visualization by bar charts.

examples. However, the correctness of displayed data is questionable. It is impossible to generalize this visualization to multi-class problems.

4.4 Representation in the ROC Space

The ROC (Receiver Operating Characteristics) (Provost and Fawcett, 2001) space is a 2-dimensional space that shows classifier (rule/rule set) performance in terms of its false positive rate (*FPr*) plotted on the X-axis, and true positive rate (*TPr*) plotted on the Y-axis. The ROC space is appropriate for measuring the success of subgroup discovery, since subgroups whose $\frac{TPr}{FPr}$ tradeoffs are close to the main diagonal (line connecting the points (0, 0) and (1, 1) in the ROC space) can be discarded as insignificant (Kavšek and Lavrač, 2006); the reason is that the rules with the $\frac{TPr}{FPr}$ ration on the main diagonal have the same distribution of covered positives and negatives (*TPr*= *FPr*) as the distribution in the entire data set. An example of five subgroups represented in the ROC space is shown in Figure 11.

Even though the ROC space is an appropriate rule visualization, it is usually used just for the evaluation of discovered rules. The ROC convex hull is the line connecting the potentially optimal subgroups. The area under the ROC convex hull (AUC, area under curve) is a measure of quality of the resulting ruleset.[7]

This visualization method is not intuitive to the end user, but is absolutely clear to every machine learning expert. The displayed data is correct, but there is no content displayed. An advantage of this method compared to the other visualization methods is that it allows the comparison of outcomes of different algorithms at the same time. The ROC space is designed for two-class problems and is therefore inappropriate for multi-class problems.

4.5 Bar Charts Visualization

The visualization by bar charts was introduced by Kralj et al. (2005). In this visualization, the purpose of the first line is to visualize the distribution of the entire example set. The area on the right represents the positive examples and the area on the left represents the negative examples of the target class. Each following line represents one subgroup. The positive and the negative examples of each subgroup are drawn below the positive and the negative examples of the entire example set. Subgroups are sorted by the relative share of positive examples (precision).

7. Note that in terms of $\frac{TPr}{FPr}$ ratio optimality, two subgroups (A1 and B2) are suboptimal, lying below the ROC convex hull.

An example of five subgroups visualized by bar charts is shown in Figure 12. It is simple, understandable and shows all the data correctly. This visualization method allows simple comparison between subgroups and is therefore useful. It is relatively straight-forward to understand and can be extended to multi-class problems. It does not display the contents of data, though.

4.6 Summary of Subgroup Visualization Methods

In this section, we (subjectively) compare the five different subgroup visualization methods by considering their intuitiveness, correctness of displayed data, usefulness, ability to ability to display contents besides the numerical properties of subgroups, (e.g., plot subgroup probability densities against the values of an attribute), and their extensibility to multi-class problems. The summary of the evaluation is presented in Table 6.

	Pie chart	Box plot	Continuous attribute	ROC	Bar chart
Intuitiveness	+	-	+	+/-	+
Correctness	-	+	-	+	+
Usefulness	-	+	+	+	+
Contents	-	-	+	-	-
Multi-class	+	-	-	-	+

Table 6: Our evaluation of subgroup visualization methods.

Two visualizations score best in Table 6 of our evaluation of subgroup visualization methods: the visualization of subgroups w.r.t. a continuous attribute and the bar chart visualization. The visualization of subgroups w.r.t. a continuous attribute is the only visualization that directly shows the contents of the data; its main shortcomings are the doubtful correctness of the displayed data and its difficulty to be extended to multi-class problems. It also requires a continuous or ordered discrete attribute in the data. The bar chart visualization combines the good properties of the pie chart and the box plot visualization. In Table 6, it only fails in displaying the contents of the data. By using the two best visualizations, one gets a very good understanding of the mining results.

To show the applicability of subgroup discovery visualizations for supervised descriptive rule discovery, the bar visualizations of results of contrast set mining, jumping emerging patterns and subgroup discovery on the survey data analysis problem of Section 2 are shown in Figures 13, 14 and 15, respectively.

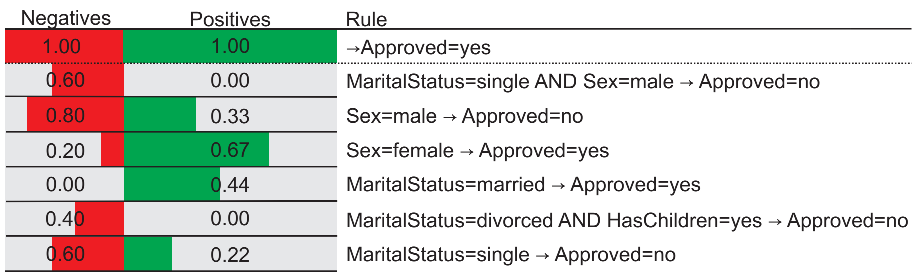


Figure 13: Bar visualization of contrast sets of Figure 3.

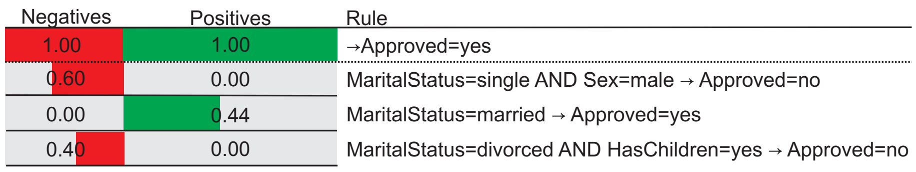


Figure 14: Bar visualization of jumping emerging patterns of Figure 4.

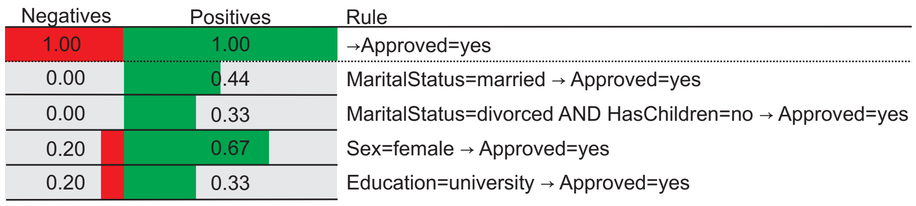


Figure 15: Bar visualization of subgroups of Figure 5 of individuals who have approved the issue.

5. Conclusions

Patterns in the form of rules are intuitive, simple and easy for end users to understand. Therefore, it is not surprising that members of different communities have independently addressed supervised descriptive rule induction, each of them solving similar problems in similar ways and developing vocabularies according to the conventions of their respective research communities.

This paper sheds a new light on previous work in this area by providing a systematic comparison of the terminology, definitions, goals, algorithms and heuristics of contrast set mining (CSM), emerging pattern mining (EPM) and subgroup discovery (SD) in a unifying framework called supervised descriptive rule discovery. We have also shown that the heuristics used in CSM and EPM can be translated into two well-known heuristics used in SD, both aiming at trading-off between coverage and distributional difference. In addition, the paper presents a critical survey of existing visualization methods, and shows that some methods used in subgroup discovery can be easily adapted for use in CSM and EPM.

Acknowledgments

The work of Petra Kralj and Nada Lavrač was funded by the project Knowledge Technologies (grant no. P2-0103) funded by the Slovene National Research Agency, and co-funded by the European 6FP project IQ - *Inductive Queries for Mining Patterns and Models* (IST-FP6-516169). Geoff Webb's contribution to this work has been supported by Australian Research Council Grant DP0772238.

References

Rakesh Agrawal, Heikki Mannila, Ramakrishnan Srikant, Hannu Toivonen, and A. Inkeri Verkamo. Fast discovery of association rules. *Advances in Knowledge Discovery and Data Mining*, pages 307–328, 1996.

Martin Atzmüller and Frank Puppe. Semi-automatic visual subgroup mining using VIKAMINE. *Journal of Universal Computer Science (JUCS), Special Issue on Visual Data Mining*, 11(11):

1752–1765, 2005.

Martin Atzmüller and Frank Puppe. SD-Map - a fast algorithm for exhaustive subgroup discovery. In *Proceedings of the 10th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD-06)*, pages 6–17, 2006.

Martin Atzmüller, Frank Puppe, and Hans-Peter Buscher. Exploiting background knowledge for knowledge-intensive subgroup discovery. In *Proceedings of the 19th International Joint Conference on Artificial Intelligence (IJCAI-05)*, pages 647–652, 2005a.

Martin Atzmüller, Frank Puppe, and Hans-Peter Buscher. Profiling examiners using intelligent subgroup mining. In *Proceedings of the 10th Workshop on Intelligent Data Analysis in Medicine and Pharmacology (IDAMAP-05)*, pages 46–51, 2005b.

Yonatan Aumann and Yehuda Lindell. A statistical theory for quantitative association rules. In *Proceedings of the 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-99)*, pages 261–270, 1999.

Stephen D. Bay. Multivariate discretization of continuous variables for set mining. In *Proceedings of the 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-2000)*, pages 315–319, 2000.

Stephen D. Bay and Michael J. Pazzani. Detecting group differences: Mining contrast sets. *Data Mining and Knowledge Discovery*, 5(3):213–246, 2001.

Roberto J. Bayardo. Efficiently mining long patterns from databases. In *Proceedings of the 1998 ACM SIGMOD International Conference on Management of Data (SIGMOD-98)*, pages 85–93, 1998.

Anne-Laure Boulesteix, Gerhard Tutz, and Korbinian Strimmer. A CART-based approach to discover emerging patterns in microarray data. *Bioinformatics*, 19(18):2465–2472, 2003.

Peter Clark and Tim Niblett. The CN2 induction algorithm. *Machine Learning*, 3(4):261–283, 1989.

William W. Cohen. Fast effective rule induction. In *Proceedings of the 12th International Conference on Machine Learning (ICML-95)*, pages 115–123, 1995.

Olena Daly and David Taniar. Exception rules in data mining. In *Encyclopedia of Information Science and Technology (II)*, pages 1144–1148. 2005.

María José del Jesus, Pedro González, Francisco Herrera, and Mikel Mesonero. Evolutionary fuzzy rule induction process for subgroup discovery: A case study in marketing. *IEEE Transactions on Fuzzy Systems*, 15(4):578–592, 2007.

Guozhu Dong and Jinyan Li. Efficient mining of emerging patterns: Discovering trends and differences. In *Proceedings of the 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-99)*, pages 43–52, 1999.

Guozhu Dong, Xiuzhen Zhang, Limsoon Wong, and Jinyan Li. CAEP: Classification by aggregating emerging patterns. In *Proceedings of the 2nd International Conference on Discovery Science (DS-99)*, pages 30–42, 1999.

Hongjian Fan and Kotagiri Ramamohanara. A bayesian approach to use emerging patterns for classification. In *Proceedings of the 14th Australasian Database Conference (ADC-03)*, pages 39–48, 2003.

Hongjian Fan and Kotagiri Ramamohanarao. Efficiently mining interesting emerging patterns. In *Proceeding of the 4th International Conference on Web-Age Information Management (WAIM-03)*, pages 189–201, 2003.

Hongjian Fan, Ming Fan, Kotagiri Ramamohanarao, and Mengxu Liu. Further improving emerging pattern based classifiers via bagging. In *Proceedings of the 10th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD-06)*, pages 91–96, 2006.

Jerome H. Friedman and Nicholas I. Fisher. Bump hunting in high-dimensional data. *Statistics and Computing*, 9(2):123–143, 1999.

Johannes Fürnkranz and Peter A. Flach. An analysis of rule evaluation metrics. In *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*, pages 202–209, 2003.

Dragan Gamberger and Nada Lavrač. Expert-guided subgroup discovery: Methodology and application. *Journal of Artificial Intelligence Research*, 17:501–527, 2002.

Dragan Gamberger, Nada Lavrač, and Dietrich Wettschereck. Subgroup visualization: A method and application in population screening. In *Proceedings of the 7th International Workshop on Intelligent Data Analysis in Medicine and Pharmacology (IDAMAP-02)*, pages 31–35, 2002.

Gemma C. Garriga, Petra Kralj, and Nada Lavrač. Closed sets for labeled data. In *Proceedings of the 10th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD-06)*, pages 163 – 174, 2006.

Robert J. Hilderman and Terry Peckham. A statistically sound alternative approach to mining contrast sets. In *Proceedings of the 4th Australia Data Mining Conference (AusDM-05)*, pages 157–172, 2005.

Jože Jenkole, Petra Kralj, Nada Lavrač, and Alojzij Sluga. A data mining experiment on manufacturing shop floor data. In *Proceedings of the 40th International Seminar on Manufacturing Systems (CIRP-07)*, 2007. 6 pages.

Branko Kavšek and Nada Lavrač. APRIORI-SD: Adapting association rule learning to subgroup discovery. *Applied Artificial Intelligence*, 20(7):543–583, 2006.

Willi Klösgen. Explora: A multipattern and multistrategy discovery assistant. *Advances in Knowledge Discovery and Data Mining*, pages 249–271, 1996.

Willi Klösgen and Michael May. Spatial subgroup mining integrated in an object-relational spatial database. In *Proceedings of the 6th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD-02)*, pages 275–286, 2002.

Willi Klösgen, Michael May, and Jim Petch. Mining census data for spatial effects on mortality. *Intelligent Data Analysis*, 7(6):521–540, 2003.

Ron Kohavi and Foster Provost, editors. *Editorial for the Special Issue on Applications of Machine Learning and the Knowledge Discovery Process, Glossary of Terms*, 1998.

Petra Kralj, Nada Lavrač, and Blaž Zupan. Subgroup visualization. In *8th International Multiconference Information Society (IS-05)*, pages 228–231, 2005.

Petra Kralj, Ana Rotter, Nataša Toplak, Kristina Gruden, Nada Lavrač, and Gemma C. Garriga. Application of closed itemset mining for class labeled data in functional genomics. *Informatica Medica Slovenica*, (1):40–45, 2006.

Petra Kralj, Nada Lavrač, Dragan Gamberger, and Antonija Krstačić. Contrast set mining for distinguishing between similar diseases. In *Proceedings of the 11th Conference on Artificial Intelligence in Medicine (AIME-07)*, pages 109–118, 2007a.

Petra Kralj, Nada Lavrač, Dragan Gamberger, and Antonija Krstačić. Contrast set mining through subgroup discovery applied to brain ischaemia data. In *Proceedings of the 11th Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining : (PAKDD-07)*, pages 579–586, 2007b.

Nada Lavrač, Bojan Cestnik, Dragan Gamberger, and Peter A. Flach. Decision support through subgroup discovery: Three case studies and the lessons learned. *Machine Learning Special issue on Data Mining Lessons Learned*, 57(1-2):115–143, 2004a.

Nada Lavrač, Branko Kavšek, Peter A. Flach, and Ljupčo Todorovski. Subgroup discovery with CN2-SD. *Journal of Machine Learning Research*, 5:153–188, 2004b.

Nada Lavrač, Petra Kralj, Dragan Gamberger, and Antonija Krstačić. Supporting factors to improve the explanatory potential of contrast set mining: Analyzing brain ischaemia data. In *Proceedings of the 11th Mediterranean Conference on Medical and Biological Engineering and Computing (MEDICON-07)*, pages 157–161, 2007.

Jinyan Li and Limsoon Wong. Identifying good diagnostic gene groups from gene expression profiles using the concept of emerging patterns. *Bioinformatics*, 18(10):1406–1407, 2002.

Jinyan Li, Guozhu Dong, and Kotagiri Ramamohanarao. Instance-based classification by emerging patterns. In *Proceedings of the 14th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD-2000)*, pages 191–200, 2000.

Jinyan Li, Guozhu Dong, and Kotagiri Ramamohanarao. Making use of the most expressive jumping emerging patterns for classification. *Knowledge and Information Systems*, 3(2):1–29, 2001.

Jinyan Li, Huiqing Liu, James R. Downing, Allen Eng-Juh Yeoh, and Limsoon Wong. Simple rules underlying gene expression profiles of more than six subtypes of acute lymphoblastic leukemia (ALL) patients. *Bioinformatics*, 19(1):71–78, 2003.

Jessica Lin and Eamonn Keogh. Group SAX: Extending the notion of contrast sets to time series and multimedia data. In *Proceedings of the 10th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD-06)*, pages 284–296, 2006.

Bing Liu, Wynne Hsu, Heng-Siew Han, and Yiyuan Xia. Mining changes for real-life applications. In *Proceedings of the 2nd International Conference on Data Warehousing and Knowledge Discovery (DaWaK-2000)*, pages 337–346, 2000.

Bing Liu, Wynne Hsu, and Yiming Ma. Discovering the set of fundamental rule changes. In *Proceedings of the 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-01)*, pages 335–340, 2001.

Michael May and Lemonia Ragia. Spatial subgroup discovery applied to the analysis of vegetation data. In *Proceedings of the 4th International Conference on Practical Aspects of Knowledge Management (PAKM-2002)*, pages 49–61, 2002.

Foster J. Provost and Tom Fawcett. Robust classification for imprecise environments. *Machine Learning*, 42(3):203–231, 2001.

J. Ross Quinlan. Induction of decision trees. *Machine Learning*, 1(1):81–106, 1986.

J. Ross Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1993.

Mondelle Simeon and Robert J. Hilderman. Exploratory quantitative contrast set mining: A discretization approach. In *Proceedings of the 19th IEEE International Conference on Tools with Artificial Intelligence - Vol.2 (ICTAI-07)*, pages 124–131, 2007.

K.K.W. Siu, S.M. Butler, T. Beveridge, J.E. Gillam, C.J. Hall, A.H. Kaye, R.A. Lewis, K. Mannan, G. McLoughlin, S. Pearson, A.R. Round, E. Schultke, G.I. Webb, and S.J. Wilkinson. Identifying markers of pathology in SAXS data of malignant tissues of the brain. *Nuclear Instruments and Methods in Physics Research A*, 548:140–146, 2005.

Hee S. Song, Jae K. Kimb, and Soung H. Kima. Mining the change of customer behavior in an internet shopping mall. *Expert Systems with Applications*, 21(3):157–168, 2001.

Arnaud Soulet, Bruno Crmilleux, and Franois Rioult. Condensed representation of emerging patterns. In *Proceedings of the 8th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD-04)*, pages 127–132, 2004.

Einoshin Suzuki. Data mining methods for discovering interesting exceptions from an unsupervised table. *Journal of Universal Computer Science*, 12(6):627–653, 2006.

Ke Wang, Senqiang Zhou, Ada W.-C. Fu, and Jeffrey X. Yu. Mining changes of classification by correspondence tracing. In *Proceedings of the 3rd SIAM International Conference on Data Mining (SDM-03)*, pages 95–106, 2003.

Geoffrey I. Webb. OPUS: An efficient admissible algorithm for unordered search. *Journal of Artificial Intelligence Research*, 3:431–465, 1995.

Geoffrey I. Webb. Discovering associations with numeric variables. In *Proceedings of the 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-01)*, pages 383–388, 2001.

Geoffrey I. Webb. Discovering significant patterns. *Machine Learning*, 68(1):1–33, 2007.

Geoffrey I. Webb, Shane M. Butler, and Douglas Newlands. On detecting differences between groups. In *Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-03)*, pages 256–265, 2003.

Dietrich Wettschereck. A KDDSE-independent PMML visualizer. In *Proceedings of 2nd Workshop on Integration Aspects of Data Mining, Decision Support and Meta-Learning (IDDM-02)*, pages 150–155, 2002.

Tzu-Tsung Wong and Kuo-Lung Tseng. Mining negative contrast sets from data with discrete attributes. *Expert Systems with Applications*, 29(2):401–407, 2005.

Stefan Wrobel. An algorithm for multi-relational discovery of subgroups. In *Proceedings of the 1st European Conference on Principles of Data Mining and Knowledge Discovery (PKDD-97)*, pages 78–87, 1997.

Stefan Wrobel. Inductive logic programming for knowledge discovery in databases. In Sašo Džeroski and Nada Lavrač, editors, *Relational Data Mining*, chapter 4, pages 74–101. 2001.

Filip Železný and Nada Lavrač. Propositionalization-based relational subgroup discovery with RSD. *Machine Learning*, 62:33–63, 2006.

Journal of Machine Learning Research 10 (2009) 405-440 Submitted 1/07; Revised 8/08; Published 2/09

Particle Swarm Model Selection

Hugo Jair Escalante HUGOJAIR@CCC.INAOEP.MX
Manuel Montes MMONTESG@INAOEP.MX
Luis Enrique Sucar ESUCAR@INAOEP.MX
Department of Computational Sciences
National Institute of Astrophysics, Optics and Electronics
Puebla, México, 72840

Editor: Isabelle Guyon and Amir Saffari

Abstract

This paper proposes the application of particle swarm optimization (*PSO*) to the problem of *full model selection, FMS,* for classification tasks. *FMS* is defined as follows: given a pool of preprocessing methods, feature selection and learning algorithms, to select the combination of these that obtains the lowest classification error for a given data set; the task also includes the selection of hyperparameters for the considered methods. This problem generates a vast search space to be explored, well suited for stochastic optimization techniques. *FMS* can be applied to any classification domain as it does not require domain knowledge. Different model types and a variety of algorithms can be considered under this formulation. Furthermore, competitive yet simple models can be obtained with *FMS*. We adopt *PSO* for the search because of its proven performance in different problems and because of its simplicity, since neither expensive computations nor complicated operations are needed. Interestingly, the way the search is guided allows *PSO* to avoid overfitting to some extend. Experimental results on benchmark data sets give evidence that the proposed approach is very effective, despite its simplicity. Furthermore, results obtained in the framework of a model selection challenge show the competitiveness of the models selected with *PSO*, compared to models selected with other techniques that focus on a single algorithm and that use domain knowledge.

Keywords: full model selection, machine learning challenge, particle swarm optimization, experimentation, cross validation

1. Introduction

Model selection is the task of picking the model that best describes a data set (Hastie et al., 2001). Since the phrase *describing a data set* can be interpreted in several different ways, the model selection task can denote diverse related problems, including: variable and feature selection (Bengio and Chapados, 2003; Guyon et al., 2006a; Guyon and Elisseeff, 2003), system identification (Voss and Feng, 2002; Nelles, 2001), parameter-hyperparameter optimization (Guyon et al., 2006b; Kim et al., 2002; Hastie et al., 2001; Cawley and Talbot, 2007b; Escalante et al., 2007), and discretization (Boullé, 2007; Hue and Boullé, 2007). In this paper we give a broader interpretation to this task and call it *full model selection* (*FMS*). The *FMS* problem consists on the following: given a pool of preprocessing methods, feature selection and learning algorithms, select the combination of these that obtains the lowest classification error for a given data set. This task also includes the selection of hyperparameters for the considered methods, resulting in a vast search space that is well suited for stochastic optimization techniques.

Adopting a broader interpretation to the model selection problem allows us to consider different model types and a variety of methods, in contrast to techniques that consider a single model type (i.e., either learning algorithm or feature selection method, but not both) and a single method (e.g., neural networks). Also, since neither prior domain knowledge nor machine learning knowledge is required, *FMS* can be applied to any classification problem without modification. This is a clear advantage over ad-hoc model selection methods that perform well on a single domain or that work for a fixed algorithm. This will help users with limited machine learning knowledge, since *FMS* can be seen as a black-box tool for model selection. Machine learning experts can also benefit from this approach. For example, several authors make use of search strategies for the selection of *candidate models* (Lutz, 2006; Boullé, 2007; Reunanen, 2007; Wichard, 2007), the *FMS* approach can be adopted for obtaining such candidate models.

One could expect a considerable loss of accuracy by gaining generality. However, this is not the case of the proposed approach since in international competitions it showed comparable performance to other techniques that were designed for a single algorithm (i.e., doing hyperparameter optimization) and to methods that took into account domain knowledge (Guyon et al., 2008). The main drawback is the computational cost to explore the vast search space, particularly for large data sets. But, we can gain efficiency without a significant loss in accuracy, by adopting a random subsampling strategy, see Section 4.3. The difficult interpretability of the selected models is another limitation of the proposed approach. However, naive users may accept to trade interpretably for ease-of-use, while expert users may gain insight in the problem at hand by analyzing the structure of the selected model (type of preprocessing chosen, number of features selected, linearity or non-linearity of the predictor).

In this paper, we propose to use particle swarm optimization (*PSO*) for exploring the full-models search space. *PSO* is a bio-inspired search technique that has shown comparable performance to that of evolutionary algorithms (Angeline, 1998; Reyes and Coello, 2006). Like evolutionary algorithms, *PSO* is useful when other techniques such as gradient descend or direct analytical discovery are not applicable. Combinatoric and real-valued optimization problems in which the optimization surface possesses many locally optimal solutions, are well suited for swarm optimization. In *FMS* it must be found the best combination of methods (for preprocessing, feature selection and learning) and simultaneously optimizing real valued functions (finding pseudo-optimal parameters for the considered methods), in consequence, the application of *PSO* is straightforward.

The methodological differences between swarm optimization and evolutionary algorithms have been highlighted by several authors (Angeline, 1998; Kennedy and Eberhart, 1995, 2001). However a difference in performance has not been demonstrated in favor of either method. Such demonstration would be a difficult task because no black-box stochastic optimization algorithm can outperform another over all optimization problems, not even over random search (Wolpert and Macready, 1997; van den Bergh, 2001). We selected *PSO* instead of evolutionary algorithms because of its simplicity and generality as no important modification was made for applying it to *FMS*. *PSO* is easier to implement than evolutionary algorithms because it only involves a single operator for updating solutions. In contrast, evolutionary algorithms require a particular representation and specific methods for cross-over, mutation, speciation and selection. Furthermore, *PSO* has been found to be very effective in a wide variety of applications, being able to produce good solutions at a very low computational cost (Gudise and Venayagamoorthy, 2003; Hernández et al., 2004; Xiaohui et al., 2003; Yoshida et al., 2001; Robinson, 2004; Kennedy and Eberhart, 2001; Reyes and Coello, 2006).

PSO is compared to pattern search (*PS*) in order to evaluate the added value of using the swarm strategy instead of another intensive search method. We consider *PS* among other search techniques because of its simplicity and proved performance in model selection (Momma and Bennett, 2002; Bi et al., 2003; Dennis and Torczon, 1994). Cross validation (*CV*) is used in both techniques for assessing the *goodness* of models. Experimental results in benchmark data give evidence that both *PSO* and *PS* are effective strategies for *FMS*. However, it was found that *PSO* outperforms *PS*, showing better convergence behavior and being less prone to overfitting. Furthermore, the proposed method was evaluated in the context of a model selection competition in which several specialized and prior-knowledge based methods for model selection were used. Models selected with *PSO* were always among the top ranking models through the different stages of the challenge (Guyon et al., 2006c, 2007, 2008; Escalante et al., 2007). During the challenge, our best entry was ranked 8th over all ranked participants, 5th among the methods that did not use domain knowledge and 2nd among the methods that used the software provided by the organizers (Guyon et al., 2006c, 2007, 2008). In this paper we outperform the latter entry while reducing the computational burden by using a subsampling strategy; our best entry is currently the top-ranked one among models that do not use prior domain knowledge and 2nd over all entries, see Section 4.3.

PSO has been widely used for parameter selection in supervised learning (Kennedy and Eberhart, 1995, 2001; Salerno, 1997; Gudise and Venayagamoorthy, 2003). However, parameter selection is related with the first level of inference in which, given a learning algorithm, the task is to find parameters for such algorithm in order to describe the data. For example, in neural networks the adjustment of weights between units according to some training data is a parameter selection problem. Hyperparameter optimization, on the other hand, is related with the second level of inference, that is, finding parameters for the methods that in turn should determine parameters for describing the data. In the neural network example, selecting the optimal number of units, the learning rate, and the number of epochs for training the network is a hyperparameter optimization problem. *FMS* is capable of operating across several levels of inference by simultaneously performing feature selection, preprocessing and classifier selection, and hyperparameter optimization for the selected methods. *PSO* has been used for hyperparameter optimization by Voss and Feng (2002), however they restricted the problem to linear systems for univariate data sets, considering one hundred data observations. In this paper we are going several steps further: we applied *PSO* for *FMS* considering non-linear models in multivariate data sets with a large number of observations.

Parallel to this work, Gorissen et al. used genetic algorithms for meta-model selection in regression tasks (Gorissen, 2007; Gorissen et al., 2008), a similar approach that emerged totally independently to our proposal. One should note, however, that this method has been used for a different task in low dimensional data sets; most results are reported for 2D data. Even with this dimensionality the method has been run for only a few iterations with a small population size. Their use of a genetic algorithm required the definition of specific operators *for each of the considered models*. Gorissen et al. considered seven different models (including neural networks and kernel methods) that required of 18 different genetic operators for creation, mutation and cross-over (Gorissen, 2007; Gorissen et al., 2008). Additionally, general operators for speciation and selection were also defined. In the present work a single operator was used for updating solutions, regardless of the considered models. This clearly illustrates the main advantage of *PSO* over genetic algorithms, namely generality and simplicity.

The main contribution of this work is experimental: we provide empirical evidence indicating that by using *PSO* we were able to perform intensive search over a huge space and succeeded in selecting competitive models without significantly overfitting. This is due to the way the search is guided in *PSO*: performing a broad search around promising solutions but not overdoing in terms of really fine optimization. This sort of search is known to help avoiding overfitting by *undercomputing* (Dietterich, 1995). Experimental results supported by some a posteriori analysis give evidence of the validity of our approach. The way we approached the model selection problem and the use of a stochastic-search strategy are also contributions. To the best of our knowledge there are no similar works that consider the *FMS* problem for classification tasks.

The rest of this paper is organized as follows. In the next section we describe the general *PSO* algorithm. In Section 3, we describe the application of *PSO* to *FMS*. Section 4 presents experimental results in benchmark data; comparing the performance of *PSO* to that of *PS* in *FMS* and analyzing the performance of *PSO* under different parameter settings; also, are described the results obtained in the framework of a model selection competition. In Section 5, we analyze mechanisms in *PSMS* that allow to select competitive models without overfitting the data. Finally, in Section 6, we present the conclusions and outline future research directions.

2. Particle Swarm Optimization (*PSO*)

PSO is a population-based search algorithm inspired by the behavior of biological communities that exhibit both individual and social behavior; examples of these communities are flocks of birds, schools of fishes and swarms of bees. Members of such societies share common goals (e.g., finding food) that are realized by exploring its environment while interacting among them. Proposed by Kennedy and Eberhart (1995), *PSO* has become an established optimization algorithm with applications ranging from neural network training (Kennedy and Eberhart, 1995; Salerno, 1997; Kennedy and Eberhart, 2001; Gudise and Venayagamoorthy, 2003; Engelbrecht, 2006) to control and engineering design (Hernández et al., 2004; Xiaohui et al., 2003; Yoshida et al., 2001; Robinson, 2004). The popularity of *PSO* is due in part to the simplicity of the algorithm (Kennedy and Eberhart, 1995; Reyes and Coello, 2006; Engelbrecht, 2006), but mainly to its effectiveness for producing good results at a very low computational cost (Gudise and Venayagamoorthy, 2003; Kennedy and Eberhart, 2001; Reyes and Coello, 2006). Like evolutionary algorithms, *PSO* is appropriate for problems with immense search spaces that present many local minima.

In *PSO* each solution to the problem at hand is called a particle. At each time t, each particle, i, has a position $\mathbf{x}_i^t =< x_{i,1}^t, x_{i,2}^t, \ldots, x_{i,d}^t >$ in the search space; where d is the dimensionality of the solutions. A set of particles $\mathbf{S} = \{\mathbf{x}_1^t, \mathbf{x}_2^t, \ldots, \mathbf{x}_m^t\}$ is called a swarm. Particles have an associated velocity value that they use for *flying* (exploring) through the search space. The velocity of particle i at time t is given by $\mathbf{v}_i^t =< v_{i,1}^t, v_{i,2}^t, \ldots, v_{i,d}^t >$, where $v_{i,k}^t$ is the velocity for dimension k of particle i at time t. Particles adjust their flight trajectories by using the following updating equations:

$$v_{i,j}^{t+1} = W \times v_{i,j}^t + c_1 \times r_1 \times (p_{i,j} - x_{i,j}^t) + c_2 \times r_2 \times (p_{g,j} - x_{i,j}^t), \quad (1)$$

$$x_{i,j}^{t+1} = x_{i,j}^t + v_{i,j}^{t+1} \quad (2)$$

where $p_{i,j}$ is the value in dimension j of the best solution found so far by particle i; $\mathbf{p}_i =< p_{i,1}, \ldots, p_{i,d} >$ is called personal best. $p_{g,j}$ is the value in dimension j of the best particle found so far in the swarm ($\mathbf{S}$); $\mathbf{p}_g =< p_{g,1}, \ldots, p_{g,d} >$ is considered the leader particle. Note that through

$\mathbf{p}_i$ and $\mathbf{p}_g$ each particle i takes into account individual (local) and social (global) information for updating its velocity and position. In that respect, $c_1, c_2 \in \mathbb{R}$ are constants weighting the influence of local and global best solutions, respectively. $r_1, r_2 \sim U[0,1]$ are values that introduce randomness into the search process. W is the so called inertia weight, whose goal is to control the impact of the past velocity of a particle over the current one, influencing the local and global exploration abilities of the algorithm. This is one of the most used improvements of *PSO* for enhancing the rate of convergence of the algorithm (Shi and Eberhart, 1998, 1999; van den Bergh, 2001). For this work we considered an adaptive inertia weight specified by a triplet $\mathbf{W} = (w_{start}, w_f, w_{end})$; where w_{start} and w_{end} are the initial and final values for W, respectively, and w_f indicates the fraction of iterations in which W is decreased. Under this setting W is decreased by $W = W - wdec$ from iteration $t = 1$ (where $W = w_{start}$) up to iteration $t = I \times w_f$ (after which $W = w_{end}$); where $w_{dec} = \frac{w_{start} - w_{end}}{I \times w_f}$ and I is the maximum number of iterations. This setting allows us to explore a large area at the start of the optimization, when W is large, and to slightly refine the search later by using a smaller inertia weight (Shi and Eberhart, 1998, 1999; van den Bergh, 2001).

An adaptive W can be likened to the temperature parameter in simulated annealing (Kirkpatrick et al., 1983); this is because, in essence, both parameters influence the global and local exploration abilities of their respective algorithms, although in different ways. A constant W is analogous to the momentum parameter p in gradient descend with momentum term (Qian, 1999), where weights are updated by considering both the current gradient and the weight change of the previous step (weighed by p). Interestingly, the inertia weight is also similar to the weight-decay constant (γ) used in machine learning to prevent overfitting. In neural networks the weights are decreased by $(1-\gamma)$ in each learning step, which is equivalent to add a penalty term into the error function that encourages the magnitude of the weights to decay towards zero (Bishop, 2006; Hastie et al., 2001); the latter penalizes complex models and can be used to obtain sparse solutions (Bishop, 2006).

The pseudo code of the *PSO* algorithm considered in this work is shown in Algorithm 1; default recommended values for the *FMS* problem are shown as well (these values are based on the analysis of Section 2.1 and experimental results from Section 4.2). The swarm is randomly initialized, considering restrictions on the values that each dimension can take. Next, the *goodness* of each particle is evaluated and $\mathbf{p}_g$, $\mathbf{p}_{1,\ldots,m}$ are initialized. Then, the iterative *PSO* process starts, in each iteration: *i*) the velocities and positions of each particle in every dimension are updated according to Equations (1) and (2); *ii*) the *goodness* of each particle is evaluated; *iii*) $\mathbf{p}_g$ and $\mathbf{p}_{1,\ldots,m}$ are updated, if needed; and *iv*) the inertia weight is decreased. This process is repeated until either a maximum number of iterations is reached or a minimum fitness value is obtained by a particle in the swarm (we used the first criterion for *FMS*); eventually, an (locally) optimal solution is found.

A fitness function is used to evaluate the aptitude (*goodness*) of candidate solutions. The definition of a specific fitness function depends on the problem at hand; in general it must reflect the proximity of the solutions to the optima. A fitness function $F : \Psi \rightarrow \mathbb{R}$, where Ψ is the space of particles positions, should return a scalar $f_{\mathbf{x}_i}$ for each particle position $\mathbf{x}_i$, indicating how far particle i is from the optimal solution to the problem at hand. For *FMS* the goal is to improve classification accuracy of full models. Therefore, any function F, which takes as input a model and returns an estimate of classification performance, is suitable (see Section 3.3).

Note that in Equation (1) every particle in the swarm knows the best position found so far by any other particle within the swarm, that is $\mathbf{p}_g$. Under this formulation a fully-connected swarm-topology is considered in which every member knows the leader particle. This topology has shown to converge faster than any other topology (Kennedy and Mendes, 2002; Reyes and Coello, 2006;

Algorithm 1 Particle swarm optimization.

```
Require: [Default recommended values for FMS]
– $\mathbf{c}_1, \mathbf{c}_2$: individual/social behavior weights; $[\mathbf{c}_1 = \mathbf{c}_2 = 2]$
– $\mathbf{m}$: swarm size; $[\mathbf{m} = 5]$
– $\mathbf{I}$: number of iterations; $[\mathbf{I} = 50]$
– $\mathbf{F}(\Psi \rightarrow \mathbb{R})$: fitness function; $[\mathbf{F}(\Psi \rightarrow \mathbb{R}) = 2-$fold $CV\ BER$ $]$
– $\mathbf{W}$: Inertia weight $\mathbf{W} = (1.2, 0.5, 0.4)$
Set decrement factor for $W$ $(w_{dec} = \frac{w_{start} - w_{end}}{I \times w_f})$
Initialize swarm $(\mathbf{S} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m\})$
Compute $f_{\mathbf{x}_{1,\ldots,m}} = F(\mathbf{x}_{1,\ldots,m})$ (Section 3.3)
Locate leader $(\mathbf{p}_g)$ and set personal bests $(\mathbf{p}_{1,\ldots,m} = \mathbf{x}_{1,\ldots,m})$
$t = 1$
while $t < I$ do
  for all $\mathbf{x}_i \in \mathbf{S}$ do
    Calculate velocity $\mathbf{v}_i$ for $\mathbf{x}_i$(Equation (1))
    Update position of $\mathbf{x}_i$ (Equation (2))
    Compute $f_{\mathbf{x}_i} = F(\mathbf{x}_i)$
    Update $\mathbf{p}_i$ (if necessary)
  end for
  Update $\mathbf{p}_g$ (if necessary)
  if $t < \lfloor I \times w_f \rfloor$ then
    $W = W - w_{dec}$
  end if
  $t^{++}$
end while
return $\mathbf{p}_g$
```

Kennedy and Eberhart, 2001; Engelbrecht, 2006). With this topology, however, the swarm is prone to converge to local minima. We tried to overcome this limitation by using the adaptive inertia weight, W.

2.1 PSO Parameters

Selecting the best parameters (W, c_1, c_2, m, I) for *PSO* is another model selection task. In the application of *PSO* for *FMS* we are dealing with a very complex problem lying in the third level of inference. Fortunately, several empirical and theoretical studies have been performed about the parameters of *PSO* from which useful information can be obtained (Shi and Eberhart, 1998, 1999; Kennedy and Mendes, 2002; Reyes and Coello, 2006; Ozcan and Mohan, 1998; Clerc and Kennedy, 2002; van den Bergh, 2001). In the rest of this section the *PSO* parameters are analyzed in order to select appropriate values for *FMS*. Later on, in Section 4.2, results of experiments with *PSO* for *FMS* under different parameter settings are presented.

We will start analyzing $c_1, c_2 \in \mathbb{R}$, the weighting factors for individual and social behavior. It has been shown that convergence is guaranteed[1] for *PSO* under certain values of c_1, c_2 (van den Bergh, 2001; Reyes and Coello, 2006; Ozcan and Mohan, 1998; Clerc and Kennedy, 2002). Most convergence studies have simplified the problem to a single one-dimensional particle, setting $\phi = c_1 \times r_1 + c_2 \times r_2$, and $\mathbf{p}_g$ and $\mathbf{p}_i$ constant (van den Bergh, 2001; Reyes and Coello, 2006; Ozcan and Mohan, 1998; Clerc and Kennedy, 2002). A complete study including the inertia weight is carried out by van den Bergh (2001). In agreement with this study the value $\phi < 3.8$ guarantees eventual convergence of the algorithm when the inertia weight W is close to 1.0. For the experiments in this work it was fixed $c_1 = c_2 = 2$, with these values we have $\phi < 3.8$ with high probability $P(\phi < 3.8) \approx 0.9$, given the uniformly distributed numbers r_1, r_2 (van den Bergh, 2001). The configuration $c_1 = c_2 = 2$ also has been proven, empirically, to be an effective choice for these parameters (Kennedy and Eberhart, 1995; Shi and Eberhart, 1998, 1999; Kennedy and Mendes, 2002).

Note, however, the restriction that W remains close to 1.0. In our experiments we considered a value of $W = 1.2$ at the beginning of the search, decreasing it during 50% (i.e., $w_f = 0.5$) of the *PSO* iterations up to the value $W = 0.4$. In consequence, it is possible that at the end of the search process the swarm may show divergent behavior. In practice, however, the latter configuration for W resulted very useful for *FMS*, see Section 4. This selection of W should not be surprising since even the configuration $\mathbf{W} = (1, 1, 0)$ has obtained better results than using a fixed W in empirical studies (Shi and Eberhart, 1998). Experimental results with different configurations of $\mathbf{W}$ for *FMS* give evidence that the configuration we selected can provide better models than using constant $\mathbf{W}$, see Section 4.2; although further experiments need to be performed in order to select the best $\mathbf{W}$ configuration for *FMS*.

With respect to m, the size of the swarm, experimental results suggest that the size of the population does not damage the performance of *PSO* (Shi and Eberhart, 1999), although slightly better results have been obtained with a large value of m, (Shi and Eberhart, 1999). Our experimental results in Section 4.2 confirm that the selection of m is not crucial, although (contrary to previous results) slightly better models are obtained by using a small swarm size. The latter is an important result for *FMS* because using a small number of particles reduces the number of fitness function evaluations, and in consequence the computational cost of the search.

Regarding the number of iterations, to the best of our knowledge, there is no work on the subject. This is mainly due to the fact that this issue depends mostly on the complexity of the problem at hand and therefore a general rule cannot be derived. For *FMS* the number of iterations should not be large to avoid oversearching (Dietterich, 1995). For most experiments in this paper we fixed $I = 100$. However, experimental results in Section 4.2 show that by running *PSO* for a smaller number of iterations is enough to obtain models of the same, and even better, generalization performance. This gives evidence that early stopping can be an useful mechanism to avoid overfitting, see Section 5.

3. Particle Swarm Model Selection

Since one of the strong advantages of *PSO* is its simplicity, its application to *FMS* is almost direct. Particle swarm full model selection (hereafter *PSMS*, that is the application of *PSO* to *FMS*) is

1. Note that in *PSO* we say that the swarm converges iff $\lim_{t \to \inf} \mathbf{p}_{gt} = \mathbf{p}$, where $\mathbf{p}$ is an arbitrary position in the search space and t indexes iterations of *PSO*. Since $\mathbf{p}$ refers to an arbitrary position, this definition does not mean either convergence to local or global optimum, but convergence to the global best position in the swarm (van den Bergh, 2001; Reyes and Coello, 2006; Engelbrecht, 2006).

ID	Object name	F	Hyperparameters	Description
1	*Ftest*	*FS*	$f_{max}, w_{min}, p_{val}, fdr_{max}$	Feature ranking according the F-statistic
2	*Ttest*	*FS*	$f_{max}, w_{min}, p_{val}, fdr_{max}$	Feature ranking according the T-statistic
3	*aucfs*	*FS*	$f_{max}, w_{min}, p_{val}, fdr_{max}$	Feature ranking according to the AUC criterion
4	*odds-ratio*	*FS*	$f_{max}, w_{min}, p_{val}, fdr_{max}$	Feature ranking according to the odds ratio statistic
5	*relief*	*FS*	$f_{max}, w_{min}, k_{num}$	Relief ranking criterion
6	*rffs*	*FS*	f_{max}, w_{min}	Random forest used as feature selection filter
7	*svcrfe*	*FS*	f_{max}	Recursive feature elimination filter using svc
8	*Pearson*	*FS*	$f_{max}, w_{min}, p_{val}, fdr_{max}$	Feature ranking according to the Pearson correlation coef.
9	*ZFilter*	*FS*	f_{max}, w_{min}	Feature ranking according to a heuristic filter
10	*gs*	*FS*	f_{max}	Forward feature selection with Gram-Schmidt orth.
11	*s2n*	*FS*	f_{max}, w_{min}	Signal-to-noise ratio for feature ranking
12	$pc-extract$	*FS*	f_{max}	Extraction of features with *PCA*
1	*normalize*	*Pre*	center	Normalization of the lines of the data matrix
2	*standardize*	*Pre*	center	Standardization of the features
3	$shift-scale$	*Pre*	$take_{log}$	Shifts and scale data
1	*bias*	*Pos*	none	Finds the best threshold for the output of the classifiers

Table 1: Feature selection (*FS*), preprocessing (*Pre*) and postprocessing (*Pos*) objects available in *CLOP*. A brief description of the methods and their hyperparameters is presented.

described by the pseudocode in Algorithm 1, in this section are presented additional details about *PSMS*: first we describe the pool of methods considered in this work; then, we describe the representation of particles and the fitness function used; finally, we briefly discuss complexity issues. The code of *PSMS* is publicly available from the following website `http://ccc.inaoep.mx/~hugojair/code/psms/`.

3.1 The Challenge Learning Object Package

In order to implement *PSMS* we need to define the models search space. For this purpose we consider the set of methods in a machine learning toolbox from which full models can be generated. Currently, there are several machine learning toolboxes, some of them publicly available (Franc and Hlavac, 2004; van der Heijden et al., 2004; Wichard and Merkwirth, 2007; Witten and Frank, 2005; Saffari and Guyon, 2006; Weston et al., 2005); even there is a track of this journal (*JMLR*) dedicated to machine learning software. This is due to the increasing interest from the machine learning community in the dissemination and popularization of this research field (Sonnenburg, 2006). The *Challenge Learning Object Package*[2] (*CLOP*) is one of such development kits distributed under the GNU license (Saffari and Guyon, 2006; Guyon et al., 2006c, 2007, 2008). *CLOP* is a *Matlab*^{R} toolbox with implementations of feature-variable selection methods and machine learning algorithms (*CLOP* also includes the *PSMS* implementation used in this work). The list of available preprocessing, feature selection and postprocessing methods in the *CLOP* toolbox is shown in Table 1; a description of the learning algorithms available in *CLOP* is presented in Table 2. One should note that this version of *CLOP* includes the methods that best performed in a model selection competition (Guyon et al., 2008; Cawley and Talbot, 2007a; Lutz, 2006).

2. See `http://clopinet.com/CLOP/`.

ID	Object name	Hyperparameters	Description
1	*zarbi*	none	Linear classifier
2	*naive*	none	Naïve Bayes
3	*klogistic*	none	Kernel logistic regression
4	*gkridke*	none	Generalized kridge (VLOO)
5	*logitboost*	units number, shrinkage, depth	Boosting with trees (R)
6	*neural*	units number, shrinkage, maxiter, balance	Neural network (Netlab)
7	*svc*	shrinkage, kernel parameters (coef0, degree, gamma)	SVM classifier
8	*kridge*	shrinkage, kernel parameters (coef0, degree, gamma)	Kernel ridge regression
9	*rf*	units number, balance, mtry	Random forest (R)
10	*lssvm*	shrinkage, kernel parameters (coef0, degree, gamma), balance	Kernel ridge regression

Table 2: Available learning objects with their respective hyperparameters in the CLOP package.

In consequence, for *PSMS* the pool[3] of methods to select from are those methods described in Tables 1 and 2. In *CLOP* a typical model consists of the *chain*, which is a grouping object that allows us to perform serial concatenation of different methods. A chain may include combinations of (several/none) feature selection algorithm(s) followed by (several/none) preprocessing method(s), in turn followed by a learning algorithm and finally (several/none) postprocessing algorithm(s). For example, the model given by:
chain$\{$*gs(*$fmax = 8$*),standardize(center=1),neural(units=10,s=0.5,balance=1,iter=10)*$\}$
uses *gs* for feature selection, *standardization* of data and a balanced *neural network* classifier with 10 hidden units, learning rate of 0.5, and trained for 10 iterations. In this work chain objects that include methods for preprocessing, feature selection and classification are considered full-models. Specifically, we consider models with at most one feature selection method, but allowing to perform preprocessing before feature selection and viceversa, see Section 3.2. The bias method was used as postprocessing in every model tested to set an optimal threshold in the output of the models in order to minimize their error. The search space in *FMS* is given by all the possible combinations of methods and hyperparameters; an infinite search space due to the real valued parameters.

3.2 Representation

In *PSO* each potential solution to the problem at hand is considered a particle. Particles are represented by their position, which is nothing but a $d-$dimensional numerical vector (d being the dimensionality of the solution). In *FMS* potential solutions are full-models, in consequence, for *PSMS* we need a way to codify a full-model by using a vector of numbers. For this purpose we propose the representation described in Equation (3), the dependence on time (t) is omitted for clarity.

$$\mathbf{x}_i = < x_{i,pre}, y_{i,1\dots N_{pre}}, x_{i,fs}, y_{i,1,\dots N_{fs}}, x_{i,sel}, x_{i,class}, y_{i,1,\dots N_{class}} > \tag{3}$$

where $x_{i,pre} \in \{1,\dots,8\}$ represents a combination of preprocessing methods. Each combination is represented by a binary vector of size 3 (i.e., the number of preprocessing methods considered), there are $2^3 = 8$ possible combinations. Each element of the binary vector represents a single preprocessing method; if the value of the k^{th} element is set to 1 then the preprocessing method with *ID* = k is used (see Table 1). For example, the first combination $< 0,0,0 >$ means *no preprocessing*; while the seventh $< 1,1,0 >$ means that this model ($\mathbf{x}_i$) uses *normalization* and *standardization* as

3. Notice that the *CLOP* package includes also the spider package (Weston et al., 2005) which in turn includes other implementations of learning algorithms and preprocessing methods. However, in this work we only used *CLOP* objects.

preprocessing. $y_{i,1\dots N_{pre}}$ codify the hyperparameters for the selected combination of preprocessing methods, $N_{pre} = 3$ because each preprocessing method has a single hyperparameter; note that the order of the preprocessing methods is fixed (i.e., *standardization* can never be performed before *normalization*), in the future we will relax this constraint. $x_{i,fs} \in \{0, \dots, 12\}$ represents the *ID* of the feature selection method used by the model (see Table 1), and $y_{i,1\dots N_{fs}}$ its respective hyperparameters; N_{fs} is set to the maximum number of hyperparameters that any feature selection method can take. $x_{i,sel}$ is a binary variable that indicates whether preprocessing should be performed before feature selection or viceversa. $x_{i,class} \in \{1, \dots, 10\}$ represents the classifier selected and $y_{i,1,\dots N_{class}}$ its respective hyperparameters; N_{class} is the maximum number of hyperparameters that a classifier can take. This numerical codification must be decoded and used with the *chain* grouping object for obtaining a full-model from a particle position $\mathbf{x}_i$. Note that the dimensionality of each particle is $d = 1 + N_{pre} + 1 + N_{fs} + 1 + 1 + N_{class} = 16$.

3.3 Fitness Function

In *FMS* it is of interest to select models that minimize classification errors on unseen data (i.e., maximizing generalization performance). Therefore, the fitness function (F) should relate a model with an estimate of its classification performance in unseen data. The simplicity of *PSMS* allows us to use any classification performance measure as F, because the method does not require derivatives. Thus, valid options for F include mean absolute error, balanced error rate, squared root error, recall, precision, area under the *ROC* curve, etcetera. For this work it was used the balanced error rate (*BER*) as F. *BER* takes into account misclassification rates in both classes, which prevents *PSMS* of selecting biased models (favoring the majority class) in imbalanced data sets. Furthermore, *BER* has been used in machine learning challenges as leading error measure for ranking participants (Guyon et al., 2007, 2008). The *BER* of model ψ is the average of the misclassifications obtained by ψ over the classes in a data set, as described in Equation (4):

$$BER(\psi) = \frac{E_+ + E_-}{2} \tag{4}$$

where E_+ and E_- are the misclassifications rates for the positive and negative classes, respectively.

The selection of the fitness function is a crucial aspect in *PSO*. However, for *PSMS*, the critical part lies in the way an estimate of generalization performance of a model (given F and training data) is obtained, and not in the fitness function itself. This is the main challenge of model selection, since error estimates using training data are very optimistic about the behavior of models on unseen data, this phenomenon is known as overfitting (Bishop, 2006; Hastie et al., 2001; Nelles, 2001). In order to overcome overfitting the *BER* was calculated using a $k-$fold cross validation *(k-fold CV)* approach (Note that the *BER* is still obtained from training data). This is the only explicit mechanism of *PSMS* to avoid overfitting. *k-fold CV* is the most used hold-out approach for model selection (Nelles, 2001; Hastie et al., 2001; Cawley and Talbot, 2007b). Using a high value for k, say $k = N$ where N is the training set size (i.e., *leave one out - CV*), the error estimate is almost unbiased, but can have high variance because the N training sets are very similar to each other (Nelles, 2001; Cawley and Talbot, 2007b); furthermore, computation time could be a serious problem (Hastie et al., 2001; Nelles, 2001). With a low value for k, the estimate can have low variance but the bias could be a problem. The selection of an optimal k is still an open problem in the model selection field. For this work were performed experiments with $k \in \{2, 5, 10\}$, but no statistically-significant difference, among these values, was found, see Section 4.2.

3.4 Computational Complexity

As we have seen, the search space in the *FMS* problem is composed by all the possible models that can be built given the considered methods and their hyperparameters. This is an infinite search space even with the restriction imposed to the values that models can take; this is the main drawback of *FMS*. However, the use of particle swarm optimization (PSO) allows us to harness the complexity of this problem. Most algorithms used for *FMS* cannot handle very big search spaces. But PSO is well suited to large search spaces: it converges fast and has a manageable computational complexity (Kennedy and Eberhart, 2001; Reyes and Coello, 2006). As we can see from Algorithm 1, *PSO* is very simple and does not involve expensive operations.

The computational expensiveness of *PSMS* is due to the fitness function we used. For each selected model the fitness function should train and evaluate such model $k-$times. Depending on the model complexity this process can be performed on linear, quadratic or higher order times. Clearly, computing the fitness function using the entire training set, as opposed to *k-fold CV*, could reduce *PSMS* complexity, although we could easily overfit the data. For a single run of *PSMS* the fitness function should be evaluated $\rho = m \times (I+1)$ times, with m being the swarm size and I the number of iterations. Suppose the computational complexity of model λ is bounded by λ_O then the computational complexity of *PSMS* will be bounded by $\rho \times k \times \lambda_O$. Because λ_O is related to the computational complexity of model λ (which depends on the size and dimensionality of the data set) this value may vary dramatically. For instance, computing the fitness function for a naïve Bayes model in a high dimensional data set takes around two seconds[4], whereas computing the same fitness function for the same data set could take several minutes if a support vector classifier is used.

In order to reduce the computational cost of *PSMS* we could introduce a complexity penalty-term into the fitness function (this is current work). A simpler alternative solution is calculating the fitness function for each model using only a small subset of the available data; randomly selected and different each time. This approach can also be useful for avoiding local minima. The subsampling heuristic was used for high-dimensional data sets and for data sets with a large number of examples. Experimental results show an important reduction of processing time, without a significant loss of accuracy, see Section 4.3. We emphasize that complexity is due to the nature of the *FMS* problem. With this approach, however, users will be able to obtain models for their data without requiring knowledge on the data or on machine learning techniques.

4. Experimental Results

In this section results of experiments with *PSMS* using benchmark data from two different sources are presented. First, we present results on a suite of benchmark machine learning data sets[5] used by several authors (Mika et al., 2000; Rätsch et al., 2001; Cawley and Talbot, 2007b); such data sets are described in Table 3, ten replications (i.e., random splits of training and testing data) of each data set were considered. These data sets were used to compare *PSO* to *PS* in the *FMS* problem (Section 4.1) and to study the performance of *PSMS* under different settings (Section 4.2). Next we applied *PSMS* to the data sets used in a model selection competition (Section 4.3). The goal of the latter experiments is to compare the performance of *PSMS* against other model selection strategies.

4. Most of the experiments were carried out on a workstation with *Pentium*^{TM} 4 processor at 2.5 GHz, and 1 gigabyte in RAM.
5. These data sets are available from `http://ida.first.fraunhofer.de/projects/bench/benchmarks.htm`.

ID	**Data set**	**Training Patterns**	**Testing Patterns**	**Input Features**
1	Breast cancer	200	77	9
2	Diabetes	468	300	8
3	Flare solar	666	400	9
4	German	700	300	20
5	Heart	170	100	13
6	Image	1300	1010	20
7	Splice	1000	2175	60
8	Thyroid	140	75	5
9	Titanic	150	2051	3

Table 3: Data sets used in the comparison of *PSO* and *PS*, ten replications (i.e., random splits of training and testing sets) for each data set were considered.

4.1 A Comparison of *PSO* and *PS*

In the first set of experiments we compared the performance of *PSO* to that of another search strategy in the *FMS* problem. The goal was to evaluate the advantages of using the swarm strategy over another intensive search method. Among the available search techniques we selected *PS* because of its simplicity and proved performance in model selection (Dennis and Torczon, 1994; Momma and Bennett, 2002; Bi et al., 2003). *PS* is a direct search method that samples points in the search space in a fixed pattern about the best solution found so far (the center of the pattern). Fitness values are calculated for the sampled points trying to find a minimizer; if a new minimum is find then the center of the pattern is changed, otherwise the search step is reduced by half; this process is iterated until a stop criteria is met. The considered *PS* algorithm described in Algorithm 2 is an adaptation of that proposed by Momma et al. for hyperparameter optimization of support vector regression (Momma and Bennett, 2002; Bi et al., 2003).

The input to *PS* is the pattern P and the search step Δ. Intuitively, P specifies the direction of the neighboring solutions that will be explored; while Δ specifies the distance to such neighboring solutions. There are several ways to generate P; for this work we used the nearest neighbor sampling pattern (Momma and Bennett, 2002). Such pattern is given numerically by $P = [\mathbf{I}_d \; -\mathbf{I}_d \; \mathbf{0}_{1\times d}^T]$, where $\mathbf{I}_d$ is the identity matrix of size d; $\mathbf{0}_{1\times d}$ is a vector of size $1_\times d$ with all zero entries; d is the dimensionality of the problem. Δ, a vector of size $1 \times d$, is the search step by which the space of solutions is explored. We defined $\Delta = \frac{\mathbf{q}_{maxvals} - \mathbf{q}_{minvals}}{2}$, where $\mathbf{q}_{maxvals}$ and $\mathbf{q}_{minvals}$ are the maximum and minimum values that the solutions can take, respectively. Each iteration of *PS* involves the evaluation of $N_c - 1$ solutions (where N_c is the number of columns of P). Solutions are updated by adding s_j ($j \in 1,\ldots,N_c - 1$) to the current-best solution $\mathbf{q}_g$; where each s_j is obtained by multiplying (element-by-element) the search step vector Δ and the j^{th} column of P. $\mathbf{q}_g$ is replaced by a new solution $\mathbf{q}_j$ only if $f_{\mathbf{q}_j} < f_{min} = f_{\mathbf{q}_g}$. The output of *PS* is $\mathbf{q}_g$, the solution with the lowest fitness value; we encourage the reader to follow the references for further details (Momma and Bennett, 2002; Dennis and Torczon, 1994).

For applying *PS* to the *FMS* problem (hereafter *PATSMS*) the solution $\mathbf{q}_g$ was initialized in the same way that each particle in the swarm was initialized for *PSMS* (i.e., randomly). The same

Algorithm 2 Pattern search. p_i is the i^{th} column of P, N_c the number of columns in P.

```
**Require:**
– $I_{ps}$: number of iterations
– $\mathbf{F}(\Psi \rightarrow \mathbb{R})$: fitness function
– $\mathbf{P}$: pattern
– $\Delta$: search step
Initialize solution $\mathbf{q}_i$ $(i = 1)$
Compute $f_{\mathbf{q}_i} = F(\mathbf{q}_i)$ (Section 3.3)
Set $\mathbf{q}_g = \mathbf{q}_i$; $f_{min} = f_{\mathbf{q}_i}$
while $i < I_{ps}$ do
  for all $p_j \in P_{1,\dots,N_c-1}$ do
    $s_j = \Delta . p_j$
    $\mathbf{q}_j = \mathbf{q}_g + s_j$
    Compute $f_{\mathbf{q}_j} = F(\mathbf{q}_j)$
    if $f_{\mathbf{q}_j} < f_{min}$ then
      Update $\mathbf{q}_g$ $(\mathbf{q}_g = \mathbf{q}_j, f_{min} = f_{\mathbf{q}_j})$
    end if
    $i^{++}$
  end for
  $\Delta = \Delta/2$
end while
return $\mathbf{q}_g$
```

representation, fitness function and restrictions on the values that each dimension can take were considered for both methods, see Section 3. Under these settings *PS* is a very competitive baseline for *PSO*.

In each experiment described below, we let *PS* and *PSO* perform the same number of fitness function evaluations, using exactly the same settings and data sets, guaranteeing a fair comparison. Since both methods use the same fitness function and perform the same number of fitness function evaluations, the difference in performance indicates the gain we have by guiding the search according to Equations (1) and (2). As recommended by Demsar, we used the Wilcoxon signed-rank test for the comparison of the resultant models (Demsar, 2006). In the following we will refer to this statistical test with 95% of confidence when mentioning statistical significance.

We compared the *FMS* ability of *PSMS* and *PATSMS* by evaluating the accuracy of the selected models at the end of the search process; that is, we compared the performance of models $\mathbf{p}_g$ and $\mathbf{q}_g$ in Algorithms 1 and 2, respectively. We also compared the performance of the solutions tried through the search. For each trial we fixed the number of iterations for *PSMS* to $I = 100$ with a swarm size of $m = 10$. In consequence, both search strategies performed $m \times (I+1) = 1010$ evaluations of the fitness function in each run. Because more than $180,000$ models were tested we used $2-$fold *CV* for computing the fitness function. In each trial the training set was used for *FMS* and the resultant model was evaluated in the test set. This process was repeated for each replica of each data set described in Table 3. Averaged results of this experiment are shown in Table 4. We show the average *CV* error obtained through the search process (*CV-BER*) and the error obtained by the selected model, at the end of the search, in the test set (*test-BER*).

ID	Data set	PATSMS test-BER	PSMS test-BER	PATSMS CV-BER	PSMS CV-BER
1	Breast-cancer	36.98$\pm$0.08	**33.59$\pm$0.12**	**32.64$\pm$0.06**	32.96$\pm$0.01
2	Diabetes	26.07$\pm$0.03	**25.37$\pm$0.02**	**25.39$\pm$0.02**	26.48$\pm$0.05
3	Flare-solar	32.87$\pm$0.02	**32.65$\pm$0.01**	**32.69$\pm$0.01**	33.13$\pm$0.01
4	German	28.65$\pm$0.02	**28.28$\pm$0.02**	**31.00$\pm$0.00**	31.02$\pm$0.00
5	Heart	19.50$\pm$0.19	**17.35$\pm$0.06**	**16.96$\pm$0.07**	19.93$\pm$0.03
6	Image	3.58$\pm$0.01	**2.50$\pm$0.01**	**11.54$\pm$0.10**	15.88$\pm$0.04
7	Splice	13.94$\pm$0.99	**9.46$\pm$0.25**	**18.01$\pm$0.05**	19.15$\pm$0.07
8	Thyroid	10.84$\pm$0.39	**5.98$\pm$0.06**	**11.15$\pm$0.20**	15.49$\pm$0.12
9	Titanic	29.94$\pm$0.00	**29.60$\pm$0.00**	**27.19$\pm$0.13**	27.32$\pm$0.13

Table 4: Average and variance of *test-BER* and *CV-BER* obtained by models selected with *PSMS* and *PATSMS*, the best results are shown in **bold**. *test-BER* is the *BER* obtained by the selected model using the test set (averaged over 10 replications of each data set). *CV-BER* is the average of *CV BER* obtained by each of the candidate solutions through the search process (averaging performed over all particles and iterations and over 10 replications of each data set) .

The performance of both search strategies is similar. *PSMS* outperformed *PATSMS* through all of the data sets at the end of the search process (Columns 3 and 4 in Table 4). The *test-BER* differences are statistically significant for all but the *flare-solar* and *german* data sets. In the latter data sets the hypothesis that models selected with *PATSMS* and *PSMS* perform equally cannot be rejected. However, note that for these data sets the models selected with *PSMS* outperformed those selected with *PATSMS* in 6 out of 10 replications. Globally, we noticed that from the 90 trials ($9-$data sets, $\times$ $10-$replications for each, see Table 3), 68.9% of the models selected with *PSMS* outperformed those obtained with *PATSMS*. While only in 22.2% of the runs *PATSMS* outperformed *PSMS*, in the rest both methods tied in performance. A statistical test over these 90 results was performed and a statistically-significant difference, favoring *PSMS*, was found. Despite *PATSMS* being a strong baseline, these results give evidence that *PSMS* outperforms *PATSMS* at the end of the search process.

Columns five and six in Table 4 show the average *BER* obtained with each strategy through the 1010 evaluations for each data set (averaged over replications). *CV-BER* reflects the behavior of the search strategies through the search process. *PATSMS* slightly outperformed *PSMS* in this aspect, though the difference is statistically significant only for the *Heart*, *image* and *thyroid* data sets. The slight superior performance of *PATSMS* in *CV-BER* is due to the pattern we used and the search procedure itself. *PATSMS* performs a finer grained search over a restricted search area around the initial solution $\mathbf{q}_g$. The latter results in a lower *CV-BER* because *PATSMS* always moves the pattern towards the local minimum nearest to the initial solution, $\mathbf{q}_g$. *PSMS,* on the other hand, explores a much larger search space because the search is not only guided by the best solution found so far ($\mathbf{p}_g$), but also by the individual best solutions of each particle ($\mathbf{p}_{1,\dots,m}$). The latter produces a higher *CV-BER* in average because, even when the *CV-BER* of the final models is low, many models of varied *CV-BER* performance are tried through the search.

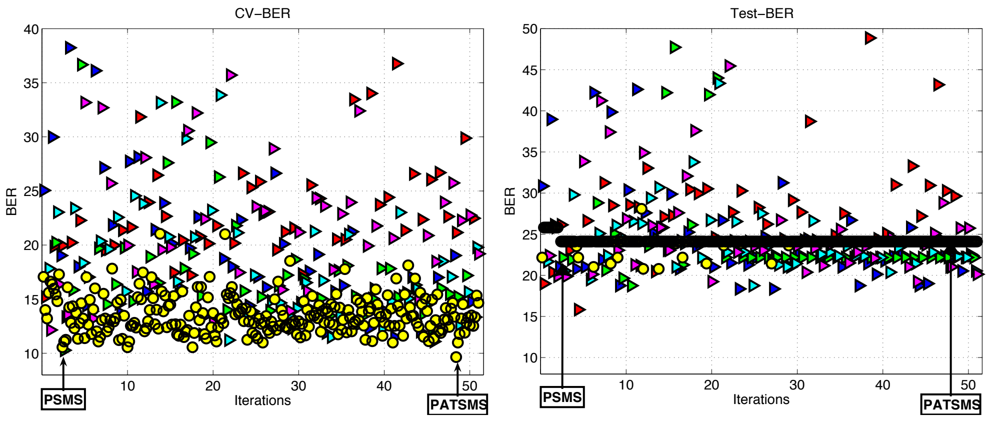


Figure 1: Performance of *PATSMS* (circles) and *PSMS* (triangles) as a function of number of iterations for an experiment with one replication of the *Heart* data set. For *PSMS* we show the performance of each particle with a different color. Left: Cross-validation Balanced Error Rate (*BER*). Right: Test set *BER*. In both plots we indicate with an arrow the model selected by each search strategy.

In Figure 1 we show the performance of the solutions tried through the search by each method for a single replication of the *Heart* data set (for clarity in the plots we used $m = 5$ and $I = 50$ for this experiment). We show the *CV* and test *BER* for every model tried through the search. It can be seen that the *CV-BER* of *PATSMS* is lower than that of *PSMS*, showing an apparent better convergence behavior (left plot in Figure 1). However, by looking the test *BER* of the models tried, it becomes evident that *PATSMS* is trapped into a local minimum since the very first iterations (right plot in Figure 1). *PSMS*, on the other hand, obtains a higher *CV-BER* through the search, though it is less prone to follow a local minimum. This is because with *PSMS* the search is guided by one global and m local solutions, which prevents *PSMS* from performing a pure local search; the latter in turn prevents *PSMS* of overfitting the data. This result gives evidence of the better convergence behavior of *PSMS*.

The model selected with *PSMS* obtained a lower *test-BER* than that selected with *PATSMS* (see right plot in Figure 1). In fact all of the solutions in the final swarm outperformed the solution obtained with *PATSMS* in *test-BER*. With *PSMS* the model of lowest test *BER* was obtained after 3 iterations of *PSMS*, giving evidence of the fast convergence of *PSO*. One should note that the *test-BER* of the worst *PSMS* solutions is higher than that of the best *PATSMS* solution. However, near the end of the search, the possibilities that *PSMS* can select a worse solution than *PATSMS* are small.

We have seen that *PATSMS* is prone to get trapped into a local minimum because it performs a fine grained search over a small area. This causes that only a few methods are considered through the search with this method. *PSMS*, on the other hand, tries a wide variety of classifiers and feature selection methods, exploring a larger search space and being able to escape from local minima. In order to analyze the diversity of the methods considered by *PSMS*, in Figure 2 we show the

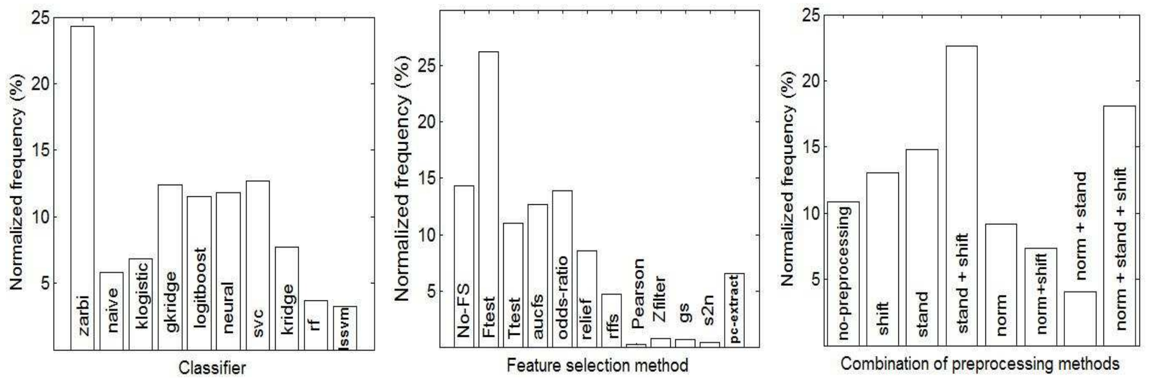


Figure 2: Normalized frequency of classifiers (left), feature selection method (middle) and combination of preprocessing methods (right) preferred by *PSMS* through the search process. In the right plot, the abbreviations *shift*, *stand* and *norm* stand for *shift-scale*, *standardization* and *normalization*, respectively. Results are normalized over the 90,900 models tried for obtaining the results from Table 4.

normalized frequency of methods preferred by *PSMS*[6] through the search. The results shown in this figure are normalized over the 90,900 models tried for obtaining the results from Table 4. From this figure, we can see that most of the classifiers and feature selection methods were considered for creating solutions. No classifier, feature selection method or combination of preprocessing methods was used for more than 27% of the models. This reflects the fact that different methods are required for different data sets and that some are equivalent. There were, however, some methods that were slightly more used than others.

The preferred classifier was the *zarbi CLOP*-object, that was used for about 23% of the models. This is a surprising result because *zarbi* is a very simple linear classifier that separates the classes by using information from the mean and variance of training examples (Golub et al., 1999). However, in 97.35% of the times that *zarbi* was used it was combined with a feature selection method. *Gkridge, svc, neural* and *logitboost* were all equally selected after *zarbi*. *Ftest* was the most used feature selection method, though most of the feature selection strategies were used. Note that *Pearson, Zfilter, gs* and *s2n* were considered only for a small number of models. The combination *standardize* + *shift-scale* was mostly used for preprocessing, although the combination *normalize* + *standardize* + *shift-scale* was also highly used. Interestingly, in 70.1% of the time preprocessing was performed before feature selection. These plots illustrate the diversity of classifiers considered by *PSMS* through the search process, showing that *PSMS* is not biased towards models that tend to perform very well individually (e.g., *logitboost, rf* or *gkridge*). Instead, *PSMS* attempts to find the best full model for each data set.

6. We do not show those preferred by *PATSMS* because the methods selected with this strategy are, most of the times, those considered in the initial solution. In our experiments we found that for each replication *PATSMS* used the same classifier and feature selection method for about 95% of the search iterations. The selection of these methods depended on the initial solution instead of the way the search space is explored or the individual performance of the methods. Therefore, no useful information can be obtained about why some methods are preferred over others, even when in average (over the 90,900 solutions tried) the histograms may look similar to those shown in Figure 2.

k	CV-BER	test-BER	test-BER*	Time
2	25.36$\pm$0.56	22.99$\pm$1.18	25.79$\pm$0.59	4166.85
5	25.18$\pm$0.59	20.50$\pm$1.53	26.30$\pm$0.46	6062.56
10	24.54$\pm$0.63	20.82$\pm$1.74	26.03$\pm$0.51	7737.11

Table 5: Average *CV-BER* (average CV result over all particles and iterations), *test-BER* (test result corresponding to the best CV result), *test-BER** (average test result over all particles and iterations) and processing time (in seconds) for different values of k in the $k-$fold *CV*. Results are averaged over a single replication of each data set described in Table 3.

4.2 Parameter Selection for *PSMS*

In this section we analyze the performance of *PSMS* under different settings. The goal is to identify mechanisms in *PSMS* that allow us obtaining competitive models and, to some extent, avoiding overfitting. For the experiments described in this section we consider a single replication for each data set. As before, we show the average *CV* error of the solutions considered during the search as well as the error of the selected model in the test set. We also show the average test set error of all solutions tried through the search *test-BER** (averaging performed over all particles and all iterations), providing information about the generalization performance of the models considered throughout the search.

4.2.1 VALUE OF K IN K-FOLD CV

First we analyze the behavior of *PSMS* for different values of k in the *CV* for computing the fitness function. We consider the values $k = [2,5,10]$. Average results of this experiment are shown in Table 5.

From this table, we can see that the performance is similar for the different values of k considered. The best results at the end of the search are obtained with $k = 5$ (column 3 in Table 5), while the best generalization performance is obtained with $k = 2$ (column 4 in Table 5). However, these differences are not statistically significant. Therefore, the null hypothesis that models selected with $k = [2,5,10]$ perform equally in *test-BER* and *test-BER** cannot be rejected. The latter is an important result because by using $k = 2$ the processing time of *PSMS* is considerably reduced. Note that for models of quadratic (or higher order) complexity, computing the fitness function with $2-$fold *CV* is even more efficient than computing the fitness function using the full training set. It is not surprising that processing time increases as we increase k (column 5 in Table 5). Although it is worth mentioning that the variance in processing time was very large (e.g., for $k = 2$ it took 7 minutes applying *PSMS* to the *titanic* data set and about five hours for the *image* data set).

4.2.2 NUMBER I OF ITERATIONS

Next we performed experiments varying the number of iterations (I), using $2-fold$ *CV* for computing the fitness function and a swarm size of $m = 10$. We considered the values $I = [10,25,50,100]$. Averaged results for this experiment are shown in Table 6 (rows 2 - 5). As we can see the best results were obtained by running *PSMS* for 50 iterations. Interestingly, models selected by running *PSMS* for 10 iterations outperformed those selected after 100 iterations in *test-BER*, though the difference

Setting	CV-BER	test-BER	test-BER*
I=10	25.33$\pm$0.33	22.17$\pm$1.81	27.64$\pm$0.52
I=25	25.29$\pm$0.33	21.88$\pm$1.68	27.59$\pm$0.49
I=50	**24.02$\pm$0.38**	**21.12$\pm$1.74**	**26.72$\pm$0.65**
I=100	24.57$\pm$0.37	22.81$\pm$1.44	27.27$\pm$0.48
m=5	**24.27$\pm$0.49**	**20.81$\pm$1.50**	**25.01$\pm$0.85**
m=10	25.07$\pm$0.34	21.64$\pm$2.04	25.99$\pm$0.74
m=20	25.09$\pm$0.34	21.76$\pm$1.84	26.00$\pm$0.64
m=40	24.82$\pm$0.43	21.45$\pm$2.13	25.96$\pm$0.78
m=50	25.32$\pm$0.44	22.54$\pm$1.65	26.11$\pm$0.77
W=(0,0,0)	**23.86$\pm$0.71**	20.40$\pm$1.71	**22.46$\pm$1.32**
W=(1.2,0.5,0.5)	24.22$\pm$0.76	**19.41$\pm$1.37**	23.38$\pm$1.34
W=(1,1,1)	27.62$\pm$0.30	21.88$\pm$1.68	27.13$\pm$0.53

Table 6: Average *CV-BER*, *test-BER* and *test-BER** for different settings of m, I and $\mathbf{W}$. The best results are shown in **bold**. Results are averaged over a single replication of each data set described in Table 3.

was not statistically significant. The difference in performance between models selected after 50 and 100 iterations was statistically significant. This result shows the fast convergence property of *PSMS* and that early stopping could be an useful mechanism to avoid overfitting, see Section 5.

4.2.3 SWARM SIZE M

In the next experiment we fixed the number of iterations to $I = 50$ and varied the swarm size as follows, $m = [5, 10, 20, 40, 50]$. Results of this experiment are shown in rows 6-10 in Table 6. This time the best result was obtained by using a swarm size of 5, however there is a statistically-significant difference only between $m = 5$ and $m = 50$. Therefore, models of comparable performance can be obtained by using $m = [5, 10, 20, 40]$. This is another interesting result because using a small swarm size reduces the number of fitness function evaluations for *PSMS* and therefore makes it more practical. An interesting result is that by using 50 iterations with any swarm size the *test-BER* is very close to the *CV-BER* estimate. Again, this provides evidence that early stopping can improve the average generalization performance of the models.

4.2.4 INERTIA WEIGHT W

We also performed experiments with different configurations for $\mathbf{W}$, the adaptive inertia weight. Each configuration is defined by a triplet $\mathbf{W} = (w_{start}, w_f, w_{end})$, whose elements indicate the starting value for W, the proportion of iterations to vary it and its final value, respectively, see Section 2. Three configurations were tried with the goal of evaluating the advantages of using an adaptive inertia weight instead of constant values. Results of this experiment are shown in rows 11-13 in Table 6. It can be seen that the best results in *CV-BER* and *Test-BER** were obtained with $\mathbf{W} = (0, 0, 0)$; the differences with the other results are statistically-significant. Under this configuration *PSMS* is not taking into account information from past velocities for updating solutions; which causes *PSMS* to converge quickly into a local minimum and refining the search around this point.

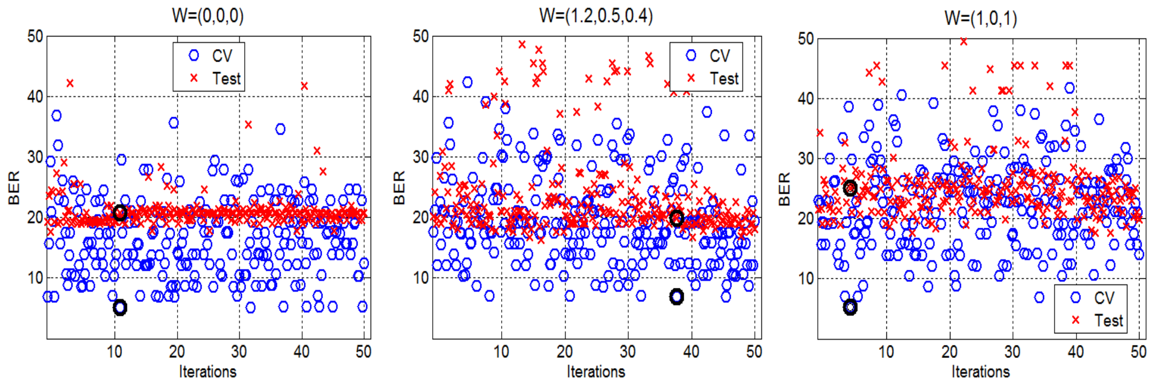


Figure 3: Algorithm performance as a function of number of iterations for different configurations of $\mathbf{W}$. *CV-BER* (circles) and *test-BER** (crosses) for the *Heart* data set. We are displaying the test and CV BER values for each particle at every time step. Since each time step involves m=5 particles, then for each iteration are displayed m=5 crosses and m=5 circles. We consider the following configurations: $\mathbf{W} = (0,0,0)$ (left), $\mathbf{W} = (1.2,0.5,0.4)$ (middle) and $\mathbf{W} = (1,0,1)$ (right). The *CV-BER* and *test-BER* of the best solution found by *PSMS* are enclosed within a bold circle.

The best result at the end of the search (column 3 in Table 6) was obtained with $\mathbf{W} = (1.2,0.5,0.4)$, the difference with the other results is statistically-significant. Under this configuration both global and local search is performed during the *PSMS* iterations; which caused higher *CV-BER* and *test-BER** than that of the first configuration, however, the generalization performance of the final model was better. The configuration $\mathbf{W} = (1,0,1)$ obtained the worst results in all of the measures; this is because under this configuration the search is never refined, since *PSMS* always takes into account the past velocity for updating solutions. The latter configuration could be a better choice for *FMS* because this way *PSMS* does not over-search around any solution; however, local search is also an important component of any search algorithm. In Figure 3 we show the *CV* and test *BER* of solutions tried during the search for the *Heart* data set under the different configurations tried. From this figure we can appreciate the fact that using constant values for W results in more local (when $\mathbf{W} = (0,0,0)$) or global search (when $\mathbf{W} = (1,0,1)$). An adaptive inertia weight, on the other hand, aims to control the tradeoff between global and local search, which results in a model with lower variance for this example. Therefore, an adaptive inertia weight seems to be a better option for *FMS*; this is because it prevents, to some extend, *PSMS* to overfit the data. However, further experiments need to be performed in order to select the best configuration for $\mathbf{W}$.

4.2.5 INDIVIDUAL (c_1) AND GLOBAL (c_2) WEIGHTS

We now analyze the performance of *PSMS* under different settings of the individual (c_1) and global (c_2) weights. We considered three configurations for c_1 and c_2; in the first one both weights have the same influence in the search (i.e., $c_1 = 2; c_2 = 2$), this was the setting used for all of the experiments reported in Section 4. In the second setting the local weight has no influence in the search (i.e., $c_1 = 0; c_2 = 2$), while in the third configuration the global weight is not considered in the search (i.e., $c_1 = 2; c_2 = 0$). We ran *PSMS* for $I = 50$ iterations with a swarm size of $m = 5$, using a single

ID	Setting	CV-BER	test-BER	test-BER*
1	$c_1 = 2; c_2 = 2$	**23.69$^+_-$0.68**	**19.72$^+_-$1.45**	**23.92$^+_-$1.16**
2	$c_1 = 0; c_2 = 2$	26.87$^+_-$0.43	22.42$^+_-$1.32	27.13$^+_-$0.55
3	$c_1 = 2; c_2 = 0$	24.99$^+_-$0.41	21.73$^+_-$1.42	25.59$^+_-$0.66

Table 7: Average *CV-BER*, *test-BER* and *test-BER** for different settings of c_1 and c_2 in Equation (1). The best results are shown in **bold**. Results are averaged over a single replication of each data set described in Table 3.

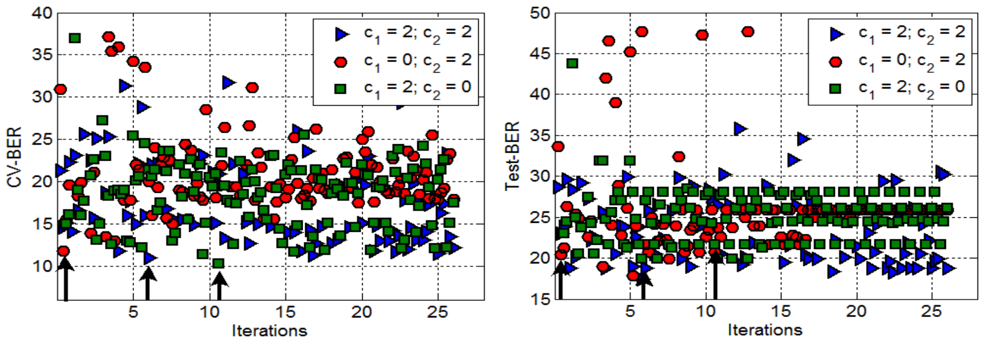


Figure 4: Performance of *PSMS* as a function of number of iterations using different settings for c_1 and c_2, see Table 7. We show the *CV-BER* (left) and *Test-BER* (right) for a single replication of the *Heart* data set. *PSMS* was ran for $I = 25$ iterations in this experiment. The models selected with each configuration are indicated with arrows.

replication for each data set described in Table 3; for the three configurations considered it was used the same adaptive inertia weight $\mathbf{W} = (1.2, 0.5, 0.4)$; averaged results of this experiment are shown in Table 7.

From this table we can see that the best performance is obtained by assigning equal weights to both factors. The difference in performance is statistically-significant over all measures with respect to the other two configurations. Therefore, by using the first configuration we can obtain solutions of better performance through and at the end of the search. More importantly, solutions of better generalization performance can be obtained with this configuration as well. The difference in performance is higher with the second configuration, where the individual-best solutions have no influence in the search; therefore, *PSMS* is searching locally around the global best solution. In the third configuration the global-best solution has no influence in the search; in consequence, the search is guided according the $m-$individual-best solutions. For illustration in Figures 4 and 5 we show the performance of *PSMS* as a function of the number of iterations for a single replication of the *Heart* data set. In Figure 4 we show the performance of *PSMS* for $I = 25$ iterations and in Figure 5 for $I = 100$ iterations.

From these figures we can see that the *CV* estimate is very similar for the different settings we considered (left plots in Figures 4 and 5). However, by looking at the performance of the solutions in the test set (right plots in Figures 4 and 5), we can appreciate that configurations 2 and 3 overfit

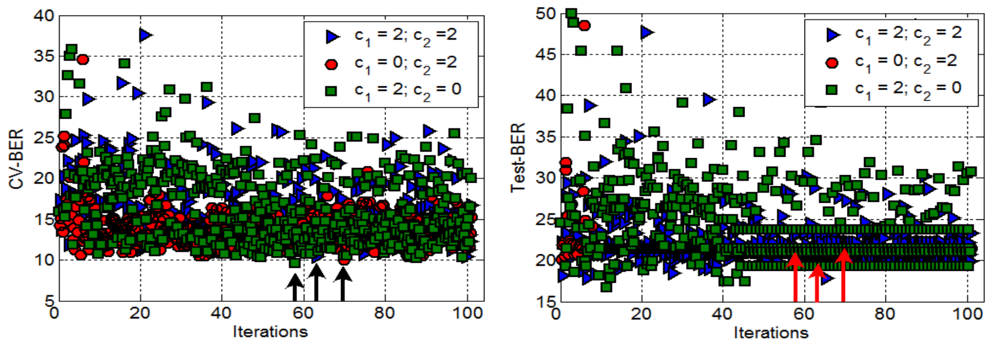


Figure 5: Performance of *PSMS* as a function of number of iterations using different settings for c_1 and c_2, see Table 7. We show the *CV-BER* (left) and *Test-BER* (right) for a single replication of the *Heart* data set. *PSMS* was ran for $I = 100$ iterations in this experiment. The models selected with each configuration are indicated with arrows.

the data (red circles and green squares). With $c_1 = 0$ we have that the $m = 5$ particles converge to single local minima, performing a fine grained search over this solution (red circles). With $c_2 = 0$ each of the $m-$particles converge to different local minima, overdoing the search over each of the m solutions (green squares). On the other hand, with the configuration $c_1 = 2, c_2 = 2$ *PSMS* is not trapped into a local minimum (blue triangles); searching around promising solutions, but without doing a fine grained search over any of them.

Better models (indicated by arrows) are selected by *PSMS* with the first configuration, even when their *CV* is higher than that of the models selected with the other configurations. This result confirms that *PSMS* is overfitting the data with the configurations 2 and 3. Note that with $I = 25$ iterations (Figure 4) the first configuration is not converging to a local minimum yet; while with $I = 100$ iterations (Figure 5) it looks like *PSMS* starts searching locally at the last iterations. This result illustrates why early stopping can be useful for obtaining better models with *PSMS*.

In order to better appreciate the generalization performance for the different configurations, in Figure 6 we plot the *CV-BER* as a function of *test-BER* for the run of *PSMS* with $I = 25$, we plot each particle with a different color.

From this figure we can see that the best model is obtained with the first configuration; for the configurations 2 and 3 the particles obtain the same *test-BER* for different solutions (middle and right plots in Figure 6). Despite the *CV* estimate is being minimized for these configurations, the *test-BER* performance of models does not improve. It is clear from the right plot in Figure 6 that with $c_2 = 0$ each particle is trapped in different local minima, doing a fine grained search over them that causes *PSMS* to overfit the data. It also can be seen from the middle plot that with $c_1 = 0$ the search is biased towards a single global-best solution (magenta circle), again, causing *PSMS* to overfit the data. On the other hand, results with the first configuration (left plot in Figure 6) show that particles do not oversearch at any solution.

4.3 Results on the Model Selection Challenge

In this section we describe experimental results of *PSMS* in the framework of a model selection competition called *agnostic learning vs. prior knowledge challenge* (*ALvsPK*) (Guyon et al., 2007,

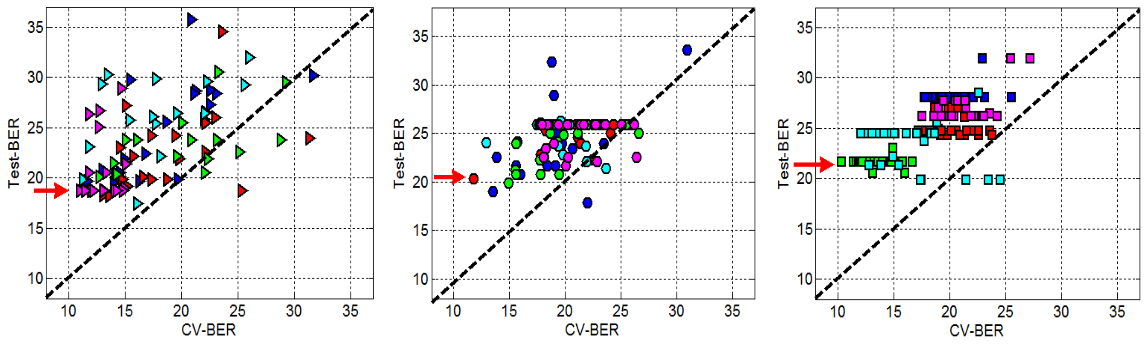


Figure 6: *Test-BER* as a function of *CV-BER* for a run of *PSMS* for $I = 25$ iterations in the *Heart* data set. Results with different configurations for c_1 and c_2 are shown. Left: $c_1 = 2$, $c_2 = 2$. Middle: $c_1 = 0, c_2 = 2$. Right: $c_1 = 2, c_2 = 0$. In each plot each particle is shown with a different color. The selected model with each configuration is indicated with an arrow.

2008). The goal of these experiments is to compare the performance of *PSMS* against other model selection strategies that work for a single algorithm or that use domain knowledge for this task. Through its different stages, the *ALvsPK* competition evaluated novel strategies for model selection as well as the added value of using prior knowledge for improving classification accuracy (Guyon et al., 2008). This sort of competitions are very useful because through them the real effectiveness of methods can be evaluated; motivating further research in the field and collaborations among participants.

4.3.1 CHALLENGE PROTOCOL AND CLOP

The rules of the challenge were quite simple, the organizers provided five data sets for binary classification together with the *CLOP* toolbox (Saffari and Guyon, 2006). The task was to obtain the model with the lowest *BER* over the five data sets on unseen data. Participants were free to elect using *CLOP* or their own learning machine implementations. The challenge is over now, although the challenge website[7] still remains open, allowing the evaluation of learning techniques and model selection methods. A complete description of the challenge and a comprehensive analysis of the results are described by Guyon et al. (2006c, 2007, 2008).

The competition was divided into two stages. The first stage, called *the model selection game* (Guyon et al., 2006c), was focused on the evaluation of pure model selection strategies. In the second stage, the goal was to evaluate the gain we can have by introducing prior knowledge into the model selection process (Guyon et al., 2007, 2008). In the latter stage participants could introduce knowledge of the data domain into the model selection process (*prior knowledge track*). Also, participants could use agnostic methods in which no domain knowledge is considered in the selection process (*agnostic track*).

The data sets used in the agnostic track of the *ALvsPK* challenge are described in Table 8, these data sets come from real domains. Data sets used for the agnostic and prior knowledge tracks were

7. See `http://www.agnostic.inf.ethz.ch/`.

different. For the agnostic track the data were preprocessed and dummy features were introduced, while for the prior knowledge track raw data were used, together with a description of the domain. We should emphasize that, although all of the approaches evaluated in the *ALvsPK* competition faced the same problem (that of choosing a model that obtains the lowest classification error for the data), such methods did not adopt the *FMS* interpretation. Most of the proposed approaches focused on a fixed machine learning technique like tree-based classifiers (Lutz, 2006), or kernel-based methods (Cawley, 2006; Pranckeviciene et al., 2007; Guyon et al., 2008), and did not take into account feature selection methods. Participants in the prior knowledge track could introduce domain knowledge. Furthermore, most of participants used their own implementations, instead of the *CLOP* toolbox.

After the challenge, the CLOP toolkit was augmented with methods, which performed well in the challenge (Cawley, 2006; Cawley and Talbot, 2007a; Lutz, 2006). These include Logitboost (Friedman et al., 2000), LSSVM (Suykens and Vandewalle, 1999), and kernel ridge regression (Saunders et al., 1998; Hastie et al., 2001).

4.3.2 COMPETITIVENESS OF PSMS

In both stages of the competition we evaluated models obtained with *PSMS* under different settings. Models obtained by *PSMS* were ranked high in the participants list, showing the competitiveness of *PSMS* for model selection (Guyon et al., 2006c, 2007, 2008; Escalante et al., 2007). Furthermore, the difference with methods that used prior knowledge was relatively small, showing that *FMS* can be a viable solution for the model selection problem without the need of investing time in introducing domain knowledge, and by considering a wide variety of methods.

The results of *PSMS* in the *ALvsPK* challenge have been partially analyzed and discussed elsewhere (Escalante et al., 2007; Guyon et al., 2007, 2008). During the challenge, our best entry (called *Corrida-final*) was ranked[8] 8^{th} over all ranked participants, 5^{th} among the methods that did not use domain knowledge and 2^{nd} among the methods that used the software provided by the organizers (Guyon et al., 2006c, 2007, 2008). For *Corrida-final* we used $k = 5$ and the full training set for computing the fitness function; we ran *PSMS* for 500 iterations for the *Ada* data set and 100 iterations for *Hiva*, *Gina* and *Sylva*. We did not applied *PSMS* to the *Nova* data set in that entry, instead we selected a model for *Nova* by trial and error. For such entry we used a version of *CLOP* where only there were available the following classifiers *zarbi, naive, neural* and *svc* (Escalante et al., 2007); also, only four feature selection methods were considered.

4.3.3 POST-CHALLENGE EXPERIMENTS

In the rest of this section we present results of *PSMS* using the augmented toolkit, including all methods described in Tables 1 and 2. In these tables we consider implementations of *logitboost, lssvm and gkridge,* which are the classifiers that won the *ALvsPK* challenge (Cawley, 2006; Cawley and Talbot, 2007a; Lutz, 2006) and were added to CLOP after the end of the challenge.

In order to efficiently apply *PSMS* to the challenge data sets we adopted a subsample strategy in which, instead of using the full training set, small subsamples of the training data were used to compute the fitness function. Each time the fitness function is computed we obtain a different random sample of size $S_{sub} = \frac{N}{SF}$, where N is the number of instances and SF is a constant that specifies the proportion of samples to be used. Subsamples are only used for the search process. At

8. See `http://www.clopinet.com/isabelle/Projects/agnostic/Results.html`.

Data set	Domain	Type	Features	Training	Validation	Testing
Ada	Marketing	Dense	48	4174	415	41471
Gina	Digits	Dense	970	3153	315	31532
Hiva	Drug discovery	Dense	1617	3845	384	38449
Nova	Text classification	Sparse binary	16969	1754	175	17537
Sylva	Ecology	Dense	216	13086	1309	130857

Table 8: Benchmark data sets used for the model selection challenges (Guyon et al., 2006c, 2007, 2008).

Data	SF	Model	Time (m)	Test-BER
Ada	1	chain({logitboost(units=469,shrinkage=0.4,depth=1),bias}	368.12	16.86
Gina	2	chain({sns(1),relief(fmax=487),gkridge,bias}	482.23	2.41
Hiva	3	chain({norm(1),rffs(fmax=1001),lssvm(gamma=0.096),bias}	124.54	28.01
Nova	1	chain({rffs(fmax=338),norm(1),std(1),sns(1),gkridge,bias}	82.12	5.27
Sylva	10	chain({sns(1),odds-ratio(fmax=60),gkridge,bias}	787.58	0.62

Table 9: Models selected with *PSMS* for the data sets of the *ALvsPK* challenge. For each data set we show the subsampling factor used (**SF**), the selected model (**Model**, some hyperparameters are omitted for clarity), the processing **Time** in minutes and the **test-BER** obtained. *sns* is for *shit-scale*, *std* is for *standardize* and *norm* is for *normalize*. See Tables 1 and 2 for a description of methods and their hyperparameters.

the end of the search the selected model is trained using the full training set (for the experiments reported in this paper we considered as training set the union of the training and validation data sets, see Table 8). Due to the dimensionality of the *Nova* data set we applied principal component analysis to this data set. Then we used the first 400 components for applying *PSMS*. We fixed $k = 2$, $I = 50$ and $m = 5$ for our experiments based on the results from previous sections. Then we ran *PSMS* for all of the data sets under the above described settings using different values for *SF*. The predictions of the resultant models were uploaded to the challenge website in order to evaluate them. Our best ranked entry in the *ALvsPK* challenge website (called *psmsx_jmlr_run_I*) is described in Table 9, and a comparison of it with the currently best-ranked entries is shown in Table 10.

We can see from Table 9 that very different models were selected by *PSMS* for each data set. This is the main advantage the *FMS* because it allows us selecting an ad-hoc model for each data set by considering different model types and a wide diversity of methods. With exception of *Ada*, the selected models included a feature selection method; this result shows the importance of feature selection methods and that some of them are more compatible than others with some classifiers; note that different numbers of features were selected for each data set. In all but the *Nova* data set preprocessing was performed before feature selection. This can be due to the fact that for *Nova* we used principal components instead of the original features. For *Ada* it was selected a *logitboost* classifier, while for *Hiva* it was selected a *lssvm* classifier with gaussian kernel. For *Gina, Nova* and *Sylva* it was selected the *gkridge* classifier; this classifier performs virtual leave-one out model selection each time it is trained (Cawley et al., 2007). Note that both *gkridge* and *logitboost* were the classifiers that best performed during the challenge (Guyon et al., 2007, 2008); this result gives evidence that *PSMS* can obtain similar and even better models, without spending time on ad-hoc modifications for each data set and without using domain knowledge.

The use of the subsampling strategy allowed to efficiently apply *PSMS* to all of the data sets. About six hours were required to obtain a competitive model for *Ada*, while about only two for the *Hiva* data set. Note that applying *PSMS* for *Hiva* using the entire training set (*Corrida-final*) took about 80 hours in the same machine (Escalante et al., 2007). During our experiments we found that the larger the subsamples the better the performance of the selected model. However, the selection of *SF* also depends on the available computer resources and time restrictions. One should note that we can increase speed of *PSMS* by caching intermediate results that can be reused (e.g., preprocessing the data sets off-line).

Despite the use of the subsampling technique the models selected with *PSMS* resulted very competitive as shown in Table 10. Currently, the *PSMS* run is the top-ranked agnostic entry in the challenge website; furthermore, the performance of this entry is superior than all but one prior-knowledge entry: *Interim-all-prior*; which is the best ranked entry overall, using "prior knowledge" or "domain knowledge". Note that the latter entry is formed of models that were designed ad-hoc for each data set, requiring much more time, effort and knowledge than *PSMS*. In average *PSMS* outperforms the best results presented in the *ALvsPK* challenge: *IJCNN07AL*, row 4 in Table 10 (Guyon et al., 2007); and the best-ranked agnostic entry (after *PSMS*): *Logitboost-with-trees*, row 5 in Table 10 (Lutz, 2006).

The performance of *PSMS* is very close to that of *IJCNN07AL*, tieing in *Sylva* and outperforming one to the other in two data sets. *PSMS* outperforms *Logitboost-with-trees* in three out of the five data sets and achieves very close performance for the other two data sets. The latter entry is ranked 10^{th} in the challenge website. It is very interesting that for the *Ada* data set the best model so far is a *logitboost* classifier with about 1000 trees (Lutz, 2006); while with *PSMS* we were able to achieve almost the same performance by using a half that number of trees, see row 2 in Table 9. This is important because simpler models of similar performance can be obtained with *PSMS*.

PSMS clearly outperformed our best-ranked entry during the challenge (row 6 in Table 10); this result gives evidence that we obtained better results by using more and better classifiers; also, the use of subsamples instead of the entire training set (when computing the fitness function), does not damage the performance of *PSMS*, although the reduction in processing time is very important. Note that for *Nova* the *PSMS* entry obtained a slightly worse result than that of *Corrida-final*; however, the model for *Nova* in *Corrida-final* was selected by trial and error which required of much more effort and time.

Results reported in this section show the efficacy of *PSMS* for model selection. Despite its simplicity it has shown comparable and even superior performance to those obtained by other model selection strategies that focused on a single learning algorithm and to methods that used prior domain knowledge for guiding the model selection process (Lutz, 2006; Reunanen, 2007; Boullé, 2007; Pranckeviciene et al., 2007; Wichard, 2007). Models selected with *PSMS* are simple and yet very competitive; furthermore, with *PSMS* no knowledge is needed on the methods to choose from, nor on the domain. In consequence, it is very easy to obtain classifiers that can achieve state-of-the-art performance without spending time on designing, developing and optimizing an ad-hoc model. Even though *PSMS* can be applied to any binary classification problem, we are not claiming it will obtain satisfactory results in every domain; however, it can be considered as a first option when dealing with binary classification tasks. It is expected that this will further improve the performance of models selected with *PSMS* if domain knowledge is used.

Entry	Description	Ada	Gina	Hiva	Nova	Sylva	Overall	Rank
Interim-all-prior	Best-PK	17.0	2.33	27.1	4.71	0.59	10.35	1th
psmsx_jmlr_run_I	PSMS	16.86	**2.41**	**28.01**	5.27	**0.62**	10.63	2nd
IJCNN07AL	Best-AL	**16.60**	3.39	28.27	**4.56**	**0.62**	10.68	4th
Logitboost-with-trees	Best-AL	**16.60**	3.53	30.18	4.69	0.78	11.15	10th
Corrida-final	Best-PSMS-ALvsPK	18.27	6.14	28.54	5.11	1.22	11.86	42th

Table 10: Comparison of models selected with *PSMS* and the best entries in the *ALvsPK* challenge data sets. We show, for reference, the best prior-knowledge entry (*Interim-all-prior*); the entry formed by the models described in Table 9 (*psmx_jmlr_run_I*); the best individual entries for each data set in the *ALvsPK* challenge(*IJCNN07AL*) (Guyon et al., 2007, 2008); the second-best entry of the agnostic track (*Logitboost-with-trees*) and our best ranked entry evaluated during the challenge *Corrida-final*. The best results in the agnostic track are shown in **bold**.

5. Discussion

In this section, we discuss the advantages and disadvantages of *PSMS* and perform a synthesis of our experiments aiming at better understanding how *PSMS* performs intensive search in hyperparameter space without overfitting the data.

5.1 Robust and Computationally Tractable Intensive Search

In Section 4 we reported experimental results that give evidence of the validity of the *PSMS* approach, demonstrating in particular that *PSMS* can outperform *PATSMS* through and at the end of the search, showing better convergence behavior and generalization performance. This is obtained at the expense of moderate additional computational complexity. This claim is supported by the theoretical analysis of the computational complexity (Section 3.4), indicating that computations are dominated by the number of learning machine trainings, and by the experiments (Section 4.2), indicating as few as 5 particles (learning machines) and 10 iterations (i.e., 50 trainings) are needed to attain the best performance. The efficiency of *PSMS* can be improved, for instance by preferably exploring learning machines, which have a lower computational cost of training. We explored successfully other heuristics, including subsampling training data, which reduced computations, at the expense of no performance degradation.

An analysis of the diversity of models tried by *PSMS* shows that this method is not biased towards models that tend to perform well individually. Investigating the operation of *PSMS* under different settings we found that the performance of *PSMS* is not significantly affected by modifying its hyperparameters. However, experimental results indicate that the use of an adaptive inertia weight may be helpful to explore the search space better.We also observed that certain parameter configurations allow the selection of competitive models, while reducing processing time. Section 5.4 provides a final set of practical recommendations.

Results of international competitions suggest that *PSMS* is competitive with model selection strategies specific to single algorithms and to strategies using prior domain knowledge. The latter is an important result because it shows that we can obtain competitive models without the need of an expert on the domain, a careful analysis to the data, or even machine learning knowledge.

5.2 Intensive Search without Overfitting

The findings summarized above provide empirical evidence suggesting *PSMS* is a reliable strategy for agnostic model selection. However, it is not obvious why *PSMS* succeeds in selecting competitive models without significantly overfitting data. Our hypothesis is that *PSMS* is able to avoid overfitting because of the way the search is guided: *PSMS* performs a broad search around promising solutions without focusing on reaching local minima. Solutions in *PSMS* are updated by taking into account information from both: the global-best solution ($\mathbf{p}_g$, weighted by c_2) and the individual-best solutions of each particle ($\mathbf{p}_{1,\dots,m}$, weighted by c_1), see Equations (1) and (2). The latter combined with an adaptive inertia weight and early stopping cause do not exaggerate the search at any local minima. Methods like *PATSMS*, on the contrary, update solutions by moving the pattern towards the local minimum nearest the initial solution, see Algorithm 2. Reaching exactly a local minimum causes *PATSMS* to learn peculiarities in the data set and does not necessarily result in obtaining a better predictive model.

The experiments we performed in Section 4.2.5 support the above conjecture. The results from Table 7 and Figures 4, 5 and 6 indicate that *PSMS* is able to avoid overfitting (to some extend) because of the way the search is guided. *PSMS* searches around good solutions without overdoing in terms of really fine-grained optimization. This sort of search can be considered as suboptimal in Dietterich's sense: *"In machine learning it is optimal to be suboptimal!"* (Dietterich, 1995). The latter statement makes reference to the well known fact that oversearching (i.e., trying to be optimal) in model selection can lead to select models that fit very well the peculiarities of the considered data set without deriving a general predictive rule (Dietterich, 1995; Jensen and Cohen, 2000; Quinlan and Cameron-Jones, 1995; Hastie et al., 2001; Loughrey and Cunningham, 2005). On the contrary, *PSMS* is able to *undercompute* because for updating solutions it considers local and global knowledge, information from past solutions (weighted by the inertia term) and randomness; note that the latter holds when a reasonable small number of iterations is performed. Furthermore, our experimental results provide empirical evidence that agrees with recent, yet traditional, explanations about why and how *PSO* works (Kennedy, 2008; Kennedy and Eberhart, 2001). Kennedy has argued the success of *PSO* is due to the fact that it performs a *"collaborative trial and error"* search (Kennedy, 2008). That is, *PSO* obtains good results mainly because the search is directed according both individual and social knowledge; the same conclusion derived from experimental results in this section. It is not surprising that distributed and collaborative computing can improve results of centralized (individualized) methods. For *FMS*, however, it is very interesting that updating solutions in a heterogeneous way allows us to avoid oversearching and, in consequence, overfitting; even when the *FMS* search space is infinite and has many local minima solutions.

5.3 Comparison with Related Work

A variety of approaches have been proposed to select parameters of specific methods for regression, classification, feature selection, discretization etcetera (Hastie et al., 2001; Bishop, 2006; Voss and Feng, 2002; Nelles, 2001; Guyon et al., 2006b; Kim et al., 2002; Hastie et al., 2001; Cawley and Talbot, 2007b; Boullé, 2007; Hue and Boullé, 2007). However, despite the potential advantages of *FMS* (namely generality, simplicity and competitiveness), this problem has been little studied because of the huge search space involved and because intensive search methods are prone to overfit the data (Gorissen et al., 2008; Escalante et al., 2007). Nevertheless, in the rest of this section we outline techniques used to avoid oversearching/overfitting in search that are applicable/related to

FMS. One should note that traditional model selection techniques just like Akaike and Bayesian information criteria, the minimum description length principle and the *VC*-dimension are not directly applicable to *FMS* and therefore they are excluded of analysis.

Grid search (with *CV*) is the widely used search-approach to model selection in practical applications (Momma and Bennett, 2002; Hsu et al., 2003). This method consists of defining a uniform grid over the search space where each point in the grid defines a solution; every point in the grid is evaluated and the point of lowest error is selected. The granularity of the grid determines both the performance of the selected solution and the efficiency of the search. A fine-grained grid may be inefficient and can lead to oversearching; while a sparse grid will result in low performance models. Note that the heterogeneousness of models and the variety of ranges for the models parameters make very difficult the application of grid search to the *FMS* problem; furthermore, the choice of an adequate granularity can be a serious problem. Other methods already used for parameter optimization that can be applied to *FMS* include: greedy search (Dietterich, 1995), random search (e.g., the bumping technique) (Hastie et al., 2001), *PS* (Bi et al., 2003; Momma and Bennett, 2002), evolutionary computation approaches (Engelbrecht, 2006; Gorissen et al., 2008; Angeline, 1998), and other swarm-optimization techniques (Kennedy and Eberhart, 2001; Engelbrecht, 2006). Note that despite one may think that exhaustive search is the best search option in model selection, this approach is impractical for most real world problems and when applicable it suffers from the oversearching phenomenon (Dietterich, 1995).

Early stopping has been widely used to prevent overfitting in search methods (Hastie et al., 2001; Engelbrecht, 2006; Loughrey and Cunningham, 2005). The goal of this heuristic is to stop searching/learning when the model starts overfitting the data. There are several variants to stop the search: after a small number of iterations, when no improvement is found after a number of iterations, when a solution of acceptable performance has been found, etcetera. A problem with early stopping is that premature stopping the algorithm would lead to selecting a solution that has not converged yet, while a late stopping of the search will cause severely overfitting the data because of oversearching. For *PSMS* we found that a small number of iterations can be enough to obtain satisfactory results in benchmark data, although the number of iterations is problem dependent; therefore, we can adopt other stopping criteria for *FMS* in the future.

Randomness has bring into play in machine learning in order avoid overfitting and to escape from local minima in search (Hastie et al., 2001; Bishop, 2006; Kirkpatrick et al., 1983; Kennedy and Eberhart, 2001). In learning algorithms, it has been successfully used to prevent overfitting and to obtain better predictors; learning methods that use randomness include bagging classifiers (Hastie et al., 2001), neural and deep belief networks (Hastie et al., 2001; Hinton et al., 2006) and randomized decision-tree algorithms (Breiman, 2001; Geurts et al., 2006). Bootstrapping is a technique (used in bagging and random forest classifiers) based on random sampling that has been widely used to estimate the generalization performance of methods as an alternative to *CV* (Hastie et al., 2001). In *PSMS* randomness played an important role because it introduces diversity into the search process and allows *PSMS* to avoid local minima. Furthermore, the subsampling strategy we used to increase the speed of *PSMS* is related to bootstrapping; in future work on *PSMS* we will explicitly consider different subsampling estimations for the selection of the final model.

As the adaptive inertia weight in *PSO*, see Section 2, there are parameters in other algorithms that aim to avoid overfitting by exploring the search space both globally and locally; examples are the temperature parameter in simulated annealing (Kirkpatrick et al., 1983) and the momentum term in on-line gradient descend and backpropagation (Qian, 1999). The ridge in ridge-regression and weight decay in neural networks training are also related to the inertia weight. Model averaging and the use of ensembles have proved to be helpful to improve predictions and avoid overfitting; this is because different models have different biases that in average result in improved performance (Hastie et al., 2001; Bishop, 2006). Future work includes in *PSMS* consists of combining particles in order to improve the performance of the swarm strategy. Finally, adding noise to the training data is another overfitting avoidance mechanism in model selection that also can be used with *PSMS*.

5.4 A Practical Guide to PSMS

In this section we describe the way *PSMS* can be put in practice in any binary classification problem. Due to the simplicity and generality of the approach below we describe a practical guide to use the *Matlab*^{R} implementation of *PSMS* (included in the *CLOP* toolbox). It is assumed that the user has available a data set (in Matlab^R format) with N samples for binary classification: a matrix $\mathbf{X}_{N\times d}$ contains the N training samples of dimensionality d and a vector $\mathbf{Y}_{N\times 1}$ their respective labels ($y_i \in [-1,1]$). After downloading and installing *CLOP* (Saffari and Guyon, 2006), *PSMS* can be applied to any data set by typing the following *Matlab*^{R} code:

```
%% load your data into the MatlabR workspace
1: >> load train_data.mat;
%%  Create A Clop Data-Object
2: >> D = data(X,Y);
%% Create A CLOP PSMS-Object with default parameters
3: >> P = psmsx;
%% Perform PSMS
4: >> [Dat, Res] = train(P, D);
%% Train the selected model with the full training set
5: >> [Odat, TrM] = train(Res.Best_Model,D);
%% Create a test data set, note that Yt can be empty
6: >> load test_data; Dt = data(Xt,Yt);
%% Test the selected and trained model on unseen data Dt
7: >> [Pred] = test(TrM, Dt);
%% Estimate the model's performance on unseen data Dt, if Yt is available
8: >> [BER] = balanced_errate(Pred.X, Pred.Y);
%% Analyze the ROC performance of the selected model
9: >> roc(Pred);
```

Note that steps 1–2 and 5–9 are associated with loading the data and the evaluation of the selected model, respectively; which are operations not attained to *PSMS*. Steps 3 and 4 will create the *PSMS* object and will start the search, respectively. Besides the selected model, the output of the search (**Res**, line 4), is a structure with useful information about the search process, see the *PSMS* documentation (Escalante, In preparation, 2009).

In Section 4.2 were presented experimental results that suggest that there is not significant difference in performance by modifying most of the *PSMS* hyperparameters. Therefore, one can choose parameter settings for *PSMS* that make practical its application without a significant decrement of performance. Below are shown recommended values for the *PSMS* hyperparameters. These parameters and other options of the current implementation can be modified very simply (Escalante, In preparation, 2009).

Recommended PSMS parameters:

Weight for individual best solution	$c_1 = 2$
Weight for global best solution	$c_2 = 2$
Adaptive inertia weight	$\mathbf{W} = (1.2, 0.5, 0.4)$
Number of iterations	$I = 50$
Swarm size	$m = 5$
Folds in *CV*	$k = 2$
Subsampling factor	$SF = 1$ (as small as possible in large data sets)

6. Conclusions

In this paper we proposed Particle Swarm Model Selection (*PSMS*), that is, the application of Particle Swarm Optimization (*PSO*) to the problem of Full Model Selection (*FMS*). Given a data set, the *FMS* problem consists of selecting the combination of preprocessing, feature selection and learning methods that obtains the lowest classification error. *FMS* also includes hyperparameter optimization for the selected methods. This approach to the model selection problem has the following advantages. First, the generality of the approach allows us to consider different model types (for preprocessing, feature selection and learning) and a variety of methods. Second, *PSMS* can be applied to any data set, since neither domain knowledge nor machine learning knowledge is required, therefore it can be considered a black-box model selection method. Third, and most importantly, competitive and yet simple models can be obtained with *PSMS*. We provide empirical evidence that shows that *PSMS* can be applied efficiently, without a significant loss of accuracy, by using a subsampling heuristic and parameter settings that reduce the computational cost.

The simplicity of *PSO* and its proven performance, comparable to that of evolutionary algorithms, make this search algorithm well suited for *FMS*. However, the application of any other stochastic optimization strategy is also possible. The main advantage of *PSO* is that a single equation for updating solutions is needed, as opposed to evolutionary algorithms where methods for representation, mutation, cross-over, speciation and selection have to be considered. Interestingly, the way the search is guided in *PSMS* allows it obtaining competitive models without significantly overfitting. Experimental results in benchmark data show superior performance of *PSMS* when compared to Pattern Search Model Selection (*PATSMS*), a direct search method that constitutes a competitive baseline.

Results obtained by models selected with *PSMS* in the framework of a model selection challenge show that it is a very competitive model selection method, despite its simplicity and generality. In such competitions, models selected with *PSMS* were always among the top ranking models, together with methods performing solely hyperparameter selection in a given model family and methods relying on prior knowledge. This demonstrates that, via the use of *PSO*, *FMS* is a viable strategy for model selection. This is remarkable because we noted in previous competitions (Guyon et al., 2005; Guyon et al., 2006b) that each data set had a different best performing method, yet researchers performing *FMS* (in an effort to find the model family best suited to a given problem) were not successful. The participants, which obtained the best results on average over all data sets restricted themselves to hyperparameter selection in one given model family. In contrast, in this paper we demonstrated the viability of *FMS* using the *PSO* search strategy. Our work paves the way to the use of intensive search techniques to perform *FMS* in the entire model space of machine learning toolkits. With the increasing availability of diverse and sophisticated machine learning toolkits and the improvements in computing power, we foresee that *FMS* will become an effective methodology.

Current work includes the use of *PSMS* for the selection of model-members for ensembles and the hierarchical application of *PSO* for *FMS* and hyperparameter optimization. *PSMS* is currently being applied to different tasks, including galaxy classification, automatic image annotation, object recognition and text classification. Future work includes the introduction of a penalty-term into the fitness function; such that (computationally) inexpensive models be favored by PSMS. The extension of *PSMS* to the multi-class classification and regression problems is another future work direction.

Acknowledgments

We would like to thank the organizers and participants of the NIPS multi-level inference workshop and model selection game, and of the *IJCNN ALvsPK* challenge. The first author thanks the UNIPEN foundation and the US National Science Foundation for travel support. We also thank editors and anonymous reviewers for their useful comments that have help us to improve this paper. We thank Dr. Eduardo Morales for his useful suggestions about the content of the paper. Last, but not least, we thank Dr. Aurelio López and INAOE for the provided support.

References

P. J. Angeline. Evolutionary optimization vs particle swarm optimization: Philosophy and performance differences. In *Proceedings of the 7th Conference on Evolutionary Programming*, volume 1447 of *LNCS*, pages 601–610, San Diego, CA, March 1998. Springer.

Y. Bengio and N. Chapados. Extensions to metric-based model selection. *Journal of Machine Learning Research*, 3:1209–1227, 2003.

J. Bi, M. Embrechts K. P. Bennett, C. M. Breneman, and M. Song. Dimensionality reduction via sparse support vector machines. *Journal of Machine Learning Research*, Mar(3):1229–1243, Mar 2003.

C. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

M. Boullé. Report on preliminary experiments with data grid models in the agnostic learning vs prior knowledge challgenge. In *Proceedings of the 20th International Joint Conference on Neural Networks*, pages 1802–1808, 2007.

L. Breiman. Random forest. *Machine Learning*, 45(1):5–32, 2001.

G. Cawley. Leave-one-out cross-validation based model selection criteria for weighted ls-svms. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN 2006)*, pages 2970–2977, Vancouver, Canada, July 2006.

G. Cawley and N. L. C. Talbot. Agnostic learning vs prior knowledge in the design of kernel machines. In *Proceedings of the 20th International Joint Conference on Neural Networks*, pages 1444–1450, Orlando, Florida, 2007a.

G. Cawley, G. Janacek, and N. L. C. Talbot. Generalised kernel machines. In *Proceedings of the 20th International Joint Conference on Neural Networks*, pages 1439–1445, Orlando, Florida, 2007.

G. C. Cawley and N. L. C. Talbot. Preventing over-fitting during model selection via bayesian regularisation of the hyper-parameters. *Journal of Machine Learning Research*, 8:841–861, April 2007b.

M. Clerc and J. Kennedy. The particle swarm: Explosion, stability and convergenge in a multi-dimensional complex space. *IEEE Transactions on on Evolutionary Computation*, 6(1):58–73, February 2002.

J. Demsar. Statistical comparisons of classfifiers over multiple data sets. *Journal of Machine Learning Research*, 7:1–30, January 2006.

J. E. Dennis and V. J. Torczon. Derivative-free pattern search methods for multidisciplinary design problems. In *Proceedings of the AIAA / USAF / NASA / ISSMO Symposium on Multidisciplinary Analysis and Optimizatino*, pages 922–932, 1994.

T. Dietterich. Overfitting and undercomputing in machine learning. *ACM Comput. Surv.*, 27(3): 326–327, 1995. ISSN 0360-0300.

A. P. Engelbrecht. *Fundamentals of Computational Swarm Intelligence*. Wiley, 2006.

H. J. Escalante. Particle swarm optimization for classifier selection: A practical guide to psms. `http://ccc.inaoep.mx/~hugojair/psms/psms_doc.pdf`, In preparation, 2009.

H. J. Escalante, M. Montes, and E. Sucar. Psms for neural networks on the ijcnn 2007 agnostic vs prior knowledge challenge. In *Proceedings of the 20th International Joint Conference on Neural Networks*, pages 1191–1197, Orlando, FL, USA., 2007.

V. Franc and V. Hlavac. The statistical pattern recognition toolbox. `http://cmp.felk.cvut.cz/cmp/software/stprtool/index.html`, 2004.

J. Friedman, T. Hastie, and R. Tibshirani. Additive logistic regression: a statistical view of boosting. *Annals of Statistics*, 28(2):337–407, 2000.

P. Geurts, D. Ernst, and L. Wehenkel. Extremely randomized trees. *Machine Learning*, 63(1):3–42, 2006. ISSN 0885-6125.

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. Bloomeld, and E. S. Lander. Molecular classification of cancer: class discovery and class prediction by gene expression monitoring. *Science*, 286: 531–537, October 1999.

D. Gorissen. Heterogeneous evolution of surrogate models. Master's thesis, Katholieke Universiteit Leuven, Belgium, June 2007.

D. Gorissen, L. De Tommasi, J. Croon, and T. Dhaene. Automatic model type selection with heterogeneous evolution: An application to rf circuit block modeling. In *IEEE Proceedings of WCCI 2008*, pages 989–996, 2008.

V.G. Gudise and G.K. Venayagamoorthy. Comparison of particle swarm optimization and backpropagation as training algorithms for neural networks. In *Proceedings of the 2003 IEEE Swarm Intelligence Symposium, 2003. (SIS03)*, pages 110–117, 2003.

I. Guyon and A. Elisseeff. An introduction to variable and feature selection. *Journal of Machine Learning Research*, 3(Mar):1157–1182, 2003.

I. Guyon, S. Gunn, A. Ben-Hur, and G. Dror. Result analysis of the nips 2003 feature selection challenge. In *Advances in Neural Information Processing Systems 17*, pages 545–552. MIT Press, Cambridge, MA, 2005.

I. Guyon, S. Gunn, M. Nikravesh, and L. Zadeh, editors. *Feature Extraction, Foundations and Applications*. Series Studies in Fuzziness and Soft Computing. Springer, 2006a.

I. Guyon, A. Saffari, G. Dror, and J. M. Buhmann. Performance prediction challenge. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN 2006)*, pages 2958–2965, Vancouver, Canada, July 2006b.

I. Guyon, A. Saffari, G. Dror, G. Cawley, and O. Guyon. Benchmark datasets and game result summary. In *NIPS Workshop on Multi-level Inference and the Model Selection Game*, Whistler, Canada, December 2006c.

I. Guyon, A. Saffari, G. Dror, and G. Cawley. Agnostic learning vs prior knowledge challenge. In *Proceedings of the 20th International Joint Conference on Neural Networks*, pages 1232–1238, Orlando, Florida, 2007.

I. Guyon, A. Saffari, G. Dror, and Gavin Cawley. Analysis of the ijcnn 2007 competition agnostic learning vs. prior knowledge. *Neural Networks*, 21(2–3):544–550, 2008.

T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer Verlag, New York, 2001.

E. Hernández, C. Coello, and A. Hernández. On the use of a population-based particle swarm optimizer to design combinational logic circuits. In *Evolvable Hardware*, pages 183–190, 2004.

G. Hinton, S. Osindero, and Y. Teh. A fast learning algorithm for deep belief nets. *Neural Computation*, 18(7):1527–1554, 2006. ISSN 0899-7667.

C. W. Hsu, C. C. Chang, and C. J. Lin. A practical guide to support vector classification. Technical report, Taipei, 2003. URL `http://www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf`.

C. Hue and M. Boullé. A new probabilistic approach in rank regression with optimal bayesian partitioning. *Journal of Machine Learning Research*, 8:2727–2754, December 2007.

D. Jensen and P Cohen. Multiple comparisons in induction algorithms. *Machine Learning*, 38(3): 309–338, 2000. ISSN 0885-6125.

J. Kennedy. How it works: Collaborative trial and error. *International Journal of Computational Intelligence Research*, 4(2):71–78, 2008.

J. Kennedy and R. Eberhart. Particle swarm optimization. In *Proceedings of the International Conference on Neural Networks*, volume IV, pages 1942–1948. IEEE, 1995.

J. Kennedy and R. Eberhart. *Swarm Intelligence*. Morgan Kaufmann, 2001.

J. Kennedy and R. Mendes. Population structure and particle swarm performance. In *Proceedings of the IEEE Congress on Evolutionary Computation (CEC 2002)*, volume 2, pages 1671–1676, 2002.

Y. Kim, N. Street, and F. Menczer. Evolutionary model selection in unsupervised learning. *Intelligent Data Analysis*, 6:531–556, 2002.

S. Kirkpatrick, C. Gelatt, and M. Vecchi. Optimization by simulated annealing. *Science*, 220(4598): 671–680, 1983.

J. Loughrey and P. Cunningham. Overfitting in wrapper-based feature subset selection: The harder you try the worse it gets. In F. Coenen M. Bramer and T. Allen, editors, *Proceedings of AI-2004, the Twenty-fourth SGAI International Conference on Innovative Techniques and Applications of Artificial Intelligence*, Research and Development in Intelligent Systems XXI, pages 33–43, 2005.

R. Lutz. Logitboost with trees applied to the wcci 2006 performance prediction challenge datasets. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN 2006)*, pages 1657– 1660, Vancouver, Canada, July 2006.

S. Mika, G. Rätsch, J. Weston, B. Schölkopf, A. J. Smola, and K.-R. Müller. Invariant feature extraction and classification in kernel spaces. In S. A. Solla, T. K. Leen, and K.-R. Müller, editors, *Advances in Neural Information Processing Systems 12*, pages 526–532, Cambridge, MA, 2000. MIT Press.

M. Momma and K. Bennett. A pattern search method for model selection of support vector regression. In *Proceedings of SIAM Conference on Data Mining*, 2002.

O. Nelles. *Nonlinear System Identification: From Classical Approaches to Neural Networks and Fuzzy Models*. Springer, 2001.

E. Ozcan and C. K. Mohan. Analysis of a simple particle swarm optimization system. In *Intelligent Engineering Systems Through Artificial Neural Networks*, pages 253–258, 1998.

E. Pranckeviciene, R. Somorjai, and M. N. Tran. Feature/model selection by the linear programming svm combined with state-of-art classifiers: What can we learn about the data. In *Proceedings of the 20th International Joint Conference on Neural Networks*, pages 1422–1428, 2007.

N. Qian. On the momentum term in gradient descent learning algorithms. *Neural Netw.*, 12(1): 145–151, 1999. ISSN 0893-6080. doi: http://dx.doi.org/10.1016/S0893-6080(98)00116-6.

J. R. Quinlan and R. M. Cameron-Jones. Oversearching and layered search in empirical learning. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence*, pages 1019–1024, 1995.

G. Rätsch, T. Onoda, and K.-R. Müller. Soft margins for adaboost. *Mach. Learn.*, 42(3):287–320, 2001. ISSN 0885-6125. doi: http://dx.doi.org/10.1023/A:1007618119488.

J. Reunanen. Model selection and assessment using cross-indexing. In *Proceedings of the 20th International Joint Conference on Neural Networks*, pages 1674–1679, 2007.

M. Reyes and C. Coello. Multi-objective particle swarm optimizers: A survey of the state-of-the-art. *International Journal of Computational Intelligence Research*, 3(2):287308, 2006.

Y. Robinson, J. Rahmat-Samii. Particle swarm optimization in electromagnetics. *IEEE Transactions on Antennas and Propagation*, 52(2):397– 407, February 2004.

A. Saffari and I. Guyon. Quickstart guide for clop. Technical report, Graz University of Technology and Clopinet, May 2006. `http://www.ymer.org/research/files/clop/QuickStartV1.0.pdf`.

J. Salerno. Using the particle swarm optimization technique to train a recurrent neural model. In *Proceedings of the Ninth International Conference on Tools with Artificial Intelligence*, pages 45–49, 1997.

C. Saunders, A. Gammerman, and V. Vovk. Ridge regression learning algorithm in dual variables. In Jude W. Shavlik, editor, *Proceedings of the 15th International Conference on Machine Learning*, pages 515–521, Madison, WI, USA, 1998.

Y. Shi and R. C. Eberhart. Parameter selection in particle swarm optimization. In *Evolutionary Programming VII*, pages 591–600, New York, 1998. Springer-Verlag.

Y. Shi and R. C. Eberhart. Emprirical study of particle swarm optimization. In *Proceedings of the Congress on Evolutionary Computation*, pages 1945–1949, Piscataway, NJ, USA, 1999. IEEE.

S. Sonnenburg. Nips workshop on machine learning open source software. `http://www2.fml.tuebingen.mpg.de/raetsch/workshops/MLOSS06/`, December 2006.

J.A.K. Suykens and J. Vandewalle. Least squares support vector machine classifiers. *Neural Processing Letters*, 9(1):293–300, 1999.

F. van den Bergh. *An Analysis of Particle Swarm Optimizers*. PhD thesis, University of Pretoria, Sudafrica, November 2001.

F. van der Heijden, R. P.W. Duin, D. de Ridder, and D. M.J. Tax. Prtools: a matlab based toolbox for pattern recognition. `http://www.prtools.org/`, 2004.

M. Voss and X. Feng. Arma model selection using particle swarm optimization and aic criteria. In *Proceedings of the 15th IFAC World Congress on Automatic Control*, 2002.

J. Weston, A. Elisseeff, G. BakIr, and F. Sinz. The spider machine learning toolbox. `http://www.kyb.tuebingen.mpg.de/bs/people/spider/`, 2005.

J. Wichard. Agnostic learning with ensembles of classifiers. In *Proceedings of the 20th International Joint Conference on Neural Networks*, pages 1753–1759, 2007.

J. Wichard and C. Merkwirth. Entool - a matlab toolbox for ensemble modeling. `http://www.j-wichard.de/entool/`, 2007.

I. H. Witten and E. Frank. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, San Francisco, 2nd edition, 2005.

D. H. Wolpert and W. G. Macready. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 4:67–82, 1997.

H. Xiaohui, R. Eberhart, and Y. Shi. Engineering optimization with particle swarm. In *Proceedings of the 2003 IEEE Swarm Intelligence Symposium, 2003. (SIS03)*, pages 53–57, 2003.

H. Yoshida, K. Kawata, Y. Fukuyama, S. Takayama, and Y. Nakanishi. A particle swarm optimization for reactive power and voltage control considering voltage security assessment. *IEEE Transactions on Power Systems*, 15(4):1232–1239, Jan 2001.

Journal of Machine Learning Research 10 (2009) 441-474 Submitted 10/07; Revised 12/08; Published 2/09

Generalization Bounds for Ranking Algorithms via Algorithmic Stability[*]

Shivani Agarwal SHIVANI@MIT.EDU
Department of Electrical Engineering & Computer Science
Massachusetts Institute of Technology
Cambridge, MA 02139, USA

Partha Niyogi NIYOGI@CS.UCHICAGO.EDU
Departments of Computer Science and Statistics
University of Chicago
Chicago, IL 60637, USA

Editor: Andre Elisseeff

Abstract

The problem of ranking, in which the goal is to learn a real-valued ranking function that induces a ranking or ordering over an instance space, has recently gained much attention in machine learning. We study generalization properties of ranking algorithms using the notion of algorithmic stability; in particular, we derive generalization bounds for ranking algorithms that have good stability properties. We show that kernel-based ranking algorithms that perform regularization in a reproducing kernel Hilbert space have such stability properties, and therefore our bounds can be applied to these algorithms; this is in contrast with generalization bounds based on uniform convergence, which in many cases cannot be applied to these algorithms. Our results generalize earlier results that were derived in the special setting of bipartite ranking (Agarwal and Niyogi, 2005) to a more general setting of the ranking problem that arises frequently in applications.

Keywords: ranking, generalization bounds, algorithmic stability

1. Introduction

A central focus in learning theory research has been the study of generalization properties of learning algorithms. Perhaps the first work in this direction was that of Vapnik and Chervonenkis (1971), who derived generalization bounds for classification algorithms based on uniform convergence. Since then, a large number of different tools have been developed for studying generalization, and have been applied successfully to analyze algorithms for both classification (learning of binary-valued functions) and regression (learning of real-valued functions), two of the most well-studied problems in machine learning.

In recent years, a new learning problem, namely that of *ranking*, has gained attention in machine learning (Cohen et al., 1999; Herbrich et al., 2000; Crammer and Singer, 2002; Joachims, 2002; Freund et al., 2003; Agarwal et al., 2005; Rudin et al., 2005; Burges et al., 2005; Cossock and Zhang, 2006; Cortes et al., 2007; Clemencon et al., 2008). In ranking, one learns a real-valued function that assigns scores to instances, but the scores themselves do not matter; instead, what is

[*]. A preliminary version of this paper (which focused on the special setting of bipartite ranking) appeared in the Proceedings of the 18th Annual Conference on Learning Theory (COLT) in 2005.

important is the relative ranking of instances induced by those scores. This problem is distinct from both classification and regression, and it is natural to ask what kinds of generalization properties hold for algorithms for this problem.

Although there have been several recent advances in developing algorithms for various settings of the ranking problem, the study of generalization properties of ranking algorithms has been largely limited to the special setting of bipartite ranking (Freund et al., 2003; Agarwal et al., 2005). In this paper, we study generalization properties of ranking algorithms in a more general setting of the ranking problem that arises frequently in applications. Our generalization bounds are derived using the notion of algorithmic stability; we show that a number of practical ranking algorithms satisfy the required stability conditions, and therefore can be analyzed using our bounds.

1.1 Previous Results

While ranking has been studied extensively in some form or another in fields as diverse as social choice theory (Arrow, 1970), statistics (Lehmann, 1975), and mathematical economics (Chiang and Wainwright, 2005), the study of ranking in machine learning is relatively new: the first paper on the subject appeared less than a decade ago (Cohen et al., 1999). Since then, however, the number of domains in which ranking has found applications has grown quickly, and as a result, ranking has gained considerable attention in machine learning and learning theory in recent years.

Some of the earlier work, by Herbrich et al. (2000) and by Crammer and Singer (2002), focused on the closely related but distinct problem of ordinal regression. Freund et al. (2003) gave one of the first learning algorithms for ranking, termed RankBoost, which was based on the principles of boosting. Since then there have been many other algorithmic developments: for example, Radlinski and Joachims (2005) have developed an algorithmic framework for ranking in information retrieval applications; Burges et al. (2005) have developed a neural network based algorithm for ranking; and Agarwal (2006) has developed an algorithmic framework for ranking in a graph-based transductive setting. More recently, there has been some interest in learning ranking functions that emphasize accuracy at the top of the ranked list; work by Rudin (2006), Cossock and Zhang (2006) and Clemencon and Vayatis (2007) falls in this category. There has also been interest in statistical analysis of ranking; in recent work, Clemencon et al. (2008) have studied statistical convergence properties of ranking algorithms—specifically, ranking algorithms based on empirical and convex risk minimization—using the theory of U-statistics.

In the paper that developed the RankBoost algorithm, Freund et al. (2003) also gave a basic generalization bound for the algorithm in the bipartite setting. Their bound was derived from uniform convergence results for the binary classification error, and was expressed in terms of the VC-dimension of a class of binary classification functions derived from the class of ranking functions searched by the algorithm. Agarwal et al. (2005) also gave a generalization bound for bipartite ranking algorithms based on uniform convergence; in this case, the uniform convergence result was derived directly for the bipartite ranking error, and the resulting generalization bound was expressed in terms of a new set of combinatorial parameters that measure directly the complexity of the class of ranking functions searched by a bipartite ranking algorithm. Agarwal and Niyogi (2005) used a different tool, namely that of algorithmic stability (Rogers and Wagner, 1978; Bousquet and Elisseeff, 2002), to obtain generalization bounds for bipartite ranking algorithms that have good stability properties. Unlike bounds based on uniform convergence, the stability-based bounds depend on properties of the algorithm rather than the function class being searched, and can be applied also to algorithms that search function classes of unbounded complexity.

As can be noted from the above discussion, the question of generalization properties of ranking algorithms has so far been investigated mainly in the special setting of bipartite ranking. There have been limited studies of generalization properties in more general settings. For example, Rudin et al. (2005) derived a margin-based bound which is expressed in terms of covering numbers and relies ultimately on a uniform convergence result; this bound is derived for a non-bipartite setting of the ranking problem, but under the restrictive distributional assumption of a "truth" function. Cortes et al. (2007) consider a different setting of the ranking problem and derive stability-based generalization bounds for algorithms in this setting. However, they also implicitly assume a "truth" function. In addition, as we discuss later in the paper, the results of Cortes et al. as stated involve some strong assumptions about the function class searched by an algorithm. These assumptions rarely hold for practical ranking algorithms, which prevents the direct application of their results. We shall discuss in Section 6 how this can be remedied.

1.2 Our Results

We use the notion of algorithmic stability to study generalization properties of ranking algorithms in a more general setting of the ranking problem than has been considered previously, and that arises frequently in applications. The notion of algorithmic stability, first studied for learning algorithms by Rogers and Wagner (1978), has been used to obtain generalization bounds for classification and regression algorithms that satisfy certain stability conditions (Bousquet and Elisseeff, 2002; Kutin and Niyogi, 2002). Here we show that algorithmic stability can be useful also in analyzing generalization properties of ranking algorithms in the setting we consider; in particular, we derive generalization bounds for ranking algorithms that have good stability properties. We show that kernel-based ranking algorithms that perform regularization in a reproducing kernel Hilbert space (RKHS) have such stability properties, and therefore our bounds can be applied to these algorithms. Our techniques are based on those of Bousquet and Elisseeff (2002); indeed, we show that the ranking error in our setting satisfies the same conditions that were used to establish the classification and regression bounds of Bousquet and Elisseeff (2002), and therefore essentially the same proof techniques can be used to analyze the ranking problem we consider. Our results generalize those of Agarwal and Niyogi (2005), which focused on bipartite ranking.

We describe the general ranking problem and the setting we consider in detail in Section 2, and define notions of stability for ranking algorithms in this setting in Section 3. Using these notions, we derive generalization bounds for stable ranking algorithms in Section 4. In Section 5 we show stability of kernel-based ranking algorithms that perform regularization in an RKHS, and apply the results of Section 4 to obtain generalization bounds for these algorithms. Section 6 provides comparisons with related work; we conclude with a discussion in Section 7.

2. The Ranking Problem

In the problem of ranking, one is given a finite number of examples of order relationships among instances in some instance space $\mathcal{X}$, and the goal is to learn from these examples a ranking or ordering over $\mathcal{X}$ that ranks accurately future instances. Examples of ranking problems arise in a variety of domains: in information retrieval, one wants to rank documents according to relevance to some query or topic; in user-preference modeling, one wants to rank books or movies according to a user's likes and dislikes; in computational biology, one wants to rank genes according to their relevance to some disease.

In the most general setting of the ranking problem, the learner is given training examples in the form of ordered pairs of instances $(x,x') \in \mathcal{X} \times \mathcal{X}$ labeled with a ranking preference $r \in \mathbb{R}$, with the interpretation that x is to be ranked higher than (preferred over) x' if $r > 0$, and lower than x' if $r < 0$ ($r = 0$ indicates no ranking preference between the two instances); the penalty for mis-ordering such a pair is proportional to $|r|$. Given a finite number of such examples $((x_1,x'_1,r_1),\ldots,(x_m,x'_m,r_m))$, the goal is to learn a real-valued ranking function $f : \mathcal{X} \to \mathbb{R}$ that ranks accurately future instances; f is considered to rank an instance $x \in \mathcal{X}$ higher than an instance $x' \in \mathcal{X}$ if $f(x) > f(x')$, and lower than x' if $f(x) < f(x')$. Thus, assuming that ties are broken uniformly at random, the expected penalty (or loss) incurred by a ranking function f on a pair of instances (x,x') labeled by r can be written as

$$|r| \left(\mathbf{I}_{\{r(f(x)-f(x'))<0\}} + \frac{1}{2}\mathbf{I}_{\{f(x)=f(x')\}} \right) ,$$

where $\mathbf{I}_{\{\phi\}}$ is 1 if ϕ is true and 0 otherwise.

A particular setting of the ranking problem that has been investigated in some detail in recent years is the *bipartite* setting (Freund et al., 2003; Agarwal et al., 2005). In the bipartite ranking problem, instances come from two categories, positive and negative; the learner is given examples of instances labeled as positive or negative, and the goal is to learn a ranking in which positive instances are ranked higher than negative ones. Formally, the learner is given a training sample (S^+,S^-) consisting of a sequence of 'positive' examples $S^+ = (x_1^+,\ldots,x_m^+)$ and a sequence of 'negative' examples $S^- = (x_1^-,\ldots,x_n^-)$, the x_i^+ and x_j^- being instances in some instance space $\mathcal{X}$, and the goal is to learn a real-valued ranking function $f : \mathcal{X} \to \mathbb{R}$ that ranks future positive instances higher than negative ones. The bipartite ranking problem is easily seen to be a special case of the general ranking problem described above, since a training sample $(S^+,S^-) \in \mathcal{X}^m \times \mathcal{X}^n$ in the bipartite setting can be viewed as consisting of mn examples of the form $(x_i^+,x_j^-,1)$, for $1 \le i \le m$, $1 \le j \le n$; in other words, mis-ranking any positive-negative pair of instances incurs a constant penalty of 1. Thus, assuming again that ties are broken uniformly at random, the expected penalty incurred by f on a positive-negative pair (x^+,x^-) is simply

$$\mathbf{I}_{\{f(x^+)-f(x^-)<0\}} + \frac{1}{2}\mathbf{I}_{\{f(x^+)=f(x^-)\}} ;$$

the penalty incurred on a pair of positive instances or a pair of negative instances is zero.

In this paper, we consider a more general setting: the learner is given examples of instances labeled by real numbers, and the goal is to learn a ranking in which instances labeled by larger numbers are ranked higher than instances labeled by smaller numbers. Such ranking problems arise frequently in practice: for example, in information retrieval, one is often given examples of documents with real-valued relevance scores for a particular topic or query; similarly, in computational biology, one often receives examples of molecular structures with real-valued biological activity scores with respect to a particular target.

Formally, the setting we consider can be described as follows. The learner is given a finite sequence of labeled training examples $S = ((x_1,y_1),\ldots,(x_m,y_m))$, where the x_i are instances in some instance space $\mathcal{X}$ and the y_i are real-valued labels in some bounded set $\mathcal{Y} \subseteq \mathbb{R}$ which we take without loss of generality to be $\mathcal{Y} = [0,M]$ for some $M > 0$, and the goal is to learn a real-valued function $f : \mathcal{X} \to \mathbb{R}$ that ranks future instances with larger labels higher than those with smaller labels. The penalty for mis-ranking a pair of instances could again be taken to be constant for all pairs; here we consider the more general case where the penalty is larger for mis-ranking a pair of instances with a greater difference between their real-valued labels. In particular, in our setting, the penalty for mis-ranking a pair of instances is proportional to the absolute difference between their real-valued labels. Thus, assuming again that ties are broken uniformly at random, the expected penalty incurred by f on a pair of instances (x,x') with corresponding real-valued labels y and y' can be written as

$$|y-y'|\left(\mathbf{I}_{\{(y-y')(f(x)-f(x'))<0\}}+\frac{1}{2}\mathbf{I}_{\{f(x)=f(x')\}}\right).$$

This problem can also be seen to be a special case of the general ranking problem described above; a training sample $S=((x_1,y_1),\ldots,(x_m,y_m))\in(\mathcal{X}\times\mathcal{Y})^m$ in this setting can be viewed as consisting of $\binom{m}{2}$ examples of the form (x_i,x_j,y_i-y_j), for $1\leq i<j\leq m$.

In studying generalization properties of learning algorithms, one usually assumes that both training examples and future, unseen examples are generated according to some underlying random process. We shall assume in our setting that all examples (x,y) (both training examples and future, unseen examples) are drawn randomly and independently according to some (unknown) distribution $\mathcal{D}$ on $\mathcal{X}\times\mathcal{Y}$.[1] The quality of a ranking function $f:\mathcal{X}\rightarrow\mathbb{R}$ can then be measured by its *expected ranking error*, which we denote by $R(f)$ and define as follows:

$$R(f) \;=\; \mathbf{E}_{((X,Y),(X',Y'))\sim\mathcal{D}\times\mathcal{D}}\left[|Y-Y'|\left(\mathbf{I}_{\{(Y-Y')(f(X)-f(X'))<0\}}+\frac{1}{2}\mathbf{I}_{\{f(X)=f(X')\}}\right)\right]. \qquad (1)$$

Note that the expected error $R(f)$ is simply the expected mis-ranking penalty of f on a pair of examples drawn randomly and independently according to $\mathcal{D}$, assuming that ties are broken uniformly at random. In practice, since the distribution $\mathcal{D}$ is unknown, the expected error of a ranking function f must be estimated from an empirically observable quantity, such as its *empirical ranking error* with respect to a sample $S=((x_1,y_1),\ldots,(x_m,y_m))\in(\mathcal{X}\times\mathcal{Y})^m$, which we denote by $\widehat{R}(f;S)$ and define as follows:

$$\widehat{R}(f;S) \;=\; \frac{1}{\binom{m}{2}}\sum_{i=1}^{m-1}\sum_{j=i+1}^{m}|y_i-y_j|\left(\mathbf{I}_{\{(y_i-y_j)(f(x_i)-f(x_j))<0\}}+\frac{1}{2}\mathbf{I}_{\{f(x_i)=f(x_j)\}}\right). \qquad (2)$$

This is simply the average mis-ranking penalty incurred by f on the $\binom{m}{2}$ pairs (x_i,x_j), where $1\leq i<j\leq m$, assuming that ties are broken uniformly at random.

A learning algorithm for the ranking problem described above takes as input a training sample $S\in\bigcup_{m=1}^{\infty}(\mathcal{X}\times\mathcal{Y})^m$ and returns as output a ranking function $f_S:\mathcal{X}\rightarrow\mathbb{R}$. For simplicity we consider only deterministic algorithms. We are concerned in this paper with generalization properties of such algorithms; in particular, we are interested in bounding the expected error of a learned ranking function in terms of an empirically observable quantity such as its empirical error on the training sample from which it is learned. The following definitions will be useful in our study.

Definition 1 (Ranking loss function) *Define a* ranking loss function *to be a function* $\ell:\mathbb{R}^{\mathcal{X}}\times(\mathcal{X}\times\mathcal{Y})\times(\mathcal{X}\times\mathcal{Y})\rightarrow\mathbb{R}^+\cup\{0\}$ *that assigns to each* $f:\mathcal{X}\rightarrow\mathbb{R}$ *and* $(x,y),(x',y')\in(\mathcal{X}\times\mathcal{Y})$ *a non-negative real number* $\ell(f,(x,y),(x',y'))$*, interpreted as the penalty or loss of* f *in its relative*

1. Cortes et al. (2007) consider a similar setting as ours; however, they assume that only instances $x\in\mathcal{X}$ are drawn randomly, and that labels $y\in\mathcal{Y}$ are then determined according to a "truth" function $f^*:\mathcal{X}\rightarrow\mathcal{Y}$.

ranking of x and x' given corresponding labels y and y'. We shall require that ℓ be symmetric with respect to (x,y) and (x',y'), that is, that $\ell(f,(x,y),(x',y')) = \ell(f,(x',y'),(x,y))$ for all $f,(x,y),(x',y')$.

Definition 2 (Expected ℓ-error) *Let $f : \mathcal{X} \to \mathbb{R}$ be a ranking function on $\mathcal{X}$. Let $\ell : \mathbb{R}^{\mathcal{X}} \times (\mathcal{X} \times \mathcal{Y}) \times (\mathcal{X} \times \mathcal{Y}) \to \mathbb{R}^+ \cup \{0\}$ be a ranking loss function. Define the* expected ℓ-error *of f, denoted by $R_\ell(f)$, as*

$$R_\ell(f) \quad = \quad \mathbf{E}_{((X,Y),(X',Y')) \sim \mathcal{D} \times \mathcal{D}} \left[\ell(f,(X,Y),(X',Y')) \right] .$$

Definition 3 (Empirical ℓ-error) *Let $f : \mathcal{X} \to \mathbb{R}$ be a ranking function on $\mathcal{X}$, and let $S = ((x_1,y_1),\ldots,(x_m,y_m)) \in (\mathcal{X} \times \mathcal{Y})^m$. Let $\ell : \mathbb{R}^{\mathcal{X}} \times (\mathcal{X} \times \mathcal{Y}) \times (\mathcal{X} \times \mathcal{Y}) \to \mathbb{R}^+ \cup \{0\}$ be a ranking loss function. Define the* empirical ℓ-error *of f with respect to S, denoted by $\widehat{R}_\ell(f;S)$, as*

$$\widehat{R}_\ell(f;S) \quad = \quad \frac{1}{\binom{m}{2}} \sum_{i=1}^{m-1} \sum_{j=i+1}^{m} \ell(f,(x_i,y_i),(x_j,y_j)) .$$

As mentioned above, one choice for a ranking loss function could be a 0-1 loss that simply assigns a constant penalty of 1 to any mis-ranked pair:

$$\ell_{0\text{-}1}(f,(x,y),(x',y')) \quad = \quad \mathbf{I}_{\{(y-y')(f(x)-f(x'))<0\}} + \frac{1}{2}\mathbf{I}_{\{f(x)=f(x')\}} .$$

The (empirical) $\ell_{0\text{-}1}$-error then simply counts the fraction of mis-ranked pairs, which corresponds to the well-known Kendall τ measure. However, the 0-1 loss effectively uses only the sign of the difference $y - y'$ between labels, and ignores the magnitude of this difference; the loss function we use, which we term the *discrete ranking loss* and denote as

$$\ell_{\text{disc}}(f,(x,y),(x',y')) \quad = \quad |y-y'| \left(\mathbf{I}_{\{(y-y')(f(x)-f(x'))<0\}} + \frac{1}{2}\mathbf{I}_{\{f(x)=f(x')\}} \right) , \tag{3}$$

takes the magnitude of this difference into account. Comparing with Eqs. (1-2) above, we see that the expected and empirical ranking errors we have defined are simply the corresponding ℓ_{disc}-errors:

$$R(f) \equiv R_{\ell_{\text{disc}}}(f) ; \;\; \widehat{R}(f;S) \equiv \widehat{R}_{\ell_{\text{disc}}}(f;S) .$$

While our focus will be on bounding the expected (ℓ_{disc}-)ranking error of a learned ranking function, our results can be used also to bound the expected $\ell_{0\text{-}1}$-error.

Several other ranking loss functions will be useful in our study; these will be introduced in the following sections as needed.

3. Stability of Ranking Algorithms

A stable algorithm is one whose output does not change significantly with small changes in the input. The input to a ranking algorithm in our setting is a training sample of the form $S = ((x_1,y_1),\ldots,(x_m,y_m)) \in (\mathcal{X} \times \mathcal{Y})^m$ for some $m \in \mathbb{N}$; we consider changes to such a sample that consist of replacing a single example in the sequence with a new example. For $1 \le i \le m$ and $(x_i',y_i') \in (\mathcal{X} \times \mathcal{Y})$, we use S^i to denote the sequence obtained from S by replacing (x_i,y_i) with (x_i',y_i').

Several different notions of stability have been used in the study of classification and regression algorithms (Rogers and Wagner, 1978; Devroye and Wagner, 1979; Kearns and Ron, 1999; Bousquet and Elisseeff, 2002; Kutin and Niyogi, 2002; Poggio et al., 2004). The notions of stability that we define below for ranking algorithms in our setting are based on those defined earlier for bipartite ranking algorithms (Agarwal and Niyogi, 2005) and are most closely related to the notions of stability used by Bousquet and Elisseeff (2002).

Definition 4 (Uniform loss stability) *Let $\mathcal{A}$ be a ranking algorithm whose output on a training sample S we denote by f_S, and let ℓ be a ranking loss function. Let $\beta : \mathbb{N} \rightarrow \mathbb{R}$. We say that $\mathcal{A}$ has* uniform loss stability β *with respect to ℓ if for all $m \in \mathbb{N}$, $S \in (\mathcal{X} \times \mathcal{Y})^m$, $1 \leq i \leq m$ and $(x_i', y_i') \in (\mathcal{X} \times \mathcal{Y})$, we have for all $(x,y), (x',y') \in (\mathcal{X} \times \mathcal{Y})$,*

$$\left| \ell(f_S, (x,y), (x',y')) - \ell(f_{S^i}, (x,y), (x',y')) \right| \quad \leq \quad \beta(m) .$$

Definition 5 (Uniform score stability) *Let $\mathcal{A}$ be a ranking algorithm whose output on a training sample S we denote by f_S. Let $\nu : \mathbb{N} \rightarrow \mathbb{R}$. We say that $\mathcal{A}$ has* uniform score stability ν *if for all $m \in \mathbb{N}$, $S \in (\mathcal{X} \times \mathcal{Y})^m$, $1 \leq i \leq m$ and $(x_i', y_i') \in (\mathcal{X} \times \mathcal{Y})$, we have for all $x \in \mathcal{X}$,*

$$|f_S(x) - f_{S^i}(x)| \quad \leq \quad \nu(m) .$$

If a ranking algorithm has uniform loss stability β with respect to ℓ, then changing an input training sample of size m by a single example leads to a difference of at most $\beta(m)$ in the ℓ-loss incurred by the output ranking function on any pair of examples $(x,y), (x',y')$. Therefore, a smaller value of $\beta(m)$ corresponds to greater loss stability. Similarly, if a ranking algorithm has uniform score stability ν, then changing an input training sample of size m by a single example leads to a difference of at most $\nu(m)$ in the score assigned by the output ranking function to any instance x. A smaller value of $\nu(m)$ therefore corresponds to greater score stability.

The term 'uniform' in the above definitions refers to the fact that the bounds on the difference in loss or score are required to hold uniformly for all training samples S (and all single-example changes to them) and for all examples $(x,y), (x',y')$ or instances x. This is arguably a strong requirement; one can define weaker notions of stability, analogous to the hypothesis stability considered by Devroye and Wagner (1979), Kearns and Ron (1999), and Bousquet and Elisseeff (2002), or the almost-everywhere stability considered by Kutin and Niyogi (2002), which would require the bounds to hold only in expectation or with high probability (and would therefore depend on the distribution $\mathcal{D}$ governing the data). However, we do not consider such weaker notions of stability in this paper; we shall show later (Section 5) that several practical ranking algorithms in fact exhibit good uniform stability properties.

4. Generalization Bounds for Stable Ranking Algorithms

In this section we give generalization bounds for ranking algorithms that exhibit good (uniform) stability properties. The methods we use are based on those of Bousquet and Elisseeff (2002), who derived such bounds for classification and regression algorithms. Our main result is the following, which bounds the expected ℓ-error of a ranking function learned by an algorithm with good uniform loss stability in terms of its empirical ℓ-error on the training sample.

Theorem 6 *Let $\mathcal{A}$ be a symmetric ranking algorithm*[2] *whose output on a training sample $S \in (\mathcal{X} \times \mathcal{Y})^m$ we denote by f_S, and let ℓ be a bounded ranking loss function such that $0 \leq \ell(f,(x,y),(x',y')) \leq B$ for all $f : \mathcal{X} \to \mathbb{R}$ and $(x,y),(x',y') \in (\mathcal{X} \times \mathcal{Y})$. Let $\beta : \mathbb{N} \to \mathbb{R}$ be such that $\mathcal{A}$ has uniform loss stability β with respect to ℓ. Then for any $0 < \delta < 1$, with probability at least $1 - \delta$ over the draw of S (according to $\mathcal{D}^m$),*

$$R_\ell(f_S) \quad < \quad \widehat{R}_\ell(f_S;S) + 2\beta(m) + (m\beta(m) + B)\sqrt{\frac{2\ln(1/\delta)}{m}}\,.$$

The proof follows the proof of a similar result for classification and regression algorithms by Bousquet and Elisseeff (2002). In particular, the random variable $R_\ell(f_S) - \widehat{R}_\ell(f_S;S)$, representing the difference between the expected and empirical ℓ-errors of the ranking function f_S learned from the random sample S, is shown to satisfy the conditions of McDiarmid's inequality (McDiarmid, 1989), as a result of which the deviation of this difference from its expected value $\mathbf{E}_{S \sim \mathcal{D}^m}[R_\ell(f_S) - \widehat{R}_\ell(f_S;S)]$ can be bounded with high probability; a bound on this expected value then allows the above result to be established. Details of the proof are provided in Appendix A.

A few remarks on the significance of the above result are in order, especially in relation to the results of Bousquet and Elisseeff (2002). As can be seen from the definitions given in the preceding two sections, the main difference in the formulation of the ranking problem as compared to the problems of classification and regression is that the performance or loss in ranking is measured on *pairs* of examples, rather than on individual examples. This means in particular that, unlike the empirical error in classification or regression, the empirical error in ranking cannot be expressed as a sum of independent random variables. Indeed, this is the reason that in deriving uniform convergence bounds for the ranking error, the standard Hoeffding inequality used to obtain such bounds in classification and regression can no longer be applied (Agarwal et al., 2005). It may initially come as a bit of a surprise, therefore, that the proof methods of Bousquet and Elisseeff (2002) carry through for ranking without siginificant change. The reason for this is that they rely on the more general inequality of McDiarmid which, in Bousquet and Elisseeff's work, is used to capture the effect of stability, but which is also powerful enough to capture the structure of the ranking error; indeed, the uniform convergence bound for the (bipartite) ranking error derived by Agarwal et al. (2005) also made use of McDiarmid's inequality. Thus, in general, any learning problem in which the empirical performance measure fits the conditions of McDiarmid's inequality should be amenable to a stability analysis similar to Bousquet and Elisseeff's, provided of course that appropriate notions of stability are defined.

Theorem 6 gives meaningful bounds when $\beta(m) = o(1/\sqrt{m})$. This means the theorem cannot be applied directly to obtain bounds on the expected ranking error, since it is not possible to have non-trivial uniform loss stability with respect to the discrete ranking loss ℓ_{disc} defined in Eq. (3). (To see this, note that unless an algorithm picks essentially the same ranking function that orders all instances the same way for all training samples of a given size m, in which case the algorithm trivially has uniform loss stability $\beta(m) = 0$ with respect to ℓ_{disc}, there must be some $S \in (\mathcal{X} \times \mathcal{Y})^m$, $1 \leq i \leq m$, $(x_i', y_i') \in (\mathcal{X} \times \mathcal{Y})$ and some $x, x' \in \mathcal{X}$ such that f_S and f_{S^i} order x, x' differently. In this case, for $y = 0$ and $y' = M$, we get $|\ell_{\text{disc}}(f_S,(x,y),(x',y')) - \ell_{\text{disc}}(f_{S^i},(x,y),(x',y'))| = M$, giving loss stability $\beta(m) = M$ with respect to ℓ_{disc}.) However, for any ranking loss ℓ that satisfies $\ell_{\text{disc}} \leq \ell$,

2. A symmetric ranking algorithm is one whose output is independent of the order of elements in the training sequence S.

Theorem 6 can be applied to ranking algorithms that have good uniform loss stability with respect to ℓ to obtain bounds on the expected ℓ-error; since in this case $R \le R_\ell$, these bounds apply also to the expected ranking error. We consider below a specific ranking loss that satisfies this condition, and with respect to which we will be able to show good loss stability of certain ranking algorithms; other ranking losses which can also be used in this manner will be discussed in later sections.

For $\gamma > 0$, let the γ *ranking loss*, denoted by ℓ_γ, be defined as follows:

$$\ell_\gamma(f,(x,y),(x',y'))$$

$$= \begin{cases} |y-y'| & \text{if } \frac{(f(x)-f(x'))\cdot\mathrm{sgn}(y-y')}{\gamma} \le 0 \\ |y-y'| - \frac{(f(x)-f(x'))\cdot\mathrm{sgn}(y-y')}{\gamma} & \text{if } 0 < \frac{(f(x)-f(x'))\cdot\mathrm{sgn}(y-y')}{\gamma} < |y-y'| \\ 0 & \text{if } \frac{(f(x)-f(x'))\cdot\mathrm{sgn}(y-y')}{\gamma} \ge |y-y'|, \end{cases}$$

where for $u \in \mathbb{R}$,

$$\mathrm{sgn}(u) = \begin{cases} 1 & \text{if } u > 0 \\ 0 & \text{if } u = 0 \\ -1 & \text{if } u < 0. \end{cases}$$

Clearly, for all $\gamma > 0$, we have $\ell_{\text{disc}} \le \ell_\gamma$. Therefore, for any ranking algorithm that has good uniform loss stability with respect to ℓ_γ for some $\gamma > 0$, Theorem 6 can be applied to bound the expected ranking error of a learned ranking function in terms of its empirical ℓ_γ-error on the training sample. The following lemma shows that, for every $\gamma > 0$, a ranking algorithm that has good uniform score stability also has good uniform loss stability with respect to ℓ_γ.

Lemma 7 *Let $\mathcal{A}$ be a ranking algorithm whose output on a training sample $S \in (\mathcal{X} \times \mathcal{Y})^m$ we denote by f_S. Let $\nu : \mathbb{N} \to \mathbb{R}$ be such that $\mathcal{A}$ has uniform score stability ν. Then for every $\gamma > 0$, $\mathcal{A}$ has uniform loss stability β_γ with respect to the γ ranking loss ℓ_γ, where for all $m \in \mathbb{N}$,*

$$\beta_\gamma(m) = \frac{2\nu(m)}{\gamma}.$$

Proof Let $m \in \mathbb{N}$, $S \in (\mathcal{X} \times \mathcal{Y})^m$, $1 \le i \le m$ and $(x'_i, y'_i) \in (\mathcal{X} \times \mathcal{Y})$. Let $(x,y),(x',y') \in (\mathcal{X} \times \mathcal{Y})$. We want to show

$$\left|\ell_\gamma(f_S,(x,y),(x',y')) - \ell_\gamma(f_{S^i},(x,y),(x',y'))\right| \le \frac{2\nu(m)}{\gamma}.$$

Now, if $\ell_\gamma(f_S,(x,y),(x',y')) = \ell_\gamma(f_{S^i},(x,y),(x',y'))$, then trivially

$$\left|\ell_\gamma(f_S,(x,y),(x',y')) - \ell_\gamma(f_{S^i},(x,y),(x',y'))\right| = 0 \le \frac{2\nu(m)}{\gamma},$$

and there is nothing to prove. Therefore assume $\ell_\gamma(f_S,(x,y),(x',y')) \ne \ell_\gamma(f_{S^i},(x,y),(x',y'))$. Without loss of generality, let $\ell_\gamma(f_S,(x,y),(x',y')) > \ell_\gamma(f_{S^i},(x,y),(x',y'))$. There are four possibilities:

(i) $\frac{(f_S(x)-f_S(x'))\cdot\mathrm{sgn}(y-y')}{\gamma} \le 0$ and $0 < \frac{(f_{S^i}(x)-f_{S^i}(x'))\cdot\mathrm{sgn}(y-y')}{\gamma} < |y-y'|$.

In this case, we have

$$\begin{aligned}
&\left|\ell_\gamma(f_S,(x,y),(x',y')) - \ell_\gamma(f_{S^i},(x,y),(x',y'))\right| \\
&\quad = \; |y-y'| - \left(|y-y'| - \tfrac{(f_{S^i}(x)-f_{S^i}(x'))\cdot\mathrm{sgn}(y-y')}{\gamma}\right) \\
&\quad \leq \; \left(|y-y'| - \tfrac{(f_S(x)-f_S(x'))\cdot\mathrm{sgn}(y-y')}{\gamma}\right) - \left(|y-y'| - \tfrac{(f_{S^i}(x)-f_{S^i}(x'))\cdot\mathrm{sgn}(y-y')}{\gamma}\right) \\
&\quad \leq \; \frac{1}{\gamma}\left(|f_S(x)-f_{S^i}(x)| + \left|f_S(x')-f_{S^i}(x')\right|\right) \\
&\quad \leq \; \frac{2\nu(m)}{\gamma}.
\end{aligned}$$

(ii) $0 < \frac{(f_S(x)-f_S(x'))\cdot\mathrm{sgn}(y-y')}{\gamma} < |y-y'|$ and $\frac{(f_{S^i}(x)-f_{S^i}(x'))\cdot\mathrm{sgn}(y-y')}{\gamma} \geq |y-y'|$.

In this case, we have

$$\begin{aligned}
&\left|\ell_\gamma(f_S,(x,y),(x',y')) - \ell_\gamma(f_{S^i},(x,y),(x',y'))\right| \\
&\quad = \; \left(|y-y'| - \tfrac{(f_{S^i}(x)-f_{S^i}(x'))\cdot\mathrm{sgn}(y-y')}{\gamma}\right) - 0 \\
&\quad \leq \; \left(|y-y'| - \tfrac{(f_S(x)-f_S(x'))\cdot\mathrm{sgn}(y-y')}{\gamma}\right) - \left(|y-y'| - \tfrac{(f_{S^i}(x)-f_{S^i}(x'))\cdot\mathrm{sgn}(y-y')}{\gamma}\right) \\
&\quad \leq \; \frac{1}{\gamma}\left(|f_S(x)-f_{S^i}(x)| + \left|f_S(x')-f_{S^i}(x')\right|\right) \\
&\quad \leq \; \frac{2\nu(m)}{\gamma}.
\end{aligned}$$

(iii) $\frac{(f_S(x)-f_S(x'))\cdot\mathrm{sgn}(y-y')}{\gamma} \leq 0$ and $\frac{(f_{S^i}(x)-f_{S^i}(x'))\cdot\mathrm{sgn}(y-y')}{\gamma} \geq |y-y'|$.

In this case, we have

$$\begin{aligned}
&\left|\ell_\gamma(f_S,(x,y),(x',y')) - \ell_\gamma(f_{S^i},(x,y),(x',y'))\right| \\
&\quad = \; |y-y'| - 0 \\
&\quad \leq \; \left(|y-y'| - \tfrac{(f_S(x)-f_S(x'))\cdot\mathrm{sgn}(y-y')}{\gamma}\right) - \left(|y-y'| - \tfrac{(f_{S^i}(x)-f_{S^i}(x'))\cdot\mathrm{sgn}(y-y')}{\gamma}\right) \\
&\quad \leq \; \frac{1}{\gamma}\left(|f_S(x)-f_{S^i}(x)| + \left|f_S(x')-f_{S^i}(x')\right|\right) \\
&\quad \leq \; \frac{2\nu(m)}{\gamma}.
\end{aligned}$$

(iv) $0 < \frac{(f_S(x)-f_S(x'))\cdot\mathrm{sgn}(y-y')}{\gamma} < |y-y'|$ and $0 < \frac{(f_{S^i}(x)-f_{S^i}(x'))\cdot\mathrm{sgn}(y-y')}{\gamma} < |y-y'|$.

In this case, we have

$$\left|\ell_\gamma(f_S,(x,y),(x',y')) - \ell_\gamma(f_{S^i},(x,y),(x',y'))\right|$$

$$\begin{aligned}
&= \left(|y-y'| - \tfrac{(f_S(x)-f_S(x'))\cdot \text{sgn}(y-y')}{\gamma}\right) - \left(|y-y'| - \tfrac{(f_{S^i}(x)-f_{S^i}(x'))\cdot \text{sgn}(y-y')}{\gamma}\right) \\
&\leq \frac{1}{\gamma}\left(|f_S(x)-f_{S^i}(x)| + \left|f_S(x')-f_{S^i}(x')\right|\right) \\
&\leq \frac{2\nu(m)}{\gamma}\,.
\end{aligned}$$

Thus in all cases, $\left|\ell_\gamma(f_S,(x,y),(x',y')) - \ell_\gamma(f_{S^i},(x,y),(x',y'))\right| \leq \frac{2\nu(m)}{\gamma}$. ■

Putting everything together, we thus get the following result which bounds the expected ranking error of a learned ranking function in terms of its empirical ℓ_γ-error for any ranking algorithm that has good uniform score stability.

Theorem 8 *Let $\mathcal{A}$ be a symmetric ranking algorithm whose output on a training sample $S \in (\mathcal{X} \times \mathcal{Y})^m$ we denote by f_S. Let $\nu : \mathbb{N} \to \mathbb{R}$ be such that $\mathcal{A}$ has uniform score stability ν, and let $\gamma > 0$. Then for any $0 < \delta < 1$, with probability at least $1-\delta$ over the draw of S (according to $\mathcal{D}^m$),*

$$R(f_S) \quad < \quad \widehat{R}_{\ell_\gamma}(f_S;S) + \frac{4\nu(m)}{\gamma} + \left(\frac{2m\nu(m)}{\gamma} + M\right)\sqrt{\frac{2\ln(1/\delta)}{m}}\,.$$

Proof The result follows by applying Theorem 6 to $\mathcal{A}$ with the ranking loss ℓ_γ (using Lemma 7), which satisfies $0 \leq \ell_\gamma \leq M$, and from the fact that $R \leq R_{\ell_\gamma}$. ■

5. Stable Ranking Algorithms

In this section we show (uniform) stability of certain ranking algorithms that select a ranking function by minimizing a regularized objective function. We start by deriving a general result for regularization-based ranking algorithms in Section 5.1. In Section 5.2 we use this result to show stability of kernel-based ranking algorithms that perform regularization in a reproducing kernel Hilbert space (RKHS). We show, in particular, stability of two such ranking algorithms, and apply the results of Section 4 to obtain generalization bounds for these algorithms. Again, the methods we use to show stability of these algorithms are based on those of Bousquet and Elisseeff (2002), who showed similar results for classification and regression algorithms. We also use these stability results to give a consistency theorem for kernel-based ranking algorithms (Section 5.2.3).

5.1 General Regularizers

Given a ranking loss function ℓ, a class $\mathcal{F}$ of real-valued functions on $\mathcal{X}$, and a regularization functional $N : \mathcal{F} \to \mathbb{R}^+ \cup \{0\}$, consider the following regularized empirical ℓ-error of a ranking function $f \in \mathcal{F}$ (with respect to a sample $S \in (\mathcal{X} \times \mathcal{Y})^m$), with regularization parameter $\lambda > 0$:

$$\widehat{R}^\lambda_\ell(f;S) \quad = \quad \widehat{R}_\ell(f;S) + \lambda N(f)\,.$$

We consider ranking algorithms that minimize such a regularized objective function, that is, ranking algorithms that, given a training sample S, output a ranking function $f_S \in \mathcal{F}$ that satisfies

$$f_S = \arg\min_{f\in\mathcal{F}} \widehat{R}_\ell^\lambda(f;S), \tag{4}$$

for some fixed choice of ranking loss ℓ, function class $\mathcal{F}$, regularizer N, and regularization parameter λ. It is worth noting that the parameter λ often depends on the sample size m; we use λ here rather than λ_m for notational simplicity. We give below a general result that will be useful for showing stability of such regularization-based algorithms.

Definition 9 (σ-admissibility) *Let $\mathcal{F}$ be a class of real-valued functions on $\mathcal{X}$. Let ℓ be a ranking loss and let $\sigma > 0$. We say that ℓ is* σ-admissible *with respect to $\mathcal{F}$ if for all $f_1, f_2 \in \mathcal{F}$ and all $(x,y),(x',y') \in (\mathcal{X}\times\mathcal{Y})$, we have*

$$\left|\ell(f_1,(x,y),(x',y')) - \ell(f_2,(x,y),(x',y'))\right| \le \sigma\Big(\left|f_1(x)-f_2(x)\right| + \left|f_1(x')-f_2(x')\right|\Big).$$

Lemma 10 *Let $\mathcal{F}$ be a convex class of real-valued functions on $\mathcal{X}$. Let ℓ be a ranking loss such that $\ell(f,(x,y),(x',y'))$ is convex in f, and let $\sigma > 0$ be such that ℓ is σ-admissible with respect to $\mathcal{F}$. Let $\lambda > 0$, and let $N : \mathcal{F}\to\mathbb{R}^+\cup\{0\}$ be a functional defined on $\mathcal{F}$ such that for all samples $S \in (\mathcal{X}\times\mathcal{Y})^m$, the regularized empirical ℓ-error $\widehat{R}_\ell^\lambda(f;S)$ has a minimum (not necessarily unique) in $\mathcal{F}$. Let $\mathcal{A}$ be a ranking algorithm defined by Eq. (4), and let $S = ((x_1,y_1),\ldots,(x_m,y_m)) \in (\mathcal{X}\times\mathcal{Y})^m$, $1 \le i \le m$, and $(x_i',y_i') \in (\mathcal{X}\times\mathcal{Y})$. For brevity, denote $f \equiv f_S$, $f^i \equiv f_{S^i}$, and let $\Delta f = f^i - f$. Then we have that for any $t \in [0,1]$,*

$$\begin{aligned} N(f) - N(f+t\Delta f) + N(f^i) - N(f^i - t\Delta f) \\ \le \frac{t\sigma}{\lambda\binom{m}{2}} \sum_{j\ne i}\Big(|\Delta f(x_i)| + 2|\Delta f(x_j)| + |\Delta f(x_i')|\Big). \end{aligned}$$

The proof follows that of a similar result for regularization-based algorithms for classification and regression (Bousquet and Elisseeff, 2002); it is based crucially on the convexity of the ranking loss ℓ. Details are provided in Appendix B.

As we shall see below, the above result can be used to establish stability of certain regularization-based ranking algorithms.

5.2 Regularization in Hilbert Spaces

Let $\mathcal{F}$ be a reproducing kernel Hilbert space (RKHS) of real-valued functions on $\mathcal{X}$, with kernel $K : \mathcal{X}\times\mathcal{X}\to\mathbb{R}$. Then the reproducing property of $\mathcal{F}$ gives that for all $f \in \mathcal{F}$ and all $x \in \mathcal{X}$,

$$|f(x)| = |\langle f, K_x\rangle_K|, \tag{5}$$

where $K_x : \mathcal{X}\to\mathbb{R}$ is defined as

$$K_x(x') = K(x,x'),$$

and $\langle\cdot,\cdot\rangle_K$ denotes the RKHS inner product in $\mathcal{F}$. In particular, applying the Cauchy-Schwartz inequality to Eq. (5) then gives that for all $f \in \mathcal{F}$ and all $x \in X$,

$$\begin{aligned} |f(x)| &\le \|f\|_K\,\|K_x\|_K \\ &= \|f\|_K\sqrt{K(x,x)}, \end{aligned} \tag{6}$$

where $\|\cdot\|_K$ denotes the RKHS norm in $\mathcal{F}$.

Further details about RKHSs can be found, for example, in expositions by Haussler (1999) and Evgeniou et al. (2000). We shall consider ranking algorithms that perform regularization in the RKHS $\mathcal{F}$ using the squared norm in $\mathcal{F}$ as a regularizer. Specifically, let $N : \mathcal{F} \rightarrow \mathbb{R}^+ \cup \{0\}$ be the regularizer defined by

$$N(f) \quad = \quad \|f\|_K^2 .$$

We show below that, if the kernel K is such that $K(x,x)$ is bounded for all $x \in \mathcal{X}$, then a ranking algorithm that minimizes an appropriate regularized error over $\mathcal{F}$, with regularizer N as defined above, has good uniform score stability.

Theorem 11 *Let $\mathcal{F}$ be an RKHS with kernel K such that for all $x \in \mathcal{X}$, $K(x,x) \leq \kappa^2 < \infty$. Let ℓ be a ranking loss such that $\ell(f,(x,y),(x',y'))$ is convex in f and ℓ is σ-admissible with respect to $\mathcal{F}$. Let $\lambda > 0$, and let $\mathcal{A}$ be a ranking algorithm that, given a training sample S, outputs a ranking function $f_S \in \mathcal{F}$ that satisfies $f_S = \arg\min_{f \in \mathcal{F}} \left\{ \widehat{R}_\ell(f;S) + \lambda \|f\|_K^2 \right\}$. Then $\mathcal{A}$ has uniform score stability ν, where for all $m \in \mathbb{N}$,*

$$\nu(m) = \frac{8\sigma\kappa^2}{\lambda m} .$$

Proof Let $m \in \mathbb{N}$, $S = ((x_1,y_1),\ldots,(x_m,y_m)) \in (\mathcal{X} \times \mathcal{Y})^m$, $1 \leq i \leq m$, and $(x_i',y_i') \in (\mathcal{X} \times \mathcal{Y})$. Applying Lemma 10 with $t = 1/2$, we get (using the notation of Lemma 10) that

$$\begin{aligned} &\|f\|_K^2 - \|f + \tfrac{1}{2}\Delta f\|_K^2 + \|f^i\|_K^2 - \|f^i - \tfrac{1}{2}\Delta f\|_K^2 \\ &\qquad \leq \frac{\sigma}{\lambda m(m-1)} \sum_{j \neq i} \Big(|\Delta f(x_i)| + 2|\Delta f(x_j)| + |\Delta f(x_i')| \Big) . \end{aligned} \tag{7}$$

Note that since $\mathcal{F}$ is a vector space, $\Delta f \in \mathcal{F}$, $(f + \frac{1}{2}\Delta f) \in \mathcal{F}$, and $(f^i - \frac{1}{2}\Delta f) \in \mathcal{F}$, so that $\|f + \frac{1}{2}\Delta f\|_K$ and $\|f^i - \frac{1}{2}\Delta f\|_K$ are well-defined. Now, we have

$$\begin{aligned} &\|f\|_K^2 - \|f + \tfrac{1}{2}\Delta f\|_K^2 + \|f^i\|_K^2 - \|f^i - \tfrac{1}{2}\Delta f\|_K^2 \\ &\qquad = \|f\|_K^2 + \|f^i\|_K^2 - \frac{1}{2}\|f + f^i\|_K^2 \\ &\qquad = \frac{1}{2}\|f\|_K^2 + \frac{1}{2}\|f^i\|_K^2 - \langle f, f^i \rangle_K \\ &\qquad = \frac{1}{2}\|\Delta f\|_K^2 . \end{aligned}$$

Combined with Eq. (7), this gives

$$\frac{1}{2}\|\Delta f\|_K^2 \quad \leq \quad \frac{\sigma}{\lambda m(m-1)} \sum_{j \neq i} \Big(|\Delta f(x_i)| + 2|\Delta f(x_j)| + |\Delta f(x_i')| \Big) .$$

Since (as noted above) $\Delta f \in \mathcal{F}$, by Eq. (6), we thus get that

$$\begin{aligned} \frac{1}{2}\|\Delta f\|_K^2 \quad &\leq \quad \frac{\sigma}{\lambda m(m-1)} \|\Delta f\|_K \sum_{j \neq i} \left(\sqrt{K(x_i,x_i)} + 2\sqrt{K(x_j,x_j)} + \sqrt{K(x_i',x_i')} \right) \\ &\leq \quad \frac{4\sigma\kappa}{\lambda m} \|\Delta f\|_K , \end{aligned}$$

which gives

$$\|\Delta f\|_K \;\leq\; \frac{8\sigma\kappa}{\lambda m}\,. \tag{8}$$

Thus, by Eqs. (6) and (8), we have for all $x \in \mathcal{X}$,

$$|f_S(x) - f_{S^i}(x)| = |\Delta f(x)| \;\leq\; \frac{8\sigma\kappa^2}{\lambda m}\,.$$

The result follows. ■

This gives the following generalization bound for kernel-based ranking algorithms:

Corollary 12 *Under the conditions of Theorem 11, we have that for any* $0 < \delta < 1$*, with probability at least* $1 - \delta$ *over the draw of* S *(according to* $\mathcal{D}^m$*), the expected ranking error of the ranking function* f_S *learned by* $\mathcal{A}$ *is bounded by*

$$R(f_S) \;<\; \widehat{R}_{\ell_1}(f_S;S) + \frac{32\sigma\kappa^2}{\lambda m} + \left(\frac{16\sigma\kappa^2}{\lambda} + M\right)\sqrt{\frac{2\ln(1/\delta)}{m}}\,.$$

Proof The result follows from Theorem 11, by applying Theorem 8 with $\gamma = 1$. ■

Under the conditions of the above results, a kernel-based ranking algorithm minimizing a regularized empirical ℓ-error also has good uniform loss stability with respect to ℓ; this follows from the following simple lemma:[3]

Lemma 13 *Let* $\mathcal{F}$ *be a class of real-valued functions on* $\mathcal{X}$*, and let* $\mathcal{A}$ *be a ranking algorithm that, given a training sample* S*, outputs a ranking function* $f_S \in \mathcal{F}$*. If* $\mathcal{A}$ *has uniform score stability* ν *and* ℓ *is a ranking loss that is* σ*-admissible with respect to* $\mathcal{F}$*, then* $\mathcal{A}$ *has uniform loss stability* β *with respect to* ℓ*, where for all* $m \in \mathbb{N}$*,*

$$\beta(m) \;=\; 2\sigma\nu(m)\,.$$

Proof Let $m \in \mathbb{N}$, $S \in (\mathcal{X} \times \mathcal{Y})^m$, $1 \leq i \leq m$ and $(x_i', y_i') \in (\mathcal{X} \times \mathcal{Y})$. Let $(x,y), (x',y') \in (\mathcal{X} \times \mathcal{Y})$. Then we have

$$\begin{aligned}
\left|\ell(f_S,(x,y),(x',y')) - \ell(f_{S^i},(x,y),(x',y'))\right| \;&\leq\; \sigma\Big(|f_S(x) - f_{S^i}(x)| + |f_S(x') - f_{S^i}(x')|\Big) \\
&\leq\; 2\sigma\nu(m)\,,
\end{aligned}$$

where the first inequality follows from σ-admissibility and the second from uniform score stability. ■

3. We note that the proof of Lemma 7 in Section 4 really amounts to showing that ℓ_γ is $\frac{1}{\gamma}$-admissible with respect to the set of all ranking functions on $\mathcal{X}$; the result then follows by the observation in Lemma 13.

Corollary 14 *Under the conditions of Theorem 11, $\mathcal{A}$ has uniform loss stability* β *with respect to ℓ, where for all $m \in \mathbb{N}$,*

$$\beta(m) = \frac{16\sigma^2\kappa^2}{\lambda m}.$$

Proof Immediate from Theorem 11 and Lemma 13. ■

This leads to the following additional bound for kernel-based ranking algorithms that minimize a regularized empirical ℓ-error where ℓ is bounded and satisfies $\ell_{\text{disc}} \leq \ell$:

Corollary 15 *Under the conditions of Theorem 11, if in addition the ranking loss ℓ satisfies $\ell_{\text{disc}}(f,(x,y),(x',y')) \leq \ell(f,(x,y),(x',y')) \leq B$ for all $f : \mathcal{X} \to \mathbb{R}$ and $(x,y),(x',y') \in (\mathcal{X} \times \mathcal{Y})$, then we have that for any $0 < \delta < 1$, with probability at least $1-\delta$ over the draw of S (according to $\mathcal{D}^m$), the expected ranking error of the ranking function f_S learned by $\mathcal{A}$ is bounded by*

$$R(f_S) \quad < \quad \widehat{R}_\ell(f_S;S) + \frac{32\sigma^2\kappa^2}{\lambda m} + \left(\frac{16\sigma^2\kappa^2}{\lambda} + B\right)\sqrt{\frac{2\ln(1/\delta)}{m}}.$$

Proof The result follows from Corollary 14, by applying Theorem 6 and the fact that $R \leq R_\ell$. ■

The results of both Corollary 12 and Corollary 15 show that a larger regularization parameter λ leads to better stability and, therefore, a tighter confidence interval in the resulting generalization bound. In particular, when σ is independent of λ, one must have $\lambda \gg \frac{1}{\sqrt{m}}$ in order for either of these bounds to be meaningful. In practice, the ranking losses ℓ minimized by kernel-based ranking algorithms tend to be larger than the loss ℓ_1, and therefore Corollary 12 tends to provide tighter bounds on the expected ranking error than Corollary 15. Below we look at two specific algorithms that minimize two different (regularized) ranking losses.

5.2.1 HINGE RANKING LOSS

Consider the following ranking loss function, which we refer to as the *hinge ranking loss* due to its similarity to the hinge loss in classification:

$$\ell_{\text{h}}(f,(x,y),(x',y')) \quad = \quad \Big(|y-y'| - (f(x)-f(x'))\cdot \text{sgn}(y-y')\Big)_+,$$

where for $a \in \mathbb{R}$,

$$a_+ \quad = \quad \begin{cases} a & \text{if } a > 0 \\ 0 & \text{otherwise.} \end{cases}$$

We consider a ranking algorithm $\mathcal{A}_{\text{h}}$ that minimizes the regularized ℓ_h-error in an RKHS $\mathcal{F}$. Specifically, let $\mathcal{A}_{\text{h}}$ be a ranking algorithm which, given a training sample $S \in (\mathcal{X} \times \mathcal{Y})^m$, outputs a ranking function $f_S \in \mathcal{F}$ that satisfies (for some fixed $\lambda > 0$)

$$f_S \quad = \quad \arg\min_{f\in\mathcal{F}} \left\{ \widehat{R}_{\ell_{\text{h}}}(f;S) + \lambda\|f\|_K^2 \right\}.$$

We note that this algorithm has an equivalent quadratic programming formulation similar to SVMs in the case of classification. In particular, the problem of minimizing $\widehat{R}^{\lambda}_{\ell_h}(f;S)$ is equivalent to that of minimizing

$$\frac{1}{2}\|f\|_K^2 + C\sum_{i=1}^{m-1}\sum_{j=i+1}^{m}\xi_{ij}$$

subject to

$$\begin{aligned}\xi_{ij} &\geq |y_i - y_j| - (f(x_i) - f(x_j)\cdot \mathrm{sgn}(y_i - y_j))\\ \xi_{ij} &\geq 0\end{aligned}$$

for all $1 \leq i < j \leq m$, where $C = 1/(\lambda m(m-1))$. The dual formulation of this problem obtained by introducing Lagrange multipliers leads to a quadratic program similar to that obtained for SVMs; for example, see the related algorithms described by Herbrich et al. (1998, 2000), Joachims (2002), Rakotomamonjy (2004), and Agarwal (2006).

It can be verified that $\ell_h(f,(x,y),(x',y'))$ is convex in f, and that ℓ_h is 1-admissible with respect to $\mathcal{F}$ (the proof of 1-admissibility is similar to the proof of Lemma 7 which, as noted earlier in Footnote 3, effectively shows $\frac{1}{\gamma}$-admissibility of the γ ranking loss ℓ_γ). Thus, if $K(x,x) \leq \kappa^2$ for all $x \in \mathcal{X}$, then from Corollary 12 we get that for any $0 < \delta < 1$, with probability at least $1-\delta$ over the draw of S (according to $\mathcal{D}^m$), the expected ranking error of the ranking function f_S learned by the above algorithm $\mathcal{A}_h$ is bounded by

$$R(f_S) \quad < \quad \widehat{R}_{\ell_1}(f_S;S) + \frac{32\kappa^2}{\lambda m} + \left(\frac{16\kappa^2}{\lambda} + M\right)\sqrt{\frac{2\ln(1/\delta)}{m}}\,.$$

5.2.2 LEAST SQUARES RANKING LOSS

Consider now the following *least squares ranking loss*:

$$\ell_{sq}(f,(x,y),(x',y')) \quad = \quad \Big(|y-y'| - \mathrm{sgn}(y-y')\cdot(f(x)-f(x'))\Big)^2\,. \tag{9}$$

Let $\mathcal{A}_{sq}$ be a ranking algorithm that minimizes the regularized ℓ_{sq}-error in an RKHS $\mathcal{F}$, that is, given a training sample $S \in (\mathcal{X}\times\mathcal{Y})^m$, the algorithm $\mathcal{A}_{sq}$ outputs a ranking function $f_S \in \mathcal{F}$ that satisfies (for some fixed $\lambda > 0$)

$$f_S \quad = \quad \arg\min_{f\in\mathcal{F}}\left\{\widehat{R}_{\ell_{sq}}(f;S) + \lambda\|f\|_K^2\right\}.$$

It can be verified that $\ell_{sq}(f,(x,y),(x',y'))$ is convex in f. Now, we claim that the effective search space of $\mathcal{A}_{sq}$ on training samples of size m is actually $\mathcal{F}_\lambda = \left\{f \in \mathcal{F} \,\middle|\, \|f\|_K^2 \leq \frac{M^2}{\lambda}\right\}$, that is, that for any training sample $S = ((x_1,y_1),\ldots,(x_m,y_m)) \in (\mathcal{X}\times\mathcal{Y})^m$, the ranking function f_S returned by $\mathcal{A}_{sq}$ satisfies $\|f_S\|_K^2 \leq \frac{M^2}{\lambda}$. To see this, note that for the zero function $f_0 \in \mathcal{F}$ which assigns $f_0(x) = 0$ for all $x \in \mathcal{X}$, we have

$$\begin{aligned}\widehat{R}_{\ell_{sq}}(f_0;S) + \lambda\|f_0\|_K^2 &= \frac{1}{\binom{m}{2}}\sum_{i=1}^{m-1}\sum_{j=i+1}^{m}(|y_i-y_j|-0)^2 + \lambda\cdot 0\\ &\leq M^2\,.\end{aligned}$$

Therefore, by definition of f_S,

$$\widehat{R}_{\ell_{\text{sq}}}(f_S;S) + \lambda \|f_S\|_K^2 \;\; \leq \;\; M^2 .$$

Since $\widehat{R}_{\ell_{\text{sq}}}(f_S;S) \geq 0$, this implies

$$\|f_S\|_K^2 \;\; \leq \;\; \frac{M^2}{\lambda} .$$

It is easily verified that the effective search space $\mathcal{F}_\lambda$ is a convex set. The following lemma shows that if $K(x,x) \leq \kappa^2$ for all $x \in \mathcal{X}$, then ℓ_{sq} is $(2M + \frac{4\kappa M}{\sqrt{\lambda}})$-admissible with respect to $\mathcal{F}_\lambda$.

Lemma 16 *Let $\mathcal{F}$ be an RKHS with kernel K such that for all $x \in \mathcal{X}$, $K(x,x) \leq \kappa^2 < \infty$. Let $\lambda > 0$, and let $\mathcal{F}_\lambda = \left\{ f \in \mathcal{F} \;\middle|\; \|f\|_K^2 \leq \frac{M^2}{\lambda} \right\}$. Then the least squares ranking loss ℓ_{sq}, defined in Eq. (9) above, is $(2M + \frac{4\kappa M}{\sqrt{\lambda}})$-admissible with respect to $\mathcal{F}_\lambda$.*

Proof First, note that for any $f \in \mathcal{F}_\lambda$ and any $x \in \mathcal{X}$, we have from Eq. (6) that

$$|f(x)| \;\leq\; \kappa \|f\|_K \;\leq\; \frac{\kappa M}{\sqrt{\lambda}} .$$

Now, let $f_1, f_2 \in \mathcal{F}_\lambda$, and let $(x,y),(x',y') \in (\mathcal{X} \times \mathcal{Y})$. Then we have,

$$\begin{aligned}
&\Big| \ell_{\text{sq}}(f_1,(x,y),(x',y')) - \ell_{\text{sq}}(f_1,(x,y),(x',y')) \Big| \\
&\quad = \;\; \Big| \Big((y-y') - (f_1(x) - f_1(x')) \Big)^2 - \Big((y-y') - (f_2(x) - f_2(x')) \Big)^2 \Big| \\
&\quad = \;\; \Big| 2(y-y') - (f_1(x) - f_1(x')) - (f_2(x) - f_2(x')) \Big| \cdot \Big| (f_2(x) - f_2(x')) - (f_1(x) - f_1(x')) \Big| \\
&\quad \leq \;\; \Big(2|y-y'| + |f_1(x)| + |f_1(x')| + |f_2(x)| + |f_2(x')| \Big) \cdot \Big(|f_1(x) - f_2(x)| + |f_1(x') - f_2(x')| \Big) \\
&\quad \leq \;\; \Big(2M + \frac{4\kappa M}{\sqrt{\lambda}} \Big) \cdot \Big(|f_1(x) - f_2(x)| + |f_1(x') - f_2(x')| \Big) ,
\end{aligned}$$

where the second equality follows from the identity $|a^2 - b^2| = |a+b| \cdot |a-b|$. ■

An analysis of the proof of Lemma 10 shows that if a regularization-based ranking algorithm minimizing a regularized ℓ-error in some convex function class $\mathcal{F}$ is such that the ranking function returned by it for training samples of a given size m always lies in some effective search space $\mathcal{F}_m \subseteq \mathcal{F}$ that is also convex, then the result of the lemma holds even when for each m, the loss function ℓ is σ_m-admissible, for some $\sigma_m > 0$, only with respect to the smaller function class $\mathcal{F}_m$. This in turn implies more general versions of Theorem 11 and Corollary 12, in which the same results hold even if a ranking algorithm minimizing a regularized ℓ-error in an RKHS $\mathcal{F}$ is such that for each m, ℓ is σ_m-admissible with respect to a convex subset $\mathcal{F}_m \subseteq \mathcal{F}$ that serves as an effective search space for training samples of size m. Thus, from Lemma 16 and the discussion preceding it, it follows that if $K(x,x) \leq \kappa^2$ for all $x \in \mathcal{X}$, then we can apply (the more general version of) Corollary 12 with $\sigma_m = (2M + \frac{4\kappa M}{\sqrt{\lambda}})$ (recall that $\lambda \equiv \lambda_m$ may depend on the sample size m) to get that for any $0 < \delta < 1$, with probability at least $1-\delta$ over the draw of S (according to $\mathcal{D}^m$), the expected ranking error of the ranking function f_S learned by the algorithm $\mathcal{A}_{\text{sq}}$ is bounded by

$$R(f_S) \quad < \quad \widehat{R}_{\ell_1}(f_S;S) + \frac{64\kappa^2 M}{\lambda m}\Big(1+\frac{2\kappa}{\sqrt{\lambda}}\Big) + \left(\frac{32\kappa^2 M}{\lambda}\Big(1+\frac{2\kappa}{\sqrt{\lambda}}\Big)+M\right)\sqrt{\frac{2\ln(1/\delta)}{m}}\,.$$

5.2.3 CONSISTENCY

We can also use the above results to show consistency of kernel-based ranking algorithms. In particular, let $R^*_\ell(\mathcal{F})$ denote the optimal expected ℓ-error in an RKHS $\mathcal{F}$ (for a given distribution $\mathcal{D}$):

$$R^*_\ell(\mathcal{F}) \quad = \quad \inf_{f\in\mathcal{F}} R_\ell(f)\,.$$

Then for bounded loss functions ℓ, we can show that with an appropriate choice of the regularization parameter λ, the expected ℓ-error $R_\ell(f_S)$ of the ranking function f_S learned by a kernel-based ranking algorithm that minimizes a regularized empirical ℓ-error in $\mathcal{F}$ converges (in probability) to this optimal value. To show this, we shall need the following simple lemma:

Lemma 17 *Let $f : \mathcal{X}\to\mathbb{R}$ be a fixed ranking function, and let ℓ be a bounded ranking loss function such that $0 \le \ell(f,(x,y),(x',y')) \le B$ for all $f : \mathcal{X}\to\mathbb{R}$ and $(x,y),(x',y') \in (\mathcal{X}\times\mathcal{Y})$. Then for any $0<\delta<1$, with probability at least $1-\delta$ over the draw of $S \in (\mathcal{X}\times\mathcal{Y})^m$ (according to $\mathcal{D}^m$),*

$$\widehat{R}_\ell(f;S) \quad < \quad R_\ell(f) + B\sqrt{\frac{2\ln(1/\delta)}{m}}\,.$$

The proof involves a simple application of McDiarmid's inequality to the random variable $\widehat{R}_\ell(f;S)$; details are provided in Appendix C. We then have the following consistency result:

Theorem 18 *Let $\mathcal{F}$ be an RKHS with kernel K such that for all $x\in\mathcal{X}$, $K(x,x)\le\kappa^2<\infty$. Let ℓ be a ranking loss such that $\ell(f,(x,y),(x',y'))$ is convex in f and ℓ is σ-admissible with respect to $\mathcal{F}$. Furthermore, let ℓ be bounded such that $0 \le \ell(f,(x,y),(x',y')) \le B$ for all $f : \mathcal{X}\to\mathbb{R}$ and $(x,y),(x',y') \in (\mathcal{X}\times\mathcal{Y})$, and let f^*_ℓ be any fixed function in $\mathcal{F}$ that satisfies*

$$R_\ell(f^*_\ell) \quad \le \quad R^*_\ell(\mathcal{F}) + \frac{1}{m}\,. \tag{10}$$

(Note that such a function f^*_ℓ exists by definition of $R^*_\ell(\mathcal{F})$.) Let $\lambda>0$, and let $\mathcal{A}$ be a ranking algorithm that, given a training sample S, outputs a ranking function $f_S\in\mathcal{F}$ that satisfies $f_S = \arg\min_{f\in\mathcal{F}}\left\{\widehat{R}_\ell(f;S)+\lambda\|f\|_K^2\right\}$. Then we have that for any $0<\delta<1$, with probability at least $1-\delta$ over the draw of S (according to $\mathcal{D}^m$),

$$R_\ell(f_S) \quad < \quad R^*_\ell(\mathcal{F}) + \lambda\|f^*_\ell\|_K^2 + \frac{1}{m} + \frac{32\sigma^2\kappa^2}{\lambda m} + \left(\frac{16\sigma^2\kappa^2}{\lambda}+2B\right)\sqrt{\frac{2\ln(2/\delta)}{m}}\,.$$

Thus, if $\lambda = o(1)$ and $\lambda = \omega(\frac{1}{\sqrt{m}})$ (for example, if $\lambda = m^{-1/4}$), then $R_\ell(f_S)$ converges in probability to the optimal value $R^*_\ell(\mathcal{F})$ (as $m\to\infty$).

Proof As in Corollary 15, we can use Corollary 14 and apply Theorem 6 with $\frac{\delta}{2}$ to get that with probability at least $1-\frac{\delta}{2}$,

$$R_\ell(f_S) \quad < \quad \widehat{R}_\ell(f_S;S) + \frac{32\sigma^2\kappa^2}{\lambda m} + \left(\frac{16\sigma^2\kappa^2}{\lambda} + B\right)\sqrt{\frac{2\ln(2/\delta)}{m}}\,. \tag{11}$$

Now,

$$\begin{aligned}\widehat{R}_\ell(f_S;S) \quad &\leq \quad \widehat{R}_\ell(f_S;S) + \lambda\|f_S\|_K^2 \\ &\leq \quad \widehat{R}_\ell(f_\ell^*;S) + \lambda\|f_\ell^*\|_K^2\,,\end{aligned}$$

where the first inequality is due to non-negativity of $\|f_S\|_K^2$ and the second inequality follows from the definition of f_S. Applying Lemma 17 to f_ℓ^* with $\frac{\delta}{2}$, we thus get that with probability at least $1-\frac{\delta}{2}$,

$$\widehat{R}_\ell(f_S;S) \quad < \quad \left(R_\ell(f_\ell^*) + B\sqrt{\frac{2\ln(2/\delta)}{m}}\right) + \lambda\|f_\ell^*\|_K^2\,. \tag{12}$$

Combining the inequalities in Eqs. (11-12), each of which holds with probability at least $1-\frac{\delta}{2}$, together with the condition in Eq. (10), gives the desired result. ■

6. Comparisons With Related Work

In the above sections we have derived stability-based generalization bounds for ranking algorithms in a setting that is more general than what has been considered previously, and have shown that kernel-based ranking algorithms that perform regularization in an RKHS in this setting satisfy the required stability conditions. In this section we discuss how our results relate to other recent studies.

6.1 Comparison with Stability Bounds for Classification/Regression

As pointed out earlier, our stability analysis of ranking algorithms is based on a similar analysis for classification and regression algorithms by Bousquet and Elisseeff (2002); therefore it is instructive to compare the generalization bounds we obtain here with those obtained in that work. Such a comparison shows that the bounds we obtain here for ranking are very similar to those obtained for classification and regression, differing only in constants (and, of course, in the precise definition of stability, which is problem-dependent). The difference in constants in the two bounds is due in part to the difference in loss functions in ranking and classification/regression, and in part to a slight difference in definitions of stability (in particular, our definitions are in terms of changes to a training sample that consist of replacing one element in the sample with a new one, while the definitions of Bousquet and Elisseeff are in terms of changes that consist of removing one element from the sample). As discussed in Section 4, the reason that we are able to obtain bounds for ranking algorithms using the same methods as those of Bousquet and Elisseeff lies in the power of McDiarmid's inequality, which was used by Bousquet and Elisseeff to capture the effect of stability but is also general enough to capture the structure of the ranking problem.

It is also instructive to compare our bounds with those obtained for bipartite ranking algorithms (Agarwal and Niyogi, 2005). In particular, the bounds for kernel-based ranking algorithms in the bipartite setting differ from our bounds in that the sample size m in our case is replaced with the term $mn/(m+n)$, where m,n denote the numbers of positive and negative examples, respectively, in the bipartite setting. This suggests that in general, the effective sample size in ranking is the ratio between the number of pairs in the training sample (mn in the bipartite setting) and the total number of examples ($m+n$); indeed, in our setting, this ratio is $\binom{m}{2}/m = m/2$, which fits our bounds.

6.2 Comparison with Uniform Convergence Bounds

While uniform convergence bounds for ranking have not been derived explicitly in the setting we consider, it is not hard to see that the techniques used to derive such bounds in the settings of Agarwal et al. (2005) and Clemencon et al. (2008) can be extended to obtain similar bounds in our setting. The crucial difference between such bounds and those derived in this paper is that uniform convergence bounds depend on the complexity of the function class searched by an algorithm. As discussed in the context of bipartite ranking (Agarwal and Niyogi, 2005), this means that such bounds can quickly become uninformative in high dimensions; in the case of function classes whose complexity cannot be bounded (such as the RKHS corresponding to a Gaussian kernel), uniform convergence bounds cannot be applied at all. In both these cases, the stability analysis provides a more useful viewpoint.

6.3 Comparison with Cortes et al. (2007)

The work that is perhaps most closely related to ours is that of Cortes et al. (2007), who study "magnitude-preserving" ranking algorithms—most of which perform regularization in an RKHS—and also use algorithmic stability to derive generalization bounds for such algorithms. The setting considered by Cortes et al. is similar to ours: the learner is given a finite sequence of labeled training examples $((x_1,y_1),\ldots,(x_m,y_m)) \in (\mathcal{X}\times\mathbb{R})^m$, and the goal is to learn a real-valued function $f:\mathcal{X}\to\mathbb{R}$ that ranks future instances with larger labels higher than those with smaller labels. The term "magnitude-preserving" comes from their view of the magnitude of the difference $(y-y')$ in the labels of two examples $(x,y),(x',y')$ as not just a penalty for mis-ranking the pair (x,x'), but as a quantity to be explicitly preserved by f, in the sense that $(f(x)-f(x'))$ be close to $(y-y')$.

There are three important differences between our work and that of Cortes et al. (2007). First, Cortes et al. implicitly assume the existence of a "truth" function: their results are derived under that assumption that only instances $x\in\mathcal{X}$ are drawn randomly, and that labels $y\in\mathcal{Y}$ are then determined according to a "truth" function $f^*:\mathcal{X}\to\mathcal{Y}$. In our work, we make the more general distributional assumption that all examples (x,y) are drawn randomly according to a joint distribution on $\mathcal{X}\times\mathcal{Y}$.

Second, Cortes et al. consider only the notion of uniform loss stability; they do not consider uniform score stability. As a consequence, the bounds they obtain for ranking algorithms performing regularization in an RKHS are similar to our bound in Corollary 15, which bounds the expected ranking error $R(f_S)$ of a learned ranking function f_S in terms of the empirical ℓ-error $\widehat{R}_\ell(f_S;S)$, where ℓ is the loss minimized by the algorithm.[4] In contrast, our bound in Corollary 12, which

4. Cortes et al. do not actually bound the expected ranking error $R(f_S)$ (which in our view is the primary performance measure of a ranking function in our setting); their results simply give bounds on the expected ℓ-error $R_\ell(f_S)$, where ℓ again is the loss minimized by the algorithm. However, for $\ell_{\text{disc}} \le \ell$, one can easily deduce bounds on $R(f_S)$ from these results.

makes use of the notion of score stability, bounds the expected error $R(f_S)$ in terms of the empirical ℓ_1-error $\widehat{R}_{\ell_1}(f_S;S)$. As mentioned in Section 5.2, the ℓ_1 loss tends to be smaller than many of the loss functions minimized by kernel-based ranking algorithms in practice (indeed, it is never greater than the hinge ranking loss ℓ_{h}), thus leading to tighter bounds for the same algorithms.

The third and most important difference between our work and that of Cortes et al. is that the results of Cortes et al. involve certain strong assumptions on the function class searched by an algorithm. In particular, they assume that the function class $\mathcal{F}$ searched by a ranking algorithm is bounded in the sense that $\exists\, M' > 0$ such that for all $f \in \mathcal{F}$ and all $x \in \mathcal{X}$, $|f(x) - f^*(x)| \leq M'$ (where, as mentioned above, $f^* : \mathcal{X} \rightarrow \mathbb{R}$ is the truth function in their setting). No RKHS $\mathcal{F}$ satisfies this condition, and so their results as stated do not apply to the kernel-based ranking algorithms they consider. On closer inspection, one notices that for the loss functions they show to be 1-admissible, they do not actually need this assumption. For the loss functions they show to be $4M'$-admissible, however, a non-trivial fix is required. We expect that an approach similar to that of separating out the effective search space, as we have done in the case of the least squares ranking loss in Section 5.2.2, should work; indeed, in the case of the least squares ranking loss, assuming a bounded label space Y, it is easy to see from the observations in Section 5.2.2 that $|f(x) - y|$ is bounded for all f in the *effective* search space (and all $(x,y) \in (\mathcal{X} \times \mathcal{Y})$).

6.4 Comparison with Rudin et al. (2005)

Rudin et al. (2005) studied a different setting of the ranking problem, in which one assumes the existence of a pair-wise "truth" function $\pi : \mathcal{X} \times \mathcal{X} \rightarrow \{0,1\}$ that assigns a binary ranking preference $\pi(x,x')$ to each pair of instances (x,x'), such that $\pi(x,x') = 1 \Rightarrow \pi(x',x) = 0$: if $\pi(x,x') = 1$, then x is to be ranked higher than x'; if $\pi(x',x) = 1$, then x' is to be ranked higher than x; and if $\pi(x,x') = \pi(x',x) = 0$, then there is no ranking preference between x and x'. The training sample given to a learner in this setting consists of a finite number of instances $x_1, \ldots, x_m$, each drawn randomly and independently according to some (unknown) distribution on $\mathcal{X}$, together with the corresponding ranking preferences $\pi(x_i,x_j)$ for $i,j \in \{1,\ldots,m\}, i \neq j$. Rudin et al. derived a generalization bound for this setting using techniques inspired by the works of Cucker and Smale (2002), Koltchinskii and Panchenko (2002), and Bousquet (2003). Their bound, expressed in terms of covering numbers, is a margin-based bound, which is of interest when the learned ranking function has zero empirical error on the training sample.

Noting that the labels y_i, y_j corresponding to a pair of instances x_i, x_j in our setting are used in our results only in the form of ranking preferences $(y_i - y_j)$, we can clearly view the setting of Rudin et al. as a special case of ours: the ranking preferences $(y_i - y_j)$, which are random variables in our setting, get replaced by the deterministic preferences $\pi(x_i,x_j)$. All quantities can be adapted accordingly; for example, the expected ranking error of a ranking function $f : \mathcal{X} \rightarrow \mathbb{R}$ in this setting (with respect to a distribution $\mathcal{D}$ on $\mathcal{X}$) becomes

$$R(f) \quad = \quad \mathbf{E}_{(X,X') \sim \mathcal{D} \times \mathcal{D}} \left[|\pi(X,X')| \left(\mathbf{I}_{\{\pi(X,X')(f(X) - f(X')) < 0\}} + \frac{1}{2} \mathbf{I}_{\{f(X) = f(X')\}} \right) \right].$$

In fact, rather than restrict ourselves to binary preferences, we can extend the setting of Rudin et al. to allow real-valued preferences, requiring that $\pi(x,x') = -\pi(x',x)$ for all $x,x' \in \mathcal{X}$; if π takes values in $[-M,M]$, we get the same results as we have derived in our setting (with the probabilities now being over the random draw of instances only).

We can also consider another extension of interest in the pair-wise truth function setting, where we may not have access to the preferences for all pairs of instances in the training sample, but rather are given preferences for only a limited number of pairs; in this case, the learner receives a sequence of instances $S = (x_1, \ldots, x_m)$, and preferences $\pi(x_i, x_j)$ for only a subset of pairs $E \subseteq \{(i,j) \mid 1 \le i < j \le m\}$. It is interesting then to consider a model in which not only the instances in S but also the pairs in E for which the ranking preferences are provided may be chosen randomly. If the instances in S are drawn randomly and independently according to some distribution $\mathcal{D}$ on $\mathcal{X}$, and the set E is drawn randomly (independently of S) from some distribution $\mathcal{E}_m$ on the set of possible (undirected) edge sets for a graph on m vertices $\{1, \ldots, m\}$ (for example, E could be obtained by including each edge with some fixed probability $0 < p_m < 1$, or by selecting at random a subset of α_m edges for some $\alpha_m \le \binom{m}{2}$), then under some natural conditions on $\mathcal{E}_m$, we can extend our techniques to obtain generalization bounds in this setting too. In the remainder of this section we discuss some details of this extension.

In particular, the empirical ranking error of a ranking function $f : \mathcal{X} \to \mathbb{R}$ in the above setting, with respect to a training sample $T = (S, E, \pi|_{(S,E)})$ where S and E are as above and $\pi|_{(S,E)}$ is the restriction of π to $\{(x_i, x_j) \mid (i,j) \in E\}$, is given by

$$\widehat{R}(f;T) \quad = \quad \frac{1}{|E|} \sum_{(i,j) \in E} |\pi(x_i, x_j)| \left(\mathbf{I}_{\{\pi(x_i,x_j)(f(x_i) - f(x_j)) < 0\}} + \frac{1}{2} \mathbf{I}_{\{f(x_i) = f(x_j)\}} \right) .$$

Given a loss function ℓ that assigns for any ranking function f, any pair of instances (x, x') and any $r \in [-M, M]$ a non-negative loss $\ell(f, x, x', r)$, the expected and empirical ℓ-errors can be defined similarly:

$$\begin{aligned} R_\ell(f) \quad &= \quad \mathbf{E}_{(X,X') \sim \mathcal{D} \times \mathcal{D}} \left[\ell(f, X, X', \pi(X, X')) \right] ; \\ \widehat{R}_\ell(f;T) \quad &= \quad \frac{1}{|E|} \sum_{(i,j) \in E} \ell(f, x_i, x_j, \pi(x_i, x_j)) . \end{aligned}$$

A ranking algorithm $\mathcal{A}$ in this setting, which given a training sample T outputs a ranking function f_T, has uniform loss stability β with respect to ℓ if for all $m \in \mathbb{N}$, $S \in \mathcal{X}^m$, $E \subseteq \{(i,j) \mid 1 \le i < j \le m\}$, $\pi : \mathcal{X} \times \mathcal{X} \to [-M, M]$, $1 \le i \le m$ and $x_i' \in \mathcal{X}$, we have for all $x, x' \in \mathcal{X}$ and all $r \in [-M, M]$,

$$\left| \ell(f_T, x, x', r) - \ell(f_{T^i}, x, x', r) \right| \quad \le \quad \beta(m) ,$$

where $T = (S, E, \pi|_{(S,E)})$ as above and $T^i = (S^i, E, \pi|_{(S^i,E)})$, with S^i being the sequence obtained from S by replacing x_i with x_i'. Then we can show the following bound for ranking algorithms with good loss stability in this setting:

Theorem 19 *Let $\pi : \mathcal{X} \times \mathcal{X} \to [-M, M]$ be a pair-wise truth function such that $\pi(x, x') = -\pi(x', x)$ for all $x, x' \in \mathcal{X}$. Let $\mathcal{A}$ be a symmetric ranking algorithm in the pair-wise truth function setting whose output on a training sample $T = (S, E, \pi|_{(S,E)})$ we denote by f_T, and let ℓ be a bounded ranking loss function such that $0 \le \ell(f, x, x', r) \le B$ for all $f : \mathcal{X} \to \mathbb{R}$, $x, x' \in \mathcal{X}$ and $r \in [-M, M]$. Let $\beta : \mathbb{N} \to \mathbb{R}$ be such that $\mathcal{A}$ has uniform loss stability β with respect to ℓ as defined above. Let $\mathcal{D}$ be any distribution on $\mathcal{X}$, and let $\{\mathcal{E}_m\}$ be any family of distributions on the sets of possible (undirected) edge sets for a graph on m vertices $\{1, \ldots, m\}$ that satisfies the following condition: $\exists\, s > 1$ and a sequence (δ_m) satisfying $\delta_m \ge 0$, $\lim_{m \to \infty} \delta_m = 0$, such that for all m, $\mathbf{P}_{E \sim \mathcal{E}_m}(|E| < m^s) \le \delta_m$. Then* *for any* $0 < \delta < 1$, *for all m large enough such that* $\delta_m \leq \frac{\delta}{2}$, *we have with probability at least* $1-\delta$ *over the draw of* (S,E) *according to* $\mathcal{D}^m \times \mathcal{E}_m$,

$$R_\ell(f_T) \quad < \quad \widehat{R}_\ell(f_T;T) + 2\beta(m) + \sqrt{\frac{1}{2}\left(4m(\beta(m))^2 + 8B\beta(m) + \frac{2B^2}{m-1} + \frac{B^2}{m^{s-1}}\right)\ln(2/\delta)}\,.$$

The proof makes use of McDiarmid's inequality and is similar to that of Theorem 6, with two main differences. First, McDiarmid's inequality is first applied to obtain a bound conditioned on an edge set E with $|E| \geq m^s$; the bound on the probability that $|E| < m^s$ then allows us to obtain the unconditional bound. Second, the constants c_k in the application of McDiarmid's inequality are different for each k, and depend on the degrees of the corresponding vertices in the graph with edge set E; the sum $\sum_k c_k^2$ is then bounded using a bound on the sum of squares of degrees in a graph due to de Caen (1998). Details of the proof are provided in Appendix D.

The condition on $\mathcal{E}_m$ in the above theorem states that with high probability (probability at least $1-\delta_m$), an edge set E drawn according to $\mathcal{E}_m$ has at least m^s edges for some $s > 1$, that is, $|E|$ is super-linear in m.[5] Thus, in the pair-wise truth function setting, we need not have access to the ranking preferences $\pi(x_i, x_j)$ for all $\binom{m}{2}$ pairs of instances in S; having the preferences for any super-linear number of pairs (where the pairs are selected independently of the instances S themselves) suffices to give a generalization bound (albeit with a correspondingly slower convergence rate, as quantified by the $\frac{1}{m^{s-1}}$ term in the bound above). Below we give examples of two families of distributions $\{\mathcal{E}_m\}$ that satisfy the above condition.

Example 1 *Fix any* $1 < s < 2$, *and let* $\mathcal{E}_m$ *be the distribution that corresponds to selecting a random subset of* α_m *edges, with* $\alpha_m = \min\left(\binom{m}{2}, \lceil m^s \rceil\right)$. *Then for large enough m,* $\alpha_m = \lceil m^s \rceil$, *and* $\mathbf{P}_{E\sim\mathcal{E}_m}(|E| < m^s) = 0$.

Example 2 *Fix any* $1 < s < 2$, *and let* $\mathcal{E}_m$ *be the distribution that corresponds to including each edge with a fixed probability* p_m, *with* $p_m = \min\left(1, \frac{8}{m^{2-s}}\right)$. *Then for large enough m,* $p_m = \frac{8}{m^{2-s}}$, *and assuming without loss of generality that* $m \geq 2$, *it is easy to show using a Chernoff bound that in this case,* $\mathbf{P}_{E\sim\mathcal{E}_m}(|E| < m^s) \leq e^{-m^s/4}$. *Indeed, for any such m, let* Z_m *be the random variable equal to the number of edges* $|E|$. *Then* Z_m *is a binomial random variable with parameters* $\binom{m}{2}$ *and* $p_m = \frac{8}{m^{2-s}}$, *and*

$$\begin{aligned}
\mathbf{E}[Z_m] &= \tbinom{m}{2} p_m \\
&= 4(m-1)m^{s-1} \\
&= 2(m^s + (m-2)m^{s-1}) \\
&\geq 2m^s\,,
\end{aligned}$$

where the inequality follows from our assumption that $m \geq 2$. *The Chernoff bound for deviations below the mean of a binomial random variable tells us that for any* $0 < \delta < 1$, $\mathbf{P}(Z_m < (1-\delta)\mathbf{E}[Z_m]) \leq e^{-\mathbf{E}[Z_m]\delta^2/2}$. *Thus we have*

$$\mathbf{P}(Z_m < m^s) \quad = \quad \mathbf{P}\Big(Z_m < \Big(1 - \frac{1}{2}\Big)2m^s\Big)$$

5. Note that in the statement of Theorem 19 we require $|E|$ to be super-linear in m by a polynomial factor; a smaller super-linear growth also leads to a generalization bound, but with a slower convergence rate.

$$
\begin{aligned}
&\leq \mathbf{P}\Big(Z_m < \Big(1 - \frac{1}{2}\Big)\mathbf{E}[Z_m]\Big) \\
&\leq e^{-\mathbf{E}[Z_m](\frac{1}{2})^2/2} \\
&\leq e^{-m^s/4}.
\end{aligned}
$$

Comparing the bound in Theorem 19 with that of Theorem 6, it is worth noting that the effective sample size in the above setting becomes m^{s-1}. This is consistent with the discussion in Section 6.1, in that the ratio between the number of pairs in the training sample and the total number of examples in this case is (with high probability) at least $m^s/m = m^{s-1}$.

7. Discussion

Our goal in this paper has been to study generalization properties of ranking algorithms in a setting where ranking preferences among instances are indicated by real-valued labels on the instances. This setting of the ranking problem arises frequently in practice and is more general than those considered previously. We have derived generalization bounds for ranking algorithms in this setting using the notion of algorithmic stability; in particular, we have shown that ranking algorithms that exhibit good stability properties also have good generalization properties, and have applied our results to obtain generalization bounds for kernel-based ranking algorithms that perform regularization in a reproducing kernel Hilbert space. Such algorithms often cannot be analyzed using uniform convergence results.

The main difference in the mathematical formulation of ranking problems as compared to classification or regression problems is that the loss function in ranking is 'pair-wise' rather than 'point-wise'. Indeed, ranking often resembles weighted 'classification on pairs', with the weights being given by the corresponding ranking preferences (although learning a real-valued function that induces a total ordering on the instance space is not quite the same as learning a binary-valued function on instance pairs that simply decides which of two instances should be ranked higher, Cohen et al. 1999; Balcan et al. 2007; Ailon and Mohri 2008). However, generalization bounds from classification cannot be applied directly to ranking, due to dependences among the instance pairs. As discussed earlier, the reason that we are able to obtain bounds for ranking algorithms using the same methods as those used by Bousquet and Elisseeff (2002) for classification and regression algorithms lies in the power of McDiarmid's inequality, which was used by Bousquet and Elisseeff to capture the effect of stability but is also general enough to capture the structure of the ranking problems we consider. A comparison of our results with those of Bousquet and Elisseeff (2002) and Agarwal and Niyogi (2005) suggests that the effective sample size in ranking is proportional to the ratio between the number of pairs in the training sample and the total number of examples; this can be smaller than the number of examples m if ranking preferences are provided for less than m^2 pairs.

The notions of uniform stability studied in this paper correspond most closely to those studied by Bousquet and Elisseeff (2002). These notions are strict in that they require changes in a sample to have bounded effect uniformly over all samples and replacements. One can define weaker notions of stability, analogous to the hypothesis stability considered by Devroye and Wagner (1979), Kearns and Ron (1999), and Bousquet and Elisseeff (2002), or the almost-everywhere stability considered by Kutin and Niyogi (2002), which would require the bounds to hold only in expectation or with high probability. Such notions would lead to a distribution-dependent treatment as opposed to the distribution-free treatment obtained with uniform stability, and it would be particularly interesting to see if making distributional assumptions in ranking can mitigate the reduced sample size effect discussed above.

For the sake of simplicity and to keep the focus on the main ideas involved in applying stability techniques to ranking, we have focused in this paper on bounding the expected ranking error in terms of the empirical ranking error. In classification and regression, stability analysis has also been used to provide generalization bounds in terms of the leave-one-out error (Bousquet and Elisseeff, 2002; Kearns and Ron, 1999), and with a slight change in definitions of stability, similar results can be obtained in the case of ranking as well. In particular, in this case we need to define (for a ranking algorithm which given a training sample S returns a ranking function f_S) a 'leave-two-out' ranking error as follows:

$$\widetilde{R}_\ell(f_S;S) \quad = \quad \frac{1}{\binom{m}{2}} \sum_{1 \le i < j \le m} \ell(f_{S^{\setminus ij}}, (x_i,y_i), (x_j,y_j)) \,,$$

where $S^{\setminus ij}$ denotes the sequence obtained by removing the ith and jth examples from a sample $S = ((x_1,y_1), \ldots, (x_m,y_m)) \in (\mathcal{X} \times \mathcal{Y})^m$. Then, defining notions of stability in terms of changes to a sample that consist of removing two examples rather than replacing one example, one can obtain very similar generalization bounds for ranking algorithms in terms of the above leave-two-out error as those we have derived in terms of the empirical error.

An open question concerns the analysis of other ranking algorithms using the algorithmic stability framework. For example, it has been shown that the AdaBoost algorithm for classification is stability-preserving, in the sense that stability of base classifiers implies stability of the final learned classifier (Kutin and Niyogi, 2001). It would be interesting if a similar result could be shown for the RankBoost ranking algorithm (Freund et al., 2003), which is based on the same principles of boosting as AdaBoost.

Finally, it is also an open question to analyze generalization properties of ranking algorithms in even more general settings of the ranking problem. For example, a more general setting would be one in which a finite number of instances $x_1, \ldots, x_m \in \mathcal{X}$ are drawn randomly and independently according to some distribution $\mathcal{D}$ on $\mathcal{X}$, and then pair-wise ranking preferences $r_{ij} \in [-M,M]$ for $i < j$ are drawn from a conditional distribution, conditioned on the corresponding pair of instances (x_i,x_j). We are not aware of any generalization bounds for such a setting.

Acknowledgments

We would like to thank the anonymous referees for constructive comments that helped improve the paper. SA is supported in part by NSF award DMS-0732334. PN thanks the NSF for financial support.

Appendix A. Proof of Theorem 6

Our main tool will be the following powerful concentration inequality of McDiarmid (1989), which bounds the deviation of any function of a sample for which a single change in the sample has limited effect.

Theorem 20 (McDiarmid 1989) *Let $X_1,\ldots,X_m$ be independent random variables, each taking values in a set A. Let $\phi : A^m \to \mathbb{R}$ be such that for each $1 \leq k \leq m$, there exists $c_k > 0$ such that*

$$\sup_{x_1,\ldots,x_m\in A, x'_k\in A} \Big|\phi(x_1,\ldots,x_m) - \phi(x_1,\ldots,x_{k-1},x'_k,x_{k+1},\ldots,x_m)\Big| \quad\leq\quad c_k\,.$$

Then for any $\varepsilon > 0$,

$$\mathbf{P}\Big(\phi(X_1,\ldots,X_m) - \mathbf{E}\left[\phi(X_1,\ldots,X_m)\right] \geq \varepsilon\Big) \quad\leq\quad e^{-2\varepsilon^2/\sum_{k=1}^m c_k^2}\,.$$

Before we apply McDiarmid's inequality to prove Theorem 6, we shall need the following technical lemma. In the following, we shall drop explicit references to distributions where clear from context, replacing, for example, $\mathbf{E}_{(X,Y)\sim\mathcal{D}}[\ldots]$ with simply $\mathbf{E}_{(X,Y)}[\ldots]$.

Lemma 21 *Let $\mathcal{A}$ be a symmetric ranking algorithm whose output on a training sample $S \in (\mathcal{X}\times\mathcal{Y})^m$ we denote by f_S, and let ℓ be a ranking loss function. Then for all $1 \leq i < j \leq m$, we have*

$$\begin{aligned}&\mathbf{E}_{S\sim\mathcal{D}^m}\Big[R_\ell(f_S) - \widehat{R}_\ell(f_S;S)\Big]\\ &\quad= \quad \mathbf{E}_{S\sim\mathcal{D}^m,((X'_i,Y'_i),(X'_j,Y'_j))\sim\mathcal{D}\times\mathcal{D}}\Big[\ell(f_S,(X'_i,Y'_i),(X'_j,Y'_j)) - \ell(f_{S^{i,j}},(X'_i,Y'_i),(X'_j,Y'_j))\Big]\,.\end{aligned}$$

Proof Denoting $S = ((X_1,Y_1),\ldots,(X_m,Y_m))$, we have by linearity of expectation,

$$\mathbf{E}_S\Big[\widehat{R}_\ell(f_S;S)\Big] \quad=\quad \frac{1}{\binom{m}{2}}\sum_{i=1}^{m-1}\sum_{j=i+1}^{m}\mathbf{E}_S\Big[\ell(f_S,(X_i,Y_i),(X_j,Y_j))\Big]\,.$$

By symmetry, the term in the summation is the same for all i,j. Therefore, for all $1 \leq i < j \leq m$, we get

$$\begin{aligned}\mathbf{E}_S\Big[\widehat{R}_\ell(f_S;S)\Big] \quad&=\quad \mathbf{E}_S\Big[\ell(f_S,(X_i,Y_i),(X_j,Y_j))\Big]\\ &=\quad \mathbf{E}_{S,((X'_i,Y'_i),(X'_j,Y'_j))}\Big[\ell(f_S,(X_i,Y_i),(X_j,Y_j))\Big]\,.\end{aligned}$$

Interchanging the roles of (X_i,Y_i) with (X'_i,Y'_i) and (X_j,Y_j) with (X'_j,Y'_j), we get

$$\mathbf{E}_S\Big[\widehat{R}_\ell(f_S;S)\Big] \quad=\quad \mathbf{E}_{S,((X'_i,Y'_i),(X'_j,Y'_j))}\Big[\ell(f_{S^{i,j}},(X'_i,Y'_i),(X'_j,Y'_j))\Big]\,.$$

Since by definition

$$\mathbf{E}_S\Big[R_\ell(f_S)\Big] \quad=\quad \mathbf{E}_{S,((X'_i,Y'_i),(X'_j,Y'_j))}\Big[\ell(f_S,(X'_i,Y'_i),(X'_j,Y'_j))\Big]\,,$$

the result follows. ■

Proof *(of Theorem 6)*
Let $\phi : (\mathcal{X}\times\mathcal{Y})^m \to \mathbb{R}$ be defined as follows:

$$\phi(S) \quad=\quad R_\ell(f_S) - \widehat{R}_\ell(f_S;S)\,.$$

We shall show that ϕ satisfies the conditions of McDiarmid's inequality (Theorem 20). Let $S = ((x_1,y_1),\dots,(x_m,y_m)) \in (\mathcal{X}\times\mathcal{Y})^m$. Then for each $1 \le k \le m$, we have for any $(x'_k,y'_k) \in (\mathcal{X}\times\mathcal{Y})$:

$$\begin{aligned}\left|\phi(S)-\phi(S^k)\right| &= \left|\left(R_\ell(f_S)-\widehat{R}_\ell(f_S;S)\right)-\left(R_\ell(f_{S^k})-\widehat{R}_\ell(f_{S^k};S^k)\right)\right| \\ &\le \left|R_\ell(f_S)-R_\ell(f_{S^k})\right|+\left|\widehat{R}_\ell(f_S;S)-\widehat{R}_\ell(f_{S^k};S^k)\right|.\end{aligned}$$

Now,

$$\begin{aligned}\left|R_\ell(f_S)-R_\ell(f_{S^k})\right| &= \left|\mathbf{E}_{((X,Y),(X',Y'))}\left[\ell(f_S,(X,Y),(X',Y'))-\ell(f_{S^k},(X,Y),(X',Y'))\right]\right| \\ &\le \mathbf{E}_{((X,Y),(X',Y'))}\left[\left|\ell(f_S,(X,Y),(X',Y'))-\ell(f_{S_k},(X,Y),(X',Y'))\right|\right] \\ &\le \beta(m),\end{aligned}$$

and

$$\begin{aligned}&\left|\widehat{R}_\ell(f_S;S)-\widehat{R}_\ell(f_{S^k};S^k)\right| \\ &\le \frac{1}{\binom{m}{2}}\sum_{1\le i<j\le m, i\ne k, j\ne k}\left|\ell(f_S,(x_i,y_i),(x_j,y_j))-\ell(f_{S^k},(x_i,y_i),(x_j,y_j))\right| \\ &\quad+\frac{1}{\binom{m}{2}}\sum_{i\ne k}\left|\ell(f_S,(x_i,y_i),(x_k,y_k))-\ell(f_{S^k},(x_i,y_i),(x'_k,y'_k))\right| \\ &\le \frac{1}{\binom{m}{2}}\sum_{1\le i<j\le m, i\ne k, j\ne k}\beta(m) \;+\; \frac{1}{\binom{m}{2}}\sum_{i\ne k}B \\ &= \frac{1}{\binom{m}{2}}\left(\left(\binom{m}{2}-(m-1)\right)\beta(m)+(m-1)B\right) \\ &< \beta(m)+\frac{2B}{m}.\end{aligned}$$

Thus we have

$$\left|\phi(S)-\phi(S^k)\right| \;\le\; 2\left(\beta(m)+\frac{B}{m}\right).$$

Therefore, applying McDiarmid's inequality to ϕ, we get for any $\varepsilon>0$,

$$\begin{aligned}\mathbf{P}_S\left(\left(R_\ell(f_S)-\widehat{R}_\ell(f_S;S)\right)-\mathbf{E}_S\left[R_\ell(f_S)-\widehat{R}_\ell(f_S;S)\right]\ge\varepsilon\right)& \\ &\le e^{-2\varepsilon^2/m(2(\beta(m)+\frac{B}{m}))^2} \\ &= e^{-m\varepsilon^2/2(m\beta(m)+B)^2}.\end{aligned}$$

Now, by Lemma 21, we have (for any $1\le i<j\le m$),

$$\begin{aligned}&\mathbf{E}_S\left[R_\ell(f_S)-\widehat{R}_\ell(f_S;S)\right] \\ &= \mathbf{E}_{S,((X'_i,Y'_i),(X'_j,Y'_j))}\left[\ell(f_S,(X'_i,Y'_i),(X'_j,Y'_j))-\ell(f_{S^{i,j}},(X'_i,Y'_i),(X'_j,Y'_j))\right] \\ &\le \mathbf{E}_{S,((X'_i,Y'_i),(X'_j,Y'_j))}\left[\left|\ell(f_S,(X'_i,Y'_i),(X'_j,Y'_j))-\ell(f_{S^i},(X'_i,Y'_i),(X'_j,Y'_j))\right|\right] \\ &\quad+\mathbf{E}_{S,((X'_i,Y'_i),(X'_j,Y'_j))}\left[\left|\ell(f_{S^i},(X'_i,Y'_i),(X'_j,Y'_j))-\ell(f_{S^{i,j}},(X'_i,Y'_i),(X'_j,Y'_j))\right|\right] \\ &\le 2\beta(m).\end{aligned}$$

Thus we get for any $\varepsilon > 0$,

$$\mathbf{P}_S \left(R_\ell(f_S) - \widehat{R}_\ell(f_S;S) - 2\beta(m) \geq \varepsilon \right) \quad \leq \quad e^{-m\varepsilon^2/2(m\beta(m)+B)^2} .$$

The result follows by setting the right hand side equal to δ and solving for ε. ■

Appendix B. Proof of Lemma 10

Proof *(of Lemma 10)*
Recall that a convex function $\phi : \mathcal{U} \to \mathbb{R}$ satisfies for all $u, v \in \mathcal{U}$ and for all $t \in [0,1]$,

$$\phi(u + t(v-u)) - \phi(u) \ \leq \ t(\phi(v) - \phi(u)) .$$

Since $\ell(f,(x,y),(x',y'))$ is convex in f, we have that $\widehat{R}_\ell(f;S)$ is convex in f. Therefore for any $t \in [0,1]$, we have

$$\widehat{R}_\ell(f + t\Delta f;S) - \widehat{R}_\ell(f;S) \quad \leq \quad t\left(\widehat{R}_\ell(f^i;S) - \widehat{R}_\ell(f;S) \right), \tag{13}$$

and also (interchanging the roles of f and f^i),

$$\widehat{R}_\ell(f^i - t\Delta f;S) - \widehat{R}_\ell(f^i;S) \quad \leq \quad t\left(\widehat{R}_\ell(f;S) - \widehat{R}_\ell(f^i;S) \right). \tag{14}$$

Adding Eqs. (13) and (14), we get

$$\widehat{R}_\ell(f + t\Delta f;S) - \widehat{R}_\ell(f;S) + \widehat{R}_\ell(f^i - t\Delta f;S) - \widehat{R}_\ell(f^i;S) \quad \leq \quad 0 . \tag{15}$$

Now, since $\mathcal{F}$ is convex, we have that $(f + t\Delta f) \in \mathcal{F}$ and $(f^i - t\Delta f) \in \mathcal{F}$. Since f minimizes $\widehat{R}^\lambda_\ell(f;S)$ in $\mathcal{F}$ and f^i minimizes $\widehat{R}^\lambda_\ell(f;S^i)$ in $\mathcal{F}$, we thus have

$$\widehat{R}^\lambda_\ell(f;S) - \widehat{R}^\lambda_\ell(f + t\Delta f;S) \quad \leq \quad 0 , \tag{16}$$

$$\widehat{R}^\lambda_\ell(f^i;S^i) - \widehat{R}^\lambda_\ell(f^i - t\Delta f;S^i) \quad \leq \quad 0 . \tag{17}$$

Adding Eqs. (15), (16) and (17), we get

$$\begin{aligned}
&\lambda\Big(N(f) - N(f + t\Delta f) + N(f^i) - N(f^i - t\Delta f) \Big) \\
&\leq \quad \widehat{R}_\ell(f^i;S) - \widehat{R}_\ell(f^i;S^i) + \widehat{R}_\ell(f^i - t\Delta f;S^i) - \widehat{R}_\ell(f^i - t\Delta f;S) \\
&= \quad \frac{1}{\binom{m}{2}} \sum_{j \neq i} \Big(\ell(f^i,(x_i,y_i),(x_j,y_j)) - \ell(f^i,(x'_i,y'_i),(x_j,y_j)) \\
&\qquad\qquad + \ell(f^i - t\Delta f,(x'_i,y'_i),(x_j,y_j)) - \ell(f^i - t\Delta f,(x_i,y_i),(x_j,y_j)) \Big) \\
&= \quad \frac{1}{\binom{m}{2}} \sum_{j \neq i} \Big(\Big(\ell(f^i,(x_i,y_i),(x_j,y_j)) - \ell(f^i - t\Delta f,(x_i,y_i),(x_j,y_j)) \Big) \\
&\qquad\qquad + \Big(\ell(f^i - t\Delta f,(x'_i,y'_i),(x_j,y_j)) - \ell(f^i,(x'_i,y'_i),(x_j,y_j)) \Big) \Big) \\
&\leq \quad \frac{t\sigma}{\binom{m}{2}} \sum_{j \neq i} \Big(|\Delta f(x_i)| + 2|\Delta f(x_j)| + |\Delta f(x'_i)| \Big),
\end{aligned}$$

where the last inequality follows by σ-admissibility. The result follows. ■

Appendix C. Proof of Lemma 17

The proof is a simple application of McDiarmid's inequality.
Proof *(of Lemma 17)*
Let $\phi : (\mathcal{X} \times \mathcal{Y})^m \to \mathbb{R}$ be defined as follows:

$$\phi(S) \;=\; \widehat{R}_\ell(f;S)\,.$$

Then by linearity of expectation,

$$\begin{aligned}\mathbf{E}_{S\sim\mathcal{D}^m}\Big[\phi(S)\Big] &= \frac{1}{\binom{m}{2}}\sum_{i=1}^{m-1}\sum_{j=i+1}^{m}\mathbf{E}_{((X_i,Y_i),(X_j,Y_j))\sim\mathcal{D}\times\mathcal{D}}\Big[\ell(f,(X_i,Y_i),(X_j,Y_j))\Big] \\ &= R_\ell(f)\,.\end{aligned}$$

We shall show that ϕ satisfies the conditions of McDiarmid's inequality (Theorem 20). Let $S = ((x_1,y_1),\ldots,(x_m,y_m)) \in (\mathcal{X}\times\mathcal{Y})^m$. Then for each $1 \le k \le m$, we have for any $(x'_k,y'_k) \in (\mathcal{X}\times\mathcal{Y})$:

$$\begin{aligned}\big|\phi(S)-\phi(S^k)\big| &= \Big|\widehat{R}_\ell(f;S)-\widehat{R}_\ell(f;S^k)\Big| \\ &\le \frac{1}{\binom{m}{2}}\sum_{i\neq k}\Big|\ell(f,(x_i,y_i),(x_k,y_k))-\ell(f,(x_i,y_i),(x'_k,y'_k))\Big| \\ &\le \frac{1}{\binom{m}{2}}(m-1)B \\ &= \frac{2B}{m}\,.\end{aligned}$$

Therefore, applying McDiarmid's inequality to ϕ, we get for any $\varepsilon > 0$,

$$\begin{aligned}\mathbf{P}_{S\sim\mathcal{D}^m}\Big(\widehat{R}_\ell(f;S)-R_\ell(f)\ge\varepsilon\Big) &\le e^{-2\varepsilon^2/m(\frac{2B}{m})^2} \\ &= e^{-m\varepsilon^2/2B^2}\,.\end{aligned}$$

The result follows by setting the right hand side equal to δ and solving for ε. ■

Appendix D. Proof of Theorem 19

The proof of this result also makes use of McDiarmid's inequality and is similar to the proof of Theorem 6 in Appendix A, with two main differences. First, McDiarmid's inequality is first applied to obtain a bound conditioned on an edge set E with $|E| \ge m^s$; the bound on the probability that $|E| < m^s$ then allows us to obtain the unconditional bound. Second, the constants c_k in the application of McDiarmid's inequality are different for each k, and depend on the degrees of the corresponding vertices in the graph with edge set E; the sum $\sum_k c_k^2$ is then bounded using a bound on the sum of squares of degrees in a graph due to de Caen (1998).
Proof *(of Theorem 19)*
Let $m \in \mathbb{N}$, and fix any edge set $E_0 \subseteq \{(i,j) \mid 1 \le i < j \le m\}$ with $|E_0| \ge m^s$. Let $\phi_{E_0} : \mathcal{X}^m \to \mathbb{R}$ be defined as follows:

$$\phi_{E_0}(S) \;=\; R_\ell(f_T)-\widehat{R}_\ell(f_T;T)\,,$$

where $T = (S, E_0, \pi|_{(S,E_0)})$. We shall show that ϕ_{E_0} satisfies the conditions of McDiarmid's inequality (Theorem 20). Let $S = (x_1, \ldots, x_m) \in \mathcal{X}^m$. Then for each $1 \leq k \leq m$, we have for any $x'_k \in \mathcal{X}$:

$$\begin{aligned}\left|\phi_{E_0}(S) - \phi_{E_0}(S^k)\right| &= \left|\Big(R_\ell(f_T) - \widehat{R}_\ell(f_T;T)\Big) - \Big(R_\ell(f_{T^k}) - \widehat{R}_\ell(f_{T^k};T^k)\Big)\right| \\ &\leq \Big|R_\ell(f_T) - R_\ell(f_{T^k})\Big| + \Big|\widehat{R}_\ell(f_T;T) - \widehat{R}_\ell(f_{T^k};T^k)\Big|.\end{aligned}$$

Now,

$$\begin{aligned}\Big|R_\ell(f_T) - R_\ell(f_{T^k})\Big| &= \Big|\mathbf{E}_{(X,X')}\Big[\ell(f_T, X, X', \pi(X,X')) - \ell(f_{T^k}, X, X', \pi(X,X'))\Big]\Big| \\ &\leq \mathbf{E}_{(X,X')}\Big[\Big|\ell(f_T, X, X', \pi(X,X')) - \ell(f_{T^k}, X, X', \pi(X,X'))\Big|\Big] \\ &\leq \beta(m),\end{aligned}$$

and

$$\begin{aligned}&\Big|\widehat{R}_\ell(f_T;T) - \widehat{R}_\ell(f_{T^k};T^k)\Big| \\ &\leq \frac{1}{|E_0|}\sum_{(i,j)\in E_0, i\neq k, j\neq k}\Big|\ell(f_T, x_i, x_j, \pi(x_i,x_j)) - \ell(f_{T^k}, x_i, x_j, \pi(x_i,x_j))\Big| \\ &\quad + \frac{1}{|E_0|}\sum_{(i,k)\in E_0}\Big|\ell(f_T, x_i, x_k, \pi(x_i,x_k)) - \ell(f_{T^k}, x_i, x'_k, \pi(x_i,x'_k))\Big| \\ &\quad + \frac{1}{|E_0|}\sum_{(k,j)\in E_0}\Big|\ell(f_T, x_k, x_j, \pi(x_k,x_j)) - \ell(f_{T^k}, x'_k, x_j, \pi(x'_k,x_j))\Big| \\ &\leq \frac{1}{|E_0|}\sum_{(i,j)\in E_0, i\neq k, j\neq k}\beta(m) + \frac{1}{|E_0|}\sum_{(i,k)\in E_0} B + \frac{1}{|E_0|}\sum_{(k,j)\in E_0} B \\ &\leq \beta(m) + \frac{d_k}{|E_0|}B,\end{aligned}$$

where d_k is the degree of vertex k in the graph with edge set E_0. Thus we have

$$\Big|\phi_{E_0}(S) - \phi_{E_0}(S^k)\Big| \leq c_k,$$

where

$$c_k = 2\beta(m) + \frac{d_k}{|E_0|}B.$$

Now,

$$\sum_{k=1}^m d_k = 2|E_0|,$$

and using a bound on the sum of squares of degrees in a graph due to de Caen (1998), we have

$$\sum_{k=1}^m d_k^2 \leq |E_0|\left(\frac{2|E_0|}{m-1} + m - 2\right).$$

Therefore,

$$\begin{aligned}
\sum_{k=1}^{m} c_k^2 &= \sum_{k=1}^{m} \left(2\beta(m) + \frac{d_k}{|E_0|} B \right)^2 \\
&= 4m(\beta(m))^2 + \frac{4B\beta(m)}{|E_0|} \sum_{k=1}^{m} d_k + \frac{B^2}{|E_0|^2} \sum_{k=1}^{m} d_k^2 \\
&\leq 4m(\beta(m))^2 + 8B\beta(m) + \frac{2B^2}{m-1} + \frac{B^2(m-2)}{|E_0|} \\
&\leq \underbrace{4m(\beta(m))^2 + 8B\beta(m) + \frac{2B^2}{m-1} + \frac{B^2}{m^{s-1}}}_{\gamma(m)},
\end{aligned}$$

where the last inequality follows since $|E_0| \geq m^s$. Thus, applying McDiarmid's inequality to ϕ_{E_0}, we get for any $\varepsilon > 0$,

$$\mathbf{P}_S \left(\left(R_\ell(f_T) - \widehat{R}_\ell(f_T;T) \right) - \mathbf{E}_S \left[R_\ell(f_T) - \widehat{R}_\ell(f_T;T) \;\middle|\; E = E_0 \right] \geq \varepsilon \;\middle|\; E = E_0 \right) \leq e^{-2\varepsilon^2/\gamma(m)}.$$

Now, as in Theorem 6, we can show that

$$\mathbf{E}_S \left[R_\ell(f_T) - \widehat{R}_\ell(f_T;T) \;\middle|\; E = E_0 \right] \leq 2\beta(m).$$

Thus we get for any $\varepsilon > 0$,

$$\mathbf{P}_S \left(R_\ell(f_T) - \widehat{R}_\ell(f_T;T) - 2\beta(m) \geq \varepsilon \;\middle|\; E = E_0 \right) \leq e^{-2\varepsilon^2/\gamma(m)}.$$

Since the above bound holds for all E_0 with $|E_0| \geq m^s$, we have

$$\begin{aligned}
\mathbf{P}_{(S,E)} \left(R_\ell(f_T) - \widehat{R}_\ell(f_T;T) - 2\beta(m) \geq \varepsilon \right) &\leq \mathbf{P}_E(|E| < m^s) + e^{-2\varepsilon^2/\gamma(m)} \\
&\leq \delta_m + e^{-2\varepsilon^2/\gamma(m)}.
\end{aligned}$$

Thus for m large enough such that $\delta_m \leq \frac{\delta}{2}$, we get

$$\mathbf{P}_{(S,E)} \left(R_\ell(f_T) - \widehat{R}_\ell(f_T;T) - 2\beta(m) \geq \varepsilon \right) \leq \frac{\delta}{2} + e^{-2\varepsilon^2/\gamma(m)}.$$

Setting the second term on the right hand side equal to $\frac{\delta}{2}$ and solving for ε gives the desired result. ■

References

Shivani Agarwal. Ranking on graph data. In *Proceedings of the 23rd International Conference on Machine Learning*, 2006.

Shivani Agarwal and Partha Niyogi. Stability and generalization of bipartite ranking algorithms. In *Proceedings of the 18th Annual Conference on Learning Theory*, 2005.

Shivani Agarwal, Thore Graepel, Ralf Herbrich, Sariel Har-Peled, and Dan Roth. Generalization bounds for the area under the ROC curve. *Journal of Machine Learning Research*, 6:393–425, 2005.

Nir Ailon and Mehryar Mohri. An efficient reduction of ranking to classification. In *Proceedings of the 21st Annual Conference on Learning Theory*, 2008.

Kenneth J. Arrow. *Social Choice and Individual Values*. Yale University Press, Second edition, 1970.

Maria-Florina Balcan, Nikhil Bansal, Alina Beygelzimer, Don Coppersmith, John Langford, and Gregory B. Sorkin. Robust reductions from ranking to classification. In *Proceedings of the 20th Annual Conference on Learning Theory*, 2007.

Olivier Bousquet. New approaches to statistical learning theory. *Annals of the Institute of Statistical Mathematics*, 55(2):371–389, 2003.

Olivier Bousquet and André Elisseeff. Stability and generalization. *Journal of Machine Learning Research*, 2:499–526, 2002.

Chris Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Greg Hullender. Learning to rank using gradient descent. In *Proceedings of the 22nd International Conference on Machine Learning*, 2005.

Alpha C. Chiang and Kevin Wainwright. *Fundamental Methods of Mathematical Economics*. McGraw-Hill Irwin, Fourth edition, 2005.

Stephan Clemencon and Nicolas Vayatis. Ranking the best instances. *Journal of Machine Learning Research*, 8:2671–2699, 2007.

Stephan Clemencon, Gabor Lugosi, and Nicolas Vayatis. Ranking and empirical minimization of U-statistics. *Annals of Statistics*, 36:844–874, 2008.

William W. Cohen, Robert E. Schapire, and Yoram Singer. Learning to order things. *Journal of Artificial Intelligence Research*, 10:243–270, 1999.

Corinna Cortes, Mehryar Mohri, and Ashish Rastogi. Magnitude-preserving ranking algorithms. In *Proceedings of 24th International Conference on Machine Learning*, 2007.

David Cossock and Tong Zhang. Subset ranking using regression. In *Proceedings of the 19th Annual Conference on Learning Theory*, 2006.

Koby Crammer and Yoram Singer. Pranking with ranking. In *Advances in Neural Information Processing Systems 14*, pages 641–647. MIT Press, 2002.

Felipe Cucker and Steve Smale. On the mathematical foundations of learning. *Bulletin of the American Mathematical Society*, 39:1–49, 2002.

Dominique de Caen. An upper bound on the sum of squares of degrees in a graph. *Discrete Mathematics*, 185:245–248, 1998.

Luc Devroye and Terry J. Wagner. Distribution-free performance bounds for potential function rules. *IEEE Transactions on Information Theory*, 25(5):601–604, 1979.

Theodoros Evgeniou, Massimiliano Pontil, and Tomaso Poggio. Regularization networks and support vector machines. *Advances in Computational Mathematics*, 13(1):1–50, 2000.

Yoav Freund, Raj Iyer, Robert E. Schapire, and Yoram Singer. An efficient boosting algorithm for combining preferences. *Journal of Machine Learning Research*, 4:933–969, 2003.

David Haussler. Convolution kernels on discrete structures. Technical Report UCSC-CRL-99-10, University of California at Santa Cruz, 1999.

Ralf Herbrich, Thore Graepel, Peter Bollmann-Sdorra, and Klaus Obermayer. Learning preference relations for information retrieval. In *Proceedings of the ICML-1998 Workshop on Text Categorization and Machine Learning*, 1998.

Ralf Herbrich, Thore Graepel, and Klaus Obermayer. Large margin rank boundaries for ordinal regression. *Advances in Large Margin Classifiers*, pages 115–132, 2000.

Thorsten Joachims. Optimizing search engines using clickthrough data. In *Proceedings of the ACM Conference on Knowledge Discovery and Data Mining*, 2002.

Michael Kearns and Dana Ron. Algorithmic stability and sanity-check bounds for leave-one-out cross-validation. *Neural Computation*, 11:1427–1453, 1999.

Vladimir Koltchinskii and Dmitry Panchenko. Empirical margin distributions and bounding the generalization error of combined classifiers. *The Annals of Statistics*, 30(1):1–50, 2002.

Samuel Kutin and Partha Niyogi. The interaction of stability and weakness in AdaBoost. Technical Report TR-2001-30, Computer Science Department, University of Chicago, 2001.

Samuel Kutin and Partha Niyogi. Almost-everywhere algorithmic stability and generalization error. In *Proceedings of the 18th Conference on Uncertainty in Artificial Intelligence*, 2002.

Erich L. Lehmann. *Nonparametrics: Statistical Methods Based on Ranks*. Holden-Day, 1975.

Colin McDiarmid. On the method of bounded differences. In *Surveys in Combinatorics 1989*, pages 148–188. Cambridge University Press, 1989.

Tomaso Poggio, Ryan Rifkin, Sayan Mukherjee, and Partha Niyogi. General conditions for predictivity in learning theory. *Nature*, 428:419–422, 2004.

Filip Radlinski and Thorsten Joachims. Query chains: Learning to rank from implicit feedback. In *Proceedings of the ACM Conference on Knowledge Discovery and Data Mining*, 2005.

Alain Rakotomamonjy. Optimizing area under ROC curves with SVMs. In *Proceedings of the ECAI-2004 Workshop on ROC Analysis in AI*, 2004.

William H. Rogers and Terry J. Wagner. A finite sample distribution-free performance bound for local discrimination rules. *Annals of Statistics*, 6(3):506–514, 1978.

Cynthia Rudin. Ranking with a p-norm push. In *Proceedings of the 19th Annual Conference on Learning Theory*, 2006.

Cynthia Rudin, Corinna Cortes, Mehryar Mohri, and Robert E. Schapire. Margin-based ranking meets boosting in the middle. In *Proceedings of the 18th Annual Conference on Learning Theory*, 2005.

Vladimir N. Vapnik and Alexey Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability and its Applications*, 16:264–280, 1971.

Journal of Machine Learning Research 10 (2009) 475-514 Submitted 3/08; Revised 7/08; Published 2/09

Controlling the False Discovery Rate of the Association/Causality Structure Learned with the PC Algorithm

Junning Li JUNNINGL@ECE.UBC.CA
Z. Jane Wang ZJANEW@ECE.UBC.CA
Department of Electrical and Computer Engineering
University of British Columbia
Vancouver, BC, V6T 1Z4
Canada

Editors: Paolo Frasconi, Kristian Kersting, Hannu Toivonen and Koji Tsuda

Abstract

In real world applications, graphical statistical models are not only a tool for operations such as classification or prediction, but usually the network structures of the models themselves are also of great interest (e.g., in modeling brain connectivity). The false discovery rate (FDR), the expected ratio of falsely claimed connections to all those claimed, is often a reasonable error-rate criterion in these applications. However, current learning algorithms for graphical models have not been adequately adapted to the concerns of the FDR. The traditional practice of controlling the type I error rate and the type II error rate under a conventional level does not necessarily keep the FDR low, especially in the case of sparse networks. In this paper, we propose embedding an FDR-control procedure into the PC algorithm to curb the FDR of the skeleton of the learned graph. We prove that the proposed method can control the FDR under a user-specified level at the limit of large sample sizes. In the cases of moderate sample size (about several hundred), empirical experiments show that the method is still able to control the FDR under the user-specified level, and a heuristic modification of the method is able to control the FDR more accurately around the user-specified level. The proposed method is applicable to any models for which statistical tests of conditional independence are available, such as discrete models and Gaussian models.

Keywords: Bayesian networks, false discovery rate, PC algorithm, directed acyclic graph, skeleton

1. Introduction

Graphical models have attracted increasing attention in the fields of data mining and machine learning in the last decade. These models, such as Bayesian networks (also called belief networks) and Markov random fields, generally represent events or random variables as vertices (also referred to as nodes), and encode conditional-independence relationships among the events or variables as directed or undirected edges (also referred to as arcs) according to the Markov properties (see Lauritzen, 1996, Chapt. 3). Of particular interest here are Bayesian networks (see Pearl, 1988, Chapt. 3.3) that encode conditional-independence relationships according to the directed Markov property (see Lauritzen, 1996, pages 46–53) with directed acyclic graphs (DAGs) (i.e., graphs with only directed edges and with no directed cycles). The directed acyclic feature facilitates the computation of Bayesian networks because the joint probability can be factorized recursively into many local conditional probabilities.

As a fundamental and intuitive tool to analyze and visualize the association and/or causality relationships among multiple events, graphical models have become more and more explored in biomedical researches, such as discovering gene regulatory networks and modelling functional connectivity between brain regions. In these real world applications, graphical models are not only a tool for operations such as classification or prediction, but often the network structures of the models themselves are also output of great interest: a set of association and/or causality relationships discovered from experimental observations. For these applications, a desirable structure-learning method needs to account for the error rate of the graphical features of the discovered network. Thus, it is important for structure-learning algorithms to control the error rate of the association/causality relationships discovered from a limited number of observations closely below a user-specified level, in addition to finding a model that fits the data well. As edges are fundamental elements of a graph, error rates related to them are of natural concerns.

In a statistical decision process, there are basically two sources of errors: the type I errors, that is, falsely rejecting negative hypotheses when they are actually true; and the type II errors, that is, falsely accepting negative hypotheses when their alternatives, the positive hypotheses are actually true. In the context of learning graph structures, a negative hypothesis could be that an edge does not exist in the graph while the positive hypothesis could be that the edge does exist. Because of the stochastic nature of random sampling, data of a limited sample size may appear to support a positive hypothesis more than a negative hypothesis even when actually the negative hypothesis is true, or vice versa. Thus it is generally impossible to absolutely prevent the two types of errors simultaneously, but has to set a threshold on a certain type of errors, or keep a balance between the them, for instance by minimizing a certain lost function associated with the errors according to the Bayesian decision theory. For example, when diagnosing cancer, to catch the potential chance of saving a patient's life, doctors probably hope that the type II error rate, that is, the probability of falsely diagnosing a cancer patient as healthy, to be low, such as less than 5%. Meanwhile, when diagnosing a disease whose treatment is so risky that may cause the loss of eyesight, to avoid the unnecessary but great risk for healthy people, doctors probably hope that the type I error rate, that is, the probability of falsely diagnosing a healthy people as affected by the disease, to be extremely low, such as less than 0.1%. Learning network structures may face scenarios similar to the two cases above of diagnosing diseases. Given data of a limited sample size, there is not an algorithm guaranteeing a perfect recovery of the structure of the underlying graphical model, and any algorithm has to compromise on the two types of errors.

For problems involving simultaneously testing multiple hypotheses, such as verifying the existence of edges in a graph, there are several different criteria for their error-rate control (see Table 2), depending on researchers' concerns or the scenario of the study. Generally there are not mathematically or technically superior relationships among different error-rate criteria if the research scenario is not specified. One error-rate criterion may be favoured in one scenario while another criterion may be right of interest in a different scenario, just as the aforementioned examples of diagnosing diseases. In real world applications, selecting the error rate of interest is largely not an abstract question "which error rate is superior over others?", but a practical question "which error rate is the researchers' concern?" For extended discussions on why there are not general superior relationships among different error-rate criteria, please refer to Appendix C, where examples of typical research scenarios, accompanied by theoretical discussions, illustrate that each of the four error-rate criteria in Table 2 may be favoured in a certain study.

The false discovery rate (FDR) (see Benjamini and Yekutieli, 2001; Storey, 2002), defined as the expected ratio of falsely discovered positive hypotheses to all those discovered, has become an important and widely used criterion in many research fields, such as bioinformatics and neuroimaging. In many real world applications that involve multiple hypothesis testing, the FDR is more reasonable than the traditional type I error rate and type II error rate. Suppose that in a pilot study researchers are selecting candidate genes for a genetic research on schizophrenia. Due to the limited funding, only a limited number of genes can be studied thoroughly in the afterward genetic research. To use the funding efficiently, researchers would hope that 95% of the candidate genes selected in the pilot study are truly associated with the disease. In this case, the FDR is chosen as the error rate of interest and should be controlled under 5%. Simply controlling the type I error rate and the type II error rate under certain levels does not necessarily keep the FDR sufficiently low, especially in the case of sparse networks. For example, suppose a gene regulatory network involves 100 genes, where each gene interacts in average with 3 others, that is, there are 150 edges in the network. Then an algorithm with the *realized* type I error rate = 5% and the *realized* power = 90% (i.e., the *realized* type II error rate = 10%) will recover a network with $150\times 90\%=135$ correct connections and $[100\times(100-1)/2-150]\times 5\% = 240$ false connections. This means that $240/(240+135) = 64\%$ of the claimed connections actually do not exist in the true network. Due to the popularity of the FDR in research practices, it is highly desirable to develop structure-learning algorithms that allow the control over the FDR on network structures.

However, current structure-learning algorithms for Bayesian networks have not been adequately adapted to explicitly controlling the FDR of the claimed "discovered" networks. Score-based search methods (see Heckerman et al., 1995) look for a suitable structure by optimizing a certain criterion of goodness-of-fit, such as the Akaike information criterion (AIC), the Bayesian information criterion (BIC), or the Bayesian Dirichlet likelihood equivalent metric (BDE), with a random walk (e.g., simulated annealing) or a greedy walk (e.g., hill-climbing), in the space of DAGs or their equivalence classes.[1] It is worth noting that the restricted case of tree-structured Bayesian networks has been optimally solved, in the sense of Kullback-Leibler divergence, with Chow and Liu (1968)'s method, and that Chickering (2002) has proved that the greedy equivalence search can identify the true equivalence class in the limit of large sample sizes. Nevertheless, scores do not directly reflect the error rate of edges, and the sample sizes in real world applications are usually not large enough to guarantee the perfect asymptotic identification.

The Bayesian approach first assumes a certain prior probability distribution over the network structures, and then estimates the posterior probability distribution of the structures after data are observed. Theoretically, the posterior probability of any structure features, such as the existence of an edge, the existence of a path, or even the existence of a sub-graph, can be estimated with the Bayesian approach. This consequently allows the control of the posterior error rate of these structure features, that is, the posterior probability of the non-existence of these features. It should be pointed out that the posterior error rate is conceptually different from those error rates such as the type I error rate, the type II error rate, and the FDR, basically because they are from different statistical perspectives. The posterior error rate is defined from the perspective of Bayesian statistics. From the Bayesian perspective, network structures are assumed to be random, according to a probability distribution, and the posterior error rate is the probability of the non-existence of certain features according to the posterior probability distribution over the network structures. Given the same data,

1. An equivalence class of DAGs is a set of DAGs that encode the same set of conditional-independence relationships according to the directed Markov property.

different posterior distributions will be derived from different prior distributions. The type I error rate, the type II error rate, and the FDR are defined from the perspective of classical statistics. From the classical perspective, there is a true, yet unknown, model behind the data, and the error rates are defined by comparing with the true model. Nevertheless, a variant of the FDR, the positive false discovery rate (pFDR) proposed by Storey (2002), can be interpreted from the Bayesian perspective (Storey, 2003).

The Bayesian approach for structure learning is usually conducted with the maximum-*a posteriori*-probability (MAP) method or the posterior-expectation method. The MAP method selects the network structure with the largest posterior probability. The optimal structure is usually searched for in a score-based manner, with the posterior probability or more often approximations to the relative posterior probability (for instance the BIC score) being the score to optimize. Cooper and Herskovits (1992) developed a heuristic greedy search algorithm called K2[2] that can finish the search in a polynomial time with respect to the number of vertices, given the order of vertices. The MAP method provides us with a single network structure, the posteriorly most probable one, but does not address error rates in the Bayesian approach.

To control errors in the Bayesian approach, the network structure should be learned with the posterior-expectation method, that is, calculating the posterior probabilities of network structures, and then deriving the posterior expectation of the existence of certain structure features. Though theoretically the posterior-expectation method can control the error rate of any structure features, in practice its capacity is largely limited for computational reasons. The number of DAGs increases super-exponentially as the number of vertices increases (Robinson, 1973). For 10 vertices, there are already about 4.2×10^{18} DAGs. Though the number of equivalence classes of DAGs is much smaller than the number of DAGs, it is still forbiddingly large, empirically asymptotically decreasing to 1/13.652 of the number of DAGs, as the number of vertices increases (Steinsky, 2003). Therefore, exact inferences of posterior probabilities are only feasible for small scale problems, or under certain additional constraints. For certain prior distributions, and given the order of vertices, Friedman and Koller (2003) have derived a formula that can be used to calculate the exact posterior probability of a structure feature with the computational complexity bounded by $O(N^{D_{in}+1})$, where N is the number of vertices and D_{in} is the upper bound of the in-degree for each vertex. Considering similar prior distributions, but without the restriction on the order of vertices, Koivisto and Sood (2004) have developed a fast exact Bayesian inference algorithm based on dynamic programming that is able to compute the exact posterior probability of a sub-network with the computational complexity bounded by $O(N2^N + N^{D_{in}+1}L(m))$, where $L(m)$ is the complexity of computing a marginal conditional likelihood from m samples. In practice, this algorithm runs fairly fast when the number of vertices is less than 25. For networks with more than 30 vertices, the authors suggested setting more restrictions or combining with inexact techniques. These two breakthroughs made exact Bayesian inferences practical for certain prior distributions. However, as Friedman and Koller (2003) pointed out, the prior distributions which facilitate the exact inference are not hypothesis equivalent (see Heckerman et al., 1995), that is, different network structures that are in the same equivalence class often have different priors. The simulation performed by Eaton and Murphy (2007) confirmed that these prior distributions deviate far from the uniform distributions. This implies that the methods cannot be applied to the widely accepted uninformative prior, that is, the uniform prior distribution over DAGs. For general problems, the posterior probability of a structure feature can be approx-

2. The algorithm is named K2 because it evolved from a system named Kutato (Herskovits and Cooper, 1990).

imated with Markov chain Monte Carlo (MCMC) methods (Madigan et al., 1995). As a versatile implementation of Bayesian inferences, the MCMC method can estimate the posterior probability given any prior probability distribution. However, MCMC usually requires intensive computation and results may depend on the initial state of the randomization.

Constraint-based approaches, such as the SGS[3] algorithm (see Spirtes et al., 2001, pages 82–83), inductive causation (IC)[4] (see Pearl, 2000, pages 49–51), and the PC[5] algorithm (see Spirtes et al., 2001, pages 84–89), are rooted in the directed Markov property, the rule by which Bayesian networks encode conditional independence. These methods first test hypotheses of conditional independence among random variables, and then combine those accepted hypotheses of conditional independence to construct a partially directed acyclic graph (PDAG) according to the directed Markov property. The computational complexity of these algorithms is difficult to analyze exactly, though for the worst case, which rarely occurs in real world applications, is surely bounded by $O(N^2 2^N)$ where N is the number of vertices. In practice, the PC algorithm and the fast-causal-inference (FCI) algorithm (see Spirtes et al., 2001, pages 142–146) can achieve polynomial time if the maximum degree of a graph is fixed. It has been proved that if the true model satisfies the faithfulness constraints (see Spirtes et al., 2001, pages 13 and 81) and all the conditional-independence/dependence relationships are correctly identified, then the PC algorithm and the IC algorithm can exactly recover the true equivalence class, and so do the FCI algorithm and the IC* algorithm[6] (see Pearl, 2000, pages 51–54) for problems with latent variables. Kalisch and Bühlmann (2007) have proved that for Gaussian Bayesian networks, the PC algorithm can consistently estimate the equivalence class of an underlying sparse DAG as the sample size m approaches infinity, even if the number of vertices N grows as fast as $O(m^\lambda)$ for any $0 < \lambda < \infty$. Yet, as in practice hypotheses of conditional independence are tested with statistical inference from limited data, false decisions cannot be entirely avoided and thus the ideal recovery cannot be achieved. In current implementations of the constraint-based approaches, the error rate of testing conditional independence is usually controlled individually for each test, under a conventional level such as 5% or 1%, without correcting the effect of multiple hypothesis testing. Therefore these implementations may fail to curb the FDR, especially for sparse graphs.

In this paper, we propose embedding an FDR-control procedure into the PC algorithm to curb the error rate of the skeleton of the learned PDAGs. Instead of individually controlling the type I error rate of each hypothesis test, the FDR-control procedure considers the hypothesis tests together to correct the effect of simultaneously testing the existence of multiple edges. We prove that the proposed method, named as the PC_{fdr}-skeleton algorithm, can control the FDR under a user-specified level at the limit of large sample sizes. In the case of moderate sample sizes (about several hundred), empirical experiments show that the method is able to control the FDR under the user-specified level, and a heuristic modification of the method is able to control the FDR more accurately around the user-specified level. Schäfer and Strimmer (2005) have applied an FDR procedure to graphical Gaussian models to control the FDR of the non-zero entries of the partial correlation matrix. Different from Schäfer and Strimmer (2005)'s work, our method, built within the frame-

3. "SGS" stands for **S**pirtes, **G**lymour and **S**cheines who invented this algorithm.
4. An extension of the IC algorithm which was named as IC* (see Pearl, 2000, pages 51–54) was previously also named as IC by Pearl and Verma (1992). Here we follow Pearl (2000).
5. "PC" stands for **P**eter Spirtes and **C**lark Glymour who invented this algorithm. A modified version of the PC algorithm which was named as PC* (see Spirtes et al., 2001, pages 89–90) was previously also named as PC by Spirtes and Glymour (1991). Here we follow Spirtes et al. (2001).
6. See footnote 4.

work of the PC algorithm, is not only applicable to the special case of Gaussian models, but also generally applicable to any models for which conditional-independence tests are available, such as discrete models.

We are particularly interested in the PC algorithm because it roots in conditional-independence relationships, the backbone of Bayesian networks, and p-values of hypothesis testing represent one type of error rates. We consider the skeleton of graphs because constraint-based algorithms usually first construct an undirected graph, and then annotate it into different types of graphs while keeping the skeleton as the same as that of the undirected one.

The PC$_{\text{fdr}}$-skeleton algorithm is not designed to replace or claimed to be superior over the standard PC algorithm, but provide the PC algorithm with the ability to control the FDR over the skeleton of the recovered network. The PC$_{\text{fdr}}$-skeleton algorithm controls the FDR while the standard PC algorithm controls the type I error rate, as illustrated in Section 3.1. Since there are no general superior relationships between different error-rate criteria, as explained in Appendix C, neither be there between the PC$_{\text{fdr}}$-skeleton algorithm and the standard PC algorithm. In research practices, researchers first decide which error rate is of interest, and then choose appropriate algorithms to control the error rate of interest. Generally they will not select an algorithm that sounds "superior" but controls the wrong error rate. Since the purpose of the paper is to provide the PC algorithm with the control over the FDR, we assume in this paper that the FDR has been selected as the error rate of interest, and selecting the error rate of interest is out of the scope of the paper.

The remainder of the paper is organized as follows. In Section 2, we review the PC algorithm, present the FDR-embedded PC algorithm, prove its asymptotic performance, and analyze its computational complexity. In Section 3, we evaluate the proposed method with simulated data, and demonstrate its real world applications to learning functional connectivity networks between brain regions using functional-magnetic-resonance-imaging (fMRI). Finally, we discuss the advantages and limitations of the proposed method in Section 4.

2. Controlling FDR with PC Algorithm

In this section, we first briefly introduce Bayesian networks and review the PC algorithm. Then, we expatiate on the FDR-embedded PC algorithm and its heuristic modification, prove their asymptotic performances, and analyze their computational complexity. At the end, we discuss other possible ideas of embedding FDR control into the PC algorithm.

2.1 Notations and Preliminaries

To assist the reading, notations frequently used in the paper are listed as follows:

$a, b, \cdots$	: vertices
$X_a, X_b, \cdots$	: variables respectively represented by vertices $a, b, \cdots$
$A, B, \cdots$	: vertex sets
$X_A, X_B, \cdots$	: variable sets respectively represented by vertex sets $A, B, \cdots$
V	: the vertex set of a graph
$N = \|V\|$	: the number of vertices of a graph
$a \rightarrow b$	: a directed edge or an ordered pair of vertices
$a \sim b$	: an undirected edge, or an unordered pair of vertices

E	: a set of directed edges
$E^{\sim}$	: the undirected edges derived from E, that is, $\{a \sim b \mid a \to b \text{ or } b \to a \in E\}$
$G = (V, E)$	: a directed graph composed of vertices in V and edges in E
$G^{\sim} = (V, E^{\sim})$	: the skeleton of a directed graph $G = (V, E)$
$\text{adj}(a, G)$	: vertices adjacent to a in graph G, that is, $\{b \mid a \to b \text{ or } b \to a \in E\}$
$\text{adj}(a, G^{\sim})$	: vertices adjacent to a in graph $G^{\sim}$, that is, $\{b \mid a \sim b \in E^{\sim}\}$
$a \perp b \mid C$	: vertices a and b are d-separated by vertex set C
$X_a \perp X_b \mid X_C$	: X_a and X_b are conditional independent given X_C
$p_{a \perp b \mid C}$	: the p-value of testing $X_a \perp X_b \mid X_C$

A Bayesian network encodes a set of conditional-independence relationships with a DAG $G = (V, E)$ according to the directed Markov property defined as follows.

Definition 1 ***the Directed Markov Property:*** *if A and B are d-separated by C where A, B and C are three disjoint sets of vertices, then X_A and X_B are conditionally independent given X_C, that is, $P(X_A, X_B|X_C) = P(X_A|X_C)P(X_B|X_C)$. (see Lauritzen, 1996, pages 46–53)*

The concept of *d-separation* (see Lauritzen, 1996, page 48) is defined as follows. A chain between two vertices a and b is a sequence $a = a_0, a_1, \ldots, a_n = b$ of distinct vertices such that $a_{i-1} \sim a_i \in E^{\sim}$ for all i=1, ... , n. Vertex b is a descendant of vertex a if and only if there is a sequence $a = a_0, a_1, \ldots, a_n = b$ of distinct vertices such that $a_{i-1} \to a_i \in E$ for all i=1, ..., n. For three disjoint subsets A, B and $C \subseteq V$, C d-separates A and B if and only if any chain π between $\forall a \in A$ and $\forall b \in B$ contains a vertex $\gamma \in \pi$ such that either

- arrows of π do not meet head-to-head at γ and $\gamma \in C$, or
- arrows of π meet head-to-head at γ and γ is neither in C nor has any descendants in C.

Moreover, a probability distribution P is *faithful* to a DAG G (see Spirtes et al., 2001, pages 13 and 81) if all and only the conditional-independence relationships derived from P are encoded by G. In general, a probability distribution may possess other independence relationships besides those encoded by a DAG.

It should be pointed out that there are often several different DAGs encoding the same set of conditional-independence relationships and they are called an *equivalence class* of DAGs. An equivalence class can be uniquely represented by a completed acyclic partially directed graph (CPDAG) (also called the essential graph in the literature) that has the same skeleton as a DAG does except that some edges are not directed (see Andersson et al., 1997).

2.2 PC Algorithm

If a probability distribution P is faithful to a DAG G, then the PC algorithm (see Spirtes et al., 2001, pages 84–89) is able to recover the equivalence class of the DAG G, given the set of the conditional-independence relationships. In general, a probability distribution may include other independence relationships besides those encoded by a DAG. The faithfulness assumption assures that the independence relationships can be perfectly encoded by a DAG. In practice, the information on conditional independence is usually unknown but extracted from data with statistical hypothesis testing. If the p-value of testing a hypothesis $X_a \perp X_b \mid X_C$ is lower than a user-specified level α

(conventionally 5% or 1%), then the hypothesis of conditional independence is rejected while the hypothesis of conditional dependence $X_a \not\perp X_b | X_C$ is accepted.

The first step of the PC algorithm is to construct an undirected graph $G^{\sim}$ whose edge directions will later be further determined with other steps, while the skeleton is kept the same as that of $G^{\sim}$. Since we restrict ourselves to the error rate of the skeleton, here we only present in Algorithm 1 the first step of the PC algorithm, as implemented in software Tetrad version 4.3.8 (see http://www.phil.cmu.edu/projects/tetrad), and refer to it as the PC-skeleton algorithm.

Algorithm 1 PC-skeleton

Input: the data X_V generated from a probability distribution faithful to a DAG G_{true},
and the significance level α for every statistical test of conditional independence
Output: the recovered skeleton $G^{\sim}$

```
1:  Form the complete undirected graph $G^{\sim}$ on the vertex set $V$.
2:  Let depth $d = 0$.
3:  repeat
4:    for each ordered pair of adjacent vertices $a$ and $b$ in $G^{\sim}$, that is, $a \sim b \in E^{\sim}$ do
5:      if $|\text{adj}(a, G^{\sim}) \setminus \{b\}| \geq d$, then
6:        for each subset $C \subseteq \text{adj}(a, G^{\sim}) \setminus \{b\}$ and $|C| = d$ do
7:          Test hypothesis $X_a \perp X_b | X_C$ and calculate the $p$-value $p_{a \perp b|C}$.
8:          if $p_{a \perp b|C} \geq \alpha$, then
9:            Remove edge $a \sim b$ from $G^{\sim}$.
10:           Update $G^{\sim}$ and $E^{\sim}$.
11:           break the for loop at line 6
12:         end if
13:       end for
14:     end if
15:   end for
16:   Let $d = d + 1$.
17: until $|\text{adj}(a, G^{\sim}) \setminus \{b\}| < d$ for every ordered pair of adjacent vertices $a$ and $b$ in $G^{\sim}$.
```

The theoretical foundation of the PC-skeleton algorithm is **Proposition 1**: if two vertices a and b are not adjacent in a DAG G, then there is a set of other vertices C that either all are neighbors of a or all are neighbors of b such that C d-separates a and b, or equivalently, $X_a \perp X_b | X_C$, according to the directed Markov property. Since two adjacent vertices are not d-separated by any set of other vertices (according to the directed Markov property), **Proposition 1** implies that a and b are not adjacent if and only if there is a d-separating C in either neighbors of a or neighbors of b. Readers should notice that the proposition does not imply that a and b are d-separated only by such a C, but just guarantees that a d-separating C can be found in either neighbors of a or neighbors b.

Proposition 1 *If vertices a and b are not adjacent in a DAG G, then there is a set of vertices C which is either a subset of* $\text{adj}(a, G) \setminus \{b\}$ *or a subset of* $\text{adj}(b, G) \setminus \{a\}$ *such that C d-separates a and b in G.* This proposition is a corollary of Lemma 5.1.1 on page 411 of book *Causation, Prediction, and Search* (Spirtes et al., 2001).

The logic of the PC-skeleton algorithm is as follows, under the assumption of perfect judgment on conditional independence. The most straightforward application of **Proposition 1** to structure learning is to exhaustively search all the possible neighbors of a and b to verify whether there is such a d-separating C to disconnect a and b. Since the possible neighbors of a and b are unknown, to guarantee the detection of such a d-separating C, all the vertices other than a and b should be searched as possible neighbors of a and b. However, such a straightforward application is very inefficient because it probably searches many unnecessary combinations of vertices by considering all the vertices other than a and b as their possible neighbors, especially when the true DAG is sparse. The PC-skeleton algorithm searches more efficiently, by keeping updating the possible neighbors of a vertex once some previously-considered possible neighbors have been found actually not connected with the vertex. Starting with a fully connected undirected graph $G^{\sim}$ (step 1), the algorithm searches for the d-separating C progressively by increasing the size of C, that is, the number of conditional variables, from zero (step 2) with the step size of one (step 16). Given the size of C, the search is performed for every vertex pair a and b (step 4). Once a C d-separating a and b is found (step 8), a and b are disconnected (step 9), and the neighbors of a and b are updated (step 10). In the algorithm, $G^{\sim}$ is continually updated, so $\text{adj}(a, G^{\sim})$ is also constantly updated as the algorithm progresses. The algorithm stops when all the subsets of the current neighbors of each vertex have been examined (step 17). The Tetrad implementation of the PC-skeleton algorithm examines an edge $a \sim b$ as two ordered pairs (a, b) and (b, a) (step 4), each time searching for the d-separating C in the neighbors of the first element of the pair (step 6). In this way, both the neighbors of a and the neighbors of b are explored.

The accuracy of the PC-skeleton algorithm depends on the discriminability of the statistical test of conditional independence. If the test can perfectly distinguish dependence from independence, then the algorithm can correctly recover the true underlying skeleton, as proved by Spirtes et al. (2001, pages 410–412). The outline of the proof is as follows. First, all the true edges will be recovered because an adjacent vertex pair $a \sim b$ is not d-separated by any vertex set C that excludes a and b. Second, if the edge between a non-adjacent vertex pair a and b has not been removed, subsets of either $\text{adj}(a, G) \setminus \{b\}$ or $\text{adj}(b, G) \setminus \{a\}$ will be searched until the C that d-separates a and b according to **Proposition 1** is found, and consequently the edge between a and b will be removed. If the judgments on conditional independence and conditional dependence are imperfect, the PC-skeleton algorithm is unstable. If an edge is mistakenly removed from the graph in the early stage of the algorithm, then other edges which are not in the true graph may be included in the graph (see Spirtes et al., 2001, page 87).

2.3 False Discovery Rate

In a statistical decision process, there are basically two sources of errors: the type I errors, that is, falsely rejecting negative hypotheses when they are actually true; and the type II errors, that is, falsely accepting negative hypotheses when their alternative, the positive hypotheses are actually true. The FDR (see Benjamini and Yekutieli, 2001) is a criterion to assess the errors when multiple hypotheses are simultaneously tested. It is the expected ratio of the number of falsely claimed positive results to that of all those claimed to be positive, as defined in Table 2. A variant of the FDR, the positive false discovery rate (pFDR), defined as in Table 2, was proposed by Storey (2002). Clearly, pFDR = FDR / $P(R_2 > 0)$, so the two measures will be similar if $P(R_2 > 0)$ is close to 1, and quite different otherwise.

Test Results	Truth		
	Negative	Positive	Total
Negative	TN (true negative)	FN (false negative)	R_1
Positive	FP (false positive)	TP (true positive)	R_2
Total	T_1	T_2	

Table 1: Results of multiple hypothesis testing, categorized according to the claimed results and the truth.

Full Name	Abbreviation	Definition
False Discovery Rate	FDR	$E(\mathrm{FP}/R_2)$ (See note *)
Positive False Discovery Rate	pFDR	$E(\mathrm{FP}/R_2 \mid R_2 > 0)$
Family-Wise Error Rate	FWER	$P(\mathrm{FP} \geq 1)$
Type I Error Rate (False Positive Rate)	α	$E(\mathrm{FP}/T_1)$
Specificity (True Negative Rate)	$1-\alpha$	$E(\mathrm{TN}/T_1)$
Type II Error Rate (False Negative Rate)	β	$E(\mathrm{FN}/T_2)$
Power (Sensitivity, True Positive Rate)	$1-\beta$	$E(\mathrm{TP}/T_2)$

Table 2: Criteria for multiple hypothesis testing. Here $E(x)$ means the expected value of x, and $P(\mathcal{A})$ means the probability of event $\mathcal{A}$. Please refer to Table 1 for related notations. * If $R_2 = 0$, FP/R_2 is defined to be 0.

The FDR is a reasonable criterion when researchers expect the "discovered" results are trustful and dependable in afterward studies. For example, in a pilot study, we are selecting candidate genes for a genetic research on Parkinson's disease. Because of the limited funding, we can only study a limited number of genes in the afterward genetic research. Thus, when selecting candidate genes in the pilot study, we hope that 95% of the selected candidate genes are truly associated with the disease. In this case, the FDR is chosen as the error rate of interest and should be controlled under 5%. Since similar situations are quite common in research practices, the FDR has been widely adopted in many research fields such as bioinformatics and neuroimaging.

In the context of learning the skeleton of a DAG, a negative hypothesis could be that a connection does not exist in the DAG, and a positive hypothesis could be that the connection exists. The FDR is the expected proportion of the falsely discovered connections to all those discovered. Learning network structures may face scenarios similar to the aforementioned pilot study, but the FDR control has not yet received adequate attention in structure learning.

Benjamini and Yekutieli (2001) have proved that, when the test statistics have positive regression dependency on each of the test statistics corresponding to the true negative hypotheses, the FDR can be controlled under a user-specified level q by Algorithm 2. In other cases of dependency, the FDR can be controlled with a simple conservative modification of the procedure by replacing H^* in Eq. (1) with $H(1+1/2,\ldots,+1/H)$. Storey (2002) has provided algorithms to control the pFDR for independent test statistics. For a review and comparison of more FDR methods, please refer to Qian and Huang (2005)'s work. It should be noted that the FDR procedures do not control the *realized* FDR of a trial under q, but control the *expected value* of the error rate when the procedures are repetitively applied to randomly sampled data.

Algorithm 2 FDR-stepup

Input: a set of p-values $\{p_i | i = 1, \ldots, H\}$, and the threshold of the FDR q
Output: the set of rejected null hypotheses

```
1: Sort the $p$-values of $H$ hypothesis tests in the ascendant order as $p_{(1)} \leq \ldots \leq p_{(H)}$.
2: Let $i = H$, and $H^* = H$ (or $H^* = H(1 + 1/2, \ldots, +1/H)$, depending on the assumption of the
   dependency among the test statistics).
3: while
       $$\frac{H^*}{i} p_{(i)} > q \quad \text{and} \quad i > 0, \qquad (1)$$
   do
4:     Let $i = i - 1$.
5: end while
6: Reject the null hypotheses associated with $p_{(1)}, \ldots, p_{(i)}$, and accept the null hypotheses associ-
   ated with $p_{(i+1)}, \ldots, p_{(H)}$.
```

Besides the FDR and the pFDR, other criteria, as listed in Table 2, can also be applied to assess the uncertainty of multiple hypothesis testing. The type I error rate is the expected ratio of the type I errors to all the negative hypotheses that are actually true. The type II error rate is the expected ratio of the type II errors to all the positive hypotheses that are actually true. The family-wise error rate is the probability that at least one of the accepted positive hypotheses are actually wrong. Generally, there are not mathematically or technically superior relationships among these error-rate criteria. Please refer to Appendix C for examples of typical research scenarios where each particular error rate is favoured.

Controlling both the type I and the type II error rates under a conventional level (such as $\alpha <$ 5% or 1% and $\beta <$ 10% or 5%) does not necessarily curb the FDR at a desired level. As shown in Eq. (2), if FP/T_1 and FN/T_2 are fixed and positive, FP/R_2 approaches 1 when T_2/T_1 is small enough. This is the case of sparse networks where the number of existing connections T_2 is much smaller than the number of non-existing connections T_1.

$$\frac{\mathrm{FP}}{R_2} = \frac{\frac{\mathrm{FP}}{T_1}}{\frac{\mathrm{FP}}{T_1} + (1 - \frac{\mathrm{FN}}{T_2})\frac{T_2}{T1}}. \qquad (2)$$

2.4 PC Algorithm with FDR

Steps 8–12 of the PC-skeleton algorithm control the type I error rate of each statistical test of conditional independence individually below a pre-defined level α, so the algorithm can not explicitly control the FDR. We propose embedding an FDR-control procedure into the algorithm to curb the error rate of the learned skeleton. The FDR-control procedure collectively considers the hypothesis tests related to the existence of multiple edges, correcting the effect of multiple hypothesis testing. The proposed method is described in Algorithm 3, and we name it as the $\mathrm{PC}_{\mathrm{fdr}}$-skeleton algorithm. Similar to the PC-skeleton algorithm, $G^{\sim}$, $\mathrm{adj}(a,G^{\sim})$ and $E^{\sim}$ are constantly updated as the algorithm progresses.

The PC$_{\mathrm{fdr}}$-skeleton and the PC-skeleton algorithms share the same search strategy, but differ on the judgment of conditional independence. The same as the PC-skeleton algorithm, the PC$_{\mathrm{fdr}}$-skeleton algorithm increases d, the number of conditional variables, from zero (step 3) with the step size of one (step 25), and also keeps updating the neighbors of vertices (steps 14 and 15) when some previously-considered possible neighbors have been considered not connected (step 13). The PC$_{\mathrm{fdr}}$-skeleton algorithm differs from the PC-skeleton algorithm on the inference of d-separation, with its steps 11–20 replacing steps 8–12 of the PC-skeleton algorithm. In the PC-skeleton algorithm, two vertices are regarded as d-separated once the conditional-independence test between them yields a p-value larger than the pre-defined significant level α. In this way, the type I error of each statistical test is controlled separately, without consideration of the effect of multiple hypothesis testing. The PC$_{\mathrm{fdr}}$-skeleton algorithm records in $p^{max}_{a\sim b}$ the up-to-date maximum p-value associated with an edge $a \sim b$ (steps 9 and 10), and progressively removes those edges whose non-existence is accepted by the FDR procedure (step 12), with $P^{max} = \{p^{max}_{a\sim b}\}_{a\neq b}$ and the pre-defined FDR level q being the input. The FDR procedure, **Algorithm 2**, is invoked at step 12, either immediately after every element of P^{max} has been assigned a valid p-value for the first time, or later once any element of P^{max} is updated.

The $p^{max}_{a\sim b}$ is the upper bound of the p-value of testing the hypothesis that a and b are d-separated by at least one of the vertex sets C searched in step 7. According to the directed Markov property, a and b are not adjacent if and only if there is a set of vertices $C \subseteq V \setminus \{a,b\}$ d-separating a and b. As the algorithm progresses, the d-separations between a and b by vertex sets $C_1,\ldots,C_K \subseteq V \setminus \{a,b\}$ are tested respectively, and consequently a sequence of p-values $p_1,\ldots,p_K$ are calculated. If we use $p^{max}_{a\sim b} = \max_{i=1}^{K} p_i$ as the statistic to test the negative hypothesis that there is, though unknown, a C_j among $C_1,\ldots,C_K$ d-separating a and b, then due to

$$P(p^{max}_{a\sim b} \leq p) = P(p_i \leq p \text{ for all } i=1,\ldots,K) \leq P(p_j \leq p) = p, \tag{3}$$

$p^{max}_{a\sim b}$ is the upper bound of the p-value of testing the negative hypothesis. Eq. (3) also clearly shows that the PC-skeleton algorithm controls the type I error rate of the negative hypothesis, since its step 8 is equivalent to "if $p^{max}_{a\sim b} < \alpha$, then … " if $p^{max}_{a\sim b}$ is recorded in the PC-skeleton algorithm.

The statistical tests performed at step 8 of the PC$_{\mathrm{fdr}}$-skeleton algorithm generally are not independent with each other, since the variables involved in two hypotheses of conditional independence may overlap. For example, conditional-independence relationships $a\perp b_1|C$ and $a\perp b_2|C$ both involve a and C. It is very difficult to prove whether elements of P^{max} have positive regression dependency or not, so rigorously the conservative modification of Algorithm 2, should be applied at step 12. However, since $p^{max}_{a\sim b}$ is probably a loose upper bound of the p-value of testing $a \nsim b$, in practice we simply apply the FDR procedure that is correct for positive regression dependency.

It should be noted that different from step 9 of the PC-skeleton algorithm, step 14 of the PC$_{\mathrm{fdr}}$-skeleton algorithm may remove edges other than just $a \sim b$, because the decisions on other edges can be affected by the updating of $p^{max}_{a\sim b}$.

A heuristic modification of the PC$_{\mathrm{fdr}}$-skeleton algorithm is to remove $p^{max}_{a\sim b}$ from P^{max} once edge $a \sim b$ has been deleted from $G^{\sim}$ at step 14. We name this modified version as the PC$_{\mathrm{fdr*}}$-skeleton algorithm. In the PC$_{\mathrm{fdr}}$-skeleton algorithm, $p^{max}_{a\sim b}$ is still recorded in P^{max} and input to the FDR procedure after the edge $a \sim b$ has been removed. This guarantees that the algorithm can asymptotically keep the FDR under the user-specified level q (see Section 2.5). The motivation of

Algorithm 3 PC$_{\text{fdr}}$-skeleton

Input: the data X_V generated from a probability distribution faithful to a DAG G_{true},
and the FDR level q for the discovered skeleton
Output: the recovered skeleton $G^{\sim}$

```
1:  Form the complete undirected graph $G^{\sim}$ on the vertex set $V$
2:  Initialize the maximum $p$-values associated with edges as
    $P^{max} = \{p^{max}_{a\sim b} = -1\}_{a \neq b}$.
3:  Let depth $d = 0$.
4:  repeat
5:    for each ordered pair of adjacent vertices $a$ and $b$ in $G^{\sim}$, that is, $a \sim b \in E^{\sim}$ do
6:      if $|\mathrm{adj}(a, G^{\sim}) \setminus \{b\}| \geq d$, then
7:        for each subset $C \subseteq \mathrm{adj}(a, G^{\sim}) \setminus \{b\}$ and $|C| = d$ do
8:          Test hypothesis $X_a \perp X_b | X_C$ and calculate the $p$-value $p_{a\perp b|C}$.
9:          if $p_{a\perp b|C} > p^{max}_{a\sim b}$, then
10:           Let $p^{max}_{a\sim b} = p_{a\perp b|C}$.
11:           if every element of $P^{max}$ has been assigned a valid $p$-value by step 10, then
12:             Run the FDR procedure, Algorithm 2, with $P^{max}$ and $q$ as the input.
13:             if the non-existence of certain edges are accepted, then
14:               Remove these edges from $G^{\sim}$.
15:               Update $G^{\sim}$ and $E^{\sim}$.
16:               if $a \sim b$ is removed, then
17:                 break the for loop at line 7.
18:               end if
19:             end if
20:           end if
21:         end if
22:       end for
23:     end if
24:   end for
25:   Let $d = d + 1$.
26: until $|\mathrm{adj}(a, G^{\sim}) \setminus \{b\}| < d$ for every ordered pair of adjacent vertices $a$ and $b$ in $G^{\sim}$.
```

* A heuristic modification at step 15 of the algorithm is to remove from P^{max} the $p^{max}_{a\sim b}$s whose associated edges have been deleted from $G^{\sim}$ at step 14, that is, to update P^{max} as $P^{max} = \{p^{max}_{a\sim b}\}_{a\sim b \in E^{\sim}}$ right after updating $E^{\sim}$ at step 15. This heuristic modification is named as the **PC$_{\text{fdr*}}$-skeleton algorithm**.

the heuristic modification is that if an edge has been eliminated, then it should not be considered in the FDR procedure any longer. Though we cannot theoretically prove the asymptotic performance of the heuristic modification in the sense of controlling the FDR, it is shown to control the FDR closely around the user-specified level in our empirical experiments and gain more detection power than that of the PC$_{\text{fdr}}$-skeleton algorithm (see Section 3).

2.5 Asymptotic Performance

Here we prove that the PC$_{\text{fdr}}$-skeleton algorithm is able to control the FDR under a user-specified level q ($q > 0$) at the limit of large sample sizes if the following assumptions are satisfied:

(A1) The probability distribution P is faithful to a DAG G_{true}.

(A2) The number of vertices is fixed.

(A3) Given a fixed significant level of testing conditional-independence relationships, the power of detecting conditional-dependence relationships with statistical tests approaches 1 at the limit of large sample sizes. (For the definition of power in hypothesis testing, please refer to Table 2.)

Assumption (A1) is generally assumed when graphical models are applied, although it restricts the probability distribution P to a certain class. Assumption (A2) is usually implicitly stated, but here we explicitly emphasize it because it simplifies the proof. Assumption (A3) may seem demanding, but actually it can be easily satisfied by standard statistical tests, such as the likelihood-ratio test introduced by Neyman and Pearson (1928), if the data are identically and independently sampled. Two statistical tests that satisfy Assumption (A3) are listed in Appendix B.

The detection power and the FDR of the PC$_{\text{fdr}}$-skeleton algorithm and its heuristic modification at the limit of large sample sizes are elucidated in Theorems 1 and 2. The detailed proofs are provided in Appendix A.

Theorem 1 *Assuming (A1), (A2) and (A3), both the PC$_{\text{fdr}}$-skeleton algorithm and its heuristic modification, the PC$_{\text{fdr*}}$-skeleton algorithm, are able to recover all the true connections with probability one as the sample size approaches infinity:*

$$\lim_{m\to\infty} P(E^{\sim}_{true} \subseteq E^{\sim}) = 1,$$

where $E^{\sim}_{true}$ denotes the set of the undirected edges derived from the true DAG G_{true}, $E^{\sim}$ denotes the set of the undirected edges recovered with the algorithms, and m denotes the sample size.

Theorem 2 *Assuming (A1), (A2) and (A3), the FDR of the undirected edges recovered with the PC$_{\text{fdr}}$-skeleton algorithm approaches a value not larger than the user-specified level q as the sample size m approaches infinity:*

$$\limsup_{m\to\infty} \mathrm{FDR}(E^{\sim}, E^{\sim}_{true}) \le q,$$

where $\mathrm{FDR}(E^{\sim}, E^{\sim}_{true})$ *is defined as*

$$\begin{cases} \mathrm{FDR}(E^{\sim}, E^{\sim}_{true}) & = & E\left[\frac{|E^{\sim}\backslash E^{\sim}_{true}|}{|E^{\sim}|}\right], \\ \text{Define } \frac{|E^{\sim}\backslash E^{\sim}_{true}|}{|E^{\sim}|} & = & 0, \quad \text{if } |E^{\sim}| = 0. \end{cases}$$

2.6 Computational Complexity

The PC$_{\text{fdr}}$-skeleton algorithm spends most of its computation on performing statistical tests of conditional independence at step 8 and controlling the FDR at step 12. Since steps 13 to 19 of the PC$_{\text{fdr}}$-skeleton algorithm play a role similar to steps 8 to 12 of the PC-skeleton algorithm do, and all the other parts of both algorithms employ the same search strategy, the computation spent by the PC$_{\text{fdr}}$-skeleton algorithm on statistical tests has the same complexity as that by the PC-skeleton algorithm. The only extra computational cost of the PC$_{\text{fdr}}$-skeleton algorithm is at step 12 for controlling the FDR.

The computational complexity of the search strategy employed by the PC algorithm has been studied by Kalisch and Bühlmann (2007) and Spirtes et al. (see 2001, pages 85–87). Here to make the paper self-contained, we briefly summarize the results as follows. It is difficult to analyze the complexity exactly, but if the algorithm stops at the depth $d = d_{max}$, then the number of conditional-independence tests required is bounded by

$$T = 2C_N^2 \sum_{d=0}^{d_{max}} C_{N-2}^d,$$

where N is the number of vertices, C_N^2 is the number of combinations of choosing 2 un-ordered and distinct elements from N elements, and similarly C_{N-2}^2 is the number of combinations of choosing from $N-2$ elements. In the worst case that $d_{max} = N-2$, the complexity is bounded by $2C_N^2 2^{N-2}$. The bound usually is very loose, because it assumes that no edge has been removed until $d = d_{max}$. In real world applications, the algorithm is very fast for sparse networks.

The computational complexity of the FDR procedure, Algorithm 2, generally is $O(\ H\log(H)$ + $H) = O(H\log(H))$ where $H = C_N^2$ is the number of input p-values. The sorting at step 1 costs $H\log(H)$ and the "while" loop from step 3 to step 5 repeats H times at most. However, if the sorted P^{max} is recorded during the computation, each time when an element of P^{max} is updated at step 10 of the PC$_{\text{fdr}}$-skeleton algorithm, the complexity of keeping the updated P^{max} sorted is only $O(H)$. With this optimization, the complexity of the FDR-control procedure is $O(H\log(H))$ at its first operation, and is $O(H)$ later. The FDR procedure is invoked only when $p_{a\perp b|C} > p_{a\sim b}^{max}$. In the worst case that $p_{a\perp b|C}$ is always larger than $p_{a\sim b}^{max}$, the complexity of the computation spent on the FDR control in total is bounded by $O(C_N^2 \log(C_N^2) + TC_N^2)$ = $O(N^2\log(N) + TN^2)$ where T is the number of performed conditional-independence tests. This is a very loose bound because it is rare that $p_{a\perp b|C}$ is always larger than $p_{a\sim b}^{max}$.

The computational complexity of the heuristic modification, the PC$_{\text{fdr*}}$-skeleton algorithm, is the same as that of the PC$_{\text{fdr}}$-skeleton algorithm, since they share the same search strategy and both employ the FDR procedure. In the PC$_{\text{fdr*}}$-skeleton algorithm, the size of P^{max} keeps decreasing as the algorithm progresses, so each operation of the FDR procedure is more efficient. However, since the PC$_{\text{fdr*}}$-skeleton algorithm adjusts the effect of multiple hypothesis testing less conservatively, it may remove less edges than the PC$_{\text{fdr}}$-skeleton algorithm does, and invokes more conditional-independence tests. Nevertheless, their complexity is bounded by the same limit in the worst case.

It is unfair to directly compare the computational time of the PC$_{\text{fdr}}$-skeleton algorithm against that of the PC-skeleton algorithm, because if the q of the former is set at the same value as the α of the latter, the former will remove more edges and perform much less statistical tests, due to its more stringent control over the type I error rate. A reasonable way is to compare the time spent on the FDR control at step 12 against that on conditional-independence tests at step 8 in each run of the PC$_{\text{fdr}}$-skeleton algorithm. If P^{max} is kept sorted during the learning process as aforementioned, then each time (except the first time) the FDR procedure just needs linear computation time (referring to the size of P^{max}) with simple operations such as division and comparing two numerical values. Thus, we suspect that the FDR procedure will not contribute much to the total computation time of the structure learning. In our simulation study in Section 3.1, the extra computation added by the FDR control was only a tiny portion, less than 0.5%, to that spent on testing conditional independence, performed with the Cochran-Mantel-Haenszel (CMH) test (see Agresti, 2002, pages 231–232), as shown in Tables 3 and 4.

2.7 Miscellaneous Discussions Discussions

An intuitive and attracting idea of adapting the PC-skeleton algorithm to the FDR control is to "smartly" determine such an appropriate threshold of the type I error rate α that will let the errors be controlled at the pre-defined FDR level q. Given a particular problem, it is very likely that the FDR of the graphs learned by the PC-skeleton algorithm is an monotonically increasing function of the pre-defined threshold α of the type I error rate. If this hypothesis is true, then there is a one-to-one mapping between α and q for the particular problem. Though we cannot prove this hypothesis rigorously, the following argument may be enlightening. Instead of directly focusing on FDR = $E(\mathrm{FP}/R_2)$ (see Table 2), the expected ratio of the number of false positives (FP) to the number of accepted positive hypotheses (R_2), we first focus on $E(\mathrm{FP})/E(R_2)$, the ratio of the expected number of false positives to the expected number of accepted positive hypotheses, since the latter is easier to link with the type I error rate according to Eq. (2), as shown in Eq. (4),

$$\frac{E(\mathrm{FP})}{E(R_2)} = \frac{E\left(\frac{\mathrm{FP}}{T_1}\right)}{E\left(\frac{\mathrm{FP}}{T_1}+(1-\frac{\mathrm{FN}}{T_2})\frac{T_2}{T1}\right)} = \frac{E\left(\frac{\mathrm{FP}}{T_1}\right)}{E\left(\frac{\mathrm{FP}}{T_1}\right)+\left(1-E\left(\frac{\mathrm{FN}}{T_2}\right)\right)\frac{T_2}{T_1}} = \frac{\alpha}{\alpha+(1-\beta)\frac{T_2}{T_1}}, \tag{4}$$

where α and β are the type I error rate and the type II error rate respectively. A sufficient condition for $E(\mathrm{FP})/E(R_2)$ being a monotonically increasing function of the type I error rate includes (I) $(1-\beta)/\alpha > \partial(1-\beta)/\partial\alpha$, (II) $T_1 > 0$ and (III) $T_2 > 0$, where $\partial(1-\beta)/\partial\alpha$ is the derivative of $(1-\beta)$ over α. If $(1-\beta)$, regarded as a function of α, is a concave curve from $(0,0)$ to $(1,1)$, then condition (I) is satisfied. Recall that $(1-\beta)$ versus α actually is the receiver operating characteristic (ROC) curve, and that with an appropriate statistic the ROC curve of a hypothesis test is usually a concave curve from $(0,0)$ to $(1,1)$, we speculate that condition (I) is not difficult to satisfy. With the other two mild conditions (II) $T_1 > 0$ and (III) $T_2 > 0$, we could expect that $E(\mathrm{FP})/E(R_2)$ is a monotonically increasing function of α. $E(\mathrm{FP})/E(R_2)$ is the ratio of the expected values of two random variables, while $E(\mathrm{FP}/R_2)$ is the expected value of the ratio of two random variables. Generally, there is not a monotonic relationship between $E(\mathrm{FP})/E(R_2)$ and $E(\mathrm{FP}/R_2)$. Nevertheless, if the average number of false positives, $E(\mathrm{FP})$, increases proportionally faster than that of the accepted positives, $E(R_2)$, we speculate that under certain conditions, the FDR = $E(\mathrm{FP}/R_2)$ also increases accordingly. Thus the FDR may be a monotonically increasing function of the threshold α of the type I error rate for the PC-skeleton algorithm.

However, even though the FDR of the PC-skeleton algorithm may decrease as the pre-defined significant level α decreases, the FDR of the PC-skeleton algorithm still cannot be controlled at the user-specified level for general problems by "smartly" choosing an α beforehand, but somehow has to be controlled in a slightly different way, such as the PC$_{\mathrm{fdr}}$-skeleton algorithm does. First, the value of such an α for the FDR control depends on the true graph, but unfortunately the graph is unknown in problems of structure learning. According to Eq. (2), the realized FDR is a function of the realized type I and type II error rates, as well as T_2/T_1, which in the context of structure learning is the ratio of the number of true connections to the number of non-existing connections. Since

T_2/T_1 is unknown, such an α cannot be determined completely in advance without any information about the true graph, but has to be estimated practically from the observed data. Secondly, the FDR method we employ is such a method that estimates the α from the data to control the FDR of multiple hypothesis testing. The output of the FDR algorithm is the rejection of those null hypotheses associated with p-values $p_{(1)}, \ldots, p_{(i)}$ (see Algorithm 2). Given $p_{(1)} \leq \ldots \leq p_{(H)}$, the output is equivalent to the rejection of all those hypotheses whose p-values are smaller than or equal to $p_{(i)}$. In other words, it is equivalent to setting $\alpha = p_{(i)}$ in the particular multiple hypothesis testing. Thirdly, the PC$_{\text{fdr}}$-skeleton algorithm is a valid solution to combining the FDR method with the PC-skeleton algorithm. Because the estimation of the α depends on p-values, and p-values are calculated one by one as the PC-skeleton algorithm progresses with hypothesis tests, the α cannot be estimated separately before the PC-skeleton algorithm starts running, but the estimation has to be embedded within the algorithm, like in the PC$_{\text{fdr}}$-skeleton algorithm.

Another idea on the FDR control in structure learning is a two-stage algorithm. The first stage is to draft a graph that correctly includes all the existing edges and their orientations but may also include non-existing edges as well. The second stage is to select the real parents for each vertex, with the FDR controlled, from the set of potential parents determined in the first stage. The advantage of this algorithm is that the selection of real parent vertices in the second stage is completely decoupled from the determination of edge orientations, because all the parents of each vertex have been correctly connected with the particular vertex in the first stage. However, a few concerns about the algorithm should be noticed before researchers start developing this two-stage algorithm. First, to avoid missing many existing edges in the first stage, a considerable number of non-existing edges may have to be included. To guarantee a perfect protection of the existing edges given any randomly sampled data, the first stage must output a graph whose skeleton is a fully connected graph. The reason for this is that the type I error rate and the type II error rate contradict each other and the latter reaches zero generally when the former approaches one (see Appendix C). Rather than protecting existing edges perfectly, the first stage should trade off between the type I and the type II errors, in favour of keeping the type II error rate low. Second, selecting parent vertices from a set of candidate vertices in the second stage, in certain sense, can be regarded as learning the structure of a sub-graph locally, in which error-rate control remains as a crucial problem. Thus erro-rate control is still involved in both of the two stages. Though this two-stage idea may not essentially reduce the problem of the FDR control to an easier task, it may break the big task of simultaneously learning all edges to many local structure-learning tasks.

3. Empirical Evaluation

The PC$_{\text{fdr}}$-skeleton algorithm and its heuristic modification are evaluated with simulated data sets, in comparison with the PC-skeleton algorithm, in the sense of the FDR, the type I error rate and the power. The PC$_{\text{fdr}}$-skeleton and the PC-skeleton algorithms are also applied to two real functional-magnetic-resonance-imaging (fMRI) data sets, to check whether the two algorithms correctly curb the error rates that they are supposed to control in real world applications.

3.1 Simulation Study

The simulated data sets are generated from eight different DAGs, shown in Figure 1, with the number of vertices N = 15, 20, 25 or 30, and the average degree of vertices D = 2 or 3. The DAGs are generated as follows:

(1) Sample $\frac{N \times D}{2}$ undirected edges from $\{a \sim b | a, b \in V \text{ and } a \neq b\}$ with equal probabilities and without replacement to compose an undirected graph $G^{\sim}_{true}$.

(2) Generate a random order $\succ$ of vertices with permutation.

(3) Orientate the edges of $G^{\sim}$ according to the order $\succ$. If a is before b in the order $\succ$, then orientate the edge $a \sim b$ as $a \rightarrow b$. Denote the orientated graph as a DAG G_{true}.

For each DAG, we associate its vertices with (conditional) binary probability distributions as follows, to extend it to a Bayesian network.

(1) Specify the strength of (conditional) dependence as a parameter $\delta > 0$.

(2) Randomly assign each vertex $a \in V$ with a dependence strength $\delta_a = 0.5\delta$ or -0.5δ, with equal possibilities.

(3) Associate each vertex $a \in V$ with a logistic regression model

$$\begin{aligned} \Delta &= \sum_{b \in \text{pa}[a]} X_b \delta_b, \\ P(X_a = 1 | X_{\text{pa}[a]}) &= \frac{\exp(\Delta)}{1 + \exp(\Delta)}, \\ P(X_a = -1 | X_{\text{pa}[a]}) &= \frac{1}{1 + \exp(\Delta)}, \end{aligned}$$

where pa$[a]$ denotes the parent vertices of a.

The parameter δ reflects the strength of dependence because if the values of all the other parent variables are fixed, the difference between the conditional probabilities of a variable $X_a = 1$ given a parent variable $X_b = 1$ and -1 is

$$\left| \text{logit}[P(X_a = 1 | X_b = 1, X_{\text{pa}[a] \setminus \{b\}})] - \text{logit}[P(X_a = 1 | X_b = -1, X_{\text{pa}[a] \setminus \{b\}})] \right| = |2\delta_b| = \delta,$$

where the logit function is defined as $\text{logit}(x) = \log(\frac{x}{1-x})$.

Since the accuracy of the PC-skeleton algorithm and its FDR versions is related to the discriminability of the statistical tests, we generated data with different values of δ (δ= 0.5, 0.6, 0.7, 0.8, 0.9 and 1.0) to evaluate the algorithms' performances with different power of detecting conditional dependence. The larger the absolute value of δ is, the easier the dependence can be detected with statistical tests. Because statistical tests are abstract queries yielding p-values about conditional independence for the structure-learning algorithms, the accuracy of the algorithms is not determined by the particular procedure of a statistical test, or a particular family of conditional probability distributions but by the discriminability of the statistical tests. Given a fixed sample size, the stronger the conditional-dependence relationships are, the higher discriminability the statistical tests have. By varying the dependence strength δ of the *binary* conditional probability distributions, we have varied the discriminability of the statistical tests, as if by varying the dependence strength of *other* probability distribution families. To let the readers intuitively understand the dependence strength of these δ values, we list as follows examples of probability pairs whose logit contrasts are equal to these δ values:

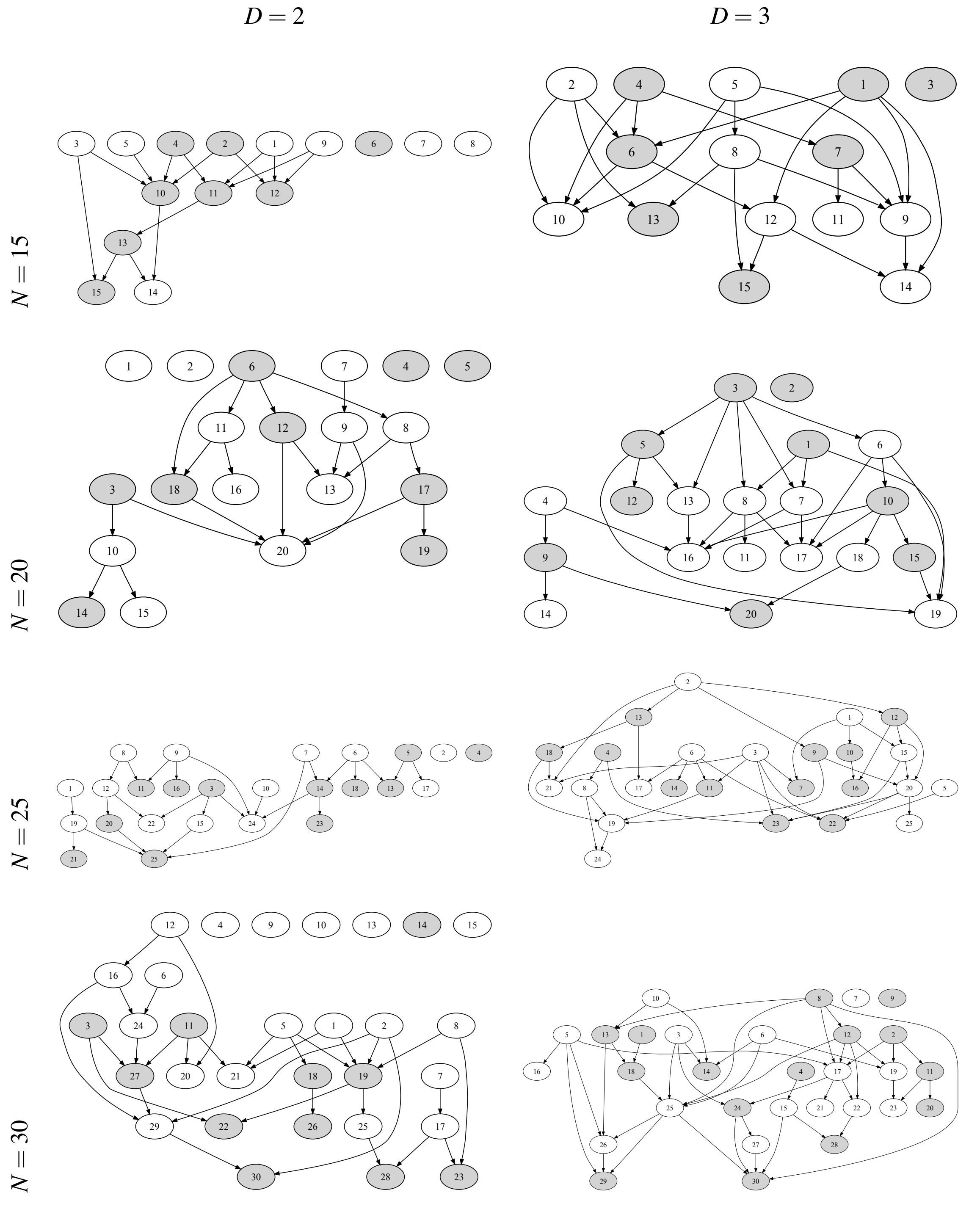


Figure 1: DAGs used in the simulation study. N denotes the number of vertices and D denotes the average degree of the vertices. Unshaded vertices are associated with positive dependence strength 0.5δ, and shaded ones are associated with negative dependence strength -0.5δ.

$$0.5 = \text{logit}(0.5622) - \text{logit}(0.4378), \quad 0.6 = \text{logit}(0.5744) - \text{logit}(0.4256),$$
$$0.7 = \text{logit}(0.5866) - \text{logit}(0.4134), \quad 0.8 = \text{logit}(0.5987) - \text{logit}(0.4013),$$
$$0.9 = \text{logit}(0.6106) - \text{logit}(0.3894), \quad 1.0 = \text{logit}(0.6225) - \text{logit}(0.3775).$$

.

In total, we performed the simulation with 48 Bayesian networks generated with all the combinations of the following parameters:

$$\begin{aligned} N &= 15, 20, 25, 30; \\ D &= 2, 3; \\ \delta &= 0.5, 0.6, 0.7, 0.8, 0.9, 1.0. \end{aligned}$$

From each Bayesian network, we repetitively generated 50 data sets each of 500 samples to estimate the statistical performances of the algorithms. A non-model-based test, the Cochran-Mantel-Haenszel (CMH) test (see Agresti, 2002, pages 231–232), was employed to the test conditional independence among random variables. Both the significant level α of the PC-skeleton algorithm and the FDR level q of the PC$_{\text{fdr}}$-skeleton algorithm and its heuristic modification were set at 5%.

Figures 2, 3 and 4 respectively show the empirical FDR, power and type I error rate of the algorithms, estimated from the 50 data sets repetitively generated from each Bayesian network, with error bars indicating the 95% confidence intervals of these estimations. The PC$_{\text{fdr}}$-skeleton algorithm controls the FDR under the user-specified level 5% for all the 48 Bayesian networks, and the PC$_{\text{fdr*}}$-skeleton algorithm steadily controls the FDR closely around 5%, while the PC-skeleton algorithm yields the FDR ranging from about 5% to about 35%, and above 15% in many cases, especially for those sparser DAGs with the average degree of vertices $D = 2$. The PC$_{\text{fdr}}$-skeleton algorithm is conservative, with the FDR notably lower than the user-specified level, while its heuristic modification controls the FDR more accurately around the user-specified level, although the correctness of the heuristic modification has not been theoretically proved. As the discriminability of the statistical tests increases, the power of all the algorithms approaches 1. When their FDR level q is set at the same value as the α of the PC-skeleton algorithm, the PC$_{\text{fdr}}$-skeleton algorithm and its heuristic modification control the type I error rate more stringently than the PC-skeleton algorithm does, so their power generally is lower than that of the PC-skeleton algorithm. Figure 4 also clearly shows, as Eq. 3 implies, that it is the type I error rate, rather than the FDR, that the PC-skeleton algorithm controls under 5%.

Figure 5 shows the average computational time spent during each run of the PC$_{\text{fdr}}$-skeleton algorithm and its heuristic modification on the statistical tests of (conditional) independence at step 8 and the FDR control at step 12. The computational time was estimated on the platform of an Intel Xeon 1.86GHz CPU and 4G RAM, and with the code implemented in Matlab R14. Tables 3 and 4 show the average ratios of the computational time spent on the FDR control to that spent on the statistical tests. The average ratios are not more than 2.57‰ for all the 48 Bayesian networks. The relatively small standard deviations, as shown in brackets in the tables, indicate that these estimated ratios are trustful. Because the PC$_{\text{fdr}}$-skeleton algorithm and its heuristic modification employ the same search strategy as the PC-skeleton algorithm does, this result evidences that the extra computation cost to achieve the control over the FDR is trivial in comparison with the computation already spent by the PC-skeleton algorithm on statistical tests.

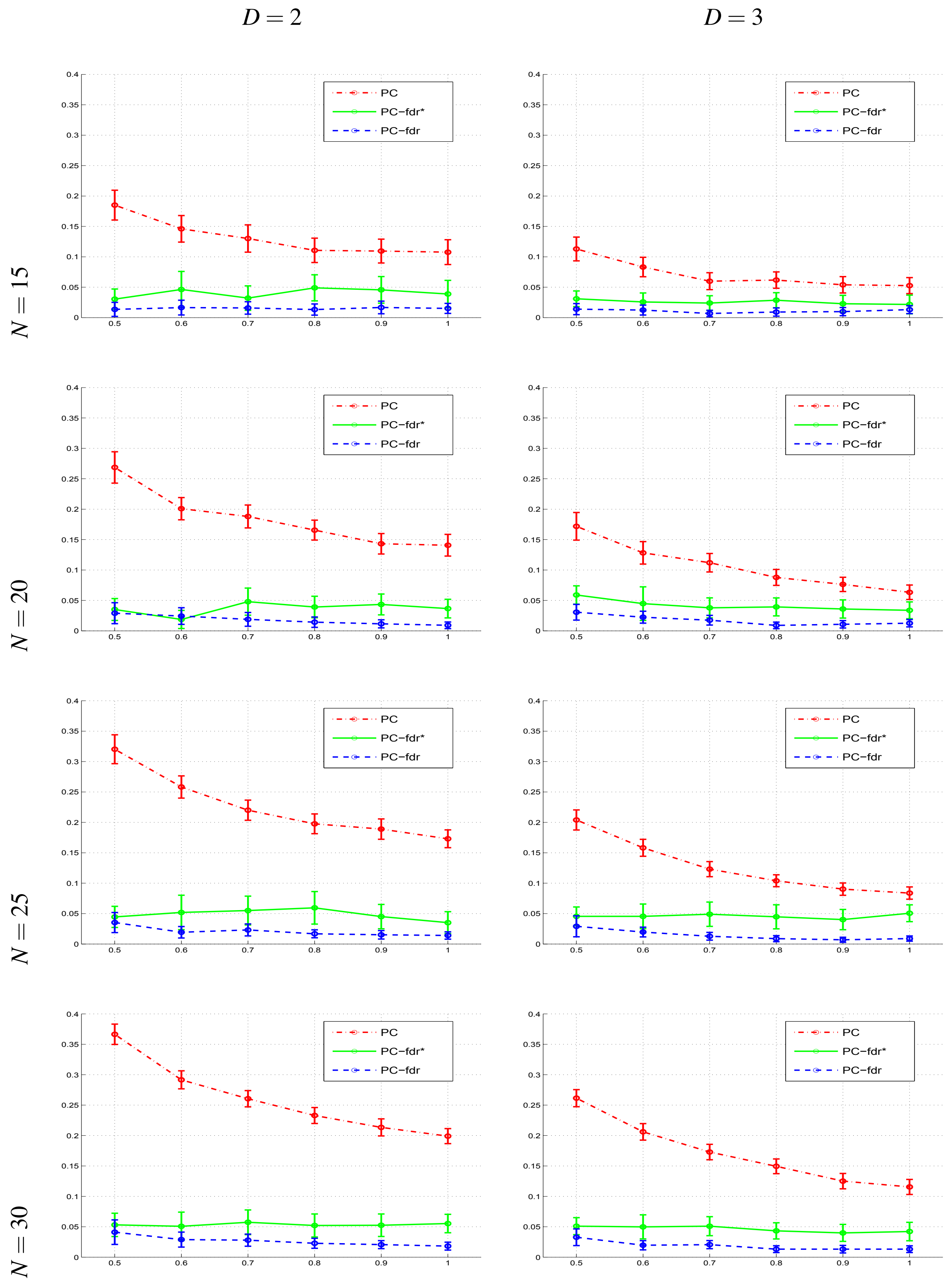


Figure 2: The FDR (with 95% confidence intervals) of the PC-skeleton algorithm, the PC_{fdr}-skeleton algorithm and the $\text{PC}_{\text{fdr}*}$-skeleton algorithm on the DAGs in Figure 1, as the dependence parameter δ shown on the x-axes increases from 0.5 to 1.0.

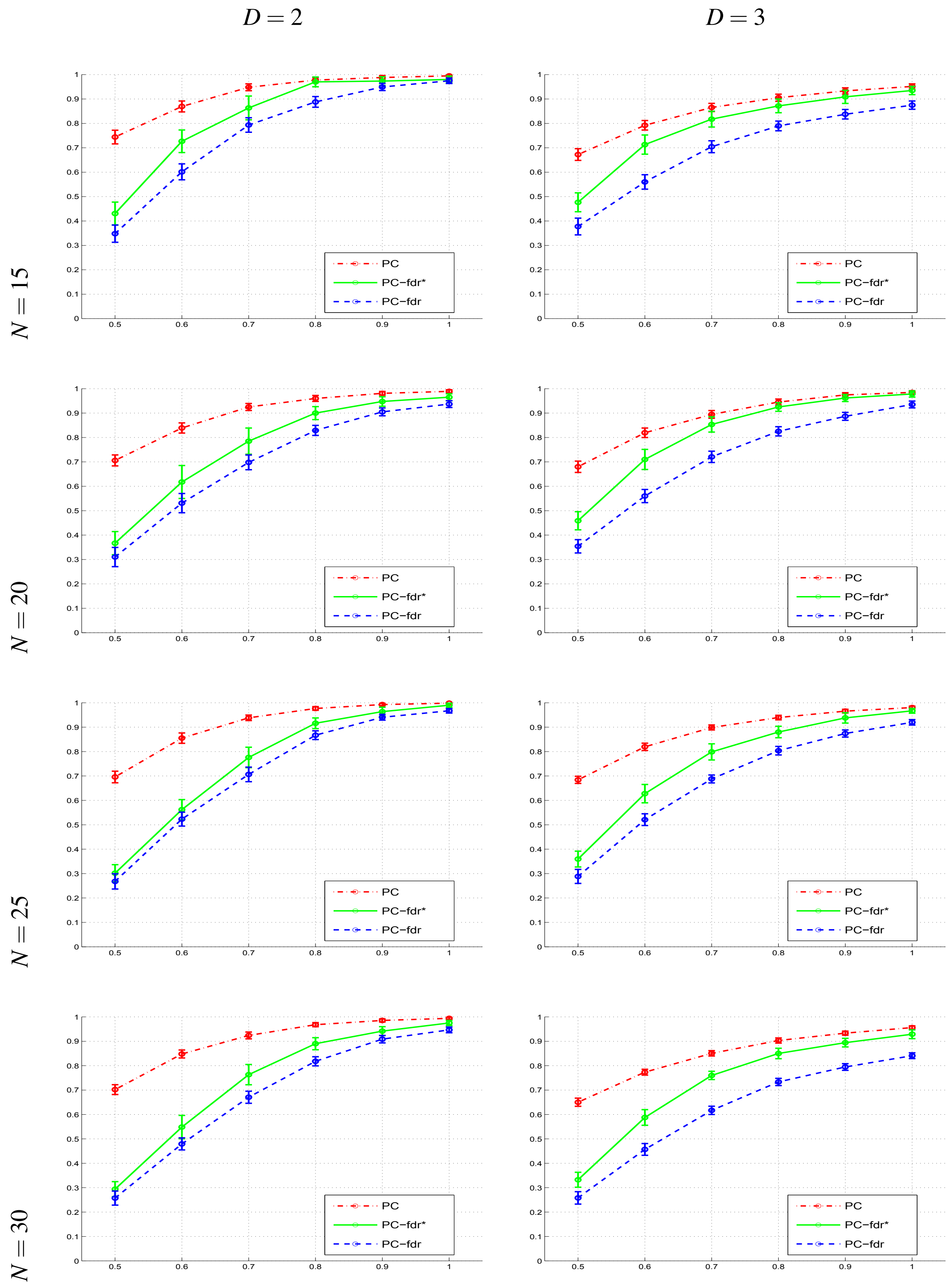


Figure 3: The power (with 95% confidence intervals) of the PC-skeleton algorithm, the $\mathrm{PC_{fdr}}$-skeleton algorithm and the $\mathrm{PC_{fdr*}}$-skeleton algorithm on the DAGs in Figure 1, as the dependence parameter δ shown on the x-axes increases from 0.5 to 1.0.

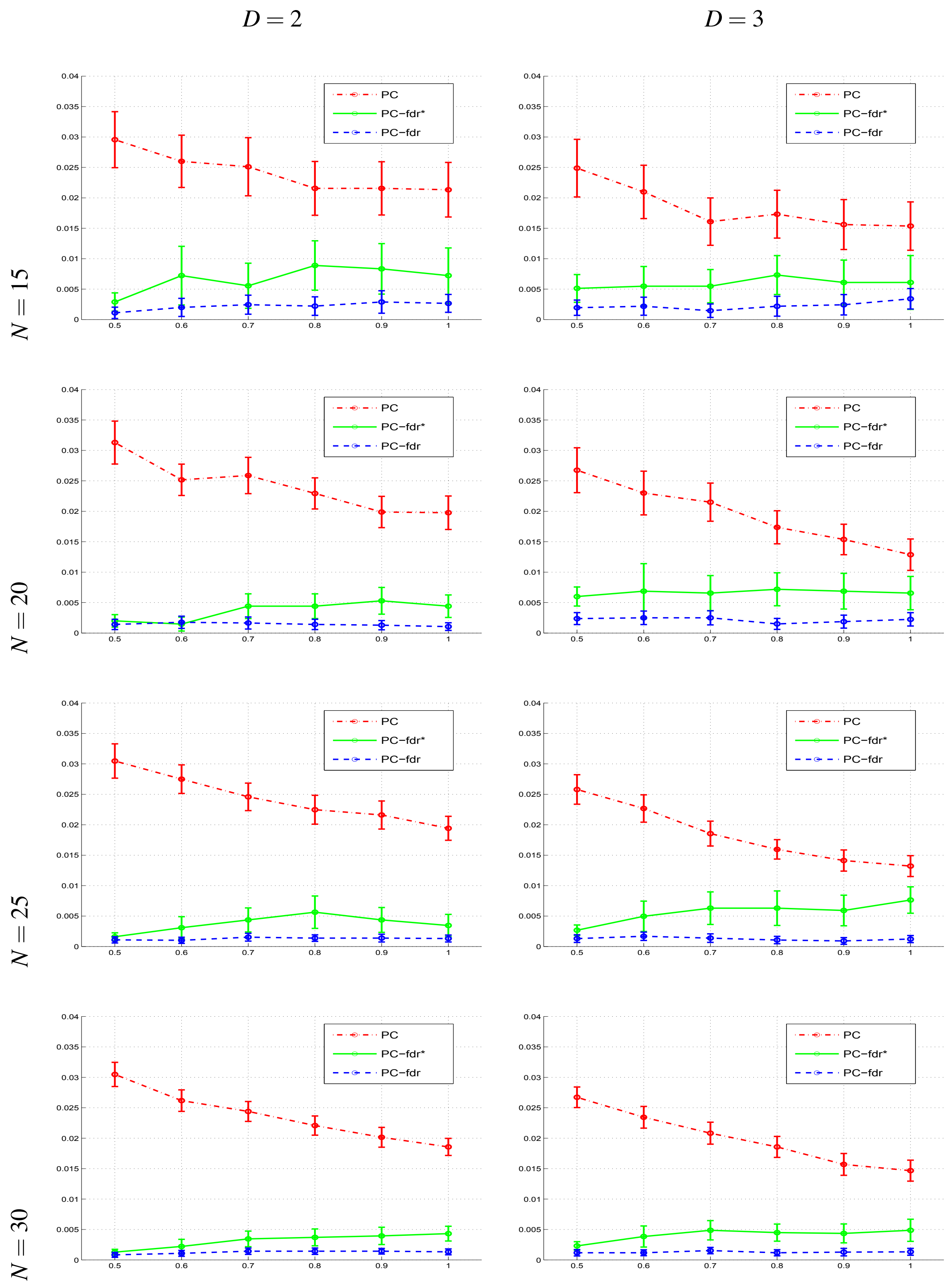


Figure 4: The type I error rates (with 95% confidence intervals) of the PC-skeleton algorithm, the PC_{fdr}-skeleton algorithm and the PC_{fdr*}-skeleton algorithm on the DAGs in Figure 1, as the dependence parameter δ shown on the x-axes increases from 0.5 to 1.0.

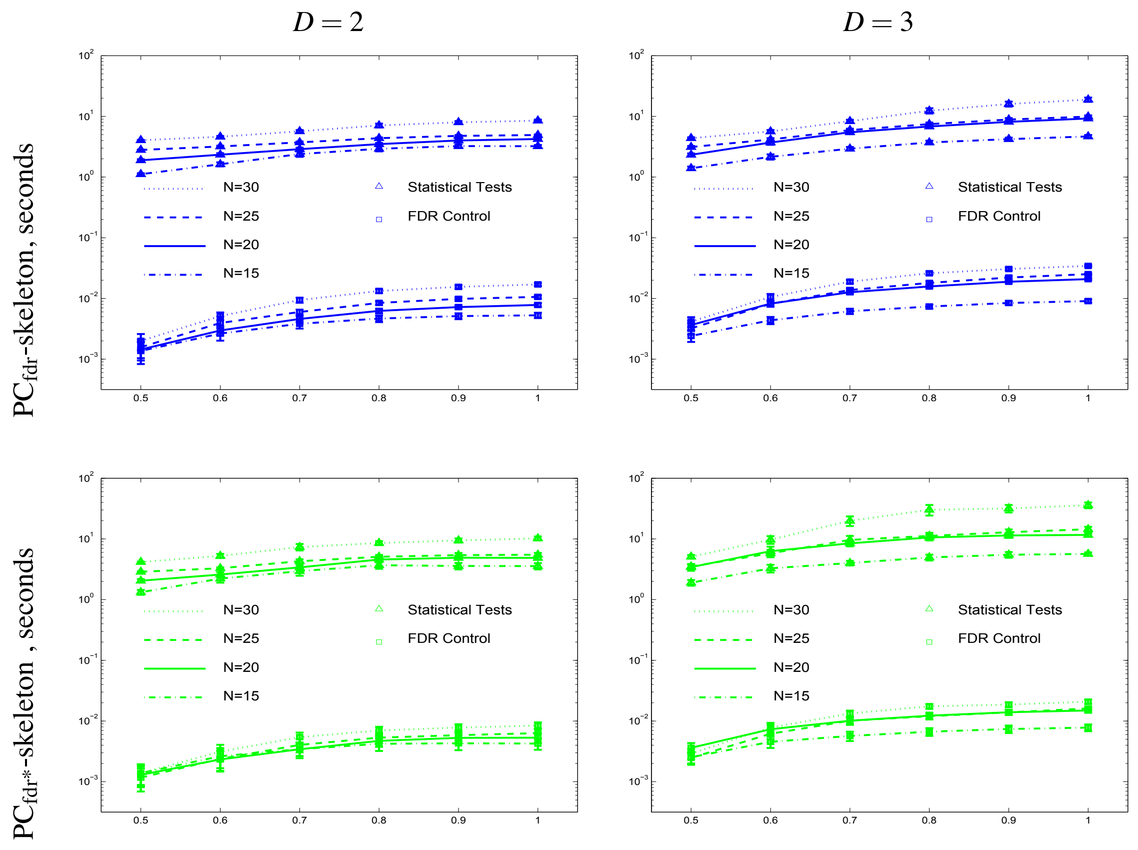


Figure 5: The average computational time (in seconds, with 95% confidence intervals) spent on the FDR control and statistical tests during each run of the PC$_{\text{fdr}}$-skeleton algorithm and its heuristic modification.

	δ	N=15	N=20	N=25	N=30
$D = 2$	0.5	1.11e-03 (3.64e-04)	7.19e-04 (2.32e-04)	5.44e-04 (1.79e-04)	4.81e-04 (1.37e-04)
	0.6	1.48e-03 (2.03e-04)	1.24e-03 (2.15e-04)	1.21e-03 (3.32e-04)	1.09e-03 (1.44e-04)
	0.7	1.58e-03 (1.68e-04)	1.61e-03 (2.01e-04)	1.59e-03 (1.31e-04)	1.64e-03 (1.09e-04)
	0.8	1.63e-03 (1.64e-04)	1.81e-03 (1.61e-04)	1.93e-03 (1.20e-04)	1.89e-03 (1.04e-04)
	0.9	1.59e-03 (1.50e-04)	1.83e-03 (1.50e-04)	2.06e-03 (1.19e-04)	1.95e-03 (9.63e-05)
	1.0	1.64e-03 (1.51e-04)	1.88e-03 (1.59e-04)	2.15e-03 (1.12e-04)	2.01e-03 (9.01e-05)
$D = 3$	0.5	1.69e-03 (3.70e-04)	1.50e-03 (2.55e-04)	9.80e-04 (1.90e-04)	9.10e-04 (1.52e-04)
	0.6	2.06e-03 (2.82e-04)	2.22e-03 (1.45e-04)	1.93e-03 (1.71e-04)	1.82e-03 (1.47e-04)
	0.7	2.11e-03 (1.84e-04)	2.36e-03 (1.24e-04)	2.31e-03 (1.19e-04)	2.29e-03 (1.12e-04)
	0.8	2.02e-03 (1.68e-04)	2.35e-03 (1.20e-04)	2.45e-03 (1.28e-04)	2.20e-03 (1.15e-04)
	0.9	2.04e-03 (1.50e-04)	2.34e-03 (9.05e-05)	2.53e-03 (1.15e-04)	2.03e-03 (1.23e-04)
	1.0	1.99e-03 (1.41e-04)	2.28e-03 (9.18e-05)	2.57e-03 (9.66e-05)	1.92e-03 (1.25e-04)

Table 3: The average ratios (with their standard deviations in brackets) of the computational time spent on the FDR control to that spent on the statistical tests during each run of the PC$_{\text{fdr}}$-skeleton algorithm.

	δ	N=15	N=20	N=25	N=30
$D=2$	0.5	8.88e-04 (2.57e-04)	5.93e-04 (2.25e-04)	3.94e-04 (1.60e-04)	3.24e-04 (1.15e-04)
	0.6	1.12e-03 (2.99e-04)	8.82e-04 (3.19e-04)	7.08e-04 (2.58e-04)	5.86e-04 (1.81e-04)
	0.7	1.17e-03 (2.76e-04)	1.04e-03 (2.92e-04)	9.02e-04 (1.51e-04)	7.45e-04 (1.37e-04)
	0.8	1.16e-03 (2.66e-04)	1.04e-03 (1.83e-04)	1.03e-03 (1.38e-04)	8.23e-04 (1.24e-04)
	0.9	1.22e-03 (2.73e-04)	1.08e-03 (1.91e-04)	1.06e-03 (8.76e-05)	8.37e-04 (1.35e-04)
	1.0	1.21e-03 (2.68e-04)	1.11e-03 (2.09e-04)	1.12e-03 (1.04e-04)	8.31e-04 (1.05e-04)
$D=3$	0.5	1.33e-03 (3.42e-04)	1.01e-03 (1.86e-04)	6.74e-04 (1.54e-04)	5.49e-04 (1.08e-04)
	0.6	1.44e-03 (3.46e-04)	1.19e-03 (1.63e-04)	1.08e-03 (2.08e-04)	7.97e-04 (8.83e-05)
	0.7	1.43e-03 (2.41e-04)	1.20e-03 (1.10e-04)	1.09e-03 (1.51e-04)	7.19e-04 (7.80e-05)
	0.8	1.36e-03 (1.38e-04)	1.17e-03 (9.36e-05)	1.10e-03 (1.20e-04)	6.48e-04 (9.32e-05)
	0.9	1.35e-03 (1.34e-04)	1.24e-03 (1.03e-04)	1.10e-03 (8.31e-05)	6.19e-04 (6.57e-05)
	1.0	1.39e-03 (1.72e-04)	1.29e-03 (1.01e-04)	1.14e-03 (9.77e-05)	5.98e-04 (4.86e-05)

Table 4: The average ratios (with their standard deviations in brackets) of the computational time spent on the FDR control to that spent on the statistical tests during each run of the $\text{PC}_{\text{fdr*}}$-skeleton algorithm.

3.2 Applications to Real fMRI Data

We applied the PC_{fdr}-skeleton and the PC-skeleton algorithms to real-world research tasks, studying the connectivity network between brain regions using functional magnetic resonance imaging (fMRI). The purpose of the applications is to check whether the two algorithms correctly curb the error rates in real world applications. The purpose of the applications is not, and also should not be, to answer the question "which algorithm, the PC_{fdr}-skeleton or the PC-skeleton, is superior?", for the following reasons. Basically, the two algorithms control different error rates between which there is not a superior relationship (see Appendix C). Secondly, the error rate of interest for a specific application is selected largely not by mathematical superiority, but by researchers' interest and the scenario of research (see Appendix C). Thirdly, the simulation study has clearly revealed the properties of and the differences (not superiority) between the two algorithms. Lastly, the approximating graphical models behind the real fMRI data are unknown, so the comparison on the real fMRI data is rough, rather than rigorous.

The two algorithms were applied to two real fMRI data sets, one including 11 discrete variables and 1300 observations, and the other including 25 continuous variables and 1098 observations. The first data set, denoted by "the bulb-squeezing data set", was collected from 10 healthy subjects each of whom was asked to squeeze a rubber bulb with their left hand at three different speeds or at a constant force, as cued by visual instruction. The data involve eleven variables: the speed of squeezing and the activities of the ten brain regions listed in Table 5. The speed of squeezing is coded as a discrete variable with four possible values: the high speed, the medium speed, the low speed, and the constant force. The activities of the brain regions are coded as discrete variables with three possible values: high activation, medium activation and low activation. The data of each subject include 130 time points. The data of the ten subjects are pooled together, so in total there are 1300 time points. For details of the data set, please refer to Li et al. (2008).

The second data set, denoted by "the sentence-picture data set", was collected from a single subject performing a cognitive task. In each trial of the task, the subject was shown in sequence an affirmative sentence and a simple picture, and then answered whether the sentence correctly

Full Name	Abbreviation
Left/Right anterior cingulate cortex	L_ACC, R_ACC
Left/Right lateral cerebellar hemispheres	L_CER, R_CER
Left/Right primary motor cortex	L_M1, R_M1
Left/Right pre-frontal cortex	L_PFC, R_PFC
Left/Right supplementary motor cortex	L_SMA, R_SMA

Table 5: Brain regions involved in the bulb-squeezing data set. The prefixes "L" or "R" in the abbreviations stand for "Left" or "Right", respectively.

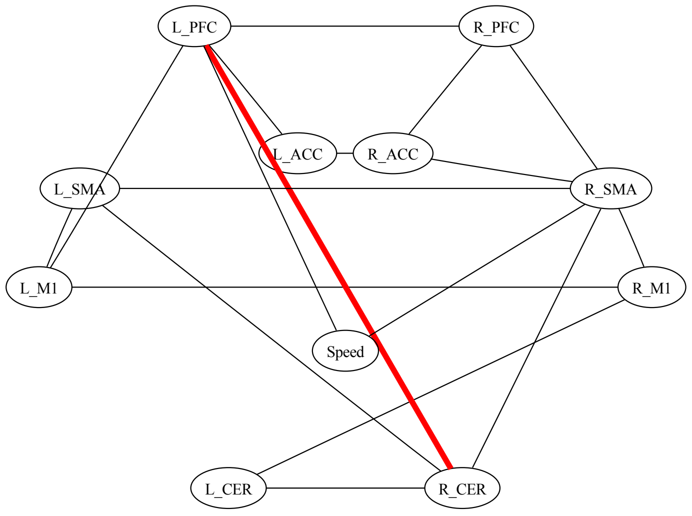


Figure 6: The networks learned from the bulb-squeezing data set, by the PC_{fdr}-skeleton and the PC-skeleton algorithms. For ease of comparison, the networks learned by the two algorithms are overlaid. Thin solid black edges are those connections detected by both the two algorithms; thick solid red edges are those connections detected only by the PC-skeleton algorithm. For the full names of the brain regions, please refer to Table 5.

described the picture. In half of the trials, the picture was presented first, followed by the sentence. In the remaining trials, the sentence was presented first, followed by the picture. The data involve the activities of 25 brain regions, as listed in Table 6, encoded as continuous variables, at 1098 time points. For details of the data set, please refer to Keller et al. (2001) and Mitchell et al. (2004).

The PC_{fdr}-skeleton and the PC-skeleton algorithms were applied to both the bulb-squeezing and the sentence-picture data sets. Both the FDR level q of the PC_{fdr}-skeleton algorithm and the type-I-error-rate level α of the PC-skeleton algorithm were set at 5%. For the bulb-squeezing data set, all of whose variables are discrete, conditional independence was tested with Pearson's Chi-square test; for the sentence-picture data set, all of whose variables are continuous, conditional independence was tested with the t-test for partial correlation coefficients (Fisher, 1924).

The networks learned from the bulb-squeezing data set and the networks learned from the sentence-picture data set are shown in Figures 6 and 7 respectively. For ease of comparison, the

Full Name	Abbreviation	
Calcarine fissure	CALC	
Left/Right dorsolateral prefrontal cortex	L_DLPFC,	R_DLPFC
Left/Right frontal eye field	L_FEF,	R_FEF
Left inferior frontal gyrus	L_IFG	
Left/Right inferior parietal lobe	L_IPL,	R_IPL
Left/Right intraparietal sulcus	L_IPS,	R_IPS
Left/Right inferior temporal lobule	L_IT,	R_IT
Left/Right opercularis	L_OPER,	R_OPER
Left/Right posterior precentral sulcus	L_PPREC,	R_PPREC
Left/Right supramarginal gyrus	L_SGA,	R_SGA
Supplementary motor cortex	SMA	
Left/Right superior parietal lobule	L_SPL,	R_SPL
Left/Right temporal lobe	L_T,	R_T
Left/Right triangularis	L_TRIA,	R_TRIA

Table 6: Brain regions involved in the sentence-picture data set. The prefixes "L" or "R" in the abbreviations stand for "Left" or "Right", respectively.

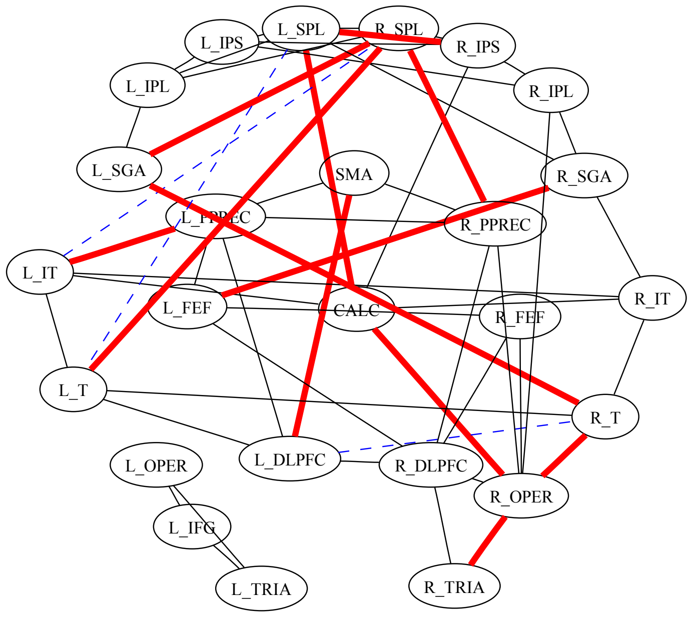


Figure 7: The networks learned from the sentence-picture data set, by the PC$_{\text{fdr}}$-skeleton and the PC-skeleton algorithms. For ease of comparison, the networks learned by the two algorithms are overlaid. Thin solid black edges are those connections detected by both the two algorithms; thin dashed blue edges are those connections detected only by the PC$_{\text{fdr}}$-skeleton algorithm; thick solid red edges are those connections detected only by the PC-skeleton algorithm. For the full names of the brain regions, please refer to Table 6.

Bulb-Squeezing

	Assumed Truth		*Realized* Detection			
	Exist	Non-Exist	Correct	False	FDR	Type I Error Rate
PC_{fdr}	17	$\frac{11*(11-1)}{2}-17=38$	17	0	**0.00%**	0.00%
PC			17	1	5.56%	**2.63%**

Sentence-Picture

	Assumed Truth		*Realized* Detection			
	Exist	Non-Exist	Correct	False	FDR	Type I Error Rate
PC_{fdr}	39	$\frac{25*(25-1)}{2}-39=261$	39	3	**7.14%**	1.14%
PC			39	12	23.5%	**4.60%**

Table 7: The *realized* error rates of the PC_{fdr}-skeleton and the PC algorithms on the bulb-squeezing and sentence-picture data sets, under the TI assumption that all and only those connections detected by both of the two algorithms truly exist.

networks learned by the two algorithms are overlaid. Thin solid black edges are those connections detected by both the two algorithms; thin dashed blue edges are those detected only by the PC_{fdr}-skeleton algorithm; thick solid red edges are those detected only by the PC-skeleton algorithm. In Figure 6, there are 17 thin solid black edges, 0 thin dashed blue edge and 1 thick solid red edge; in Figure 7, there are 39 thin solid black edges, 3 thin dashed blue edges and 12 thick solid red edges.

The results intuitively, though not rigorously, support our expectation of the performances of the two algorithms in real world applications. First, since the data sets are relatively large, with the sample sizes more than 1000, it is expected that both algorithms will recover many of the existing connections, and consequently the networks recovered by the two algorithms may share many common connections. This is consistent with the fact that in Figures 6 and 7 there are many thin solid black edges, that is, the connections recovered by both algorithms.

Second, since the PC_{fdr}-skeleton algorithm is designed to control the FDR while the PC-skeleton algorithm to control the type I error rate, it is expected that the two algorithms will control the corresponding error rate under or around the pre-defined level, which is 5% in this study. To verify whether the error rates were controlled as expected, we need to know which connections really exist and which do not. Unfortunately, this is very difficult for real data sets, because unlike the simulated data, the true models behind the real data are unknown, and in the literature, researchers usually tend to report evidences supporting the existence of connections rather than supporting the non-existence. However, since the sample sizes of the two data sets are relatively large, more than 1000, we can speculate that both of the two algorithms have recovered most of the existing connections. Extrapolating this speculation a bit, we intuitively assume that those connections detected by both of the two algorithms truly exist while all the others do not. In other words, we assume that all and only the thin black edges in the figures truly exist. We refer to this assumption as the "True Intersection" (TI) assumption. The statistics about Figures 6 and 7, under the TI assumption, are listed in Table 7. The *realized* FDR of the PC_{fdr}-skeleton algorithm on the bulb-squeezing and sentence-picture data sets are 0.00% and 7.14%, respectively; the *realized* type I error rate of the PC-skeleton algorithm on the bulb-squeezing and sentence-picture data sets are 2.63% and 4.60%, respectively. Considering that the *realized* error rate, as a statistic extracted from just a trial, may slightly deviate from its expected value, these results, derived under the TI assumption, support that the two algorithms controlled the corresponding error rate under the pre-defined level 5%.

Third, according to Eq. (2), the sparser and the larger the true network is, the higher the FDR of the PC-skeleton algorithm will be. For the bulb-squeezing data set, there are 11 vertices, and under the TI assumption, 17 existing connections and 38 non-existing connections. In this case, the *realized* FDR of the PC-skeleton algorithm is only 5.56% (Table 7). For the sentence-picture data set, there are 25 vertices, and under the TI assumption, 39 existing connections and 261 non-existing connections. In this case, the *realized* FDR of the PC-skeleton algorithm rises to 23.5% (Table 7). This notable increase of the *realized* FDR is consistent with the prediction based on Eq. (2).

It should be noted that the preceding arguments are rough rather than rigorous, since they are based on the TI assumption rather than the true models behind the data. However, because the true models behind the real data are unknown, the TI assumption is a practical and intuitive approach to assess the performance of the two algorithms in the two real world applications.

4. Conclusions and Discussions

We have proposed a modification of the PC algorithm, the $\mathrm{PC_{fdr}}$-skeleton algorithm, to curb the false discovery rate (FDR) of the skeleton of the learned Bayesian networks. The FDR-control procedure embedded into the PC algorithm collectively considers the hypothesis tests related to the existence of multiple edges, correcting the effect of multiple hypothesis testing. Under mild assumptions, it is proved that the $\mathrm{PC_{fdr}}$-skeleton algorithm can control the FDR under a user-specified level q ($q > 0$) at the limit of large sample sizes (see Theorem 2). In the cases of moderate sample size (about several hundred), empirical experiments have shown that the method is still able to control the FDR under the user-specified level. The $\mathrm{PC_{fdr*}}$-skeleton algorithm, a heuristic modification of the proposed method, has shown better performance in the simulation study, steadily controlling the FDR closely around the user-specified level and gaining more detection power, although its asymptotic performance has not been theoretically proved. Both the $\mathrm{PC_{fdr}}$-skeleton algorithm and its heuristic modification can asymptotically recover all the edges of the true DAG (see Theorem 1). The idea of controlling the FDR can be extended to other constraint-based methods, such as the inductive causation (IC) algorithm (see Pearl, 2000, pages 49–51) and the fast-causal-inference (FCI) algorithm (see Spirtes et al., 2001, pages 142–146).

The simulation study has also shown that the extra computation spent on achieving the FDR control is almost negligible when compared with that already spent by the PC algorithm on statistical tests of conditional independence. The computational complexity of the new algorithm is closely comparable with that of the PC algorithm.

As a modification based on the PC algorithm, the proposed method is modular, consisting of the PC search strategy, statistical tests of conditional independence and an FDR-control procedure. Different statistical tests and FDR-control procedures can be “plugged in”, depending on the data type and the statistical model. Thus, the method is applicable to any models for which statistical tests of conditional independence are available, such as discrete models and Gaussian models.

It should be noted that the $\mathrm{PC_{fdr}}$-skeleton algorithm is not proposed to replace the PC-skeleton algorithm. Instead, it provides an approach to controlling the FDR, a certain error-rate criterion for testing the existence of multiple edges. When multiple edges are involved in structure learning, there are different applicable error-rate criteria, such as those listed in Table 2. The selection of these criteria depends on researchers’ interest and the scenarios of studies, which is beyond the scope of this paper. When the FDR is applied, the PC$_{\text{fdr}}$-skeleton algorithm is preferable; when the type I error rate is applied, the PC-skeleton algorithm is preferable. The technical difference between the two algorithms is that the PC$_{\text{fdr}}$-skeleton algorithm adaptively adjusts the type I error rate according to the sparseness of the network to achieve the FDR control, while the PC-skeleton algorithm fixes the type I error rate.

Currently the FDR control is applied only to the skeleton of the graph, but not to the directions of the edges yet. The final output of the PC algorithm is a partially directed acyclic graph that uniquely represents an equivalence class of DAGs, so a possible improvement for the PC$_{\text{fdr}}$-skeleton algorithm is to extend the FDR control to the directions of the recovered edges. Because both type I and type II errors may lead to wrong directions in the later steps of the PC algorithm, minimizing direction errors may lead to a related, yet different, error-control task.

The asymptotic performance of the PC$_{\text{fdr}}$-skeleton algorithm has only been proved under the assumption that the number of vertices is fixed. Its behavior when both the number of vertices and the sample size approach infinity has not been studied yet. Kalisch and Bühlmann (2007) proved that for Gaussian Bayesian networks, the PC algorithm consistently recovers the equivalence class of the underlying sparse DAG, as the sample size m approaches infinity, even if the number of vertices N grows as quickly as $O(m^{\lambda})$ for any $0 < \lambda < \infty$. Their idea is to adaptively decrease the type I error rate α of the PC-skeleton algorithm as both the number of vertices and the sample size increase. It is desirable to study whether similar behavior can be achieved with the PC$_{\text{fdr}}$-skeleton algorithm if the FDR level q is adjusted appropriately as the sample size increases.

A Matlab® package of the PC$_{\text{fdr}}$-skeleton algorithm and its heuristic modification is downloadable at www.junningli.org/software.

Acknowledgments

The authors thank Dr. Martin J. McKeown for sharing the functional magnetic resonance imaging (fMRI) data (Li et al., 2008) and helpful discussions.

Appendix A. Proof of Theorems

To assist the reading, we list below notations frequently used in the proof:

$G^{\sim}_{true}$: the skeleton of the true underlying Bayesian network.

$\mathcal{A}_{a\sim b}$: the event that edge $a \sim b$ is in the graph recovered by the PC$_{\text{fdr}}$-skeleton algorithm.

$\mathcal{A}_{E^{\sim}_{true}}$: $\mathcal{A}_{E^{\sim}_{true}} = \bigcap_{a\sim b \in E^{\sim}_{true}} \mathcal{A}_{a\sim b}$, the joint event that all the edges in $G^{\sim}_{true}$, the skeleton of the true DAG, are recovered by the PC$_{\text{fdr}}$-skeleton algorithm.

$E^{\nsim}_{true}$: the set of the undirected edges that are not in $G^{\sim}_{true}$.

$p_{a\sim b}$: the value of $p^{max}_{a\sim b}$ when the PC$_{\text{fdr}}$-skeleton algorithm stops.

$C^{*}_{a\sim b}$: a certain vertex set that d-separates a and b in G_{true} and that is also a subset of either $\text{adj}(a, G^{\sim}_{true}) \setminus \{b\}$ or $\text{adj}(b, G^{\sim}_{true}) \setminus \{a\}$, according to Proposition 1. $C^{*}_{a\sim b}$ is defined only for vertex pairs that are not adjacent in the true DAG G_{true}.

$p^*_{a\sim b}$: the p-value of testing $X_a \perp X_b | X_{C^*_{a\sim b}}$. The conditional-independence relationship may not be really tested during the process of the PC$_{\text{fdr}}$-skeleton algorithm, but $p^*_{a\sim b}$ can still denote the value as if the conditional-independence relationship was tested.

H^*: the value in Eq. (1) that is either H or $H(1+1/2,\ldots,+1/H)$, depending on the assumption of the dependency of the p-values.

Lemma 1 *If as m approaches infinity, the probabilities of K events $\mathcal{A}_i(m),\cdots,\mathcal{A}_K(m)$ approach 1 at speed*

$$P(\mathcal{A}_i(m)) = 1 - o(\beta(m))$$

where $\lim_{m\to\infty} \beta(m) = 0$ and K is a finite integer, then the probability of the joint of all these events approaches 1 at speed

$$P\left(\bigcap_{i=1}^{K} \mathcal{A}_i(m)\right) \geq 1 - Ko(\beta(m))$$

as m approaches infinity.

Proof

$\because \bigcap_{i=1}^{K} \mathcal{A}_i(m) = \overline{\bigcup_{i=1}^{K} \overline{\mathcal{A}}_i(m)}$.

$\therefore P\left(\bigcap_{i=1}^{K} \mathcal{A}_i(m)\right) = 1 - P\left(\bigcup_{i=1}^{K} \overline{\mathcal{A}}_i(m)\right) \geq 1 - \sum_{i=1}^{K} P(\overline{\mathcal{A}}_i(m))$

$= 1 - \sum_{i=1}^{K} [1 - P(\mathcal{A}_i(m))] = 1 - \sum_{i=1}^{K} o(\beta(m)) = 1 - Ko(\beta(m)).$ ■

Corollary 1 *If $\mathcal{A}_i(m),\cdots,\mathcal{A}_K(m)$ are a finite number of events whose probabilities each approach 1 as m approaches infinity:*

$$\lim_{m\to\infty} P(\mathcal{A}_i(m)) = 1,$$

then the probability of the joint of all these events approaches 1 as m approaches infinity:

$$\lim_{m\to\infty} P\left(\bigcap_{i=1}^{K} \mathcal{A}_i(m)\right) = 1.$$

Lemma 2 *If there are F ($F \geq 1$) false hypotheses among H tested hypotheses, and the p-values of the all the false hypotheses are smaller than or equal to $\frac{F}{H^*}q$, where H^* is either H or $H(1+1/2,\ldots,+1/H)$, depending on the assumption of the dependency of the p-values, then all the F false hypotheses will be rejected by the FDR procedure, Algorithm 2.*

Proof

Let p_i ($i = 1,\cdots,H$) denote the p-value of the ith hypothesis, p_f denote the maximum of the p-values of the F false hypotheses, and r_f denote the rank of p_f in the ascending order of $\{p_i\}_{i=1,\cdots,H}$.

$\because p_f$ is the maximum of the p-values of the F false hypotheses.

$\therefore r_f = |\{p_i | p_i \leq p_f\}| \geq F$.

$\therefore \frac{H^*}{r_f} p_f \leq \frac{H^*}{F} p_f$.

$\because p_f \leq \frac{F}{H^*}q$.
$\therefore \frac{H^*}{r_f}p_f \leq \frac{H^*}{F}p_f \leq q$.
$\therefore$ Hypotheses with p-values not greater than p_f will be rejected.
$\because$ The p-values of the F false hypotheses are not greater than p_f.
$\therefore$ All the F false hypotheses will be rejected by the FDR procedure, Algorithm 2. ■

Proof of **Theorem 1**
If there is not any edge in the true DAG G_{true}, then the proof is trivially $E^{\sim}_{true} = \emptyset \subseteq E^{\sim}$. In the following part of the proof, we assume $E^{\sim}_{true} \neq \emptyset$. For the PC$_{\text{fdr}}$-skeleton algorithm and its heuristic modification, whenever the FDR procedure, Algorithm 2, is invoked, $p^{max}_{a\sim b}$ is always less than $\max_{C \in V\setminus\{a,b\}}\{p_{a\perp b|C}\}$, and the number of p-values input to the FDR algorithm is always not more than C^2_N. Thus, according to Lemma 2, if

$$\max_{a\sim b\in E^{\sim}_{true}}\left\{\max_{C\in V\setminus\{a,b\}}\{p_{a\perp b|C}\}\right\} \leq \frac{|E^{\sim}_{true}|}{C^2_N\sum_{i=1}^{C^2_N}\frac{1}{i}}q, \tag{5}$$

then all the true connections will be recovered by the PC$_{\text{fdr}}$-skeleton algorithm and its heuristic modification. Let $\mathcal{A}'_{a\perp b|C}$ denote the event

$$p_{a\perp b|C} \leq \frac{|E^{\sim}_{true}|}{C^2_N\sum_{i=1}^{C^2_N}\frac{1}{i}}q,$$

$\mathcal{A}'_{E^{\sim}_{true}}$ denote the event of Eq. (5), and $\mathcal{A}_{E^{\sim}_{true}}$ denote the event that all the true connections are recovered by the PC$_{\text{fdr}}$-skeleton algorithm and its heuristic modification.
$\because \mathcal{A}'_{E^{\sim}_{true}}$ is a sufficient condition for $\mathcal{A}_{E^{\sim}_{true}}$, according to Lemma 2.
$\therefore \mathcal{A}_{E^{\sim}_{true}} \supseteq \mathcal{A}'_{E^{\sim}_{true}}$.
$\therefore P(\mathcal{A}_{E^{\sim}_{true}}) \geq P(\mathcal{A}'_{E^{\sim}_{true}})$.
$\because \mathcal{A}'_{E^{\sim}_{true}}$ is the joint of a limited number of events as

$$\mathcal{A}'_{E^{\sim}_{true}} = \bigcap_{a\sim b\in E^{\sim}_{true}}\ \bigcap_{C\subseteq V\setminus\{a,b\}} \mathcal{A}'_{a\perp b|C},$$

and $\lim_{m\to\infty} P(\mathcal{A}'_{a\perp b|C}) = 1$ according to Assumption (A3).
$\therefore$ According to Corollary 1, $\lim_{m\to\infty} P(\mathcal{A}'_{E^{\sim}_{true}}) = 1$.
$\therefore 1 \geq \lim_{m\to\infty} P(\mathcal{A}_{E^{\sim}_{true}}) \geq \lim_{m\to\infty} P(\mathcal{A}'_{E^{\sim}_{true}}) = 1$.
$\therefore \lim_{m\to\infty} P(\mathcal{A}_{E^{\sim}_{true}}) = 1$. ■

Lemma 3 *Given any FDR level $q > 0$, if the p-value vector $P = [p_1, \cdots, p_H]$ input to Algorithm 2 is replaced with $P' = [p'_1, \cdots, p'_H]$, such that (1) for the those hypotheses that are rejected when P is the input, p'_i is equal to or less than p_i, and (2) for all the other hypotheses, p'_i can be any value between 0 and 1, then the set of rejected hypotheses when P' is the input is a superset of those rejected when P is the input.*

Proof
Let R and R' denote the sets of the rejected hypotheses when P and P' are respectively input to the FDR procedure.

If $R = \emptyset$, then the proof is trivially $R' \supseteq \emptyset = R$.

If $R \neq \emptyset$, let us define $\alpha = \max_{i \in R} p_i$ and $\alpha' = \max_{i \in R} p'_i$. Let $r = |R|$ denote the rank of α in the ascending order of P and r' denote the rank of α' in the ascending order of P'.
$\because p'_i \leq p_i$ for all $i \in R$.
$\therefore \alpha' \leq \alpha$.
$\because \alpha' = \max_{i \in R} p'_i$.
$\therefore r' \geq |R| = r$.
$\because \frac{H^*}{r}\alpha \leq q$.
$\therefore \frac{H^*}{r'}\alpha' \leq \frac{H^*}{r}\alpha \leq q$.
$\therefore$ When P' is the input, hypotheses with p'_i smaller than or equal to α' will be rejected.
$\because p'_i \leq \alpha', \forall i \in R$.
$\therefore R \subseteq R'$, equivalently $R' \supseteq R$. ■

Corollary 2 *Given any FDR level $q > 0$, if the p-value vector $P = [p_1, \cdots, p_H]$ input to Algorithm 2 is replaced with $P' = [p'_1, \cdots, p'_H]$ such that $p'_i \leq p_i$ for all $i = 1, \cdots, H$, then the set of rejected hypotheses when P' is the input is a superset of those rejected when P is the input.*

Proof of **Theorem 2**
Let $E^{\sim}_{stop}$ and $E^{\nsim}_{stop}$ denote the undirected edges respectively recovered and removed by the PC$_{\text{fdr}}$-skeleton algorithm when the algorithm stops. Let sequence $P^{max}_1, \cdots, P^{max}_K$ denote the values of P^{max} when the FDR procedure is invoked at step 12 as the algorithm progresses, in the order of the update process of P^{max}, and let $E^{\nsim}_k$ denote the set of removable edges indicated by the FDR procedure, with P^{max}_k as the input. $E^{\nsim}_k$ may include edges that have already been removed.
$\because$ The PC$_{\text{fdr}}$-skeleton algorithm accumulatively removes edges in $E^{\nsim}_k$.
$\therefore E^{\nsim}_{stop} = \bigcup_{k=1}^{K} E^{\nsim}_k$.
$\because P^{max}$ is updated increasingly at step 10 of the algorithm.
$\therefore$ According to **Corollary 2**, $E^{\nsim}_1 \subseteq \cdots \subseteq E^{\nsim}_K$.
$\therefore E^{\nsim}_{stop} = \bigcup_{k=1}^{K} E^{\nsim}_k = E^{\nsim}_K$.

Let $P = \{p_{a \sim b}\}$ denote the value of P^{max} when the PC$_{\text{fdr}}$-skeleton algorithm stops.
$\because$ The FDR procedure is invoked whenever P^{max} is updated.
$\therefore$ The value of P^{max} does not change after the FDR procedure is invoked for the last time.
$\therefore P = P^{max}_K$.
$\therefore E^{\sim}_{stop}$ is the same as the edges recovered by directly applying the FDR procedure to P.

The theorem is proved through comparing the result of the PC$_{\text{fdr}}$-skeleton algorithm with that of applying the FDR procedure to a virtual p-value set constructed from P. The virtual p-value set P^* is defined as follows.

For a vertex pair $a \sim b$ that is not adjacent in the true DAG G_{true}, let $C^*_{a \sim b}$ denote a certain vertex set that d-separates a and b in G_{true} and that is also a subset of either $\text{adj}(a, G^{\sim}_{true}) \setminus \{b\}$ or

$\text{adj}(b, G^{\sim}_{true}) \setminus \{a\}$. Let us define $P^* = \{p^*_{a\sim b}\}$ as:

$$p^*_{a\sim b} = \begin{cases} p_{a\perp b|C^*_{a\sim b}} & : \quad a\sim b \in E^{\not\sim}_{true}, \\ p_{a\sim b} & : \quad a\sim b \in E^{\sim}_{true}. \end{cases}$$

Though $p_{a\perp b|C^*_{a\sim b}}$ may not be actually calculated during the process of the algorithm, $p_{a\perp b|C^*_{a\sim b}}$ still can denote the value as if it was calculated. Let us design a virtual algorithm, called $Algorithm^*$, that recovers edges by just applying the FDR procedure to P^*, and let $E^{\sim *}$ denote the edges recovered by this virtual algorithm. This algorithm is virtual and impracticable because the calculation of P^* depends on the unknown $E^{\sim}_{true}$, but this algorithm exists because $E^{\sim}_{true}$ exists. For any vertex pair a and b that is not adjacent in G_{true}:
$\because$ X_a and X_b are conditional independent given $X_{C^*_{a\sim b}}$.
$\therefore$ $p_{a\perp b|C^*_{a\sim b}}$ follows the uniform distribution on [0, 1].
$\therefore$ The FDR of $Algorithm^*$ is under q.

When all the true edges are recovered by the PC$_{\text{fdr}}$-skeleton algorithm, that is, $E^{\sim}_{true} \subseteq E^{\sim}_{stop}$, the conditional independence between X_a and X_b given $X_{C^*_{a\sim b}}$ is tested for all the falsely recovered edges $a\sim b \in E^{\not\sim}_{true} \bigcap E^{\sim}_{stop}$, because for these edges, subsets of $\text{adj}(a, G_{true}) \setminus \{b\}$ and subsets of $\text{adj}(a, G_{true}) \setminus \{b\}$ have been exhaustively searched and $C^*_{a\sim b}$ is one of them. Therefore, $p_{a\sim b} \geq p^*_{a\sim b}$ for all $a\sim b \in E^{\sim}_{stop}$ when event $\mathcal{A}_{E^{\sim}_{true}}$ happens. Consequently, according to Lemma 3, if event $\mathcal{A}_{E^{\sim}_{true}}$ happens, $E^{\sim}_{stop} \subseteq E^{\sim *}$.

Let $q(E^{\sim})$ denote the realized FDR of reporting $E^{\sim}$ as the recovered skeleton of the true DAG:

$$q(E^{\sim}) = \begin{cases} \frac{|E^{\sim} \bigcap E^{\not\sim}_{true}|}{|E^{\sim}|} & : \quad E^{\sim} \neq \emptyset, \\ 0 & : \quad E^{\sim} = \emptyset. \end{cases}$$

The FDRs of the PC$_{\text{fdr}}$-skeleton algorithm and $Algorithm^*$ are $E[q(E^{\sim}_{stop})]$ and $E[q(E^{\sim *})]$ respectively. Here $E[x]$ means the expected value of x.

$\because$ $E[q(E^{\sim}_{stop})] = E[q(E^{\sim}_{stop})|\mathcal{A}_{E^{\sim}_{true}}]P(\mathcal{A}_{E^{\sim}_{true}}) + E[q(E^{\sim}_{stop})|\overline{\mathcal{A}}_{E^{\sim}_{true}}]P(\overline{\mathcal{A}}_{E^{\sim}_{true}})$
$\leq Q + P(\overline{\mathcal{A}}_{E^{\sim}_{true}})$, where $Q = E[q(E^{\sim}_{stop})|\mathcal{A}_{E^{\sim}_{true}}]P(\mathcal{A}_{E^{\sim}_{true}})$.
$\therefore$ $\limsup_{m\to\infty} E[q(E^{\sim}_{stop})] \leq \limsup_{m\to\infty} Q + \limsup_{m\to\infty} P(\overline{\mathcal{A}}_{E^{\sim}_{true}})$.
$\because$ $\lim_{m\to\infty} P(\mathcal{A}_{E^{\sim}_{true}}) = 1$, according to **Theorem 1**.
$\therefore$ $\limsup_{m\to\infty} P(\overline{\mathcal{A}}_{E^{\sim}_{true}}) = \lim_{m\to\infty} P(\overline{\mathcal{A}}_{E^{\sim}_{true}}) = 0$.
$\therefore$ $\limsup_{m\to\infty} E[q(E^{\sim}_{stop})] \leq \limsup_{m\to\infty} Q$.
$\because$ $Q \leq E[q(E^{\sim}_{stop})]$.
$\therefore$ $\limsup_{m\to\infty} Q \leq \limsup_{m\to\infty} E[q(E^{\sim}_{stop})]$.
$\therefore$ $\limsup_{m\to\infty} E[q(E^{\sim}_{stop})] = \limsup_{m\to\infty} Q = \limsup_{m\to\infty} E[q(E^{\sim}_{stop})|\mathcal{A}_{E^{\sim}_{true}}]P(\mathcal{A}_{E^{\sim}_{true}})$.

Similarly, $\limsup_{m\to\infty} E[q(E^{\sim *})] = \limsup_{m\to\infty} E[q(E^{\sim *})|\mathcal{A}_{E^{\sim}_{true}}]P(\mathcal{A}_{E^{\sim}_{true}})$.

$\because$ Given event $\mathcal{A}_{E^{\sim}_{true}}$, $E^{\sim}_{true} \subseteq E^{\sim}_{stop} \subseteq E^{\sim *}$.

$\therefore$ Given event $\mathcal{A}_{E^{\sim}_{true}}$,

$$q(E^{\sim}_{stop}) = \frac{|E^{\sim}_{stop}| - |E^{\sim}_{true}|}{|E^{\sim}_{stop}|} = 1 - \frac{|E^{\sim}_{true}|}{|E^{\sim}_{stop}|} \leq 1 - \frac{|E^{\sim}_{true}|}{|E^{\sim *}|} = \frac{|E^{\sim *}| - |E^{\sim}_{true}|}{|E^{\sim *}|} = q(E^{\sim *}).$$

$\therefore \limsup_{m\to\infty} E[q(E^{\sim}_{stop})|\mathcal{A}_{E^{\sim}_{true}}]P(\mathcal{A}_{E^{\sim}_{true}}) \leq \limsup_{m\to\infty} E[q(E^{\sim *})|\mathcal{A}_{E^{\sim}_{true}}]P(\mathcal{A}_{E^{\sim}_{true}})$.
$\therefore \limsup_{m\to\infty} E[q(E^{\sim}_{stop})] \leq \limsup_{m\to\infty} E[q(E^{\sim *})]$.
$\because$ $Algorithm^*$ controls the FDR under q.
$\therefore E[q(E^{\sim *})] \leq q$.
$\therefore \limsup_{m\to\infty} E[q(E^{\sim *})] \leq q$.
$\therefore \limsup_{m\to\infty} E[q(E^{\sim}_{stop})] \leq q$. ■

Appendix B. Statistical Tests with Asymptotic Power Equal to One

Assumption (A3) on the asymptotic power of detecting conditional dependence appears demanding, but actually the detection power of several standard statistical tests approaches one as the number of identically and independently sampled observations approaches infinity. Listed as follows are two statistical tests satisfying Assumption (A3) for Gaussian models or discrete models.

B.1 Fisher's z Transformation on Sample Partial-correlation-coefficients for Gaussian Models

In multivariate Gaussian models, X_a and X_b are conditional independent given X_C if and only if the partial-correlation-coefficient of X_a and X_b given X_C is zero (see Lauritzen, 1996, pages 129–130). The partial-correlation-coefficient ρ is defined as:

$$\begin{aligned} \rho &= \frac{\text{Cov}\,[Y_a,Y_b]}{\sqrt{\text{Var}[Y_a]\text{Var}[Y_b]}}, \\ Y_a &= X_a - < W_a, X_C >, \\ Y_b &= X_b - < W_b, X_C >, \\ W_a &= \arg\min_w E[(X_a - < w, X_C >)^2], \\ W_b &= \arg\min_w E[(X_b - < w, X_C >)^2]. \end{aligned}$$

The sample partial-correlation-coefficient $\hat{\rho}$ can be calculated from m i.i.d. samples $[x_{ai}, x_{bi}, x_{Ci}]$ $(i = 1, \cdots, m)$ as:

$$\begin{aligned}
\hat{\rho} &= \frac{\frac{1}{m}\sum_{i=1}^{m}[(\hat{y}_{ai}-\bar{y}_a)(\hat{y}_{bi}-\bar{y}_b)]}{\sqrt{\frac{1}{m}\sum_{i=1}^{m}(\hat{y}_{ai}-\bar{y}_a)^2\frac{1}{m}\sum_{i=1}^{m}(\hat{y}_{bi}-\bar{y}_b)^2}},\\
\bar{y}_a &= \frac{1}{m}\sum_{i=1}^{m}\hat{y}_{ai},\\
\bar{y}_b &= \frac{1}{m}\sum_{i=1}^{m}\hat{y}_{bi},\\
\hat{y}_{ai} &= x_{ai}- <\hat{W}_a, x_{Ci}>,\\
\hat{y}_{bi} &= x_{bi}- <\hat{W}_b, x_{Ci}>,\\
\hat{W}_a &= \arg\min_w \sum_{i=1}^{m}(x_{ai}- <w, x_{Ci}>)^2,\\
\hat{W}_b &= \arg\min_w \sum_{i=1}^{m}(x_{bi}- <w, x_{Ci}>)^2.
\end{aligned}$$

The asymptotic distribution of $z(\hat{\rho})$, where $z(x)$, the Fisher's z transformation (see Fisher, 1915), is defined as

$$z(x) = \frac{1}{2}\log\frac{1+x}{1-x},$$

is the normal distribution with mean $z(\rho)$ and variance $1/(m-|C|-3)$ (see Anderson, 1984, pages 120–134). When the type I error rate is kept lower than α, the power of detecting $\rho \neq 0$ with Fisher's z transformation is the probability that $\sqrt{m-|C|-3}\ z(\hat{\rho})$ falls in the range $(-\infty, \Phi^{-1}(\alpha/2)]$ or $[\Phi^{-1}(1-\alpha/2), +\infty)$, where Φ is the cumulative distribution function of the standard normal distribution and Φ^{-1} is its inverse function. Without loss of generality, we assume the true partial-correlation-coefficient ρ is greater than zero, then the asymptotic power is

$$\begin{aligned}
&\lim_{m\to\infty}\text{Power} \geq \lim_{m\to\infty} P\left(\sqrt{m-|C|-3}\ z(\hat{\rho}) \geq \Phi^{-1}(1-\alpha/2)\right)\\
&= \lim_{m\to\infty}\left(1-\Phi[\Phi^{-1}(1-\alpha/2)-\sqrt{m-|C|-3}\ z(\rho)]\right) = (1-\Phi[-\infty]) = 1.
\end{aligned}$$

B.2 The Likelihood-ratio Test Generally Applicable to Nested Models

The likelihood ratio is the ratio of the maximum likelihood of a restricted model to that of a saturated model (see Neyman and Pearson, 1928). Let $f(x, \theta)$ denote the probability density function of a random vector x parametrized with $\theta = [\theta^1, \cdots, \theta^k]$. The null hypothesis restricts θ to a set Ω specified with r $(r \leq k)$ constraints

$$\xi_1(\theta) = \xi_2(\theta) = \cdots = \xi_r(\theta) = 0.$$

Given i.i.d. observations $x_1, \cdots, x_m$, let $L(\theta)$ denote the likelihood function

$$L(\theta) = \prod_{i=1}^{m} f(x_i, \theta).$$

The likelihood ratio Λ given the observations is defined as

$$\Lambda = \frac{\sup L(\theta)}{\sup\limits_{\theta\in\Omega} L(\theta)}.$$

Wald (1943) has proved that under certain assumptions on $f(x,\theta)$ and $\xi_1(\theta),\cdots,\xi_r(\theta)$, the limit distribution of the statistic $-2log\Lambda$ is the χ_r^2 distribution with r degrees of freedom if the null hypothesis true. If the null hypothesis is not true, the distribution of $-2log\Lambda$ approaches the non-central $\chi_r^2(\lambda)$ distribution with r degrees of freedom and the non-central parameter

$$\begin{aligned}\lambda &= mD(\theta) \geq 0,\\ D(\theta) &= \sum_{i=1}^{k}\sum_{j=1}^{k} \frac{\xi_i(\theta)\xi_j(\theta)}{\sum\limits_{p=1}^{k}\sum\limits_{q=1}^{k} \frac{\frac{\partial \xi_i}{\partial \theta^p}\frac{\partial \xi_j}{\partial \theta^p}}{E\left[-\frac{\partial^2 f(x,\theta)}{\partial \theta^p \partial \theta^q}\right]}}.\end{aligned}$$

If $D(\theta) > 0$, then $\lim\limits_{m\to\infty} \lambda = \infty$. Let t ($t < \infty$) denote the threshold of rejecting the null hypothesis with type I error rate under α ($\alpha > 0$). The asymptotic power of detecting a θ that is not in Ω and whose $D(\theta)$ is greater than 0 is $\lim\limits_{\lambda\to\infty} P(\chi_r^2(\lambda) > t)$. The mean and the variance of the $\chi_r^2(\lambda)$ distribution is $u = r+\lambda$ and $\sigma^2 = 2(r+2\lambda)$, respectively. When λ is large enough,

$$P\left(\chi_r^2(\lambda) > t\right) \geq P\left(t < \chi_r^2(\lambda) < u+(u-t)\right) = 1 - P\left(|\chi_r^2(\lambda) - u| \geq u-t\right).$$

According to Chebyshev's inequality,

$$P\left(|\chi_r^2(\lambda) - u| \geq u-t\right) \leq \frac{\sigma^2}{(u-t)^2} = \frac{2(r+2\lambda)}{(r+\lambda-t)^2}.$$

$\therefore$ When λ is large enough, $P(\chi_r^2(\lambda) > t) \geq 1 - \frac{2(r+2\lambda)}{(r+\lambda-t)^2}$.
$\because \lim\limits_{m\to\infty} \lambda = \infty$ and both r and t are fixed.
$\therefore \lim\limits_{m\to\infty} \frac{2(r+2\lambda)}{(r+\lambda-t)^2} = 0$.
$\therefore \lim\limits_{m\to\infty} P(\chi_r^2(\lambda) > t) = 1$.

Appendix C. Error Rates of Interest

Statistical decision processes usually involve choices between negative hypotheses and their alternatives, positive hypotheses. In the decision, there are basically two sources of errors: the type I errors, that is, falsely rejecting negative hypotheses when they are actually true; and the type II errors, that is, falsely accepting negative hypotheses when their alternatives, the positive hypotheses are actually true. In the context of learning graph structures, a negative hypothesis could be that an edge does not exist in the graph, while the positive hypothesis could be that the edge does exist. It is generally impossible to absolutely prevent the two types of errors simultaneously, because observations of a limited sample size may appear to support a positive hypothesis more than a negative hypothesis even when actually the negative hypothesis is true, or vice versa, due to the stochastic nature of random sampling. Moreover, the two types of errors generally contradict each other. Given a fixed sample size and a certain statistic extracted from the data, decreasing the type I errors will increase the type II errors, and vice versa. To guarantee the absolute prevention of the type I errors in any situations, one must accept all negative hypotheses, which will generally lead the type II error rate to be one, and vice versa. The contradiction between the two types of errors is clearly revealed by the monotone increase of receiver operating characteristic (ROC) curves. Thus the errors must be controlled by setting a threshold on a certain type of errors, or trading off between them, for instance, by minimizing a certain lost function associated with the errors according to the Bayesian decision theory.

Rooted in the two types of errors, there are several different error-rate criteria (as listed in Table 2) for problems involving simultaneously testing multiple hypotheses, such as verifying the existence of edges in a graph. The type I error rate is the expected ratio of the type I errors to all the negative hypotheses that are actually true; the type II error rate is the expected ratio of the type II errors to all the positive hypotheses that are actually true; the false discovery rate (FDR) (see Benjamini and Yekutieli, 2001; Storey, 2002), is the expected ratio of falsely accepted positive hypotheses to all those accepted positive hypotheses; the family-wise error rate is the probability that at least one of the accepted positive hypotheses is actually wrong.

Generally, there are no mathematically or technically superior relationships among these error-rate criteria. Each of these error rates may be favoured in certain research scenarios. For example:

- We are diagnosing a dangerous disease whose treatment is so risky that may cause the loss of eyesight. Due to the great risk of the treatment, we hope that less than 0.1% of healthy people will be falsely diagnosed as patients of the disease. In this case, the type I error rate should be controlled under 0.1%.

- We are diagnosing cancer patients. Because failure in detecting the disease will miss the potential chance to save the patient's life, we hope that 95% of the cancer patients will be correctly detected. In this case, the type II error rate should be controlled under 5%.

- In a pilot study, we are selecting candidate genes for a genetic research on Parkinson's disease. Because of the limited funding, we can only study a limited number of genes in the afterward genetic research, so when selecting candidate genes in the pilot study, we hope that 95% of the selected candidate genes are truly associated with the disease. In this case, the FDR will be chosen as the error rate of interest and should be controlled under 5%.

- We are selecting electronic components to make a device. Any error in any component will cause the device to run out of order. To guarantee the device functions well with a probability higher than 99%, the family-wise error rate should be controlled under 1%.

In these examples, the particular error-rate criteria are selected by reasons beyond mathematical or technical superiority, but by the researchers' interest, to minimize a certain lost function associated with the errors according to the Bayesian decision theory. Learning network structures in real world applications may face scenarios similar to the above examples.

The excellent discrimination between negative hypotheses and positive hypotheses cannot be achieved by "smartly" setting a threshold on a "superior" error-rate criterion. Setting a threshold on a certain type of error rate is just choosing a cut-point on the ROC curve. If the ROC curve is not sharp enough, any cut-point on the curve away from the ends (0,0) and (1,1) still leads to considerable errors. To discriminate more accurately between a negative hypothesis and a positive hypothesis, one must design a better statistic or increase the sample size to achieve a sharper ROC curve.

References

A. Agresti. *Categorical Data Analysis (2nd edition)*. John Wiley & Sons, Inc., 2002.

T. W. Anderson. *An Introduction to Multivariate Statistical Analysis (2nd edition)*. John Wiley & Sons, Inc., 1984.

S. A. Andersson, D. Madigan, and M. D. Perlman. A characterization of Markov equivalence classes for acyclic digraphs. *The Annals of Statistics*, 25(2):505–541, 1997.

Y. Benjamini and D. Yekutieli. The control of the false discovery rate in multiple testing under dependency. *The Annals of Statistics*, 29(4):1165–1188, 2001.

D. M. Chickering. Optimal structure identification with greedy search. *Journal of Machine Learning Research*, 3:507–554, 2002.

C. Chow and C. Liu. Approximating discrete probability distributions with dependence trees. *Information Theory, IEEE Transactions on*, 14(3):462–467, 1968.

G. F. Cooper and E. Herskovits. A Bayesian method for the induction of probabilistic networks from data. *Machine Learning*, 9(4):309–347, 1992.

D. Eaton and K. Murphy. Bayesian structure learning using dynamic programming and MCMC. In *Proceedings of the Twenty-Third Conference on Uncertainty in Artificial Intelligence*, 2007.

R. A. Fisher. Frequency distribution of the values of the correlation coefficients in samples from an indefinitely large population. *Biometrika*, 10(4):507–521, 1915.

R. A. Fisher. The distribution of the partial correlation coefficient. *Metron*, 3:329–332, 1924.

N. Friedman and D. Koller. Being Bayesian about network structure. A Bayesian approach to structure discovery in Bayesian networks. *Machine Learning*, 50(1):95–125, 2003.

D. Heckerman, D. Geiger, and D. M. Chickering. Learning Bayesian networks: The combination of knowledge and statistical data. *Machine Learning*, 20(3):197–243, 1995.

E. H. Herskovits and G. F. Cooper. Kutato: An entropy-driven system for the construction of probabilistic expert systems from databases. In *Proceedings of the Sixth Conference on Uncertainty in Artificial Intelligence*, pages 54–62, 1990.

M. Kalisch and P. Bühlmann. Estimating high-dimensional directed acyclic graphs with the PC-algorithm. *Journal of Machine Learning Research*, 8:613–636, 2007.

T. A. Keller, M. A. Just, and V. A. Stenger. Reading span and the time-course of cortical activation in sentence-picture verification. In *Annual Convention of the Psychonomic Society*, 2001.

M. Koivisto and K. Sood. Exact Bayesian structure discovery in Bayesian networks. *Journal of Machine Learning Research*, 5:549–573, 2004.

S. L. Lauritzen. *Graphical Models*. Clarendon Press, Oxford University Press, Oxford, New York, 1996.

J. Li, Z. J. Wang, S. J. Palmer, and M. J. McKeown. Dynamic Bayesian network modelling of fMRI: A comparison of group analysis methods. *NeuroImage*, 41:398–407, 2008.

D. Madigan, J. York, and D. Allard. Bayesian graphical models for discrete data. *International Statistical Review*, 63(2):215–232, 1995.

T. M. Mitchell, R. Hutchinson, R. S. Niculescu, F. Pereira, X. Wang, M. Just, and S. Newman. Learning to decode cognitive states from brain images. *Machine Learning*, 57(1):145–175, 2004.

J. Neyman and E. S. Pearson. On the use and interpretation of certain test criteria for purposes of statistical inference: Part I. *Biometrika*, 20A:175–240, 1928.

J. Pearl. *Probabilistic Reasoning in Intelligent Systems*. Morgan Kaufmann, 1988.

J. Pearl. *Causality*. Cambridge University Press, 2000.

J. Pearl and T. S. Verma. A statistical semantics for causation. *Statistics and Computing*, 2(2): 91–95, 1992.

H. Qian and S. Huang. Comparison of false discovery rate methods in identifying genes with differential expression. *Genomics*, 86(4):495–503, 2005.

R. W. Robinson. Counting labeled acyclic digraphs. In *New Directions in the Theory of Graphs*, pages 239–273. Academic Press, 1973.

J. Schäfer and K. Strimmer. An empirical Bayes approach to inferring large-scale gene association networks. *Bioinformatics*, 21(6):754–764, 2005.

P. Spirtes and C. Glymour. An algorithm for fast recovery of sparse causal graphs. *Social Science Computer Review*, 9:62–72, 1991.

P. Spirtes, C. Glymour, and R. Scheines. *Causation, Prediction, and Search*. The MIT Press, 2001.

B. Steinsky. Enumeration of labelled chain graphs and labelled essential directed acyclic graphs. *Discrete Mathematics*, 270(1-3):267–278, 2003.

J. D. Storey. A direct approach to false discovery rates. *Journal of the Royal Statistical Society. Series B (Statistical Methodology)*, 64(3):479–498, 2002.

J. D. Storey. The positive false discovery rate: A Bayesian interpretation and the q-value. *The Annals of Statistics*, 31(6):2013–2035, 2003.

A. Wald. Tests of statistical hypotheses concerning several parameters when the number of observations is large. *Transactions of the American Mathematical Society*, 54(3):426–482, 1943.

Journal of Machine Learning Research 10 (2009) 515-554 Submitted 12/07; Revised 8/08; Published 2/09

Identification of Recurrent Neural Networks by Bayesian Interrogation Techniques

Barnabás Póczos[*] POCZOS@CS.UALBERTA.CA
András Lőrincz ANDRAS.LORINCZ@ELTE.HU
Department of Information Systems, Eötvös Loránd University
Pázmány P. sétány 1/C, Budapest H-1117, Hungary

Editor: Zoubin Ghahramani

Abstract

We introduce novel online Bayesian methods for the identification of a family of noisy recurrent neural networks (RNNs). We present Bayesian active learning techniques for stimulus selection given past experiences. In particular, we consider the unknown parameters as stochastic variables and use A-optimality and D-optimality principles to choose optimal stimuli. We derive myopic cost functions in order to maximize the information gain concerning network parameters at each time step. We also derive the A-optimal and D-optimal estimations of the additive noise that perturbs the dynamical system of the RNN. Here we investigate myopic as well as non-myopic estimations, and study the problem of simultaneous estimation of both the system parameters and the noise. Employing conjugate priors our derivations remain approximation-free and give rise to simple update rules for the online learning of the parameters. The efficiency of our method is demonstrated for a number of selected cases, including the task of controlled independent component analysis.

Keywords: active learning, system identification, online Bayesian learning, A-optimality, D-optimality, infomax control, optimal design

1. Introduction

When studying systems in *interactive* and *online* fashion, it is of high relevance to facilitate fast information gain during the interaction (Fedorov, 1972; Cohn, 1994). As an example, consider experiments aiming at the description of the receptive field of different neurons. These experiments look for those stimuli that maximize the response of the given neuron (deCharms et al., 1998; Földiák, 2001). Neurons, however, might change due to the investigation, so the minimization of interaction is highly desired. Different techniques have been developed to speed up the identification procedure. One approach searches for stimulus distribution that maximizes mutual information between stimulus and response (Machens et al., 2005). A recent technique assumes that the unknown system belongs to the family of generalized linear models (Lewi et al., 2007) and treats the parameters as probabilistic variables. Then the goal is to find the optimal stimuli by maximizing mutual information between the parameter set and the system's response.

This example motivates our interest in active learning (MacKay, 1992; Cohn et al., 1996; Fukumizu, 1996; Sugiyama, 2006) of noisy recurrent artificial neural networks (RNNs), when we have the freedom to interrogate the network and to measure its responses.

[*]. Present address: Department of Computing Science, University of Alberta, Athabasca Hall, Edmonton, Canada, T6G 2E8

In active learning, the training set may be modified by the learning process itself based on the progress experienced so far. The goal of this modification is to maximize the expected improvement of the precision of the estimation. This idea can for example be used to improve generalization capability in regression and classification tasks or to better estimate hidden parameters. Theoretical concepts have been formulated in the fields of Optimal Experimental Design, or Optimal Bayesian Design (Kiefer, 1959; Fedorov, 1972; Steinberg and Hunter, 1984; Toman and Gastwirth, 1993; Pukelsheim, 1993).

Although active learning is in the focus of current research interest, some relevant theoretical issues are still unresolved. While there are promising studies showing that active learning may outperform uniform sampling under certain conditions (Freund et al., 1997; Seung et al., 1992), in other cases it has been proven that active learning has no advantage over non-adaptive algorithms. For example, this is the case in compressed sensing (Castro et al., 2006a) and also for certain function classes in the area of function approximation (Castro et al., 2006b). Even more problematic is the observation that active learning heuristics may be less efficient than uniform sampling in some situations (Schein, 2005).

There are several forms of active learning. The most relevant difference is in the definition of the value of information. One of the simplest heuristics is the Uncertainty Sampling (US): US suggests that in regression or in classification tasks one should choose those training examples, which have the largest uncertainty in the value of the function or in the label of the class, respectively (Lewis and Catlett, 1994; Lewis and Gale, 1994; Cohn et al., 1996). Although several US versions exist with different measure of the uncertainty itself, they all lack robustness. The Query by Committee method improves upon robustness (Seung et al., 1992; Freund et al., 1997): the committee of a few models are trained on the existing training set and the next query points are selected to reduce the disagreement among these models. The method of Roy and McCallum (2001) minimizes the direct error, that is, it tries to choose training points to minimize the expected classification error directly.

In the literature there are other approaches, including decision theory based methods. The original ideas were worked out in Raiffa and Schlaifer (1961) and Lindley (1971). The objective in this method family is to choose the design such that the predicted value of a given utility function become maximal. Numerous utility functions have been proposed. For example, if we aim to estimate the unknown parameter $\boldsymbol{\theta}$, then one possible direction is the minimization of, for example, the entropy or the standard deviation of the posterior distribution. If we minimize the entropy then we arrive at the D-optimality principle (Bernardo, 1979; Stone, 1959). This principle is equivalent to the information maximization method (also known as infomax principle) of Lewi et al. (2007). If we intend to minimize the standard deviation then the result is the A-optimality principle (Duncan and DeGroot, 1976). A special case is called the c-optimality principle (Chaloner, 1984) when the goal is to estimate a linear projection of parameter $\boldsymbol{\theta}$ ($\mathbf{c}^T\boldsymbol{\theta}$). There exist a number of other methods, called alphabetical optimality and utility functions. For a review see, for example, Chaloner and Verdinelli (1995). Although the original ideas belong to the field of optimal experimental design, they have appeared also in active learning recently (MacKay, 1992; Tong and Koller, 2000; Schein and Ungar, 2007).

Today, active learning is present almost in all fields of machine learning and there are many popular applications on diverse areas, including Gaussian Processes (Krause and Guestrin, 2007), Artificial Neural Networks (Fukumizu, 2000), Support Vector Machines (Tong and Koller, 2001b), Generalized Linear Models (Bach, 2007; Lewi et al., 2007), Logistic Regression (Schein, 2005), learning the parameters and structure of Bayes nets (Tong and Koller, 2000, 2001a) and Hidden Markov Models (Anderson and Moore, 2005).

Our framework is similar to the generalized linear model (GLM) approach used by Lewi et al. (2007): we would like to choose interrogating, or '*control*' inputs in order to (i) identify the parameters of the network and (ii) estimate the additive noise efficiently. From now on, we use the terms *control* and *interrogation* interchangeably; control is the conventional expression, whereas the word interrogation expresses our aims better. We apply online Bayesian learning (Opper and Winther, 1999; Solla and Winther, 1999; Honkela and Valpola, 2003; Ghahramani, 2000). For Bayesian methods, prior updates often lead to intractable posterior distributions such as a mixture of exponentially numerous distributions. Here, we show that, for the model studied in this paper, computations are both tractable and approximation-free. Further, the emerging learning rules are simple. We also show that different stimuli are needed for the same RNN model depending on whether the goal is to estimate the weights of the RNN or the additive perturbation (referred to as 'driving noise').

In this article we investigate the D-optimality, as well as the A-optimality principles. To the best of our knowledge, neither of them has been applied to the typical non-spiking stochastic artificial recurrent neural network model that we treat here.

The contribution of this paper can be summarized as follows: We use A-optimality and D-optimality principles and derive cost functions and algorithms for (i) the learning of parameters of the stochastic RNN and (ii) the estimation of its driving noise. We show that, (iii) using the D-optimality interrogation technique, these two tasks are incoherent in the myopic (i.e., single step look-ahead) control scheme: signals derived from this principle for parameter estimation are sub-optimal (basically the worst possible) for the estimation of the driving noise and vice versa. (iv) We show that for the case of noise estimation task the two principles, that is, A- and D-optimality principles result in the same cost function. (v) For the A-optimality case, we derive equations for the joined estimation of the noise and the parameters. On the contrary, we show also that (vi) D-optimality cannot be applied on the same joined task. For the case of noise estimation, (vii) a non-myopic multiple step look-ahead heuristics is introduced and we demonstrate its applicability through numerical experiments.

The paper is structured as follows: In Section 2 we introduce our model. Section 3 concerns the Bayesian equations of the RNN model. In Section 4 optimal control for parameter identification is derived from the D-optimality principle. Section 5 is about the same task, but using the A-optimality principle instead. Section 6 deals with our second task, when the goal is the estimation of the driving noise of the RNN. Here we treat the D-optimality principle. Section 7 is about the same problem, but for the A-optimality principle. We combine the two tasks for both optimality principles in Section 8 and consider the cost functions for the joined estimation of the parameters and the driving noise. All of these considerations concern myopic algorithms. In Section 9 a non-myopic heuristics is introduced for the noise estimation task. Section 10 contains our numerical experiments for a number of cases, including independent component analysis. The paper ends with a short discussion and some conclusions (Section 11). Technical details of the derivations can be found in the Appendix.

2. The Model

Let $P(\mathbf{e}) = \mathcal{N}_{\mathbf{e}}(\mathbf{m},\mathbf{V})$ denote the probability density of a normally distributed stochastic variable $\mathbf{e}$ with mean $\mathbf{m}$ and covariance matrix $\mathbf{V}$. Let us assume that we have d simple computational units called '*neurons*' in a recurrent neural network:

$$\mathbf{r}_{t+1} \quad = \quad g\left(\sum_{i=0}^{I}\mathbf{F}_i\mathbf{r}_{t-i} + \sum_{j=0}^{J}\mathbf{B}_j\mathbf{u}_{t+1-j} + \mathbf{e}_{t+1}\right), \tag{1}$$

where $\{\mathbf{e}_t\}$, the driving noise of the RNN, denotes temporally independent and identically distributed (i.i.d.) stochastic variables and $P(\mathbf{e}_t) = \mathcal{N}_{\mathbf{e}_t}(\mathbf{0},\mathbf{V})$, $\mathbf{r}_t \in \mathbb{R}^d$ represents the observed activities of the neurons at time t. Let $\mathbf{u}_t \in \mathbb{R}^c$ denote the control signal at time t. The neural network is formed by the weighted delays represented by matrices $\mathbf{F}_i$ $(i = 0,\ldots,I)$ and $\mathbf{B}_j$ $(j = 0,\ldots,J)$, which connect neurons to each other and also the control components to the neurons, respectively. Control can also be seen as the means of interrogation, or the stimulus to the network (Lewi et al., 2007). We assume that function $g : \mathbb{R}^d \to \mathbb{R}^d$ in (1) is known and invertible. The computational units, the neurons, sum up weighted previous neural activities as well as weighted control inputs. These sums are then passed through identical non-linearities according to Eq. (1). Our goal is to estimate the parameters $\mathbf{F}_i \in \mathbb{R}^{d\times d}$ $(i = 0,\ldots,I)$, $\mathbf{B}_j \in \mathbb{R}^{d\times c}$ $(j = 0,\ldots,J)$ and the covariance matrix $\mathbf{V}$, as well as the driving noise $\mathbf{e}_t$ by means of the control signals.

In artificial neural network terms, (1) is in the form of *rate code models*. This is the typical form for RNNs, but there are methods to approximate rate code description with spike codes and vice versa. For the case of RNNs, the best is to compare Liquid State Machine, a spike code model of Maass et al. (2002) with the Echo State Network, the corresponding rate code model of Jaeger (2001). Rate code, very crudely, is the low pass filtered spike code, whereas spike code can be seen as the response of integrate-and-fire neurons. We show that analytic cost functions emerge for the rate code RNN model. Due to the applied conjugate priors, we can calculate the high dimensional integrals involved in our derivations, and hence these derivations remain approximation-free and give rise to simple update rules.

3. Bayesian Approach

Here we embed the estimation task into the Bayesian framework. First, we introduce the following notations: $\mathbf{x}_{t+1} = [\mathbf{r}_{t-I};\ldots;\mathbf{r}_t;\mathbf{u}_{t-J+1};\ldots;\mathbf{u}_{t+1}]$, $\mathbf{y}_{t+1} = g^{-1}(\mathbf{r}_{t+1})$, $\mathbf{A} = [\mathbf{F}_I,\ldots,\mathbf{F}_0,\mathbf{B}_J,\ldots,\mathbf{B}_0] \in \mathbb{R}^{d\times m}$. With these notations, model (1) reduces to a linear equation

$$\mathbf{y}_t \quad = \quad \mathbf{A}\mathbf{x}_t + \mathbf{e}_t. \tag{2}$$

In order to estimate the unknown quantities (parameter matrix $\mathbf{A}$, noise $\mathbf{e}_t$ and its covariance matrix $\mathbf{V}$) in an online fashion, we rely on Bayes' method. We assume that prior knowledge is available and we update our posteriori knowledge on the basis of the observations. Control will be chosen at each instant to provide maximal expected information concerning the quantities we have to estimate. Starting from an arbitrary prior distribution of the parameters the posterior distribution needs to be computed. This latter distribution, however, can be highly complex, so approximations are applied. For example, assumed density filtering, when the computed posterior is projected to simpler distributions, has been suggested (Boyen and Koller, 1998; Minka, 2001; Opper and Winther, 1999). In order to avoid approximations, we apply the method of conjugated priors (Gelman et al., 2003). For matrix **A** we assume a matrix valued normal distribution prior.

For the case of D-optimality principle, we shall use the inverted Wishart (IW) distribution as our prior for covariance matrix **V**. This is the most general known conjugate prior distribution for the covariance matrix of a normal distribution at present. For A-optimality, however, we keep the derivations simple and assume that the covariance matrix has diagonal structure. In turn, we replaced the IW assumption on the prior with the distribution of the Product of Inverted Gammas (PIG).

We define the normally distributed matrix valued stochastic variable $\mathbf{A} \in \mathbb{R}^{d\times m}$ by using the following quantities: $\mathbf{M} \in \mathbb{R}^{d\times m}$ is the expected value of **A**. $\mathbf{V} \in \mathbb{R}^{d\times d}$ is the covariance matrix of the rows, and $\mathbf{K} \in \mathbb{R}^{m\times m}$ is the so-called precision parameter matrix that we shall modify in accordance with the Bayesian update. Matrix **K** contains the estimations of the 'Bayesian trainer' about the precision of parameters in **A**. Informally, matrix **K** behaves as the inverse of a covariance matrix. Upon each observation, matrix **K** is updated. The larger the eigenvalues of this matrix, the smaller the variance ellipsoids of the posteriori estimations are.

Both **K** and **V** are positive semi-definite matrices. The density function of the stochastic variable **A** is defined as:

$$\mathcal{N}_{\mathbf{A}}(\mathbf{M},\mathbf{V},\mathbf{K}) = \frac{|\mathbf{K}|^{d/2}}{|2\pi\mathbf{V}|^{m/2}}\exp(-\frac{1}{2}tr((\mathbf{A}-\mathbf{M})^T\mathbf{V}^{-1}(\mathbf{A}-\mathbf{M})\mathbf{K})),$$

where tr, $|\cdot|$, and superscript T denote the trace operation, the determinant, and transposition, respectively (see, e.g., Gupta and Nagar, 1999; Minka, 2000). We assume that $\mathbf{Q} \in \mathbb{R}^{d\times d}$ is a positive definite matrix and $n > 0$. Using these notations, the density of the Inverted Wishart distribution with parameters **Q** and n is as follows (Gupta and Nagar, 1999):

$$\mathcal{IW}_{\mathbf{V}}(\mathbf{Q},n) = \frac{1}{Z_{n,d}}\frac{1}{|\mathbf{V}|^{(d+1)/2}}\left|\frac{\mathbf{V}^{-1}\mathbf{Q}}{2}\right|^{n/2}\exp(-\frac{1}{2}tr(\mathbf{V}^{-1}\mathbf{Q})),$$

where $Z_{n,d} = \pi^{d(d-1)/4}\prod_{i=1}^{d}\Gamma((n+1-i)/2)$ and $\Gamma(.)$ denotes the gamma function.

Similarly, let $\mathbf{V} = diag(\mathbf{v}) \in \mathbb{R}^{d\times d}$ diagonal covariance matrix with $0 < \mathbf{v} \in \mathbb{R}^d$ diagonal values. With the slight abuse of notation we will use later the $\mathbf{v} = diag(\mathbf{V}) \in \mathbb{R}^d$ term, too. Then the density of PIG is defined as

$$\mathcal{PIG}_{\mathbf{V}}(\boldsymbol{\alpha},\boldsymbol{\beta}) = \prod_{i=1}^{d}\frac{\beta_i^{\alpha_i}}{\Gamma(\alpha_i)}v_i^{-\alpha_i-1}\exp(-\frac{\beta_i}{v_i}),$$

where $\alpha_i > 0$ and $\beta_i > 0$ are the shape and scale parameters respectively.

Now, one can rewrite model (2) as follows:

$$\begin{aligned} P(\mathbf{A}|\mathbf{V}) &= \mathcal{N}_{\mathbf{A}}(\mathbf{M},\mathbf{V},\mathbf{K}), &(3)\\ P(\mathbf{e}_t|\mathbf{V}) &= \mathcal{N}_{\mathbf{e}_t}(\mathbf{0},\mathbf{V}), &(4)\\ P(\mathbf{y}_t|\mathbf{A},\mathbf{x}_t,\mathbf{V}) &= \mathcal{N}_{\mathbf{y}_t}(\mathbf{A}\mathbf{x}_t,\mathbf{V}), &(5)\end{aligned}$$

and $P(\mathbf{V}) = \mathcal{PIG}_{\mathbf{V}}(\boldsymbol{\alpha},\boldsymbol{\beta})$ or $P(\mathbf{V}) = \mathcal{IW}_{\mathbf{V}}(\mathbf{Q},n)$ depending on whether we want to use A- or D-optimality.

4. D-Optimality Approach for Parameter Learning

Let us compute the D-optimal parameter estimation strategy for our RNN given by (1) and rewritten into (3)-(5). Let us introduce two shorthands; $\boldsymbol{\theta} = \{\mathbf{A}, \mathbf{V}\}$, and $\{\mathbf{x}\}_i^j = \{\mathbf{x}_i, \dots, \mathbf{x}_j\}$. We choose the control value in (1) at each instant to provide maximal expected information concerning the unknown parameters. Assuming that $\{\mathbf{x}\}_1^t$, $\{\mathbf{y}\}_1^t$ are given, according to the infomax principle our goal is to compute

$$\arg\max_{\mathbf{u}_{t+1}} I(\boldsymbol{\theta}, \mathbf{y}_{t+1}; \{\mathbf{x}\}_1^{t+1}, \{\mathbf{y}\}_1^t), \tag{6}$$

where $I(a,b;c)$ denotes the mutual information of stochastic variables a and b for fixed parameters c. Let $H(a|b;c)$ denote the conditional entropy of variable a conditioned on variable b and for fixed parameter c. Note that

$$I(\boldsymbol{\theta}, \mathbf{y}_{t+1}; \{\mathbf{x}\}_1^{t+1}, \{\mathbf{y}\}_1^t) = H(\boldsymbol{\theta}; \{\mathbf{x}\}_1^{t+1}, \{\mathbf{y}\}_1^t) - H(\boldsymbol{\theta}|\mathbf{y}_{t+1}; \{\mathbf{x}\}_1^{t+1}, \{\mathbf{y}\}_1^t),$$

holds (Cover and Thomas, 1991) and $H(\boldsymbol{\theta}; \{\mathbf{x}\}_1^{t+1}, \{\mathbf{y}\}_1^t) = H(\boldsymbol{\theta}; \{\mathbf{x}\}_1^t, \{\mathbf{y}\}_1^t)$ is independent from $\mathbf{u}_{t+1}$, hence our task is reduced to the evaluation of the following quantity:

$$\begin{aligned} &\arg\min_{\mathbf{u}_{t+1}} H(\boldsymbol{\theta}|\mathbf{y}_{t+1}; \{\mathbf{x}\}_1^{t+1}, \{\mathbf{y}\}_1^t) = \\ &= \arg\min_{\mathbf{u}_{t+1}} - \int d\mathbf{y}_{t+1} P(\mathbf{y}_{t+1}|\{\mathbf{x}\}_1^{t+1}, \{\mathbf{y}\}_1^t) \int d\boldsymbol{\theta} P(\boldsymbol{\theta}|\{\mathbf{x}\}_1^{t+1}, \{\mathbf{y}\}_1^{t+1}) \log P(\boldsymbol{\theta}|\{\mathbf{x}\}_1^{t+1}, \{\mathbf{y}\}_1^{t+1}). \end{aligned} \tag{7}$$

In order to solve this minimization problem we need to evaluate $P(\mathbf{y}_{t+1}|\{\mathbf{x}\}_1^{t+1}, \{\mathbf{y}\}_1^t)$, the posterior $P(\boldsymbol{\theta}|\{\mathbf{x}\}_1^{t+1}, \{\mathbf{y}\}_1^{t+1})$, and the entropy of the posterior, that is $\int d\boldsymbol{\theta} P(\boldsymbol{\theta}|\{\mathbf{x}\}_1^{t+1}, \{\mathbf{y}\}_1^{t+1}) \log P(\boldsymbol{\theta}|\{\mathbf{x}\}_1^{t+1}, \{\mathbf{y}\}_1^{t+1})$, where $P(a|b)$ denotes the conditional probability of variable a given condition b. The main steps of these computations are presented below.

Assume that the *a priori* distributions $P(\mathbf{A}|\mathbf{V}, \{\mathbf{x}\}_1^t, \{\mathbf{y}\}_1^t) = \mathcal{N}(\mathbf{A}|\mathbf{M}_t, \mathbf{V}, \mathbf{K}_t)$ and $P(\mathbf{V}|\{\mathbf{x}\}_1^t, \{\mathbf{y}\}_1^t) = \mathcal{IW}_{\mathbf{V}}(\mathbf{Q}_t, n_t)$ are known. Then the posterior distribution of $\boldsymbol{\theta}$ is:

$$\begin{aligned} P(\mathbf{A}, \mathbf{V}|\{\mathbf{x}\}_1^{t+1}, \{\mathbf{y}\}_1^{t+1}) &= \frac{P(\mathbf{y}_{t+1}|\mathbf{A}, \mathbf{V}, \mathbf{x}_{t+1}) P(\mathbf{A}|\mathbf{V}, \{\mathbf{x}\}_1^t, \{\mathbf{y}\}_1^t) P(\mathbf{V}|\{\mathbf{x}\}_1^t, \{\mathbf{y}\}_1^t)}{P(\mathbf{y}_{t+1}|\{\mathbf{x}\}_1^{t+1}, \{\mathbf{y}\}_1^t)}, \\ &= \frac{\mathcal{N}_{\mathbf{y}_{t+1}}(\mathbf{A}\mathbf{x}_{t+1}, \mathbf{V}) \mathcal{N}_{\mathbf{A}}(\mathbf{M}_t, \mathbf{V}, \mathbf{K}_t) \mathcal{IW}_{\mathbf{V}}(\mathbf{Q}_t, n_t)}{\int_{\mathbf{A}} \int_{\mathbf{V}} \mathcal{N}_{\mathbf{y}_{t+1}}(\mathbf{A}\mathbf{x}_{t+1}, \mathbf{V}) \mathcal{N}_{\mathbf{A}}(\mathbf{M}_t, \mathbf{V}, \mathbf{K}_t) \mathcal{IW}_{\mathbf{V}}(\mathbf{Q}_t, n_t)}. \end{aligned}$$

This expression can be rewritten in a more useful form: let $\mathbf{K} \in \mathbb{R}^{m \times m}$ and $\mathbf{Q} \in \mathbb{R}^{d \times d}$ be positive definite matrices. Let $\mathbf{A} \in \mathbb{R}^{d \times m}$, and let us introduce the density function of the matrix valued Student-t distribution (Kotz and Nadarajah, 2004; Minka, 2000) as follows:

$$\mathcal{T}_{\mathbf{A}}(\mathbf{Q}, n, \mathbf{M}, \mathbf{K}) = \frac{|\mathbf{K}|^{d/2}}{\pi^{dm/2}} \frac{Z_{n+m,d}}{Z_{n,d}} \frac{|\mathbf{Q}|^{n/2}}{|\mathbf{Q} + (\mathbf{A} - \mathbf{M})\mathbf{K}(\mathbf{A} - \mathbf{M})^T|^{(m+n)/2}}.$$

Now, we need the following lemma:

Lemma 4.1

$$\begin{aligned} \mathcal{N}_{\mathbf{y}}(\mathbf{A}\mathbf{x}, \mathbf{V}) \mathcal{N}_{\mathbf{A}}(\mathbf{M}, \mathbf{V}, \mathbf{K}) \mathcal{IW}_{\mathbf{V}}(\mathbf{Q}, n) = \mathcal{N}_{\mathbf{A}}((\mathbf{M}\mathbf{K} + \mathbf{y}\mathbf{x}^T)(\mathbf{x}\mathbf{x}^T + \mathbf{K})^{-1}, \mathbf{V}, \mathbf{x}\mathbf{x}^T + \mathbf{K}) \times \\ \times \mathcal{IW}_{\mathbf{V}} \left(\mathbf{Q} + (\mathbf{y} - \mathbf{M}\mathbf{x})(1 - \mathbf{x}^T(\mathbf{x}\mathbf{x}^T + \mathbf{K})^{-1}\mathbf{x})(\mathbf{y} - \mathbf{M}\mathbf{x})^T, n+1 \right) \times \\ \times \mathcal{T}_{\mathbf{y}} \left(\mathbf{Q}, n, \mathbf{M}\mathbf{x}, 1 - \mathbf{x}^T(\mathbf{x}\mathbf{x}^T + \mathbf{K})^{-1}\mathbf{x} \right). \end{aligned}$$

The proof can be found in the Appendix.

Using this lemma, we can compute the posterior probabilities. We introduce the following quantities:

$$\begin{aligned}
\gamma_{t+1} &= 1-\mathbf{x}_{t+1}^T(\mathbf{x}_{t+1}\mathbf{x}_{t+1}^T+\mathbf{K}_t)^{-1}\mathbf{x}_{t+1}, && (8)\\
n_{t+1} &= n_t+1, &&\\
\mathbf{M}_{t+1} &= (\mathbf{M}_t\mathbf{K}_t+\mathbf{y}_{t+1}\mathbf{x}_{t+1}^T)(\mathbf{x}_{t+1}\mathbf{x}_{t+1}^T+\mathbf{K}_t)^{-1}, && (9)\\
\mathbf{Q}_{t+1} &= \mathbf{Q}_t+(\mathbf{y}_{t+1}-\mathbf{M}_t\mathbf{x}_{t+1})\,\gamma_{t+1}\,(\mathbf{y}_{t+1}-\mathbf{M}_t\mathbf{x}_{t+1})^T\,. && (10)
\end{aligned}$$

For the posterior probabilities we have determined that

$$\begin{aligned}
P(\mathbf{A}|\mathbf{V},\{\mathbf{x}\}_1^{t+1},\{\mathbf{y}\}_1^{t+1}) &= \mathcal{N}_{\mathbf{A}}(\mathbf{M}_{t+1},\mathbf{V},\mathbf{x}_{t+1}\mathbf{x}_{t+1}^T+\mathbf{K}_t), && (11)\\
P(\mathbf{V}|\{\mathbf{x}\}_1^{t+1},\{\mathbf{y}\}_1^{t+1}) &= \mathcal{IW}_{\mathbf{V}}(\mathbf{Q}_{t+1},n_{t+1}), && (12)\\
P(\mathbf{y}_{t+1}|\{\mathbf{x}\}_1^{t+1},\{\mathbf{y}\}_1^{t}) &= \mathcal{T}_{\mathbf{y}_{t+1}}(\mathbf{Q}_t,n_t,\mathbf{M}_t\mathbf{x}_{t+1},\gamma_{t+1})\,. &&
\end{aligned}$$

Now we are in the position to compute the entropy of the posterior distribution of $\boldsymbol{\theta}=\{\mathbf{A},\mathbf{V}\}$ using the following lemma:

Lemma 4.2 *The entropy of a stochastic variable with density function* $P(\mathbf{A},\mathbf{V}) = \mathcal{N}_{\mathbf{A}}(\mathbf{M},\mathbf{V},\mathbf{K})\,\mathcal{IW}_{\mathbf{V}}(\mathbf{Q},n)$ *assumes the form* $-\frac{d}{2}\ln|\mathbf{K}|+(\frac{m+d+1}{2})\ln|\mathbf{Q}|+f_{1,1}(d,n)$*, where* $f_{1,1}(d,n)$ *depends only on* d *and* n.

The proof can be found in the Appendix.

Lemmas 4.1 and 4.2 lead to the following corollary:

Corollary 4.3 *For the entropy of a stochastic variable with posterior distribution* $P(\mathbf{A},\mathbf{V}|\mathbf{x},\mathbf{y})$ *it holds that*

$$H(\mathbf{A},\mathbf{V};\mathbf{x},\mathbf{y}) = -\frac{d}{2}\ln|\mathbf{x}\mathbf{x}^T+\mathbf{K}|+f_{1,1}(d,n)+(\frac{m+d+1}{2})\ln|\mathbf{Q}+(\mathbf{y}-\mathbf{M}\mathbf{x})\gamma(\mathbf{y}-\mathbf{M}\mathbf{x})^T|.$$

We note that the following lemma also holds:

Lemma 4.4

$$\int \mathcal{T}_{\mathbf{y}}(\mathbf{Q},n,\boldsymbol{\mu},\gamma)\ln|\mathbf{Q}+(\mathbf{y}-\boldsymbol{\mu})\gamma(\mathbf{y}-\boldsymbol{\mu})^T|d\mathbf{y}$$

is independent from both $\boldsymbol{\mu}$ *and* γ*,*

and thus we can compute the conditional entropy expressed in (7):

Lemma 4.5

$$H(\mathbf{A},\mathbf{V}|\mathbf{y};\mathbf{x}) = \int p(\mathbf{y}|\mathbf{x})H(\mathbf{A},\mathbf{V};\mathbf{x},\mathbf{y})d\mathbf{y} = -\frac{d}{2}\ln|\mathbf{x}\mathbf{x}^T+\mathbf{K}|+f_{1,2}(\mathbf{Q},n,m),$$

where $f_{1,2}(\mathbf{Q},n,m)$ *depends only on* $\mathbf{Q}$*,* n *and* m.

Collecting all the terms, we arrive at the following *intriguingly simple* expression

$$\begin{aligned}\mathbf{u}_{t+1}^{opt} &= \arg\min_{\mathbf{u}_{t+1}} \int p\left(\mathbf{y}_{t+1}|\{\mathbf{x}\}_1^{t+1},\{\mathbf{y}\}_1^t\right) H(\mathbf{A},\mathbf{V}|\{\mathbf{x}\}_1^{t+1},\{\mathbf{y}\}_1^t,\mathbf{y}_{t+1})d\mathbf{y}_{t+1},\\ &= \arg\min_{\mathbf{u}_{t+1}} -\frac{d}{2}\ln|\mathbf{x}_{t+1}\mathbf{x}_{t+1}^T+\mathbf{K}_t| = \arg\max_{\mathbf{u}_{t+1}} \mathbf{x}_{t+1}^T\mathbf{K}_t^{-1}\mathbf{x}_{t+1},\end{aligned} \tag{13}$$

where

$$\mathbf{x}_{t+1} \doteq [\mathbf{r}_{t-I};\ldots;\mathbf{r}_t;\mathbf{u}_{t-J+1};\ldots;\mathbf{u}_{t+1}],$$

and we used that $|\mathbf{x}\mathbf{x}^T+\mathbf{K}| = |\mathbf{K}|(1+\mathbf{x}^T\mathbf{K}^{-1}\mathbf{x})$ according to the Matrix Determinant Lemma (Harville, 1997). We assume a bounded domain $\mathcal{U}$ for the control, which is necessary to keep the maximization procedure of (13) finite. This is, however, a reasonable condition for all practical applications. So,

$$\mathbf{u}_{t+1}^{opt} = \arg\max_{\mathbf{u}_{t+1}\in\mathcal{U}} \mathbf{x}_{t+1}^T\mathbf{K}_t^{-1}\mathbf{x}_{t+1}, \tag{14}$$

In what follows D-optimal control will be referred to as *'infomax interrogation scheme'*. The steps of our algorithm are summarized in Table 1.

```
Control Calculation
    $\mathbf{u}_{t+1} = \arg\max_{\mathbf{u}\in\mathcal{U}} \hat{\mathbf{x}}_{t+1}^T\mathbf{K}_t^{-1}\hat{\mathbf{x}}_{t+1}$
    where $\hat{\mathbf{x}}_{t+1} = [\mathbf{r}_{t-I};\ldots;\mathbf{r}_t;\mathbf{u}_{t-J+1};\ldots;\mathbf{u}_t;\mathbf{u}]$
    set $\mathbf{x}_{t+1} = [\mathbf{r}_{t-I};\ldots;\mathbf{r}_t;\mathbf{u}_{t-J+1};\ldots;\mathbf{u}_t;\mathbf{u_{t+1}}]$
Observation
    observe $\mathbf{r}_{t+1}$, and let $\mathbf{y}_{t+1} = g^{-1}(\mathbf{r}_{t+1})$
Bayesian update
    $\mathbf{M}_{t+1} = (\mathbf{M}_t\mathbf{K}_t + \mathbf{y}_{t+1}\mathbf{x}_{t+1}^T)(\mathbf{x}_{t+1}\mathbf{x}_{t+1}^T+\mathbf{K}_t)^{-1}$
    $\mathbf{K}_{t+1} = \mathbf{x}_{t+1}\mathbf{x}_{t+1}^T + \mathbf{K}_t$
    $n_{t+1} = n_t + 1$
    $\gamma_{t+1} = 1 - \mathbf{x}_{t+1}^T(\mathbf{x}_{t+1}\mathbf{x}_{t+1}^T+\mathbf{K}_t)^{-1}\mathbf{x}_{t+1}$
    $\mathbf{Q}_{t+1} = Q_t + (\mathbf{y}_{t+1}-\mathbf{M}_t\mathbf{x}_{t+1})\gamma_{t+1}(\mathbf{y}_{t+1}-\mathbf{M}_t\mathbf{x}_{t+1})^T$
```

Table 1: Pseudocode of the algorithm

Computation of the inverse $(\mathbf{x}_{t+1}\mathbf{x}_{t+1}^T+\mathbf{K}_t)^{-1}$ in Table 1 can be simplified considerably by the following recursion: let $\mathbf{P}_t = \mathbf{K}_t^{-1}$, then according to the Sherman-Morrison formula (Golub and Van Loan, 1996)

$$\mathbf{P}_{t+1} = (\mathbf{x}_{t+1}\mathbf{x}_{t+1}^T+\mathbf{K}_t)^{-1} = \mathbf{P}_t - \frac{\mathbf{P}_t\mathbf{x}_{t+1}\mathbf{x}_{t+1}^T\mathbf{P}_t}{1+\mathbf{x}_{t+1}^T\mathbf{P}_t\mathbf{x}_{t+1}}. \tag{15}$$

In this expression matrix inversion disappears and only a real number is inverted instead.

5. A-Optimality Approach for Parameter Learning

The D-optimality principle aims to minimize the expected posteriori entropy of the parameters. A-optimality principle differs; it measures the uncertainty by means of the variance and not the entropy. Thus, instead of (7), the A-optimal objective function is as follows:

$$\mathbf{u}_{t+1}^{opt} = \arg\min_{\mathbf{u}_{t+1}} \int d\mathbf{y}_{t+1} P(\mathbf{y}_{t+1}|\{\mathbf{x}\}_1^{t+1},\{\mathbf{y}\}_1^{t}) tr Var[\boldsymbol{\theta}|\{\mathbf{x}\}_1^{t+1},\{\mathbf{y}\}_1^{t+1}]. \tag{16}$$

where $Var[\boldsymbol{\theta}|\mathcal{F}]$ denotes the conditional covariance matrix of $\boldsymbol{\theta}$ given the condition $\mathcal{F}$.

To keep the calculations simple, in this case we use $\mathcal{PIG}_{\mathbf{V}}(\boldsymbol{\alpha}_t,\boldsymbol{\beta}_t)$ prior distribution for the covariance matrix instead of $\mathcal{IW}_{\mathbf{V}}(\mathbf{Q}_t,n_t)$. Using the notations of (8)-(10), the posterior distributions assume the following forms:

Lemma 5.1

$$\begin{aligned}
P(\mathbf{A}|\mathbf{V},\{\mathbf{x}\}_1^{t+1},\{\mathbf{y}\}_1^{t+1}) &= \mathcal{N}_{\mathbf{A}}(\mathbf{M}_{t+1},\mathbf{V},\mathbf{x}_{t+1}\mathbf{x}_{t+1}^T+\mathbf{K}_t), && (17)\\
P(\mathbf{V}|\{\mathbf{x}\}_1^{t+1},\{\mathbf{y}\}_1^{t+1}) &= \mathcal{PIG}_{\mathbf{V}}(\boldsymbol{\alpha}_{t+1},\boldsymbol{\beta}_{t+1}),\\
P(\mathbf{y}_{t+1}|\{\mathbf{x}\}_1^{t+1},\{\mathbf{y}\}_1^{t}) &= \prod_{i=1}^{d} \mathcal{T}_{(\mathbf{y}_{t+1})_i}\left((\boldsymbol{\beta}_t)_i, 2(\boldsymbol{\alpha}_t)_i, (\mathbf{M}_t\mathbf{x}_{t+1})_i, \frac{\gamma_{t+1}}{2}\right), && (18)
\end{aligned}$$

where we used the shorthands

$$\begin{aligned}
(\boldsymbol{\alpha}_{t+1})_i &= (\boldsymbol{\alpha}_t)_i + 1/2,\\
(\boldsymbol{\beta}_{t+1})_i &= (\boldsymbol{\beta}_t)_i + ((\mathbf{y}_{t+1})_i - (\mathbf{M}_t\mathbf{x}_{t+1})_i)^2\frac{\gamma_{t+1}}{2}. && (19)
\end{aligned}$$

The proof can be found in the Appendix.

Given that $P(\mathbf{V}|\{\mathbf{x}\}_1^{t+1},\{\mathbf{y}\}_1^{t+1})$ belongs to the $\mathcal{PIG}$ family we can calculate the quantity $Var(\mathbf{V}|\{\mathbf{x}\}_1^{t+1},\{\mathbf{y}\}_1^{t+1})$ (Gelman et al., 2003):

$$tr\left(Var[\mathbf{V}|\{\mathbf{x}\}_1^{t+1},\{\mathbf{y}\}_1^{t+1}]\right) = \sum_{i=1}^{d} \frac{(\boldsymbol{\beta}_{t+1})_i}{((\boldsymbol{\alpha}_{t+1})_i-1)^2((\boldsymbol{\alpha}_{t+1})_i-2)}.$$

We will need the following lemma:

Lemma 5.2

$$\begin{aligned}
tr\left(Var[\mathbf{A}|\{\mathbf{x}\}_1^{t+1},\{\mathbf{y}\}_1^{t+1}]\right) &= tr\left(\left(\mathbf{K}_t+\mathbf{x}_{t+1}\mathbf{x}_{t+1}^T\right)^{-1}\right) E[tr\mathbf{V}|\{\mathbf{x}\}_1^{t+1},\{\mathbf{y}\}_1^{t+1}],\\
&= tr\left(\left(\mathbf{K}_t+\mathbf{x}_{t+1}\mathbf{x}_{t+1}^T\right)^{-1}\right)\sum_{i=1}^{d}\frac{(\boldsymbol{\beta}_{t+1})_i}{(\boldsymbol{\alpha}_{t+1})_i-1}.
\end{aligned}$$

The proof can be found in the Appendix.

Now we can elaborate on the A-optimal cost function for parameter estimation (16):

$$\begin{aligned}
&\int d\mathbf{y}_{t+1} P(\mathbf{y}_{t+1}|\{\mathbf{x}\}_1^{t+1},\{\mathbf{y}\}_1^{t}) trVar[\boldsymbol{\theta}|\{\mathbf{x}\}_1^{t+1},\{\mathbf{y}\}_1^{t+1}] = && (20)\\
=\ & \int d\mathbf{y}_{t+1}\prod_{i=1}^{d}\mathcal{T}_{(\mathbf{y}_{t+1})_i}\left((\boldsymbol{\beta}_t)_i, 2(\boldsymbol{\alpha}_t)_i, (\mathbf{M}_t\mathbf{x}_{t+1})_i, \frac{\gamma_{t+1}}{2}\right)\times\\
&\times\left(tr\left(\left(\mathbf{K}_t+\mathbf{x}_{t+1}\mathbf{x}_{t+1}^T\right)^{-1}\right)\sum_{i=1}^{d}\frac{(\boldsymbol{\beta}_{t+1})_i}{(\boldsymbol{\alpha}_{t+1})_i-1}+\sum_{i=1}^{d}\frac{(\boldsymbol{\beta}_{t+1})_i}{((\boldsymbol{\alpha}_{t+1})_i-1)^2((\boldsymbol{\alpha}_{t+1})_i-2)}\right),\\
=\ & tr\left(\left(\mathbf{K}_t+\mathbf{x}_{t+1}\mathbf{x}_{t+1}^T\right)^{-1}\right) f_{2,1}(\boldsymbol{\alpha}_{t+1},\boldsymbol{\beta}_t)+f_{2,2}(\boldsymbol{\alpha}_{t+1},\boldsymbol{\beta}_t),
\end{aligned}$$

where $f_{2,1}$ and $f_{2,2}$ depend only on $\boldsymbol{\alpha}_{t+1}$, and $\boldsymbol{\beta}_t$. Here we used (18), (19) and Lemma 4.4.

Applying again the Sherman-Morrison formula (15) and the fact that $tr[\mathbf{K}_t^{-1}\mathbf{x}_{t+1}\mathbf{x}_{t+1}^T\mathbf{K}_t^{-1}] = tr[\mathbf{x}_{t+1}^T\mathbf{K}_t^{-1}\mathbf{K}_t^{-1}\mathbf{x}_{t+1}]$, we arrive at the following expression for A-optimal parameter estimation:

$$\mathbf{u}_{t+1}^{opt} = \arg\max_{\mathbf{u}_{t+1}\in\mathcal{U}} \frac{\mathbf{x}_{t+1}^T\mathbf{K}_t^{-1}\mathbf{K}_t^{-1}\mathbf{x}_{t+1}}{1+\mathbf{x}_{t+1}^T\mathbf{K}_t^{-1}\mathbf{x}_{t+1}}, \tag{21}$$

which is a hyperbolic programming task.

We can conclude that while in the D-optimality case the task is to minimize expression $|(\mathbf{K}_t + \mathbf{x}_{t+1}\mathbf{x}_{t+1}^T)^{-1}|$, the A-optimality principle is concerned with the minimization of $tr[(\mathbf{K}_t+\mathbf{x}_{t+1}\mathbf{x}_{t+1}^T)^{-1}]$.

6. D-Optimality Approach for Noise Estimation

One might wish to compute the optimal control for estimating noise $\mathbf{e}_t$ in (1), instead of the identification problem above. Based on (1) and because

$$\mathbf{e}_{t+1} = \mathbf{y}_{t+1} - \sum_{i=0}^{I}\mathbf{F}_i\mathbf{r}_{t-i} - \sum_{j=0}^{J}\mathbf{B}_j\mathbf{u}_{t+1-j}, \tag{22}$$

one might think that the best strategy is to use the optimal infomax control of Table 1, since it provides good estimations for parameters $\mathbf{A} = [\mathbf{F}_I,\ldots,\mathbf{F}_0,\mathbf{B}_J,\ldots,\mathbf{B}_0]$ and so for noise $\mathbf{e}_t$.

Another—and different—thought is the following. At time $t+1$, let the estimation of the noise be $\hat{\mathbf{e}}_{t+1} = \mathbf{y}_{t+1} - \sum_{i=0}^{I}\hat{\mathbf{F}}_i^t\mathbf{r}_{t-i} - \sum_{j=0}^{J}\hat{\mathbf{B}}_j^t\mathbf{u}_{t+1-j}$, where $\hat{\mathbf{F}}_i^t$ (i=0,...,I), and $\hat{\mathbf{B}}_j^t$ (j=0,...,J) denote the estimations of $\mathbf{F}$ and $\mathbf{B}$ respectively.

Using (22), we have that

$$\mathbf{e}_{t+1} - \hat{\mathbf{e}}_{t+1} = \sum_{i=0}^{I}(\mathbf{F}_i - \hat{\mathbf{F}}_i^t)\mathbf{r}_{t-i} + \sum_{j=0}^{J}(\mathbf{B}_j - \hat{\mathbf{B}}_j^t)\mathbf{u}_{t+1-j}. \tag{23}$$

This hints that the control should be $\mathbf{u}_t = \mathbf{0}$ for all times in order to get rid of the error contribution of matrix $\mathbf{B}_j$ in (23).

Straightforward D-optimality considerations oppose the utilization of objective (6) for the present task. One can optimize, instead, the following quantity:

$$\arg\max_{\mathbf{u}_{t+1}} I(\mathbf{e}_{t+1},\mathbf{y}_{t+1};\{\mathbf{x}\}_1^{t+1},\{\mathbf{y}\}_1^t).$$

In other words, for the estimation of the noise we want to design a control signal $\mathbf{u}_{t+1}$ such that the next output is the best from the point of view of greedy optimization of mutual information between the next output $\mathbf{y}_{t+1}$ and the noise $\mathbf{e}_{t+1}$. It is easy to show that this task is equivalent to the following optimization problem:

$$\arg\min_{\mathbf{u}_{t+1}} \int d\mathbf{y}_{t+1} P(\mathbf{y}_{t+1}|\{\mathbf{x}\}_1^{t+1},\{\mathbf{y}\}_1^t) H(\mathbf{e}_{t+1};\{\mathbf{x}\}_1^{t+1},\{\mathbf{y}\}_1^{t+1}), \tag{24}$$

where $H(\mathbf{e}_{t+1};\{\mathbf{x}\}_1^{t+1},\{\mathbf{y}\}_1^{t+1}) = H(\mathbf{A}\mathbf{x}_{t+1};\{\mathbf{x}\}_1^{t+1},\{\mathbf{y}\}_1^{t+1})$, because $\mathbf{e}_{t+1} = \mathbf{y}_{t+1} - \mathbf{A}\mathbf{x}_{t+1}$.

In practice, we perform this optimization in an appropriate domain $\mathcal{U}$. After some mathematical calculation we can prove that the D-optimal interrogation scheme for noise estimation gives rise to the following control:

Lemma 6.1

$$\mathbf{u}_{t+1}^{opt} = \arg\min_{\mathbf{u}_{t+1}\in\mathcal{U}} \mathbf{x}_{t+1}^T \mathbf{K}_t^{-1} \mathbf{x}_{t+1}. \tag{25}$$

The proof of this lemma can be found in the Appendix.

It is worth noting that this D-optimal cost function for noise estimation and the D-optimal cost function derived for parameter estimation in (13) are not compatible with each other. Estimating one of them quickly will necessarily delay the estimation of the other.

We shall show later (Section 9) that for large t values, expression (25) gives rise to control values close to $\mathbf{u}_t = \mathbf{0}$.

7. A-Optimality Approach for Noise Estimation

Instead of (24), our task is to compute the following quantity:

$$\arg\min_{\mathbf{u}_{t+1}} \int d\mathbf{y}_{t+1} P(\mathbf{y}_{t+1}|\{\mathbf{x}\}_1^{t+1},\{\mathbf{y}\}_1^{t})\, tr\left(Var[\mathbf{e}_{t+1}|\{\mathbf{x}\}_1^{t+1},\{\mathbf{y}\}_1^{t+1}]\right). \tag{26}$$

We will apply the identity

$$\begin{aligned}\mathcal{N}_{\mathbf{A}\mathbf{x}_{t+1}}&\left(\mathbf{M}_{t+1}\mathbf{x}_{t+1},\mathbf{V},\left(\mathbf{x}_{t+1}^T\mathbf{K}_{t+1}^{-1}\mathbf{x}_{t+1}\right)^{-1}\right)\mathcal{PIG}_{\mathbf{V}}(\boldsymbol{\alpha}_{t+1},\boldsymbol{\beta}_{t+1}) = \\ =\prod_{i=1}^{d}\mathcal{T}_{(\mathbf{A}\mathbf{x}_{t+1})_i}&\left((\boldsymbol{\beta}_{t+1})_i,2(\boldsymbol{\alpha}_{t+1})_i,(\mathbf{M}_{t+1}\mathbf{x}_{t+1})_i,\left(\mathbf{x}_{t+1}^T\frac{\mathbf{K}_{t+1}^{-1}}{2}\mathbf{x}_{t+1}\right)^{-1}\right)\times \\ \times\mathcal{PIG}_{\mathbf{V}}&\left(\boldsymbol{\alpha}_{t+1}+1,\boldsymbol{\beta}_{t+1}+diag[(\mathbf{y}_{t+1}-\mathbf{M}_t\mathbf{x}_{t+1})\frac{\gamma_{t+1}}{2}(\mathbf{y}_{t+1}-\mathbf{M}_t\mathbf{x}_{t+1})^T]\right),\end{aligned}$$

which can be proven by using Lemma A.1. We can simplify (26) by noting that

$$tr\left(Var[\mathbf{e}_{t+1}|\{\mathbf{x}\}_1^{t+1},\{\mathbf{y}\}_1^{t+1}]\right) = tr\left(Var[\mathbf{A}\mathbf{x}_{t+1}|\{\mathbf{x}\}_1^{t+1},\{\mathbf{y}\}_1^{t+1}]\right).$$

We also take advantage of the fact that

$$Var_{\mathbf{V}}[E[\mathbf{A}\mathbf{x}_{t+1}|\mathbf{V},\{\mathbf{x}\}_1^{t+1},\{\mathbf{y}\}_1^{t+1}]] = Var_{\mathbf{V}}[\mathbf{M}_{t+1}\mathbf{x}_{t+1}] = 0,$$

and proceed as

$$\begin{aligned}E_{\mathbf{V}}[trVar(\mathbf{A}\mathbf{x}_{t+1}|\mathbf{V},\{\mathbf{x}\}_1^{t+1},\{\mathbf{y}\}_1^{t+1})] &= E_{\mathbf{V}}[tr(\mathbf{V}\otimes\mathbf{x}_{t+1}^T\mathbf{K}_{t+1}^{-1}\mathbf{x}_{t+1}|\{\mathbf{x}\}_1^{t+1},\{\mathbf{y}\}_1^{t+1}],\\ &= tr(E[\mathbf{V}|\{\mathbf{x}\}_1^{t+1},\{\mathbf{y}\}_1^{t+1}])\mathbf{x}_{t+1}^T\mathbf{K}_{t+1}^{-1}\mathbf{x}_{t+1},\\ &= \mathbf{x}_{t+1}^T\mathbf{K}_{t+1}^{-1}\mathbf{x}_{t+1}\sum_{i=1}^{d}\frac{(\boldsymbol{\beta}_{t+1})_i}{(\boldsymbol{\alpha}_{t+1})_i-1},\end{aligned}$$

where $\otimes$ denotes the Kronecker product. The law of total variance says that

$$Var[\mathbf{A}\mathbf{x}] = Var[E[\mathbf{A}\mathbf{x}|\mathbf{V}]] + E[Var[\mathbf{A}\mathbf{x}|\mathbf{V}]],$$

and hence

$$tr\,Var[\mathbf{A}\mathbf{x}_{t+1}|\{\mathbf{x}\}_1^{t+1},\{\mathbf{y}\}_1^{t+1}] = \mathbf{x}_{t+1}^T\mathbf{K}_{t+1}^{-1}\mathbf{x}_{t+1}\sum_{i=1}^{d}\frac{(\boldsymbol{\beta}_{t+1})_i}{(\boldsymbol{\alpha}_{t+1})_i-1}.$$

There is another way to arrive at the same result. One can apply (37) with Lemma A.1 and use the fact that the covariance matrix of a $\mathcal{T}_{\mathbf{x}}(\beta,\alpha,\boldsymbol{\mu},\mathbf{K})$ distributed variable is $\frac{\beta\mathbf{K}^{-1}}{\alpha-2}$ (Gelman et al., 2003). That is, we have

$$trVar[\mathbf{A}\mathbf{x}_{t+1}|\{\mathbf{x}\}_1^{t+1},\{\mathbf{y}\}_1^{t+1}] = \sum_{i=1}^{d}\left(\frac{(\boldsymbol{\beta}_{t+1})_i\left(\mathbf{x}_{t+1}^T\frac{\mathbf{K}_{t+1}^{-1}}{2}\mathbf{x}_{t+1}\right)}{2(\boldsymbol{\alpha}_{t+1})_i-2}\right).$$

Now, we can proceed as

$$\begin{aligned}
\int d\mathbf{y}_{t+1}P(\mathbf{y}_{t+1}|\{\mathbf{x}\}_1^{t+1},\{\mathbf{y}\}_1^{t})tr\left(Var[\mathbf{e}_{t+1}|\{\mathbf{x}\}_1^{t+1},\{\mathbf{y}\}_1^{t+1}]\right) &= \\
\int d\mathbf{y}_{t+1}P(\mathbf{y}_{t+1}|\{\mathbf{x}\}_1^{t+1},\{\mathbf{y}\}_1^{t})tr(E[\mathbf{V}|\{\mathbf{x}\}_1^{t+1},\{\mathbf{y}\}_1^{t+1}])\mathbf{x}_{t+1}^T\mathbf{K}_{t+1}^{-1}\mathbf{x}_{t+1} &= \\
\mathbf{x}_{t+1}^T\mathbf{K}_{t+1}^{-1}\mathbf{x}_{t+1}\int d\mathbf{y}_{t+1}P(\mathbf{y}_{t+1}|\{\mathbf{x}\}_1^{t+1},\{\mathbf{y}\}_1^{t})\sum_{i=1}^{d}\frac{(\boldsymbol{\beta}_{t+1})_i}{(\boldsymbol{\alpha}_{t+1})_i-1} &= \\
\mathbf{x}_{t+1}^T\mathbf{K}_{t+1}^{-1}\mathbf{x}_{t+1}\sum_{i=1}^{d}f_4((\boldsymbol{\alpha}_{t+1})_i,(\boldsymbol{\beta}_t)_i), &
\end{aligned} \tag{27}$$

where we used (18), (19) and Lemma 4.4 again. Applying the Sherman-Morrison formula one can see that the task is the same as in (25).

8. Joint Parameter and Noise Estimation

So far we wanted to optimize the control in order to speed-up learning of either the parameters of the dynamics or the noise. In this section we investigate the A- and D-optimality principles for the joint parameter and noise estimation task.

8.1 A-optimality

According to the A-optimality principle, the joined objective for parameter and noise estimation is given as:

$$\int d\mathbf{y}_{t+1}P(\mathbf{y}_{t+1}|\{\mathbf{x}\}_1^{t+1},\{\mathbf{y}\}_1^{t})\,trVar[vec(\mathbf{A}),diag(\mathbf{V}),\mathbf{e}_{t+1}|\{\mathbf{x}\}_1^{t+1},\{\mathbf{y}\}_1^{t+1}].$$

By means of (20), (27) and Lemma 5.2, it is equivalent to:

$$\int d\mathbf{y}_{t+1}P(\mathbf{y}_{t+1}|\{\mathbf{x}\}_1^{t+1},\{\mathbf{y}\}_1^{t})E[tr\mathbf{V}|\mathbf{x}_1^{t+1},\mathbf{y}_1^{t+1}]\,tr\left(\left(\mathbf{K}_t+\mathbf{x}_{t+1}\mathbf{x}_{t+1}^T\right)^{-1}+\mathbf{x}_{t+1}^T\mathbf{K}_{t+1}^{-1}\mathbf{x}_{t+1}\right).$$

From here, one can prove the following lemma in a few steps:

Lemma 8.1 *The A-optimality principle in the joined parameter and noise estimation task gives rise to the following choice for control:*

$$\mathbf{u}_{t+1}^{opt} = \arg\max_{\mathbf{u}_{t+1}\in\mathcal{U}}\frac{1+\mathbf{x}_{t+1}^T\mathbf{K}_t^{-1}\mathbf{K}_t^{-1}\mathbf{x}_{t+1}}{1+\mathbf{x}_{t+1}^T\mathbf{K}_t^{-1}\mathbf{x}_{t+1}}. \tag{28}$$

The proof can be found in the Appendix.

Thus, the task is a hyperbolic programming task, similar to (21).

8.2 D-optimality

One of the most salient differences between A-optimality and D-optimality is that for D-optimality we have

$$H(X,Y) = H(X|Y) + H(Y),$$

however, for A-optimality the corresponding equation does not hold in general, because:

$$trVar(X,Y) \neq E_Y[trVar(X|Y)] + trVar(Y).$$

An implication—as we shall see below—is that we cannot use the D-optimality principle for the joint parameter and noise estimation task. For D-optimality our cost function would be

$$\arg\min_{\mathbf{u}_{t+1}} \int d\mathbf{y}_{t+1} P(\mathbf{y}_{t+1}|\{\mathbf{x}\}_1^{t+1}, \{\mathbf{y}\}_1^{t}) H(\mathbf{A}, \mathbf{V}, \mathbf{e}_{t+1}|\{\mathbf{x}\}_1^{t+1}, \{\mathbf{y}\}_1^{t+1}),$$

but the following equality holds:

$$H(\mathbf{A}, \mathbf{V}, \mathbf{e}_{t+1}|\{\mathbf{x}\}_1^{t+1}, \{\mathbf{y}\}_1^{t+1}) = H(\mathbf{A}, \mathbf{V}|\{\mathbf{x}\}_1^{t+1}, \{\mathbf{y}\}_1^{t+1}) + H(\mathbf{e}_{t+1}|\mathbf{A}, \mathbf{V}\{\mathbf{x}\}_1^{t+1}, \{\mathbf{y}\}_1^{t+1}),$$

and since $\mathbf{e}_{t+1} = \mathbf{y}_{t+1} - \mathbf{A}\mathbf{x}_{t+1}$, therefore the last term $H(\mathbf{e}_{t+1}|\mathbf{A}, \mathbf{V}\{\mathbf{x}\}_1^{t+1}, \{\mathbf{y}\}_1^{t+1}) = -\infty$. The first term is a finite real number, thus we can conclude that the D-optimality cost function is constant $-\infty$, and therefore the D-optimality principle does not suit the joint parameter and noise estimation task.

9. Non-myopic Optimization

Until now, we considered myopic methods for the optimization of control, that is, we aimed to determine the optimum of the objective only for the next step. In this section, we show a non-myopic heuristics for the noise estimation task (25).

The optimization of the derived objective function, $\mathbf{x}_{t+1}^T \mathbf{K}_t^{-1} \mathbf{x}_{t+1}$, is simple, provided that $\mathbf{K}_t$ is fixed during the optimization of $\mathbf{u}_{t+1}$. If so, then the optimization task is quadratic. To see this, let us partition matrix $\mathbf{K}_t$ as follows:

$$\mathbf{K}_t = \begin{pmatrix} \mathbf{K}_t^{11} & \mathbf{K}_t^{12} \\ \mathbf{K}_t^{21} & \mathbf{K}_t^{22} \end{pmatrix},$$

where $\mathbf{K}_t^{11} \in \mathbb{R}^{d\times d}$,$\mathbf{K}_t^{21} \in \mathbb{R}^{m-d\times d}$, $\mathbf{K}_t^{22} \in \mathbb{R}^{m-d\times m-d}$. It is easy to see that if domain $\mathcal{U}$ in (25) is large enough then

$$\mathbf{u}_{t+1}^{opt} = (\mathbf{K}_t^{22})^{-1} \mathbf{K}_t^{21} \mathbf{r}_t. \tag{29}$$

It occurs, however, that the objective $\mathbf{x}_{t+1}^T \mathbf{K}_t^{-1} \mathbf{x}_{t+1}$ may be improved by considering multiple-step lookaheads. In this case matrix $\mathbf{K}_t$ can be subject to changes in $\mathbf{x}_{t+1}^T \mathbf{K}_t^{-1} \mathbf{x}_{t+1}$, because it depends on previous control inputs $\mathbf{u}_1, \ldots, \mathbf{u}_t$ derived from previous optimization steps.

We propose a two-step heuristics for the long-term minimization of expression $\mathbf{x}_{t+1}^T \mathbf{K}_t^{-1} \mathbf{x}_{t+1}$. During the first τ-step long stage, we focus only on the minimization of the quantity $|\mathbf{K}_t^{-1}|$. Then, if this quantity $|\mathbf{K}_t^{-1}|$ becomes small, we start the second stage: we consider $\mathbf{K}_t^{-1}$ as given and search for control that minimizes quantity $\mathbf{x}_{t+1}^T \mathbf{K}_t^{-1} \mathbf{x}_{t+1}$, so we now apply the rule of (29). Thus, this method 'sacrifices' the first τ steps in order to achieve smaller costs later; this heuristic optimization is non-myopic. More formally, we use the strategy of Table 1 in the first τ steps in order to decrease quantity $|\mathbf{K}_t^{-1}|$ fast. Then after this τ-steps, we switch to the control method of (29). This will decrease the cost function (25) further. We will call this non-greedy interrogation heuristics introduced for noise estimation *'τ-infomax noise interrogation'*. This non-myopic heuristics admits that parameter estimation of the dynamics is the prerequisite of noise estimation, because improper parameter estimation makes apparent noise, and thus the heuristics sacrifices τ steps for parameter estimation.

In Section 10 we will empirically show that using this non-myopic strategy, after τ steps we can achieve smaller cost values in (25)—as well as better performance in parameter estimation—than using the greedy competitors. The compromise is that in the first τ steps the performance of the non-myopic control can be worse than that of the other control methods.

We note that in the τ-infomax noise interrogation, for large switching time τ and for large t values, $|\mathbf{K}_t^{22}|$ will be large, and hence—according to (29)—the optimal $\mathbf{u}_t$ for interrogation will be close to $\mathbf{0}$. (In Section 10 we will show this empirically.) A reasonable approximation of the 'τ-infomax noise interrogation' is to use the control given in Table 1 for τ steps and to switch to *zero-interrogation* onwards. This scheme will be called the *'τ-zero interrogation'* scheme.

10. Numerical Illustrations

We illustrate by numerical simulations the power of A- and D-optimizations.

10.1 Generated Data

This section provides numerical experiments for parameter and noise estimations on artificially generated toy problems.

10.1.1 PARAMETER ESTIMATION

We investigated the parameter estimation capability of the D- and A-optimal interrogation. Matrix $\mathbf{F} \in \mathbb{R}^{d \times d}$ has been generated as a random orthogonal matrix multiplied by 0.9 so that the magnitudes of its eigenvalues remained below 1. Random matrix $\mathbf{B} \in \mathbb{R}^{d \times c}$ was generated from standard normal distribution. Elements of the diagonal covariance matrix $\mathbf{V}$ of noise $\mathbf{e}_t$ were generated from the uniform distribution over $[0,1]$. The process is stable under these conditions.

To study whether or not the D- and A-optimal interrogations are able to estimate the true parameters we measured the averages of the squared deviations of the true matrices $\mathbf{F}$ and $\mathbf{B}$ and the means of their posterior estimations, respectively. The square roots of these estimations are the mean squared errors (MSE). One might use other options to measure performance. For example, L_2 norm could be replaced by the L_1 norm and the variance of the posterior estimations could also be added as the complementary information for the bias.

We examined the following strategies: (i) D-optimal control of Table 1 with $\mathcal{U} = [-\delta, \delta]^c$, which defines a c-dimensional hypercube. The value of δ was set to 50. (ii) A-optimal control of (21) with the same $\mathcal{U}$, (iii) zero control: $\mathbf{u}_t = \mathbf{0} \in \mathbb{R}^c \ \forall t$, (iv) random control: $u_t \in [-\delta, \delta]^c$ generated randomly from the uniform distribution in the c-dimensional hypercube, (v) control defined by (25) for noise estimation, called 'noise control', (vi) 25-zero control and (vii) 75-zero control defined in Section 9.

For solving the quadratic problem of (14) and (25) we used a subspace trust-region procedure, which is based on the interior-reflective Newton method described by Coleman and Li (1996). Its implementation is available in the Matlab Optimization toolbox. However, the optimization task in (21) is more involved: Generally, the optimization of a constrained hyperbolic programming task is quite difficult. We tried the gradient ascent method, but its convergence appeared to be very slow and we got poor results. In this case, it was more efficient to apply a simplex method as follows: we know that the optimal solution of (21) lies at the boundary of $\mathcal{U}$. Thus, we chose one corner of hypercube $\mathcal{U}$ randomly with uniform distribution and moved greedily to the neighboring corners with the best improvement in the objective. This procedure was iterated until convergence. The method was efficient for our special simple optimization domain.

We investigated two distinct cases. In the first case we set $d = 10 < c = 40$; the dimension of the observations is smaller than the dimension of the control. By contrast, in the other case we set $d = 40 > c = 10$. Results are shown in Fig. 1 (a-b) and in Fig. 2 (a-b). We separated the MSE values of matrices $\mathbf{B}$ and $\mathbf{F}$. According to the figure, zero control may give rise to early and sudden drops in the MSE values of matrix $\mathbf{F}$. Not surprisingly, however, zero control is unable to estimate matrix $\mathbf{B}$. For both types of matrices as well as for $d < c$ and for $d > c$, the D-optimal procedure produced the smallest MSE after about 50 online estimations, but the A-optimal method reached very similar levels only a few iterations later. As can be expected, τ-zero control, which is identical to D-optimal control in the first τ steps fell behind D-optimal control after τ since it changes the objective and estimates the noise and not the parameters afterwards.

For statistical significance studies, we introduced the concept of average correlation curves. We use Fig. 1 to explain this concept. There are 7 curves in Fig. 1 each representing the averages of 25 computer runs. Error bars make the curves incomprehensible and they hide the correlations that may be present between the errors. We note that the relative order of the curves is of interest for us. However, it is possible that in each run the relative order of the curves was the same and the overlap of the error bars—which originates from the large differences between the individual runs—hides this important piece of information. We treat this problem as follows. In each time instant $1 \leq t \leq 250$ and for all $1 \leq i < j \leq 25$ we compute the empirical (linear, or rank) correlation of the 7 curves of the i^{th} and j^{th} experiment and take the average of the $25 \times 24/2 = 300$ values. The most significant case gives rise to 1 for each of the 300 correlations, that is, the 25 experiments agree in the height of the curves at that time instant, or in their relative orderings for the case of rank correlation. If there is any single experiment out of 25 that produces different heights or orders then the average correlation becomes smaller than 1. For randomly chosen curves the average correlation is 0. Results can be seen in Fig. 1 (c-d) and Fig. 2 (c-d) for linear Pearson and for Kendal rank correlations, respectively. The curves demonstrate that after about 50 steps, the correlations, in particular the linear correlation is almost 1. This means that the curves behaved similarly in a considerable portion of the experiments. The slightly different picture shown by the linear correlation and the rank correlation could be due to the fact that the performance of the A and D-optimal control is very similar after some time, and their ordering may change often, thus giving rise to changes in the ranks in different experiments.

10.1.2 NOISE ESTIMATIONS

In Section 7 we showed that A- and D-optimality principles result in the same cost function.

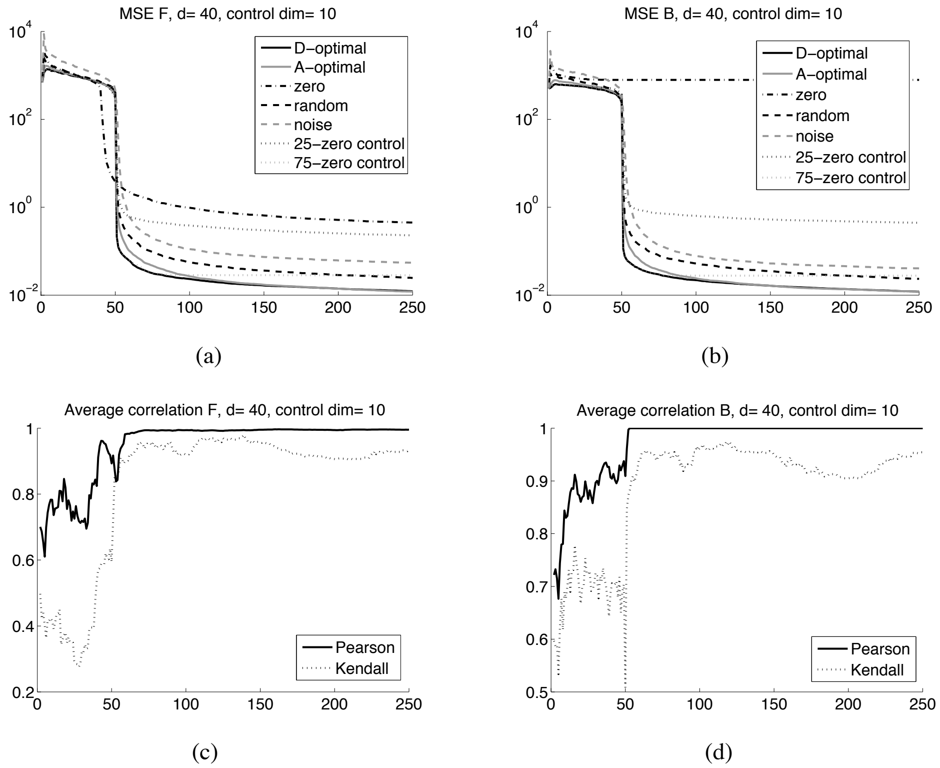


Figure 1: Mean Square Error of the estimated parameters for different control strategies and the significance of the curves. Magnitude of MSE as a function of time is averaged for 25 runs. Dimension of the control is 10. $\mathbf{F} \in \mathbb{R}^{40\times 40}$, $\mathbf{B} \in \mathbb{R}^{40\times 10}$. (a): MSE of the estimated matrix $\hat{\mathbf{F}}$. (b): MSE of the estimated matrix $\hat{\mathbf{B}}$. (c): The average correlation curves for the estimation of $\mathbf{F}$. (d): The average correlation curves for the estimation of $\mathbf{B}$. For details see the text.

We investigated the noise estimation capability of the interrogation in (25) for four cases. The first set of experiments illustrates that the estimation of driving noise $\mathbf{e}_t$ for large τ values barely differs if we replace the τ-*infomax noise interrogation* with the τ-*zero interrogation* scheme. Parameters were the same as above and the MSE of the noise estimation was computed. Results are shown in Fig. 3: for the case of $\tau = 21$, cost function (25) of the τ-zero interrogation is higher than that of τ-infomax interrogation. However, for values $\tau = 51$ and 81 the performances of the two schemes are approximately identical. Given that τ-zero and τ-infomax noise interrogation behave similarly for large τ values, we compare the τ-zero interrogation scheme with other schemes in our numerical experiments.

In the second experiment we investigated the problem of noise estimation on a toy problem. Parameters were set as in Section 10.1.1, and the following strategies were compared: zero control, infomax control, random control and τ-zero control for different τ values. Results are shown in Fig. 4. It is clear from the figure that neither the zero control, nor the infomax (D-optimal) control of Table 1 work for this case. If we want to have minimal MSE in approximately τ steps then the best strategy is to apply the τ-zero strategy, that is, the strategy of Table 1 up to τ steps and then to switch to zero control. Note, however, that parameter estimation requires to keep control values non-negligible forever (Table 1).

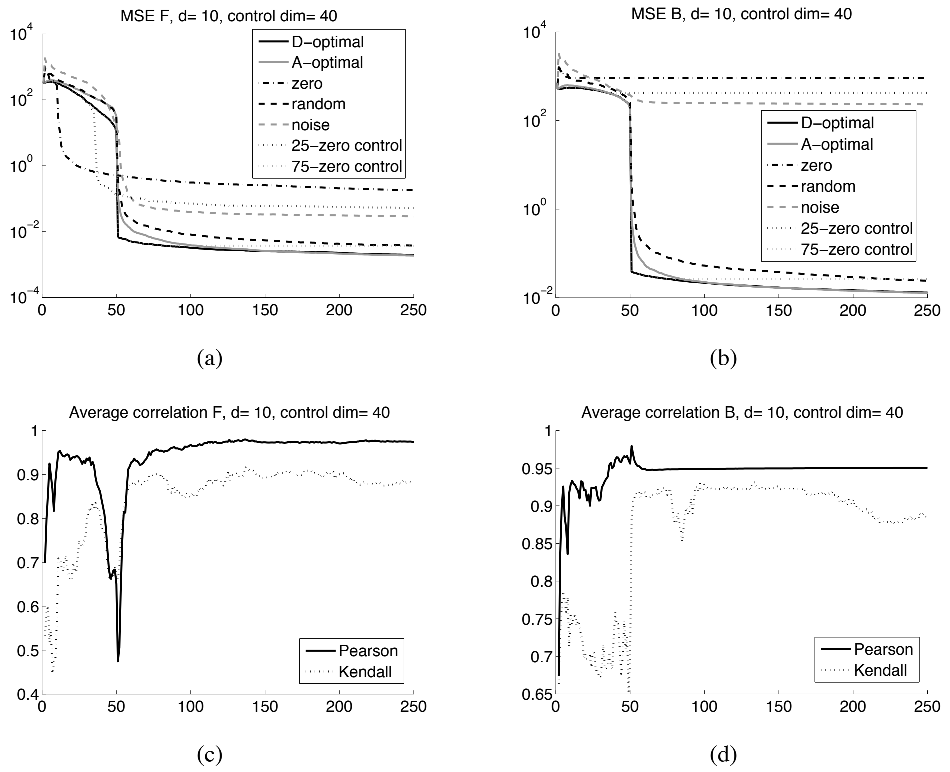
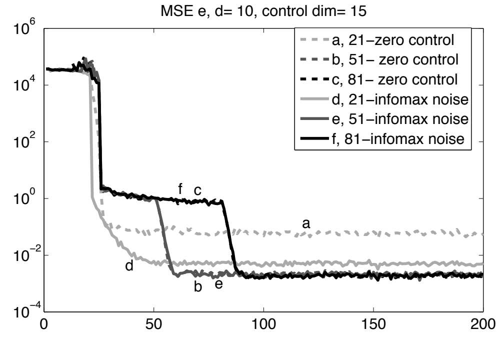
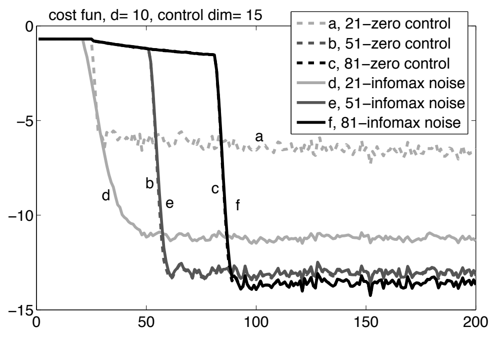


Figure 2: Mean Square Error of the estimated parameters for different control strategies and the significance of the curves. Magnitude of MSE as a function of time is averaged for 25 runs. Dimension of the control is 40. $\mathbf{F} \in \mathbb{R}^{10\times 10}$, $\mathbf{B} \in \mathbb{R}^{10\times 40}$. (a): MSE of the estimated matrix $\hat{\mathbf{F}}$. (b): MSE of the estimated matrix $\hat{\mathbf{B}}$. (c): The average correlation curves for the estimation of $\mathbf{F}$. (d): The average correlation curves for the estimation of $\mathbf{B}$. For details see the text.

In the third experiment we used numerical tools to support our the arguments we made in Section 9. We investigate the D-optimal, the zero, the random, and the greedy noise control of (25), as well as the 71-zero and 101-zero controls. Results show that if we may sacrifice the first τ steps, then the non-myopic $\tau - zero$ control gives rise to the smallest MSE for the estimated noise and the smallest values for the cost function (25) *after* τ steps considering all studied control methods. Figure 5a shows the MSE of the estimated driving noise, whereas Fig. 5b depicts the cost $\mathbf{x}_{t+1}^T \mathbf{K}_t^{-1} \mathbf{x}_{t+1}$. Figure 5c is about the time dependence of $\log|\mathbf{K}_t|$ that supports our argument in Section 9, namely, it may be worth to sacrifice steps at the beginning to quickly decrease $|\mathbf{K}_t^{-1}|$ (i.e., to decrease $\log|\mathbf{K}_t|$) in order to estimate (25) efficiently later. The problem we studied was the same as before, except that $d = 25$ and $c = 25$ were applied.

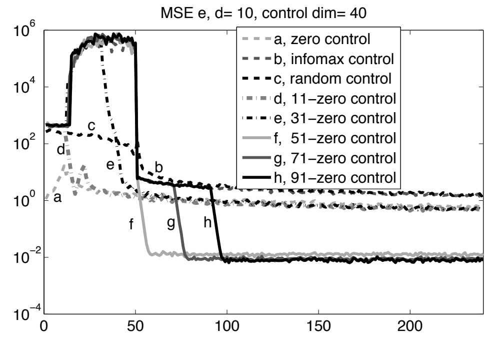
The fourth experiment illustrates the efficiency of the approximation of the noise for the case when our assumptions on $\mathbf{e}_t$ are not fulfilled. Here noise $\mathbf{e}_t$ was neither Gaussian nor i.i.d. 'Noise' $\mathbf{e}_t$ was chosen as equidistant points smoothly 'walking' along a 3 dimensional spiral curve as a function of time (Fig. 6a). Dimensions of observation and control were 3 and 15, respectively. Results are shown in Fig. 6. Neither random control, nor infomax interrogation of Table 1 (Fig. 6c), nor zero control (Fig. 6d) could produce reasonable estimation. However, the τ-zero interrogation scheme

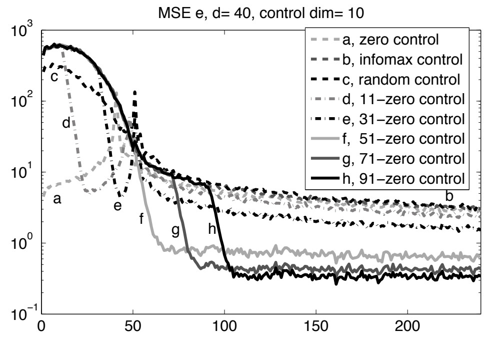




(a) (b)

Figure 3: Comparing τ-infomax noise and τ-zero interrogations. The curves are averaged for 50 runs. Dimension of the control is 15 and the dimension of the observation is 10. (a): MSE of the estimated noise (b): Cost function as given in (25). 'τ-infomax noise' (τ-zero) means that up to step number τ strategy of Table 1 applies and then the control of Eq. (29) is followed.





(a) (b)

Figure 4: Mean Square Error of the estimated noise for different control strategies. Magnitude of MSE as a function of time is averaged for 20 runs. (a): Dimension of the control is 40. $\mathbf{F} \in \mathbb{R}^{40\times 40}$, $\mathbf{B} \in \mathbb{R}^{40\times 10}$. (b): Dimension of the control is 10. $\mathbf{F} \in \mathbb{R}^{10\times 10}$, $\mathbf{B} \in \mathbb{R}^{10\times 40}$. 'τ-zero' means that up to step number τ the strategy illustrated in Table 1 was applied and then zero control followed.

produced a good approximation for large enough τ values (Fig. 6e). Details of this illustration are shown in Fig. 6f.

10.1.3 Joint Parameter and Noise Estimations

In Section 8 we showed that the objective of the D-optimality principle is constant for the joined parameter and noise estimation task. However, A-optimality principle provides sensible cost function (Eq. (28)). Unfortunately, it leads to a hyperbolic programming task. This optimization is hard in most cases. One can estimate the complexity of the objective by inspecting lower dimensional cases. We show the negative logarithm of (28) for the 2 dimensional case for different $\mathbf{K}$ matrices (Fig. 7). In one of the cases the null vector corresponds to the minimum, whereas in the other case the minimum is at a boundary point of the optimization domain. Also, the cost functions appear to be flat in a large part of their domains, rendering gradient based methods ineffective.

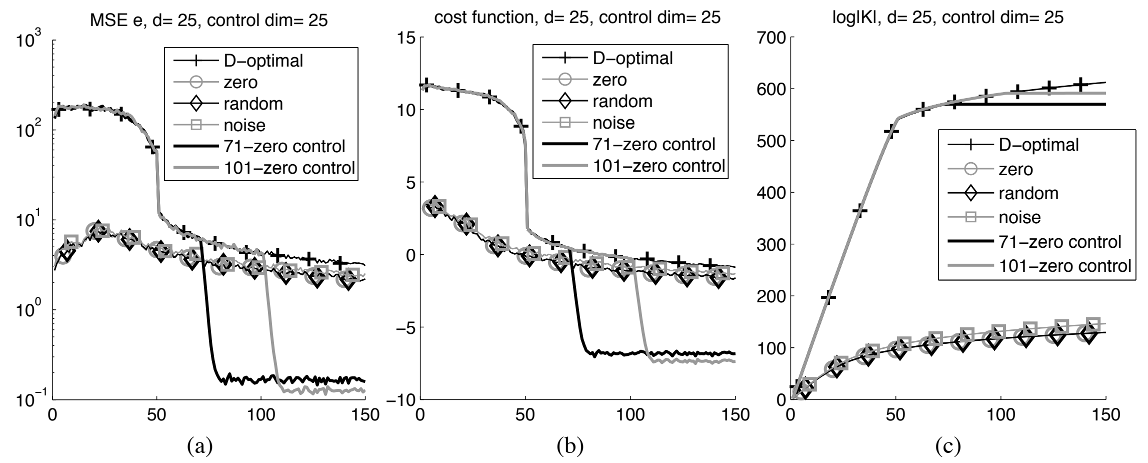


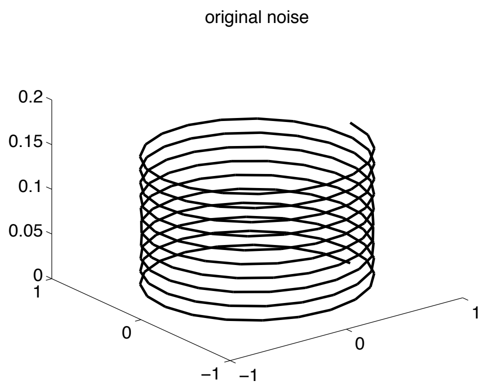
Figure 5: Empirical study on non-myopic controls for noise estimation. We sacrifice the first τ steps to achieve better MSE and smaller cost function. The curves are averaged over 25 runs. The dimension of the control and the dimension of the observation is 25. (a): MSE of noise estimation for different control strategies. (b): $\mathbf{x}_{t+1}^T \mathbf{K}_t^{-1} \mathbf{x}_{t+1}$ cost function for different control strategies. (c): $\log|\mathbf{K}_t|$ function for different control strategies.

Studies were conducted on the problem family of 10.1.1 for observation dimension $d = 15$ and control dimension $c = 30$. For the optimization, we modified the simplex method that we used for the hyperbolic task before. The single difference is that upon convergence, the best value was compared with the value of the objective at the 0 point and we chose the better one for control. We have compared this strategy with the parameter estimation strategy of the D-optimality principle, with zero control strategy, with random control strategy, and with $\tau - \textit{zero}$ control for several τ values. Results are shown in Fig. 8. The figure indicates that control derived from the A-optimality principle (28) provides superior MSE results at approximately 45 iterations and then onwards compared to the other *myopic* techniques, however its performance was slightly worse than the *non-myopic* τ-zero control for τ values larger than an appropriate threshold.

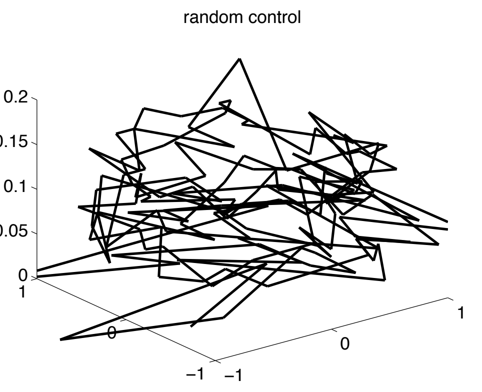
Inspecting the optimal control series of the winner, we found that the algorithm chooses control values from the boundaries of the hypercube in the first 45 or so iterations. Then up to about 130 iterations it is switching between zero control and controls on the boundaries, but eventually it uses zero controls only. That is, the A-optimality principle is able to detect the need for the switch from high control values (to determine the parameters of the dynamics) to zero control values for noise estimation. This automated switching behavior is a special advantage of the A-optimality principle.

10.2 Controlled Independent Component Analysis

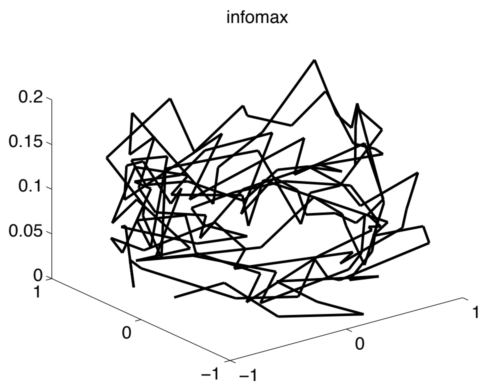
In this section we study the usefulness of our methods for auto-regressive (AR) hidden processes with independent driving sources and we would like to find the independent causes that drive the processes. This task belongs to independent component analysis (ICA) (Jutten and Hérault, 1991; Comon, 1994; Hyvärinen et al., 2001). Informally, we assume that our sources are doubly covered: they are the driving noise processes of AR processes which can not be directly observed due to the mixing with an unknown matrix. We will study our methods for this particular example and we



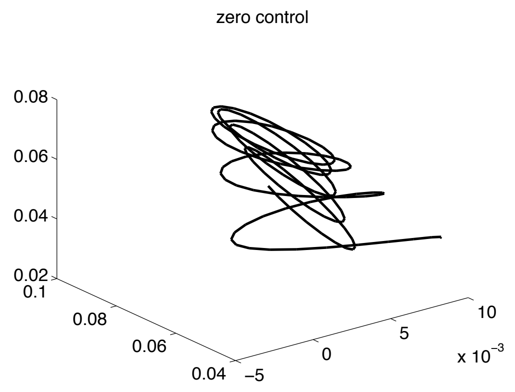
(a)



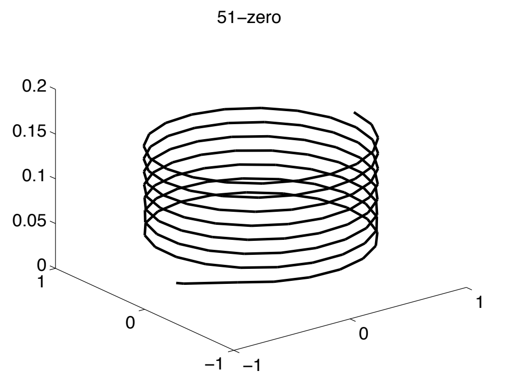
(b)



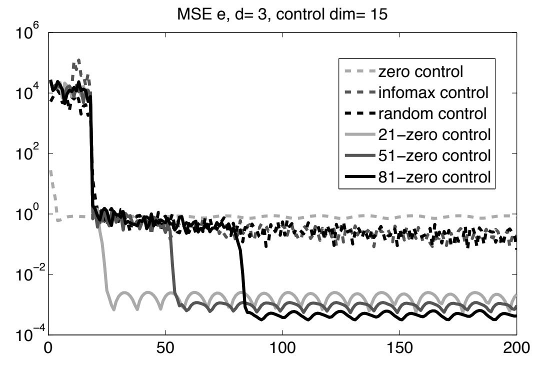
(c)



(d)



(e)



(f)

Figure 6: Different control strategies for non i.i.d. noise. (a): original noise. (b-e): estimated noise using random, infomax, zero, 51-zero strategy, respectively. (f): MSE for different control strategies. In (b-e), estimations of the first 51 time steps are not shown.

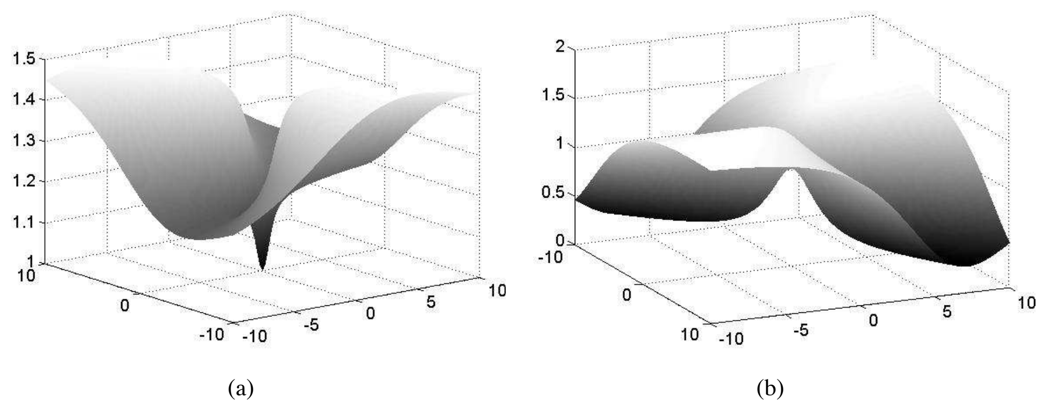


Figure 7: Negative logarithm of the objective function for the joint parameter and noise estimation task for different **K** matrices. (a) the minimum point is in zero, (b) the minimum point is on the boundary.

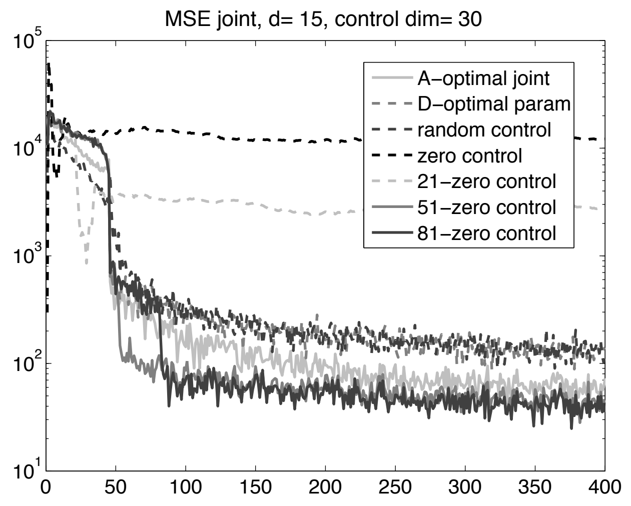


Figure 8: MSE of the joint parameter and noise estimation task. Comparisons between joint parameter and noise estimation using A-optimality principle, parameter estimation using D-optimality principle, random control, zero control and $\tau - zero$ control for different τ values. MSE values are averaged for 20 experiments.

.

assume that the processes can be exogenously controlled. Such processes are called ARX processes where X stands for letter x of the word eXogenous.

The 'classical' ICA task is as follows: we are given temporally i.i.d. signals $\mathbf{e}_t \in \mathbb{R}^d$ $(t = 1, 2, \ldots, T)$ with statistically independent coordinates. We are unable to measure them directly, but their mixture $\mathbf{r}_t = \mathbf{C}\mathbf{e}_t$ is available for observation, where $\mathbf{C} \in \mathbb{R}^{d \times d}$ is an unknown invertible matrix. The task is to measure the observable signals and to estimate both mixing matrix $\mathbf{C}$ and sources $\mathbf{e}_t$.

There are several generalizations of this problem. Hyvärinen (1998) has introduced an algorithm to solve the ICA task even if the hidden sources are AR processes, whereas Szabó and Lőrincz (2008) generalized this problem for ARX processes in the following way: Processes $\tilde{\mathbf{e}}_t \in \mathbb{R}^d$ are given and they are statistically independent for the different coordinates and are temporally i.i.d signals. They generate ARX process $\mathbf{s}_t$ by means of parameters $\tilde{\mathbf{F}} \in \mathbb{R}^{d\times d}, \tilde{\mathbf{B}} \in \mathbb{R}^{d\times c}$:

$$\mathbf{s}_{t+1} = \sum_{i=0}^{I} \tilde{\mathbf{F}}_i \mathbf{s}_{t-i} + \sum_{j=0}^{J} \tilde{\mathbf{B}}_j \mathbf{u}_{t+1-j} + \tilde{\mathbf{e}}_{t+1}. \tag{30}$$

We assume that ARX process $\mathbf{s}_t$ can not be observed directly, but its mixture

$$\mathbf{r}_t = \mathbf{C}\mathbf{s}_t \tag{31}$$

is observable, where mixing matrix $\mathbf{C} \in \mathbb{R}^{d\times d}$ is invertible, *but unknown*. Our task is to estimate the original independent processes also called sources, noises or 'causes', that is, $\tilde{\mathbf{e}}_t$, the hidden process $\mathbf{s}_t$ and mixing matrix $\mathbf{C}$ from observations $\mathbf{r}_t$. It is easy to see that (30) and (31) can be rewritten into the following form

$$\mathbf{r}_{t+1} = \sum_{i=0}^{I} \mathbf{C}\tilde{\mathbf{F}}_i \mathbf{C}^{-1} \mathbf{r}_{t-i} + \sum_{j=0}^{J} \mathbf{C}\tilde{\mathbf{B}} \mathbf{u}_{t+1-j} + \mathbf{C}\tilde{\mathbf{e}}_{t+1}. \tag{32}$$

Using notations $\mathbf{F}_i = \mathbf{C}\tilde{\mathbf{F}}_i\mathbf{C}^{-1}, \mathbf{B}_j = \mathbf{C}\tilde{\mathbf{B}}_j, \mathbf{e}_{t+1} = \mathbf{C}\tilde{\mathbf{e}}_{t+1}$, (32) takes the form of the model (1) that we are studying with function g being the identity matrix. The only difference is that in ICA tasks $\mathbf{e}_t$ is assumed to be non-Gaussian, whereas in our derivations we always used the Gaussian assumption. In our studies, however, we found that the different control methods can be useful for non-Gaussian noise, too. Furthermore, the Central Limit Theorem says that the mixture of the variables $\tilde{\mathbf{e}}_t$, that is, $\mathbf{C}\tilde{\mathbf{e}}_t$ approximates Gaussian distributions, provided that the number of mixed variables is large enough.

In our numerical experiments we studied the following special case:

$$\mathbf{r}_{t+1} \quad = \quad \mathbf{F}\mathbf{r}_t + \mathbf{B}\mathbf{u}_{t+1} + \mathbf{C}\mathbf{e}_{t+1},$$

where the dimension of the noise was 3, the dimension of the control was 15. Matrices $\mathbf{F}$ and $\mathbf{B}$ were generated the same way as before, matrix $\mathbf{C}$ was a randomly chosen orthogonal mixing, noise sources $\mathbf{e}_{t+1}$ were chosen from the benchmark tasks of the fastICA toolbox[1] (Hyvärinen, 1999). We compared 5 different control methods (zero control, D-optimal control developed for parameter estimation, random control, A-optimal control developed for joint estimation of parameters and noise, as well as the τ-zero control with τ=81 that we developed for noise estimation). Comparisons are executed by first estimating the noise ($\mathbf{C}\mathbf{e}_{t+1}$) for times $T = 1,\ldots,1000$ and then applying the JADE ICA algorithm (Cardoso, 1999) for the estimation of the noise components ($\mathbf{e}_{t+1}$). Estimation was executed in each fiftieth steps, but only for the preceding 300 elements of the time series.

The quality of separation is evaluated by means of the Amari-error (Amari et al., 1996) as follows. Let $\mathbf{W} \in \mathbb{R}^{d\times d}$ be the estimated demixing matrix, and let $\mathbf{G} := \mathbf{W}\mathbf{C} \in \mathbb{R}^{d\times d}$. In case

1. Found at `http://www.cis.hut.fi/projects/ica/fastica/`.

of perfect separation the matrix **G** is a scaled permutation matrix. The Amari-error evaluates the quality of the separation by measuring the 'distance' of matrix **G** from permutation matrices:

$$r(\mathbf{G}) = \frac{1}{2d(d-1)}\sum_{i=1}^{d}\left(\frac{\sum_{j=1}^{d}|G_{ij}|}{\max_j |G_{ij}|}-1\right)+\frac{1}{2d(d-1)}\sum_{j=1}^{d}\left(\frac{\sum_{i=1}^{d}|G_{ij}|}{\max_i |G_{ij}|}-1\right).$$

The Amari-error $r(G)$ has the property that $0 \leq r(\mathbf{G}) \leq 1$, and $r(\mathbf{G}) = 0$ if and only if **G** is a permutation matrix.

Results are shown in Fig. 9. In short, τ-zero control performs slightly better than the joint parameter and noise estimation using the A-optimality principle. We note that for A-optimality design one does not have to worry about the duration of the parameter estimation. The performance of the other methods were considerably worse, especially for early times.

Most importantly, we found that if we use Bayesian methods for noise separation in ARX problems then it is worth to interrogate the system actively to improve the efficiency of the estimation.

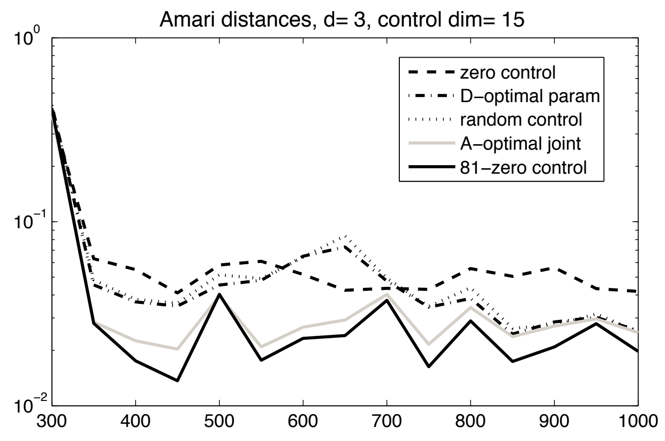


Figure 9: ARX-ICA experiment. Amari-error as a function of time for different control methods. Curves show the means of 100 experiments.

10.3 Model of the Furuta Pendulum

This section is concerned with more realistic simulations and investigate the robustness of our approach. We use a model of the Furuta pendulum (e.g., Yamakita et al., 1995) as our example, In this case, conditions of the theorems are not fulfilled and the task—in our formulation—can not be represented with a few matrices. In this simulation, we studied the D-optimality principle and compared it with the random control method. We were interested in the parameter estimation task in this example.

The two-segment Furuta pendulum problem (e.g., Yamakita et al., 1995; Gäfvert, 1998) was used. The pendulum has two links. Configuration of the pendulum is determined by the length of the links and by two angles. Dynamics of the pendulum are also determined by the different masses, that is, the masses of the links and the mass of the end effector as well as by the two motors, which are able to rotate the horizontal link and the swinging link in both directions. The angles of the horizontal and the swinging links are denoted by φ and θ, respectively (Fig. 10). Parameters

Name of parameter	Value	Unit	Notation
Angle of swinging link		rad	θ
Angle of horizontal link		rad	ϕ
Mass of horizontal link	0.072	kg	m_a
Mass of vertical link	0.00775	kg	m_p
Mass of the weight	0.02025	kg	M
Length of horizontal link	0.25	m	l_a
Length of vertical link	0.4125	m	l_p
Coulomb friction	0.015	Nm	τ_S
Coulomb stiction	0.01	Nm	τ_C
Maximal rotation speed for both links	2	$\frac{\text{rotation}}{s}$	
Approx. zero angular speed for swinging link	0.02	$\frac{rad}{s}$	$\dot{\phi}_\varepsilon$
Time intervals between interrogations	100	ms	
Maximum control value	0.05	Nm	δ

Table 2: Parameters of the Physical Model

of computer illustrations are provided in Table 2 for the sake of reproducibility. The state of the pendulum is given by ϕ, θ, $\dot{\phi}$ and $\dot{\theta}$. The magnitude of angular speeds $\dot{\phi}$ and $\dot{\theta}$ was restricted to 2 rotations/s, that is, to the interval $[-2\frac{\text{rot}}{\text{s}}, 2\frac{\text{rot}}{\text{s}}]$. For the equations of the dynamics and the details of the parameters, see, for example, the related technical report (Gäfvert, 1998).

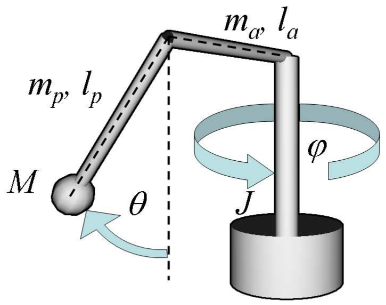


Figure 10: Furuta pendulum and notations of the different parameters. m: mass; l: length, M: mass of the end effector, subscript a: horizontal link, subscript p: swinging link, ϕ: angle of horizontal link, θ: angle of swinging link

.

The pendulum is a continuous dynamical system that we observe in discrete time steps. Furthermore, we assume that our observations are limited; we have only 144 low resolution and overlapping sensors for observing angles ϕ and θ. In each time step these sensors form our $\mathbf{r}(t) \in \mathbb{R}^{144}$ observations, which were simulated as follows: Space of angles ϕ and θ is $[0,2\pi) \times [0,2\pi)$, which we divided into $12 \times 12 = 144$ squared domains of equal sizes. There is a Gaussian sensor at the center of each domain. Each sensor gives maximal response 1 when angles θ and ϕ of the pendulum are in the center of the respective sensor, whereas the response decreased according to the Gaussian function. For example, for the i^{th} $(1 \leq i \leq 144)$ sensor characterized by angles θ_i, ϕ_i response y_i scaled as $y_i = \frac{1}{\sqrt{2\pi}\sigma}\exp(-\frac{(\theta-\theta_i)^2+(\phi-\phi_i)^2}{2\sigma^2})$ and the value of σ was set to 1.58 in radians. Sensors were crude but noise-free; no noise was added to the sensory outputs. The inset at label 4 of Fig. 11 shows the outputs of the sensors in a typical case. Sensors satisfied periodic boundary conditions; if sensor S was centered around zero degree in any of the directions, then it sensed both small (around 0 radian) and large (around 2π radian) angles. We note that the outputs of the 144 domains are arranged for purposes of visualization; the underlying geometry of the sensors is hidden for the learning algorithm.

We observed these $\mathbf{r}_t \in \mathbb{R}^{144}$ quantities and then calculated the $\mathbf{u}_{t+1} \in \mathbb{R}^2$ D-optimal control using the algorithm of Table 1, where we approximated the pendulum with the model $\tilde{\mathbf{r}}_{t+1} = \mathbf{F}\mathbf{r}_t + \mathbf{B}\mathbf{u}_{t+1}$, $\mathbf{F} \in \mathbb{R}^{144\times 144}$, $\mathbf{B} \in \mathbb{R}^{144\times 2}$. Components of vector $\mathbf{u}_{t+1}$ controlled the 2 actuators of the angles separately. Maximal magnitude of each control signal was set to 0.05 Nm. Clearly we do not know the best parameters for $\mathbf{F}$ and $\mathbf{B}$ in this case, so we studied the prediction error and the number of visited domains instead. This procedure is detailed below.

First, we note that the angle of the swinging link and the angular speeds are important from the point of view of the prediction of the dynamics, whereas the angle of the horizontal link can be neglected. Thus, for the investigation of the learning process, we used the 3D space determined by $\dot{\phi}, \theta$ and $\dot{\theta}$. As was mentioned above, angular speeds were restricted to the $[-2\frac{\text{rot}}{\text{s}}, 2\frac{\text{rot}}{\text{s}}]$ domain. We divided each angular speed domain into 12 equal regions. We also used the 12-fold division of angle θ. Counting the domains, we had $12 \times 12 \times 12 = 1,728$ rectangular block shaped domains. Our algorithm provides estimations for $\hat{\mathbf{F}}_t$ and $\hat{\mathbf{B}}_t$ in each instant. We can use them to compute the predicted observation vector $\hat{\mathbf{r}}_{t+1} = \hat{\mathbf{F}}_t\mathbf{r}_t + \hat{\mathbf{B}}_t\mathbf{u}_{t+1}$. An example is shown in inset at label 4 of Fig. 11. We investigated the $\|\mathbf{r}_{t+1} - \hat{\mathbf{r}}_{t+1}\|$ prediction error (see Fig. 11) *cumulated over these domains* as follows. For each of the 1,728 domain, we set the initial error value at 30, a value somewhat larger than the maximal error we found in the computer runs. Therefore the cumulated error at start was $1,728 \times 30 = 51,840$.

The D-optimal algorithm does two things simultaneously: (i) it explores new domains, and (ii) it decreases the errors in the domains already visited. Thus, we measured the cumulated prediction errors during learning and corrected the estimation at each step. So, if our cumulated error estimation at time t was $e(t) = \sum_{k=1}^{1,728} e_k(t)$ and the pendulum entered the i^{th} domain at time $t+1$, then we set $e_k(t+1) = e_k(t)$ for all $k \neq i$ and $e_i(t+1)$ at $e_i(t+1) = \|\mathbf{r}_{t+1} - \hat{\mathbf{r}}_{t+1}\|$. Then we computed the new cumulated prediction error, that is, $e(t+1) = \sum_{k=1}^{1,728} e_k(t+1)$.

We compared the random and the D-optimality interrogation schemes. We show two sets of figures, Figs. 12a and 12b, as well as Figs. 12c and 12d. The upper set depicts the results for the full set of the 1,728 domains. It is hard for the random control to guide the pendulum to the upper domain, so we also investigated how the D-optimal control performs here. We computed the performance for cases when the swinging link was above vertical, that is for 864 domains (Figs. 12c and 12d).

For the full domain the number of visited domains is 456 (26%) and 818 (47%) for the random control and the D-optimal control, respectively after 5,000 control steps (Fig. 12a). The error drops by 13,390 (26%) and by 24,040 (46%), respectively (Fig. 12b). While the D-optimal controlled pendulum visited more domains and achieved smaller errors, the domain-wise estimation error is about the same for the domains visited; both methods gained about 29.4% per domains.

We can compute the same quantities for the upper domains as well. The number of visited upper domains is 9 and 114 for the random control and for the D-optimal control, respectively (Fig. 12c). The decrease of error is 265 and 3,342, respectively (Fig. 12d). In other words, D-optimal control gained 29.3% in each domain on average, whereas random control, on average, gained 29.4 %,

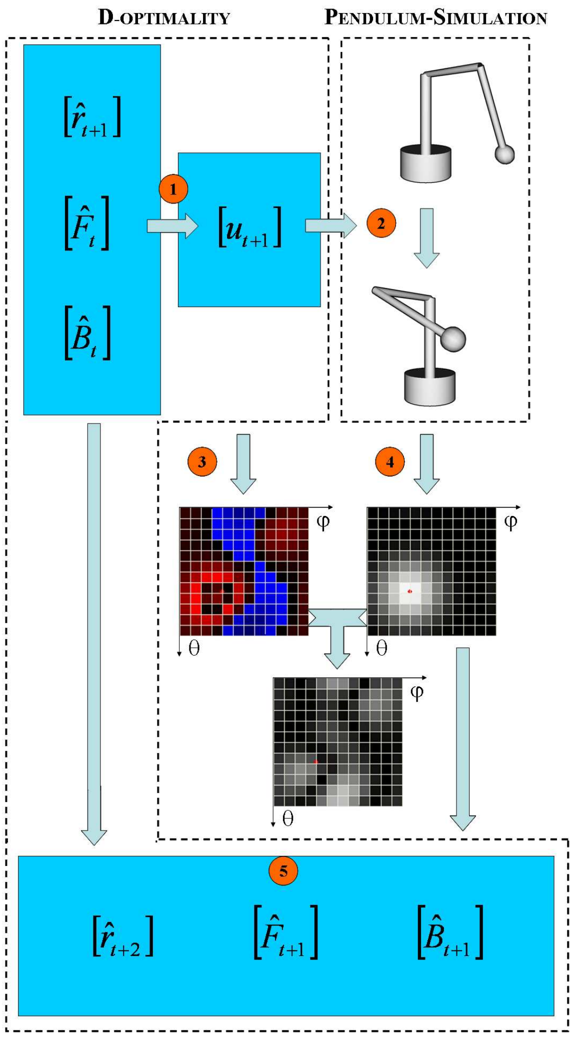


Figure 11: Scheme of D-optimal interrogation. (1) Control $\mathbf{u}_{t+1}$ is computed from D-optimal principle, (2) control acts upon the pendulum, (3) signals predicted before control step, (4) sensory information after control step. Difference between (3) and (4) is used for the computation of the cumulated prediction error. (5) Parameters were updated according to the pseudocode of Table 1. For more details, see text.

which are very close to the previous values in both cases. In this experiment D-optimal control gains more information concerning the system to be identified by visiting new domains.

This observation is further emphasized by the following data: The D-optimal algorithm discovered 37 new domains in the last 500 steps of the 5,000 step experiment. Out of these 37 domains, 20 (17) were discovered in the lower (upper) domain. By contrast, the random algorithm discovered 9 domains, out of which 5 (4) was in the lower (upper) domain. That is, D-optimality principle has a similar (roughly fourfold) lead in both the upper and lower domains, although the complexity of the task is different and the relative number of available volumes is also different in these two domains.

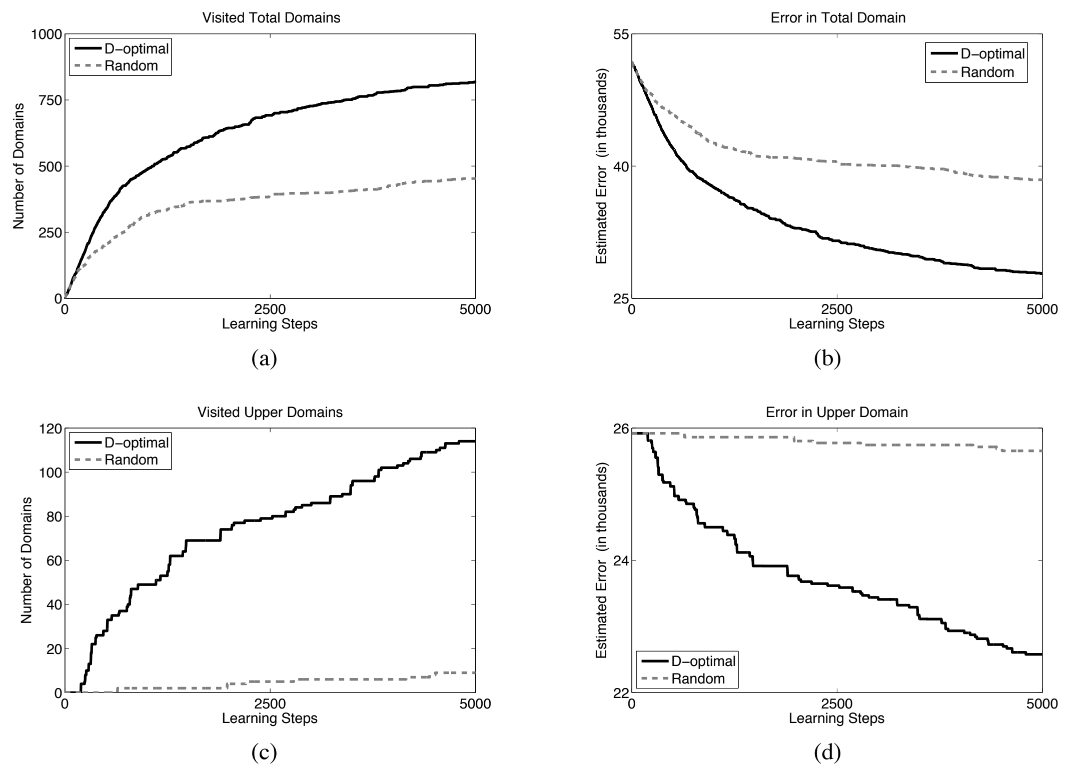


Figure 12: Furuta experiments driven by random and D-optimality controls. Solid (dotted) line: D-optimal (random) case. (a-b): Number of domains is 1728. (a): Visited domains, (b): upper bound for cumulated estimation error in all domains, (c-d): Number of domains is 864. (c): visited domains for swing angle above horizontal, (d): upper bound for cumulated estimation error for domains with swing angle above vertical. For more details, see text.

11. Discussion and Conclusions

We have treated the identification problem of recurrent neural networks as defined by the model detailed in (1). We applied active learning to solve this task. In particular, we studied the learning properties of the online A-optimality and D-optimality principles for parameter and noise estimations. We note that the D-optimal interrogation scheme is also called InfoMax control in the literature by Lewi et al. (2007). This name originates from the cost function that optimizes the mutual information.

In the generalized linear model (GLM) used by Lewi et al. (2007) $\mathbf{r}_{t+1}$ is drawn from an exponential family distribution with link function g and

$$E[\mathbf{r}_{t+1}] = g\left(\sum_{i=0}^{I} \mathbf{F}_i \mathbf{r}_{t-i} + \sum_{j=0}^{J} \mathbf{B}_j \mathbf{u}_{t+1-j}\right)$$

expected value. This model can be rewritten as

$$\mathbf{r}_{t+1} = g\left(\sum_{i=0}^{I} \mathbf{F}_i \mathbf{r}_{t-i} + \sum_{j=0}^{J} \mathbf{B}_j \mathbf{u}_{t+1-j}\right) + \mathbf{e}_{t+1}, \tag{33}$$

where $\{\mathbf{e}_t\}$ is a special noise process with $\mathbf{0} \in \mathbb{R}^d$ mean. The elements of this error series are independent of each other, but usually they are *not* identically distributed. The authors modeled spiking neurons and assumed that the main source of the noise is this spiking, which appears at the output of the neurons and adds linearly to the neural activity. They investigated the case in which the observed quantity $\mathbf{r}_t$ had a Poisson distribution. Unfortunately, in this model Bayesian equations become intractable and the estimation of the posterior may be corrupted, because the distribution is projected to the family of normal distributions at each instant. A serious problem with this approach is that the extent of the information loss caused by this approximation is not known. Our stochastic RNN model

$$\mathbf{r}_{t+1} = g\left(\sum_{i=0}^{I} \mathbf{F}_i \mathbf{r}_{t-i} + \sum_{j=0}^{J} \mathbf{B}_j \mathbf{u}_{t+1-j} + \mathbf{e}_{t+1}\right),$$

differs only slightly from the GLM model of (33), but it has considerable advantages, as we discuss it later. Note that the two models assume the same form if function g is the identity matrix and if the noise distribution is normal.

Our model is very similar to the well-studied non-linear Wiener (Celka et al., 2001) and Hammerstein (Pearson and Pottmann, 2000; Abonyi et al., 2000) systems. The Hammerstein model develops according to the following dynamics

$$\mathbf{r}_{t+1} = \sum_{i=0}^{I} \mathbf{F}_i \mathbf{r}_{t-i} + \sum_{j=0}^{J} \mathbf{B}_j g(\mathbf{u}_{t+1-j}) + \mathbf{e}_{t+1}.$$

The dynamics of the Wiener system is

$$\mathbf{r}_{t+1} = g\left(\sum_{i=0}^{I} \mathbf{F}_i g^{-1}(\mathbf{r}_{t-i}) + \sum_{j=0}^{J} \mathbf{B}_j \mathbf{u}_{t+1-j} + \mathbf{e}_{t+1}\right),$$

where we assumed that function g is invertible.

The Wiener and the Hammerstein systems have found applications in a broad range of areas, including financial predictions to the modeling of chemical processes. These models are special cases of non-linear ARX (NARX) models (Billings and Leontaritis, 1981). They are popular, because they belong to the simplest non-linear systems. Using block representation, they are simply the compositions of a static non-linear function and a dynamic ARX system. As a result, their properties can be investigated in a relatively simple manner and they are still able to model a large class of sophisticated non-linear phenomena.

Interesting comparisons between Wiener and Hammerstein systems can be found in Bai (2002), Aguirre et al. (2005), and Haber and Unbehauen (1990). We note that our Bayesian interrogation methods can be easily transferred to both the Wiener and to the Hammerstein systems.

Bayesian designs of different kinds were derived for the linear regression problem by Verdinelli (2000):

$$\mathbf{y} = \mathbf{X}\boldsymbol{\theta} + \mathbf{e}, \tag{34}$$
$$P(\mathbf{e}) = \mathcal{N}_{\mathbf{e}}(0, \sigma^2 \mathbf{I}).$$

This problem is similar to ours ((3)-(5)), but while the goal of Verdinelli (2000) was to find an optimal design for the explanatory variables $\boldsymbol{\theta}$, we were concerned with the parameter ($\mathbf{X}$ in 34) and

	Parameter	Noise	Joint
D-optimal	$\max_{\mathbf{u}_{t+1}\in\mathcal{U}} \mathbf{x}_{t+1}^T\mathbf{K}_t^{-1}\mathbf{x}_{t+1}$	$\min_{\mathbf{u}_{t+1}\in\mathcal{U}} \mathbf{x}_{t+1}^T\mathbf{K}_t^{-1}\mathbf{x}_{t+1}$	N/A
A-optimal	$\max_{\mathbf{u}_{t+1}\in\mathcal{U}} \frac{\mathbf{x}_{t+1}^T\mathbf{K}_t^{-1}\mathbf{K}_t^{-1}\mathbf{x}_{t+1}}{1+\mathbf{x}_{t+1}^T\mathbf{K}_t^{-1}\mathbf{x}_{t+1}}$	$\min_{\mathbf{u}_{t+1}\in\mathcal{U}} \mathbf{x}_{t+1}^T\mathbf{K}_t^{-1}\mathbf{x}_{t+1}$	$\max_{\mathbf{u}_{t+1}\in\mathcal{U}} \frac{1+\mathbf{x}_{t+1}^T\mathbf{K}_t^{-1}\mathbf{K}_t^{-1}\mathbf{x}_{t+1}}{1+\mathbf{x}_{t+1}^T\mathbf{K}_t^{-1}\mathbf{x}_{t+1}}$

Table 3: Cost functions in the parameter, noise, and in the joined parameter noise estimation task.

the noise (**e**) estimation tasks. In Verdinelli's paper inverted gamma prior and vector-valued normal distribution were assumed on the isotropic noise and on the explanatory variables, respectively. By contrast, we were interested in the matrix-valued coefficients and in general, non-isotropic noises. We used matrix-valued normal distribution for the coefficients, and in the D-optimal case we applied inverted Wishart distribution for the covariance matrix of the noise. Due to the properties of the inverted Wishart distribution, the noise covariance matrix is not restricted to the isotropic form. However, in the A-optimal case, we kept the derivations simple and we applied product of inverted gamma distributions for the covariance matrix as the conjugate prior.

The Bayesian online learning framework allowed us to derive analytic results for the myopic optimization of the parameters as well as the driving noise. In the *D-optimal* case the optimal interrogation strategies for parameter (14) and noise estimation (25) appeared in attractive, intriguingly simple quadratic forms. We have shown that these two tasks are incompatible with each other. Parameter and noise estimations require the maximization and the minimization of expression $\mathbf{x}_{t+1}^T\mathbf{K}_t^{-1}\mathbf{x}_{t+1}$, respectively. We have shown also that D-optimality can not be applied to the joined estimation of parameters and noise, because the corresponding cost function is constant.

For the *A-optimality* principle, we found that the objective of the noise estimation task is identical to that of the D-optimality principle. In this case, we were able to derive sensible results for the joined estimation of parameters and noise, and received hyperbolic optimization task. We also received a similar hyperbolic optimization task for the parameter estimation problem. The optimization of this task is non-trivial. For a simple hyper-cube domain we put forth a heuristics based on the simplex method. The different cost functions are summarized in Table 3.

We found empirically in the *parameter learning* task that the different objectives—that is, the minimization of $|(\mathbf{K}_t+\mathbf{x}_{t+1}\mathbf{x}_{t+1}^T)^{-1}|$ and $tr[(\mathbf{K}_t+\mathbf{x}_{t+1}\mathbf{x}_{t+1}^T)^{-1}]$, the results of the D-optimality and A-optimality principles, respectively—exhibit similar performances (Fig. 1, Fig. 2). However, D-optimality has slightly better performance and it is easier to compute, since we need to solve a quadratic problem only as opposed to a hyperbolic one. One of the reasons for the similar performance could be that the corresponding matrices are positive definite and thus they are diagonally dominant. In this case both the trace and the determinant are dominated by diagonal elements of the matrices.

In the *noise estimation* task, however, cost functions of the A- and D-optimality principles are the same. The main difficulty here is the non-myopic optimization of this cost function (Fig. 3, Fig. 4, Fig. 5, Fig. 6). The problem of non-greedy optimization of the full task has been left open for both the A-optimality and the D-optimality principles. For the noise estimation task, we suggested a heuristic solution that we called τ-infomax noise interrogation. Numerical experiments served to show that τ-infomax noise interrogation overcomes several other estimation strategies. The novel τ-infomax noise interrogation uses the D-optimal interrogation of Table 1 up to τ-steps, and applies the noise estimation control detailed in (25) afterwards. This heuristics decreases the estimation error of the coefficients of matrices $\mathbf{F}$ and $\mathbf{B}$ up to time τ and thus—upon turning off the explorative D-optimization—tries to minimize the estimation error of the value of the noise at time $\tau+1$. We introduced the τ-zero interrogation scheme and showed that it is a good approximation of the τ-infomax noise scheme for large τ values.

In the *joint noise parameter estimation* task the D-optimal principle leads to a meaningless constant valued cost function. The A-optimal objective is a hyperbolic programming task, which is difficult to optimize. Still, its myopic optimization gave us better results than the myopic D-optimal objective derived for parameter estimation only. Interestingly, myopic A-optimal interrogations behaved similarly to the non-myopic τ-zero control (D-optimal parameter estimation up to τ steps, then zero control) with emergent automatic τ selection. However, these myopic results were somewhat worse than the non-myopic results from the τ-zero interrogation, when τ was larger than an appropriate threshold (Fig. 8).

In the field of active-learning, non myopic optimization is one of the most challenging tasks so most studies are concerned with greedy optimization only. Still, greedy optimization has produced valuable results in many cases. A few studies are concerned with non-greedy optimization of active learning, but only for certain special cases (Krause and Guestrin, 2007; Rangarajan et al., 2007). This field is in a rather early stage at the moment.

We illustrated the working of the algorithm on artificial databases both for the parameter estimation problem and for the noise estimation task. In the first set of experiments the database satisfied the conditions of our theorems. We studied the robustness of the algorithm and studied the noise estimation problem for a situation in which the conditions of our theorems were not satisfied, namely when the noise was neither Gaussian nor i.i.d. In particular, we studied the ARX problem family, when the hidden driving sources are non-Gaussian and statistically independent i.i.d. processes. We found that active control can speed-up the learning process. Function g was the identity function in these experiments.

We have also started to characterize the algorithm for a problem closer to reality. We chose the two-link Furuta pendulum task in these studies. We used a crude discretization for the pendulum, where the underlying dynamics and the low-dimensional nature of the problem are both hidden. Clearly we could express neither the ideal parameters for this case nor the noise that arose as a result of the disctretization. Thus we have studied the volume explored by the D-optimality method as well as the magnitude of the prediction errors.

The pendulum problem demonstrated that D-optimality maximizes mutual information by exploring new areas without significant compromise in the precision of estimation in the visited domains. The discovery rate is in favor of the D-optimality algorithm, which has a significant lead in both the frequently visited and the rarely visited domains, although the task is different and the relative number of available volumes is also different in these domains.

Our method treats the identification problem of non-linear ARX systems. We plan to generalize the method to NARMAX systems in the future. Such systems have several application fields, including financial modeling and the modeling of gas turbines (Chiras et al., 2001). One may try to apply these principles in a broad variety of fields, including selective laser chemistry Rabitz (2003), or the analysis of brain signals ‘controlled’ by visual inputs, for example, in brain-computer interfaces Vaughan et al. (2003).

Finally, it seems desirable to determine the conditions under which the algorithms derived from the optimality principles are both consistent and efficient. The tractable form of our approximation-free results is promising in this respect.

Acknowledgments

We thank the anonymous reviewers for their valuable comments. We are grateful to György Flórea and László A. Jeni for running the pendulum simulations, and to Gábor Szirtes for his valuable comments. This research has been supported by the EC NEST 'Perceptual Consciousness: Explication and Testing' grant under contract 043261. Opinions and errors in this manuscript are the author's responsibility, they do not necessarily reflect the opinions of the EC or other project members.

Appendix A.

In this section we provide the technical details of our derivations.

A.1 Proof of Lemma 4.1

It is easy to show that the following equations hold:

$$\begin{aligned}\mathcal{N}_{\mathbf{y}}(\mathbf{Ax},\mathbf{V})\mathcal{N}_{\mathbf{A}}(\mathbf{M},\mathbf{V},\mathbf{K}) &= \mathcal{N}_{\mathbf{A}}(\mathbf{M}^+,\mathbf{V},\mathbf{xx}^T+\mathbf{K})\mathcal{N}_{\mathbf{y}}(\mathbf{Mx},\mathbf{V},\gamma),\\ \mathcal{N}_{\mathbf{A}}(\mathbf{M},\mathbf{V},\mathbf{K})\mathcal{IW}_{\mathbf{V}}(\mathbf{Q},n) &= \mathcal{IW}_{\mathbf{V}}(\mathbf{Q}+\mathbf{H},n+m)\,\mathcal{T}_{\mathbf{A}}(\mathbf{Q},n,\mathbf{M},\mathbf{K}),\end{aligned}$$

where $\mathbf{M}^+ = (\mathbf{MK}+\mathbf{yx}^T)(\mathbf{xx}^T+\mathbf{K})^{-1}$, $\gamma = 1-\mathbf{x}^T(\mathbf{xx}^T+\mathbf{K})^{-1}\mathbf{x}$, $\mathbf{H} = (\mathbf{A}-\mathbf{M})\mathbf{K}(\mathbf{A}-\mathbf{M})^T$ for the sake of brevity. Then we have

$$\mathcal{N}_{\mathbf{y}}(\mathbf{Mx},\mathbf{V},\gamma)\mathcal{IW}_{\mathbf{V}}(\mathbf{Q},n) = \mathcal{IW}_{\mathbf{V}}(\mathbf{Q}+(\mathbf{y}-\mathbf{Mx})\gamma(\mathbf{y}-\mathbf{Mx})^T,n+1)\mathcal{T}_{\mathbf{y}}(\mathbf{Q},n,\mathbf{Mx},\gamma),$$

and the statement of the lemma follows.

A.2 Proof of Lemma 4.2

Let $vec(\mathbf{A})$ denote a vector of dm dimensions where the $(d(i-1)+1)^{th},\ldots,(id)^{th}$ $(1\le i\le m)$ elements of this vector are equal to the elements of the i^{th} column of matrix $\mathbf{A}\in\mathbb{R}^{d\times m}$ in the appropriate order. Let $\otimes$ denote the Kronecker-product. It is known that for $P(\mathbf{A}) = \mathcal{N}_{\mathbf{A}}(\mathbf{M},\mathbf{V},\mathbf{K})$, $P(vec(\mathbf{A})) = \mathcal{N}_{vec(\mathbf{A})}(vec(\mathbf{M}),\mathbf{V}\otimes\mathbf{K}^{-1})$ holds (Minka, 2000). Using the well-known formula for the entropy of a multivariate and normally distributed variable (Cover and Thomas, 1991) and applying the relation $|\mathbf{V}\otimes\mathbf{K}^{-1}| = |\mathbf{V}|^m/|\mathbf{K}|^d$, we have that

$$H(\mathbf{A};\mathbf{V}) = \frac{1}{2}\ln|\mathbf{V}\otimes\mathbf{K}^{-1}| + \frac{dm}{2}\ln(2\pi e) = \frac{m}{2}\ln|\mathbf{V}| - \frac{d}{2}\ln|\mathbf{K}| + \frac{dm}{2}\ln(2\pi e).$$

By exploiting certain properties of the Wishart distribution, we can compute the entropy of distribution $\mathcal{IW}_{\mathbf{V}}(\mathbf{Q},n)$. The density of the Wishart distribution is defined by

$$\mathcal{W}_{\mathbf{V}}(\mathbf{Q},n) = \frac{1}{Z_{n,d}}|\mathbf{V}|^{(n-d-1)/2}\left|\frac{\mathbf{Q}^{-1}}{2}\right|^{n/2}\exp\left(-\frac{1}{2}tr(\mathbf{VQ}^{-1})\right).$$

Let Ψ denote the digamma function, and let $f_{1,3}(d,n) = -\sum_{i=1}^{d}\Psi(\frac{n+1-i}{2})) - d\ln 2$. Replacing $\mathbf{V}^{-1}$ with $\mathbf{S}$, we have for the Jacobian that $|\frac{d\mathbf{V}}{d\mathbf{S}}| = |\frac{d\mathbf{S}^{-1}}{d\mathbf{S}}| = |\mathbf{S}|^{-(d+1)}$ (Gupta and Nagar, 1999). To proceed we use that $E_{\mathcal{W}_{\mathbf{S}}(\mathbf{Q},n)}\mathbf{S} = n\mathbf{Q}$, and $E_{\mathcal{W}_{\mathbf{S}}(\mathbf{Q},n)} \ln|\mathbf{S}| = \ln|\mathbf{Q}| - f_{1,3}(d,n)$, (Beal, 2003) and substitute them into $E_{\mathcal{IW}_{\mathbf{V}}(\mathbf{Q},n)} \ln|\mathbf{V}|$, and $E_{\mathcal{IW}_{\mathbf{V}}(\mathbf{Q},n)} tr(\mathbf{Q}\mathbf{V}^{-1})$:

$$\begin{aligned}
E_{\mathcal{IW}_{\mathbf{V}}(\mathbf{Q},n)} \ln|\mathbf{V}| &= \int \frac{1}{Z_{n,d}} \frac{1}{|\mathbf{V}|^{(d+1)/2}} \left|\frac{\mathbf{V}^{-1}\mathbf{Q}}{2}\right|^{n/2} \exp\left(-\frac{1}{2}tr(\mathbf{V}^{-1}\mathbf{Q})\right) \ln|\mathbf{V}| d\mathbf{V}, \\
&= -\int \frac{1}{Z_{n,d}} |\mathbf{S}|^{(d+1)/2} \left|\frac{\mathbf{S}\mathbf{Q}}{2}\right|^{n/2} \exp\left(-\frac{1}{2}tr(\mathbf{S}\mathbf{Q})\right) \ln|\mathbf{S}||\mathbf{S}|^{-d-1} d\mathbf{S}, \\
&= -\int \frac{1}{Z_{n,d}} |\mathbf{S}|^{(n-d-1)/2} \left|\frac{\mathbf{Q}}{2}\right|^{n/2} \exp\left(-\frac{1}{2}tr(\mathbf{S}\mathbf{Q})\right) \ln|\mathbf{S}| d\mathbf{S}, \\
&= -E_{\mathcal{W}_{\mathbf{S}}(\mathbf{Q}^{-1},n)} \ln|\mathbf{S}|, \\
&= \ln|\mathbf{Q}| + f_{1,3}(d,n). \qquad (35)
\end{aligned}$$

One can also show that

$$\begin{aligned}
E_{\mathcal{IW}_{\mathbf{V}}(\mathbf{Q},n)} tr(\mathbf{Q}\mathbf{V}^{-1}) &= \int \frac{1}{Z_{n,d}} \frac{1}{|\mathbf{V}|^{(d+1)/2}} \left|\frac{\mathbf{V}^{-1}\mathbf{Q}}{2}\right|^{n/2} \exp\left(-\frac{1}{2}tr(\mathbf{V}^{-1}\mathbf{Q})\right) tr(\mathbf{Q}\mathbf{V}^{-1}) d\mathbf{V}, \\
&= \int \frac{1}{Z_{n,d}} |\mathbf{S}|^{(d+1)/2} \left|\frac{\mathbf{S}\mathbf{Q}}{2}\right|^{n/2} \exp\left(-\frac{1}{2}tr(\mathbf{S}\mathbf{Q})\right) tr(\mathbf{Q}\mathbf{S})|\mathbf{S}|^{-d-1} d\mathbf{S}, \\
&= \int \frac{1}{Z_{n,d}} |\mathbf{S}|^{(n-d-1)/2} \left|\frac{\mathbf{Q}}{2}\right|^{n/2} \exp\left(-\frac{1}{2}tr(\mathbf{S}\mathbf{Q})\right) tr(\mathbf{Q}\mathbf{S}) d\mathbf{S}, \\
&= E_{\mathcal{W}_{\mathbf{S}}(\mathbf{Q}^{-1},n)} tr(\mathbf{Q}\mathbf{S}), \\
&= tr(\mathbf{Q}\mathbf{Q}^{-1}n) = nd. \qquad (36)
\end{aligned}$$

We calculate the entropy of stochastic variable $\mathbf{V}$ with distribution $\mathcal{IW}_{\mathbf{V}}(\mathbf{Q},n)$. It follows from Eq. (35) and Eq. (36) that

$$\begin{aligned}
H(\mathbf{V}) &= -E_{\mathcal{IW}_{\mathbf{V}}(\mathbf{Q},n)} \left[-\ln(Z_{n,d}) + \frac{n}{2}\ln\left|\frac{\mathbf{Q}}{2}\right| - \frac{n+d+1}{2}\ln|\mathbf{V}| - \frac{1}{2}tr(\mathbf{V}^{-1}\mathbf{Q})\right], \\
&= \ln(Z_{n,d}) - \frac{n}{2}\ln\left|\frac{\mathbf{Q}}{2}\right| + \frac{n+d+1}{2}\left[\ln|\mathbf{Q}| - \sum_{i=1}^{d}\Psi(\frac{n+1-i}{2})) - d\ln 2\right] + \frac{nd}{2}, \\
&= \frac{d+1}{2}\ln|\mathbf{Q}| + f_{1,4}(d,n),
\end{aligned}$$

where $f_{1,4}(d,n)$ depends only on d and n.

Given the results above, we complete the computation of entropy $H(\mathbf{A},\mathbf{V})$ as follows:

$$\begin{aligned}
H(\mathbf{A},\mathbf{V}) &= H(\mathbf{A}|\mathbf{V}) + H(\mathbf{V}) = H(\mathbf{V}) + \int d\mathbf{V}\, \mathcal{IW}_{\mathbf{V}}(\mathbf{Q},n) H(\mathbf{A};\mathbf{V}), \\
&= \int d\mathbf{V}\, \mathcal{IW}_{\mathbf{V}}(\mathbf{Q},n) \left(\frac{m}{2}\ln|\mathbf{V}| - \frac{d}{2}\ln|\mathbf{K}| + \frac{dm}{2}\ln(2\pi e)\right) + H(\mathbf{V}), \\
&= -\frac{d}{2}\ln|\mathbf{K}| + \frac{dm}{2}\ln(2\pi e) + \frac{m}{2}[\ln|\mathbf{Q}| + f_{1,3}(d,n)] + \frac{d+1}{2}\ln|\mathbf{Q}| + f_{1,4}(d,n), \\
&= -\frac{d}{2}\ln|\mathbf{K}| + (\frac{m+d+1}{2})\ln|\mathbf{Q}| + f_{1,1}(d,n).
\end{aligned}$$

This is exactly what was claimed in Lemma 4.2.

A.3 Proof of Lemma 5.1

The proof is analogous to the D-optimality case. We need the following lemma:

Lemma A.1 *Let* $\mathbf{V}$ *diagonal positive definite matrix. Let* $\mathbf{X}_{(i,:)}$ *denote the* i^{th} *row of matrix* $\mathbf{X}$. *Let* $\eta_i = (\mathbf{A}_{(i,:)} - \mathbf{M}_{(i,:)})\mathbf{K}(\mathbf{A}_{(i,:)} - \mathbf{M}_{(i,:)})^T$, $\forall i = 1,\dots,d$. $\boldsymbol{\eta} \in \mathbb{R}^d$. *Then the following statement holds:*

$$\mathcal{N}_{\mathbf{A}}(\mathbf{M},\mathbf{V},\mathbf{K})\mathcal{PIG}_{\mathbf{V}}(\boldsymbol{\alpha},\boldsymbol{\beta}) = \mathcal{PIG}_{\mathbf{V}}(\boldsymbol{\alpha}+m/2,\boldsymbol{\beta}+\boldsymbol{\eta}/2)\prod_{i=1}^{d}\mathcal{T}_{\mathbf{A}_{(i,:)}}(\beta_i, 2\alpha_i, \mathbf{M}_{(i,:)}, \frac{\mathbf{K}}{2}).$$

Proof:

$$\begin{aligned}
&\mathcal{N}_{\mathbf{A}}(\mathbf{M},\mathbf{V},\mathbf{K})\mathcal{PIG}_{\mathbf{V}}(\boldsymbol{\alpha},\boldsymbol{\beta}) = \\
&= \frac{|\mathbf{K}|^{d/2}}{|2\pi\mathbf{V}|^{m/2}}\exp(-\frac{1}{2}tr(\mathbf{V}^{-1}(\mathbf{A}-\mathbf{M})\mathbf{K}(\mathbf{A}-\mathbf{M})^T))\prod_{i=1}^{d}\frac{\beta_i^{\alpha_i}}{\Gamma(\alpha_i)}v_i^{-\alpha_i-1}\exp(-\frac{\beta_i}{v_i}),\\
&= \frac{|\mathbf{K}|^{d/2}}{\prod_{i=1}^{d}(2\pi)^{m/2}v_i^{m/2}}\exp(-\frac{1}{2}\sum_{i=1}^{d}\frac{1}{v_i}\eta_i)\prod_{i=1}^{d}\frac{\beta_i^{\alpha_i}}{\Gamma(\alpha_i)}v_i^{-\alpha_i-1}\exp(-\frac{\beta_i}{v_i}),\\
&= \frac{|\mathbf{K}|^{d/2}}{\prod_{i=1}^{d}(2\pi)^{m/2}}\prod_{i=1}^{d}\frac{\beta_i^{\alpha_i}}{\Gamma(\alpha_i)}v_i^{-\alpha_i-m/2-1}\exp(-\frac{\beta_i+\eta_i/2}{v_i}),\\
&= \frac{|\mathbf{K}|^{d/2}}{\prod_{i=1}^{d}(2\pi)^{m/2}}\prod_{i=1}^{d}\frac{\beta_i^{\alpha_i}}{\Gamma(\alpha_i)}\frac{(\beta_i+\eta_i/2)^{\alpha_i+m/2}}{(\beta_i+\eta_i/2)^{\alpha_i+m/2}}\frac{\Gamma(\alpha_i+m/2)}{\Gamma(\alpha_i+m/2)},v_i^{-\alpha_i-m/2-1}\exp(-\frac{\beta_i+\eta_i/2}{v_i}),\\
&= \frac{|\mathbf{K}|^{d/2}}{\prod_{i=1}^{d}(2\pi)^{m/2}}\prod_{i=1}^{d}\frac{\beta_i^{\alpha_i}\Gamma(\alpha_i+m/2)/\Gamma(\alpha_i)}{(\beta_i+\eta_i/2)^{\alpha_i+m/2}}\prod_{i=1}^{d}\frac{(\beta_i+\eta_i/2)^{\alpha_i+m/2}}{\Gamma(\alpha_i+m/2)}v_i^{-\alpha_i-m/2-1}\exp(-\frac{\beta_i+\eta_i/2}{v_i}),\\
&= \prod_{i=1}^{d}\frac{|\mathbf{K}|^{1/2}}{(2\pi)^{m/2}}\frac{\Gamma(\alpha_i+m/2)}{\Gamma(\alpha_i)}\frac{\beta_i^{\alpha_i}}{(\beta_i+\eta_i/2)^{\alpha_i+m/2}}\mathcal{PIG}_{\mathbf{v}}(\boldsymbol{\alpha}+m/2,\boldsymbol{\beta}+\boldsymbol{\eta}/2),\\
&= \prod_{i=1}^{d}\frac{|\frac{\mathbf{K}}{2}|^{1/2}}{\pi^{m/2}}\frac{\Gamma(\alpha_i+m/2)\beta_i^{\alpha_i}}{\Gamma(\alpha_i)(\beta_i+(\mathbf{A}_{(i,:)}-\mathbf{M}_{(i,:)})\frac{\mathbf{K}}{2}(\mathbf{A}_{(i,:)}-\mathbf{M}_{(i,:)})^T)^{\alpha_i+m/2}}\mathcal{PIG}_{\mathbf{v}}(\boldsymbol{\alpha}+m/2,\boldsymbol{\beta}+\boldsymbol{\eta}/2),\\
&= \mathcal{PIG}_{\mathbf{v}}(\boldsymbol{\alpha}+m/2,\boldsymbol{\beta}+\boldsymbol{\eta}/2)\prod_{i=1}^{d}\mathcal{T}_{\mathbf{A}_{(i,:)}}(\beta_i,2\alpha_i,\mathbf{M}_{(i,:)},\frac{\mathbf{K}}{2}).
\end{aligned}$$

Lemma A.2

$$\begin{aligned}
\mathcal{N}_{\mathbf{y}}(\mathbf{Ax},\mathbf{V})\mathcal{N}_{\mathbf{A}}(\mathbf{M},\mathbf{V},\mathbf{K})\mathcal{PIG}_{\mathbf{V}}(\boldsymbol{\alpha},\boldsymbol{\beta}) &= \mathcal{N}_{\mathbf{A}}((\mathbf{MK}+\mathbf{yx}^T)(\mathbf{xx}^T+\mathbf{K})^{-1},\mathbf{V},\mathbf{xx}^T+\mathbf{K})\times\\
&\times\mathcal{PIG}_{\mathbf{V}}\left(\boldsymbol{\alpha}+1/2,\boldsymbol{\beta}+diag\,(\mathbf{y}-\mathbf{Mx})\frac{(1-\mathbf{x}^T(\mathbf{xx}^T+\mathbf{K})^{-1}\mathbf{x})}{2}(\mathbf{y}-\mathbf{Mx})^T\right)\times\\
&\times\prod_{i=1}^{d}T_{y_i}\left(\beta_i,2\alpha_i,(\mathbf{Mx})_i,\frac{1-\mathbf{x}^T(\mathbf{xx}^T+\mathbf{K})^{-1}\mathbf{x}}{2}\right).
\end{aligned}$$

Proof:

$$\begin{aligned}
\mathcal{N}_{\mathbf{y}}(\mathbf{Ax},\mathbf{V})\mathcal{N}_{\mathbf{A}}(\mathbf{M},\mathbf{V},\mathbf{K})\mathcal{PIG}_{\mathbf{V}}(\boldsymbol{\alpha},\boldsymbol{\beta}) &= \mathcal{N}_{\mathbf{A}}((\mathbf{MK}+\mathbf{yx}^T)(\mathbf{xx}^T+\mathbf{K})^{-1},\mathbf{V},\mathbf{xx}^T+\mathbf{K})\times\\
&\mathcal{N}_{\mathbf{y}}((\mathbf{Mx},\mathbf{V},1-\mathbf{x}^T(\mathbf{xx}^T+\mathbf{K})^{-1}\mathbf{x})\times\\
&\mathcal{PIG}_{\mathbf{V}}(\boldsymbol{\alpha},\boldsymbol{\beta}).
\end{aligned}$$

A.4 Proof of Lemma 5.2

We can see that $Var_{\mathbf{V}}[E[\mathbf{A}|\mathbf{V},\{\mathbf{x}\}_1^{t+1},\{\mathbf{y}\}_1^{t+1}]] = Var_{\mathbf{V}}[\mathbf{M}_{t+1}] = 0$, and

$$\begin{aligned} E_{\mathbf{V}}[trVar(\mathbf{A}|\mathbf{V},\{\mathbf{x}\}_1^{t+1},\{\mathbf{y}\}_1^{t+1})] &= E_{\mathbf{V}}[tr(\mathbf{V}\otimes(\mathbf{K}_t+\mathbf{x}_{t+1}\mathbf{x}_{t+1}^T)^{-1})|\{\mathbf{x}\}_1^{t+1},\{\mathbf{y}\}_1^{t+1}], \\ &= tr(E[\mathbf{V}|\{\mathbf{x}\}_1^{t+1},\{\mathbf{y}\}_1^{t+1}])tr[(\mathbf{K}_t+\mathbf{x}_{t+1}\mathbf{x}_{t+1}^T)^{-1}], \\ &= tr[(\mathbf{K}_t+\mathbf{x}_{t+1}\mathbf{x}_{t+1}^T)^{-1}]\sum_{i=1}^d \frac{(\boldsymbol{\beta}_{t+1})_i}{(\boldsymbol{\alpha}_{t+1})_i-1}, \end{aligned}$$

where we used that $P(\mathbf{V}|\{\mathbf{x}\}_1^{t+1},\{\mathbf{y}\}_1^{t+1}) = \mathcal{PIG}_{\mathbf{V}}(\boldsymbol{\alpha}_{t+1},\boldsymbol{\beta}_{t+1})$.

The law of total variance says that $Var[\mathbf{A}] = Var[E[\mathbf{A}|\mathbf{V}]] + E[Var[\mathbf{A}|\mathbf{V}]]$, hence

$$trVar[\mathbf{A}|\{\mathbf{x}\}_1^{t+1},\{\mathbf{y}\}_1^{t+1}] = tr(\mathbf{K}_t+\mathbf{x}_{t+1}\mathbf{x}_{t+1}^T)^{-1}\sum_{i=1}^d \frac{(\boldsymbol{\beta}_{t+1})_i}{(\boldsymbol{\alpha}_{t+1})_i-1}.$$

A.5 Proof of Lemma 6.1

To compute (24) we need the following lemma (Minka, 2000):

Lemma A.3 *If* $P(\mathbf{A}) = \mathcal{N}_{\mathbf{A}}(\mathbf{M},\mathbf{V},\mathbf{K})$*, then* $P(\mathbf{A}\mathbf{x}) = \mathcal{N}_{\mathbf{A}\mathbf{x}}\left(\mathbf{M}\mathbf{x},\mathbf{V},\left(\mathbf{x}^T\mathbf{K}^{-1}\mathbf{x}\right)^{-1}\right)$.

Applying this lemma and using (11) we have that

$$P(\mathbf{A}\mathbf{x}_{t+1}|\mathbf{V},\{\mathbf{x}\}_1^{t+1},\{\mathbf{y}\}_1^{t+1}) = \mathcal{N}_{\mathbf{A}\mathbf{x}_{t+1}}\left(\mathbf{M}_{t+1}\mathbf{x}_{t+1},\mathbf{V},\left(\mathbf{x}_{t+1}^T\mathbf{K}_{t+1}^{-1}\mathbf{x}_{t+1}\right)^{-1}\right). \tag{37}$$

We introduce the notations

$$\begin{aligned} \tilde{K}_{t+1} &= \left(\mathbf{x}_{t+1}^T\mathbf{K}_{t+1}^{-1}\mathbf{x}_{t+1}\right)^{-1}\in\mathbb{R}, \\ \lambda_{t+1} &= 1+(\mathbf{A}\mathbf{x}_{t+1}-\mathbf{M}_{t+1}\mathbf{x}_{t+1})^T(\tilde{K}_{t+1}\mathbf{Q}_{t+1}^{-1})(\mathbf{A}\mathbf{x}_{t+1}-\mathbf{M}_{t+1}\mathbf{x}_{t+1})\in\mathbb{R}. \end{aligned} \tag{38}$$

Exploit the fact that

$$\mathcal{N}_{\mathbf{A}}(\mathbf{M},\mathbf{V},\mathbf{K})\,\mathcal{IW}_{\mathbf{V}}(\mathbf{Q},n) = \mathcal{IW}_{\mathbf{V}}(\mathbf{Q}+\mathbf{H},n+m)\,\mathcal{T}_{\mathbf{A}}(\mathbf{Q},n,\mathbf{M},\mathbf{K}),$$

and use (12) for the posterior distribution (37) and get

$$\begin{aligned} P(\mathbf{A}\mathbf{x}_{t+1}|\{\mathbf{x}\}_1^{t+1},\{\mathbf{y}\}_1^{t+1}) &= \mathcal{T}_{\mathbf{A}\mathbf{x}_{t+1}}\left(\mathbf{Q}_{t+1},n_{t+1},\mathbf{M}_{t+1}\mathbf{x}_{t+1},\tilde{K}_{t+1}\right), \\ &= \pi^{-d/2}|\tilde{K}_{t+1}^{-1}\mathbf{Q}_{t+1}|^{-1/2}\frac{\Gamma(\frac{n_{t+1}+1}{2})}{\Gamma(\frac{n_{t+1}+1-d}{2})}\lambda_{t+1}^{\frac{n_{t+1}+1}{2}}. \end{aligned}$$

The Shannon-entropy of this distribution according to Zografos and Nadarajah (2005) can be written as:

$$H(\mathbf{A}\mathbf{x}_{t+1};\{\mathbf{x}\}_1^{t+1},\{\mathbf{y}\}_1^{t+1}) = f_{3,1}(d,n_{t+1})+\frac{d}{2}\log|\tilde{K}_{t+1}^{-1}|+\log|\mathbf{Q}_{t+1}|,$$

where

$$f_{3,1}(d,n_{t+1}) = -\log\frac{\Gamma(\frac{n_{t+1}+1}{2})}{\pi^{d/2}\Gamma(\frac{n_{t+1}+1-d}{2})}+\frac{n_{t+1}+1}{2}\left(\Psi\left(\frac{n_{t+1}+1}{2}\right)-\Psi\left(\frac{n_{t+1}+1-d}{2}\right)\right).$$

Using the notations introduced in (10) and in (38), the above expressions can be transcribed as follows:

$$
\begin{aligned}
H(\mathbf{A}\mathbf{x}_{t+1};\{\mathbf{x}\}_1^{t+1},\{\mathbf{y}\}_1^{t+1}) &= f_{3,1}(d,n_{t+1}) - \frac{d}{2}\log|\tilde{K}_{t+1}| + \log|\mathbf{Q}_{t+1}|,\\
&= f_{3,1}(d,n_{t+1}) + \frac{d}{2}\log|\mathbf{x}_{t+1}^T(\mathbf{K}_t+\mathbf{x}_{t+1}\mathbf{x}_{t+1}^T)^{-1}\mathbf{x}_{t+1}| + \log|\mathbf{Q}_{t+1}|,\\
&= f_{3,1}(d,n_{t+1}) + \frac{d}{2}\log|\mathbf{x}_{t+1}^T(\mathbf{K}_t+\mathbf{x}_{t+1}\mathbf{x}_{t+1}^T)^{-1}\mathbf{x}_{t+1}| +\\
&\quad + \log|\mathbf{Q}_t + (\mathbf{y}_{t+1}-\mathbf{M}_t\mathbf{x}_{t+1})\gamma_{t+1}(\mathbf{y}_{t+1}-\mathbf{M}_t\mathbf{x}_{t+1})^T|.
\end{aligned}
$$

Now, we are in a position to calculate (24) by applying Lemma 4.4 as before. We get that

$$
\begin{aligned}
&\int d\mathbf{y}_{t+1}P(\mathbf{y}_{t+1}|\{\mathbf{x}\}_1^{t+1},\{\mathbf{y}\}_1^t)H(\mathbf{e}_{t+1};\{\mathbf{x}\}_1^{t+1},\{\mathbf{y}\}_1^{t+1}) =\\
&= f_{3,2}(\mathbf{Q}_t,n_{t+1}) + \frac{d}{2}\log|\mathbf{x}_{t+1}^T(\mathbf{K}_t+\mathbf{x}_{t+1}\mathbf{x}_{t+1}^T)^{-1}\mathbf{x}_{t+1}|,
\end{aligned}
$$

where $f_{3,2}(\mathbf{Q}_t,n_{t+1})$ depends only on $\mathbf{Q}_t$, and n_{t+1}. We can proceed as follows

$$
\begin{aligned}
\arg\max_{\mathbf{u}_{t+1}} I(\mathbf{e}_{t+1},\mathbf{y}_{t+1};\{\mathbf{x}\}_1^{t+1},\{\mathbf{y}\}_1^t) &= \arg\min_{\mathbf{u}_{t+1}}\log|\mathbf{x}_{t+1}^T(\mathbf{K}_t+\mathbf{x}_{t+1}\mathbf{x}_{t+1}^T)^{-1}\mathbf{x}_{t+1}|,\\
&= \arg\min_{\mathbf{u}_{t+1}}\log\left|\mathbf{x}_{t+1}^T\left(\mathbf{K}_t^{-1} - \frac{\mathbf{K}_t^{-1}\mathbf{x}_{t+1}\mathbf{x}_{t+1}^T\mathbf{K}_t^{-1}}{1+\mathbf{x}_{t+1}^T\mathbf{K}_t^{-1}\mathbf{x}_{t+1}}\right)\mathbf{x}_{t+1}\right|,\\
&= \arg\min_{\mathbf{u}_{t+1}}\log\left|\frac{\mathbf{x}_{t+1}^T\mathbf{K}_t^{-1}\mathbf{x}_{t+1}}{1+\mathbf{x}_{t+1}^T\mathbf{K}_t^{-1}\mathbf{x}_{t+1}}\right|,\\
&= \arg\min_{\mathbf{u}_{t+1}}\mathbf{x}_{t+1}^T\mathbf{K}_t^{-1}\mathbf{x}_{t+1}.
\end{aligned}
$$

This is exactly what we were to prove.

A.6 Proof of Lemma 8.1

One needs to compute the value of the integral

$$
\int d\mathbf{y}_{t+1}P(\mathbf{y}_{t+1}|\{\mathbf{x}\}_1^{t+1},\{\mathbf{y}\}_1^t)E[tr\mathbf{V}|\mathbf{x}_1^{t+1},\mathbf{y}_1^{t+1}]tr\left(\left(\mathbf{K}_t+\mathbf{x}_{t+1}\mathbf{x}_{t+1}^T\right)^{-1}+\mathbf{x}_{t+1}^T\mathbf{K}_{t+1}^{-1}\mathbf{x}_{t+1}\right).
$$

However,

$$
\begin{aligned}
&\int d\mathbf{y}_{t+1}P(\mathbf{y}_{t+1}|\{\mathbf{x}\}_1^{t+1},\{\mathbf{y}\}_1^t)E[tr\mathbf{V}|\mathbf{x}_1^{t+1},\mathbf{y}_1^{t+1}]\\
&= \int d\mathbf{y}_{t+1}\prod_{i=1}^d \mathcal{T}_{(\mathbf{y}_{t+1})_i}\left((\boldsymbol{\beta}_t)_i, 2(\boldsymbol{\alpha}_t)_i, (\mathbf{M}_t\mathbf{x}_{t+1})_i, \frac{\gamma_{t+1}}{2}\right)\sum_{i=1}^d\frac{(\boldsymbol{\beta}_{t+1})_i}{(\boldsymbol{\alpha}_{t+1})_i-1},
\end{aligned}
$$

and depends only on the values of α_{t+1} and β_t as a result of (19) and Lemma 4.4, and is independent of the value of $\mathbf{x}_{t+1}$. Thus, we arrive at the minimization of the following expression:

$$
\begin{aligned}
tr\left[\left(\mathbf{K}_t^{-1}-\frac{\mathbf{K}_t^{-1}\mathbf{x}_{t+1}\mathbf{x}_{t+1}^T\mathbf{K}_t^{-1}}{1+\mathbf{x}_{t+1}^T\mathbf{K}_t^{-1}\mathbf{x}_{t+1}}\right)+\mathbf{x}_{t+1}^T\left(\mathbf{K}_t^{-1}-\frac{\mathbf{K}_t^{-1}\mathbf{x}_{t+1}\mathbf{x}_{t+1}^T\mathbf{K}_t^{-1}}{1+\mathbf{x}_{t+1}^T\mathbf{K}_t^{-1}\mathbf{x}_{t+1}}\right)\mathbf{x}_{t+1}\right]& \\
=\left[tr(\mathbf{K}_t^{-1})-tr\frac{\mathbf{K}_t^{-1}\mathbf{x}_{t+1}\mathbf{x}_{t+1}^T\mathbf{K}_t^{-1}}{1+\mathbf{x}_{t+1}^T\mathbf{K}_t^{-1}\mathbf{x}_{t+1}}+\frac{\mathbf{x}_{t+1}^T\mathbf{K}_t^{-1}\mathbf{x}_{t+1}}{1+\mathbf{x}_{t+1}^T\mathbf{K}_t^{-1}\mathbf{x}_{t+1}}\right]&, \\
=\left[tr(\mathbf{K}_t^{-1})-\frac{\mathbf{x}_{t+1}^T\mathbf{K}_t^{-1}\mathbf{K}_t^{-1}\mathbf{x}_{t+1}}{1+\mathbf{x}_{t+1}^T\mathbf{K}_t^{-1}\mathbf{x}_{t+1}}+\frac{\mathbf{x}_{t+1}^T\mathbf{K}_t^{-1}\mathbf{x}_{t+1}}{1+\mathbf{x}_{t+1}^T\mathbf{K}_t^{-1}\mathbf{x}_{t+1}}\right]&, \\
=\frac{1+\mathbf{x}_{t+1}^T\mathbf{K}_t^{-1}\mathbf{K}_t^{-1}\mathbf{x}_{t+1}}{1+\mathbf{x}_{t+1}^T\mathbf{K}_t^{-1}\mathbf{x}_{t+1}}&\ .
\end{aligned}
$$

References

J. Abonyi, R. Babuska, A. Ayala-Botto, A. Szeifert, and L. Nagy. Identification and control of nonlienar systems using Hammerstein-models. *Ind. Eng. Chem. Res*, 39:4302–4314, 2000.

L. A. Aguirre, M. C. S. Coelho, and M. V. Correa. On the interpretation and practice of dynamical differences between Hammerstein and Wiener models. In *IEE. Proc. of Control Theory Application*, volume 152, pages 349–354, 2005.

S. Amari, A. Cichocki, and H. H. Yang. A new learning algorithm for blind signal separation. In *Advances in Neural Information Processing Systems*, volume 8, pages 757–763, Cambridge, MA, 1996. MIT Press.

B. Anderson and A. Moore. Active learning for hidden Markov models: objective functions and algorithms. In *ICML '05: Proceedings of the 22nd international conference on Machine learning*, pages 9–16, New York, NY, USA, 2005. ACM.

F. R. Bach. Active learning for misspecified generalized linear models. In B. Schölkopf, J. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems*, volume 19, pages 65–72, Cambridge, MA, 2007. MIT Press.

E. W. Bai. A blind approach to the Hammerstein-Wiener model identification. *Automatica*, 38: 967–979, 2002.

M. J. Beal. *Variational Algorithms for Approximate Bayesian Inference*. PhD thesis, Gatsby Computational Neuroscience Unit, University College London, 2003.

J. M. Bernardo. Expected information as expected utility. *The Annals of Statistics*, 7(3):686–690, 1979.

S. A. Billings and I. J. Leontaritis. Identifiaction of nonlinear systems using parametric estimation techniques. In *IEE. Proc. of Control and its Applications*, pages 183–187, 1981.

X. Boyen and D. Koller. Tractable inference for complex stochastic processes. In *Fourteenth Conference on Uncertainty in Artificial Intelligence*, pages 33–42, 1998.

J. F. Cardoso. High-order contrasts for independent component analysis. *Neural Computation*, 11 (1):157–192, 1999.

R. Castro, J. Jaupt, and R. Nowak. Compressed sensing vs. active learning. In *ICASSP 2006, IEEE International Conference on Acoustics, Speech and Signal Processing*, volume 3, 2006a.

R. Castro, R. Willett, and R. 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, 2006b.

P. Celka, N. J. Bershad, and J. Vesin. Stochastic gradient identification of polynomial Wiener systems: Analysis and application. *IEEE Trans. Signal Process.*, 49(2):301–313, 2001.

K. Chaloner. Optimal bayesian experimental design for linear models. *The Annals of Statistics*, 12 (1):283–300, 1984.

K. Chaloner and I. Verdinelli. Bayesian experimental design: A review. *Statist. Sci.*, 10:273–304, 1995.

N. Chiras, C. Evans, and D. Rees. Nonlinear gas turibne modeling using NARMAX structures. *IEEE Trans. Instrum. Meas.*, 50(4):893–898, 2001.

D. A. Cohn. Neural network exploration using optimal experiment design. In *Advances in Neural Information Processing Systems*, volume 6, pages 679–686, 1994.

D. A. Cohn, Z. Ghahramani, and M. I. Jordan. Active learning with statistical models. *Journal of Artificial Intelligence Research*, 4:129–145, 1996.

T. F. Coleman and Y. Li. A reflective newton method for minimizing a quadratic function subject to bounds on some of the variables. *SIAM Journal on Optimization*, 6(4):1040–1058, 1996.

P. Comon. Independent component analysis, a new concept? *Signal Processing*, 36(3):287–314, 1994.

T. M. Cover and J. A. Thomas. *Elements of Information Theory*. Wiley-Interscience, 1991.

R. C. deCharms, D. T. Blake, and M. M. Merzenich. Optimizing sound features for cortical neurons. *Science*, 280:1439–1444, 1998.

G. Duncan and M. H. DeGroot. A mean squared error approach to optimal design theory. In *Proceedings of the 1976 Conference on Information: Sciences and Systems*, pages 217–221. The Johns Hopkins University, 1976.

V. V. Fedorov. *Theory of Optimal Experiments*. Academic Press, New York, 1972.

P. Földiák. Stimulus optimization in primary visual cortex. *Neurocomputing*, 38–40:1217–1222, 2001.

Y. Freund, H. S. Seung, E. Shamir, and N. Tishby. Selective sampling using the query by committee algorithm. *Machine Learning*, 28:133–168, 1997.

K. Fukumizu. Active learning in multilayer perceptrons. In David S. Touretzky, Michael C. Mozer, and Michael E. Hasselmo, editors, *Advances in Neural Information Processing Systems*, volume 8, pages 295–301. The MIT Press, 1996.

K. Fukumizu. Statistical active learning in multilayer perceptrons. *IEEE Transactions on Neural Networks*, 11(1):17–26, 2000.

M. Gäfvert. Modelling the Furuta pendulum. Technical report ISRN LUTFD2/TFRT–7574–SE, Department of Automatic Control, Lund University, Sweden, April 1998.

A. Gelman, J. B. Carlin, H. S. Stern, and D. B. Rubin. *Bayesian Data Analysis*. CRC Press, 2nd erdition, 2003.

Z. Ghahramani. Online variational Bayesian learning, 2000. Slides from talk presented at NIPS 2000 workshop on Online Learning.

G. H. Golub and C. F. Van Loan. *Matrix Computations*. Johns Hopkins, Baltimore, MD, 3rd ed. edition, 1996.

A. K. Gupta and D. K. Nagar. *Matrix Variate Distributions*, volume 104 of *Monographs and Surveys in Pure and Applied Mathematics*. Chapman and Hall/CRC, 1999.

R. Haber and H. Unbehauen. Structure identification of nonlinear dynamic systems– a survey on input/output approaches. *Automatica*, 26(4):651–677, 1990.

D. A. Harville. *Matrix Algebra From a Statistician's Perspective*. Springer-Verlag, 1997.

A. Honkela and H. Valpola. On-line variational Bayesian learning. In *4th International Symposium on Independent Component Analysis and Blind Signal Separation*, pages 803–808, 2003.

A. Hyvärinen. Fast and robust fixed-point algorithms for independent component analysis. *IEEE Trans. on Neural Networks*, (10):626–634, 1999.

A. Hyvärinen, J. Karhunen, and E. Oja. *Independent Component Analysis*. John Wiley, New York, 2001. URL `http://www.cis.hut.fi/projects/ica/book/`.

A. Hyvärinen. Independent component analysis for time-dependent stochastic processes. In *Proc. of ICANN'98, International Conference on Artificial Neural Networks, Skövde, Sweden*, pages 541–546, 1998.

H. Jaeger. Short term memory in echo state networks. GMD Report, 152, Fraunhofer AIS, 2001. http://publica.fraunhofer.de/starweb/pub08/en/index.htm.

C. Jutten and J. Hérault. Blind separation of sources: An adaptive algorithm based on neuromimetic architecture. *Signal Processing*, 24:1–10, 1991.

J. Kiefer. Optimum experimental designs. *Journal of the Royal Statistical Society, Series B*, 21: 272–304, 1959.

S. Kotz and S. Nadarajah. *Multivariate T-Distributions and Their Applications*. Cambridge University Press, 2004.

A. Krause and C. Guestrin. Nonmyopic active learning of gaussian processes: an exploration-exploitation approach. In *ICML '07: Proceedings of the 24th international conference on Machine learning*, pages 449–456, New York, NY, USA, 2007. ACM.

J. Lewi, R. Butera, and L. Paninski. Real-time adaptive information-theoretic optimization of neurophysiology experiments. In *Advances in Neural Information Processing Systems*, volume 19, 2007.

D. D. Lewis and J. Catlett. Heterogeneous uncertainty sampling for supervised learning. In William W. Cohen and Haym Hirsh, editors, *Proceedings of ICML-94, 11th International Conference on Machine Learning*, pages 148–156, New Brunswick, US, 1994. Morgan Kaufmann Publishers, San Francisco, US.

D. D. Lewis and W. A. Gale. A sequential algorithm for training text classifiers. In *Proceedings of SIGIR-94, 17th ACM International Conference on Research and Development in Information Retrieval*, pages 3–12, Dublin, IE, 1994. Springer Verlag, Heidelberg, DE.

D. V. Lindley. *Bayesian Statistics: A Review*. SIAM, 1971.

W. Maass, T. Natschläger, and H. Markram. Real-time computing without stable states: A new framework for neural computation based on perturbations. *Neural Computation*, 14:2531–2560, 2002.

C. K. Machens, T. Gollisch, O. Kolesnikova, and A. V. M. Herz. Testing the efficiency of sensory coding with optimal stimulus ensembles. *Neuron*, 47:447–456, 2005.

D. J. C. MacKay. Information-based objective functions for active data selection. *Neural Computation*, 4(4):590–604, 1992.

T. Minka. Bayesian linear regression, 2000. MIT Media Lab note.

T. Minka. *A Family of Algorithms for Approximate Bayesian Inference*. PhD thesis, MIT Media Lab, MIT, 2001.

M. Opper and O. Winther. A Bayesian approach to online learning. In *Online Learning in Neural Networks*. Cambridge University Press, 1999.

R. K. Pearson and M. Pottmann. Gray-box identification of block-oriented nonlnear models. *Journal of Process Control*, 10:301–315, 2000.

F. Pukelsheim. *Optimal Design of Experiments*. John Wiley & Sons, 1993.

H. Rabitz. Shaped laser pulses as reagents. *Science*, 299:525–527, 2003.

H. Raiffa and R. Schlaifer. *Applied Statistical Decision Theory*. Boston, MIT Press, 1961.

R. Rangarajan, R. Raich, and A. O. Hero. Optimal sequential energy allocation for inverse problems. *IEEE Journal of Selected Topics in Signal Processing*, 1:67–78, 2007.

N. Roy and A. McCallum. Toward optimal active learning through sampling estimation of error reduction. In *Proc. 18th International Conference on Machine Learning*, pages 441–448. Morgan Kaufmann, San Francisco, CA, 2001.

A. I. Schein. *Active Learning for Logistic Regression*. PhD thesis, University of Pennsylvania, 2005.

A. I. Schein and L. H. Ungar. Active learning for logistic regression: an evaluation. *Machine Learning*, 68(3):235–265, 2007.

H. S. Seung, M. Opper, and H. Sompolinsky. Query by committee. In *Computational Learning Theory*, 1992.

S. Solla and O. Winther. Optimal perceptron learning: An online Bayesian approach. In *Online Learning in Neural Networks*. Cambridge University Press, 1999.

D. M. Steinberg and W.G. Hunter. Experimental design: review and comment. *Technometrixs*, 26: 71–97, 1984.

M. Stone. Application of a measure of information to the design and comparison of regression experiments. *Ann. Math. Statist*, 30(1):55–70, 1959.

M. Sugiyama. Active learning in approximately linear regression based on conditional expectation of generalization error. *The Journal of Machine Learning Research*, 7:141–166, 2006.

Z. Szabó and A. Lőrincz. Towards independent subspace analysis in controlled dynamical systems. ICA Research Network International Workshop, 2008.

B. Toman and J. L. Gastwirth. Robust Bayesian experimental design and estimation for analysis of variance models using a class of normal mixtures. *Journal of statistical planning and inference*, 35(3):383–398, 1993.

S. Tong and D. Koller. Active learning for parameter estimation in Bayesian networks. In *Advances in Neural Information Processing Systems*, pages 647–653, 2000.

S. Tong and D. Koller. Active learning for structure in Bayesian networks. In *Proceedings of the International Joint Conference on Artificial Intelligence*, 2001a.

S. Tong and D. Koller. Support vector machine active learning with applications to text classification. *Journal of Machine Learning Research*, pages 45–66, 2001b.

T. M. Vaughan, W. J. Heetderks, L. J. Trejo, W. Z. Rymer, M. Weinrich, M. M. Moore, A. Kübler, B. H. Dobkin, N. Birbaumer, E. Donchin, E. W. Wolpaw, and J. R. Wolpaw. Brain-computer interface technology: a review of the Second International Meeting. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 11:94–109, 2003.

I. Verdinelli. A note on Bayesian design for the normal linear model with unknown error variance. *Biometrika*, 87:222–227, 2000.

M. Yamakita, M. Iwashiro, Y. Sugahara, and K. Furuta. Robust swing-up control of double pendulum. In *American Control Conference*, volume 1, pages 290–295, 1995.

K. Zografos and S. Nadarajah. Expressions for Rényi and Shannon entropies for multivariate distributions. *Statistics and Probability Letters*, 71(1):71–84, 2005.

Journal of Machine Learning Research 10 (2009) 555-568 Submitted 5/08; Revised 9/08; Published 3/09

On the Consistency of Feature Selection using Greedy Least Squares Regression

Tong Zhang[*] TZHANG@STAT.RUTGERS.EDU
Statistics Department
110 Frelinghuysen Road
Rutgers University, NJ 08854

Editor: Bin Yu

Abstract

This paper studies the feature selection problem using a greedy least squares regression algorithm. We show that under a certain irrepresentable condition on the design matrix (but independent of the sparse target), the greedy algorithm can select features consistently when the sample size approaches infinity. The condition is identical to a corresponding condition for Lasso.

Moreover, under a sparse eigenvalue condition, the greedy algorithm can reliably identify features as long as each nonzero coefficient is larger than a constant times the noise level. In comparison, Lasso may require the coefficients to be larger than $O(\sqrt{s})$ times the noise level in the worst case, where s is the number of nonzero coefficients.

Keywords: greedy algorithm, feature selection, sparsity

1. Introduction

We are interested in the statistical feature selection problem for least squares regression. Let $X = [\mathbf{x}_1, \ldots, \mathbf{x}_d] \in \mathbb{R}^{n \times d}$ be an $n \times d$ data matrix with $\mathbf{x}_j \in \mathbb{R}^n$ $(j = 1, \ldots, d)$ as its columns. Assume that the response vector $\mathbf{y} = [y_1, \ldots, y_n] \in \mathbb{R}^n$ is generated from a sparse linear combination of the basis vectors $\{\mathbf{x}_j\}$ plus a zero-mean stochastic noise vector $\mathbf{z} \in \mathbb{R}^n$:

$$\mathbf{y} = X\bar{\beta} + \mathbf{z} = \sum_{j=1}^{d} \bar{\beta}_j \mathbf{x}_j + \mathbf{z}, \tag{1}$$

where most coefficients $\bar{\beta}_j$ equal zero. The goal of feature selection is to identify the set of nonzeros $\{j : \bar{\beta}_j \neq 0\}$, where $\bar{\beta} = [\bar{\beta}_1, \ldots, \bar{\beta}_d]$. The purpose of this paper is study the performance of greedy least squares regression for feature selection.

The following notations are used throughout the paper. Given $\beta \in \mathbb{R}^d$, define

$$\mathrm{supp}(\beta) = \{j : \beta_j \neq 0\}.$$

Given $\mathbf{x} \in \mathbb{R}^n$ and $\bar{F} \subset \{1, \ldots, d\}$, let

$$\hat{\beta}_X(\bar{F}, \mathbf{x}) = \min_{\beta \in \mathbb{R}^d} \frac{1}{n} \|X\beta - \mathbf{x}\|_2^2 \quad \text{subject to} \quad \mathrm{supp}(\beta) \subset \bar{F}.$$

*. Supported by NSF grant DMS-0706805.

That is, $\hat{\beta}_X(\bar{F},\mathbf{x})$ is the least squares solution with coefficients restricted to $\bar{F}$.

Given $\bar{F} \in \{1,\ldots,d\}$, we let $X_{\bar{F}}$ be the $n \times |\bar{F}|$ matrix that is the restriction of columns of X to $\bar{F}$. That is, $X_{\bar{F}}$'s columns are the basis functions $\mathbf{x}_j$ with $j \in \bar{F}$ arranged in the ascending order. The following quantity, appeared in Tropp (2004), is important in our analysis (also see Wainwright, 2006):

$$\mu_X(\bar{F}) = \max_{j \notin \bar{F}} \|(X_{\bar{F}}^T X_{\bar{F}})^{-1} X_{\bar{F}}^T \mathbf{x}_j\|_1.$$

We also define for all $\bar{F} \subset \{1,\ldots,d\}$

$$\rho_X(\bar{F}) = \inf\left\{\frac{1}{n}\|X\beta\|_2^2/\|\beta\|_2^2 : \mathrm{supp}(\beta) \subset \bar{F}\right\}.$$

This quantity is the smallest eigenvalue of the restricted design matrix $\frac{1}{n}X_{\bar{F}}^T X_{\bar{F}}$, which has also appeared in previous work such as Wainwright (2006) and Zhao and Yu (2006). The requirement that $\rho_X(F)$ is bounded away from zero for small $|F|$ is often referred to as the sparse eigenvalue condition (or the restricted isometry condition).

2. Related Work

The feature selection problem of estimating $\mathrm{supp}(\bar{\beta})$ from observation $\mathbf{y}$ defined in (1) has attracted significant attention in recent years. One of the frequently used method for feature selection is Lasso, which solves the following L_1 regularization problem:

$$\hat{\beta} = \arg\min_{\beta}\left[\frac{1}{n}\left\|\sum_{j=1}^{d}\beta_j\mathbf{x}_j - \mathbf{y}\right\|_2^2 + \lambda\|\beta\|_1\right], \tag{2}$$

where $\lambda > 0$ is an appropriately chosen regularization parameter.

The effectiveness of feature selection using Lasso was established in Zhao and Yu (2006) (also see Meinshausen and Buhlmann, 2006) under irrepresentable conditions that depend on the signs of the true target $\mathrm{sgn}(\bar{\beta})$. Results established in this paper have cruder forms that depend only on the design matrix but not the sparse target $\bar{\beta}$. Such conditions have also been studied by Zhao and Yu (2006) (also see Wainwright, 2006).

In addition to Lasso, greedy algorithms have also been widely used for feature selection. Greedy algorithms for least squares regression are called matching pursuit in the signal processing community (Mallat and Zhang, 1993). The particular algorithm analyzed in this paper (some time referred to as orthogonal matching pursuit or OMP) is presented in Figure 1. The algorithm is often called forward greedy selection in the machine learning literature.

This paper investigates the behavior of greedy least squares algorithm in Figure 1 for feature selection under the stochastic noise model (1). Our result extends that of Tropp (2004), which only considered the situation without stochastic noise. It was shown by Tropp (2004) that $\mu_X(\bar{F}) < 1$ is sufficient for the greedy algorithm to identify the correct feature set $\mathrm{supp}(\bar{\beta})$ when the noise vector $\mathbf{z} = 0$. The main contribution of this paper is to generalize Tropp's analysis to handle non-zero sub-Gaussian stochastic noise vectors. In particular, we will establish conditions on $\min_{j\in\mathrm{supp}(\bar{\beta})}|\bar{\beta}_j|$ and the stopping criterion ϵ in Figure 1, so that the algorithm finds the correct feature set $\mathrm{supp}(\bar{\beta})$. The selection of stopping criterion ϵ in the greedy algorithm is equivalent to selecting an appropriate

```
Input: $X = [\mathbf{x}_1, \ldots, \mathbf{x}_d] \in \mathbb{R}^{n \times d}$, $\mathbf{y} \in \mathbb{R}^n$ and $\varepsilon > 0$
Output: $F^{(k)}$ and $\beta^{(k)}$
let $\tilde{\mathbf{x}}_j = \mathbf{x}_j / \|\mathbf{x}_j\|_2$ be normalized basis $(j = 1, \ldots, d)$
let $F^{(0)} = \emptyset$ and $\beta^{(0)} = 0$
for $k = 1, 2, \ldots$
  let $i^{(k)} = \arg\max_i |\tilde{\mathbf{x}}_i^T (X\beta^{(k-1)} - \mathbf{y})|$
  if $(|\tilde{\mathbf{x}}_{i^{(k)}}^T (X\beta^{(k-1)} - \mathbf{y})| \leq \varepsilon)$ break
  let $F^{(k)} = \{i^{(k)}\} \cup F^{(k-1)}$
  let $\beta^{(k)} = \hat{\beta}_X(F^{(k)}, \mathbf{y})$
end
```

Figure 1: Greedy Least Squares Regression (OMP)

regularization condition λ in the Lasso formulation (2), which is necessary both in theory and in practice. The condition on $\min_{j \in \text{supp}(\bar{\beta})} |\bar{\beta}_j|$ also naturally appears in the analysis of Lasso (Zhao and Yu, 2006). In fact, our result shows that the condition of $\min_{j \in \text{supp}(\bar{\beta})} |\bar{\beta}_j|$ required for greedy algorithm is weaker than the corresponding condition for Lasso.

The greedy algorithm analysis employed in this paper is a combination of an observation by Tropp (2004, see Lemma 11) and some technical lemmas for the behavior of greedy least squares regression by Zhang (2008), which are included in Appendix A for completeness. Note that Zhang (2008) only studied a forward-backward procedure, but not the more standard forward greedy algorithm considered here. In this paper, both the employment of the condition $\mu_X(\bar{F}) < 1$ and the proof in Appendix B are new.

As we shall see in this paper, the condition $\mu_X(\bar{F}) \leq 1$ is necessary for the success of the forward greedy procedure. It is worth mentioning that Lasso is consistent in parameter estimation under a weaker sparse eigenvalue condition, even if the condition $\mu_X(\bar{F}) \leq 1$ fails (which means Lasso may not estimate the true feature set correctly): for example, see Meinshausen and Yu (2008) and Zhang (2009). Although similar results may be obtained for greedy least squares regression, when the condition $\mu_X(\bar{F}) \leq 1$ fails, it was shown by Zhang (2008) that the performance of greedy algorithm can be improved by incorporating backward steps. In contrast, results in this paper show that if the design matrix satisfies the additional condition $\mu_X(\bar{F}) < 1$, then the standard forward greedy algorithm will be successful without complicated backward steps.

3. Feature Selection using Greedy Least Squares Regression

We would like to establish conditions under which the forward greedy algorithm in Figure 1 never makes any mistake (with large probability), and thus suitable for feature selection. For convenience, we state an assumption before stating the theoretical result.

Assumption 1 *Assume that*

- *The basis functions are normalized such that* $\frac{1}{n}\|\mathbf{x}_j\|_2^2 = 1$ *for all* $j = 1, \ldots, d$.
- *The target function is truly sparse: there exists* $\bar{\beta} \in \mathbb{R}^d$ *with* $\bar{F} = \text{supp}(\bar{\beta})$ *such that* $\mathbf{E}\mathbf{y} = X\bar{\beta}$.
- $\mu_X(\bar{F}) < 1$ *and* $\rho_X(\bar{F}) > 0$.

- $\mathbf{y} = [y_i]_{i=1,\ldots,n}$ *are independent (but not necessarily identically distributed) sub-Gaussians: there exists* $\sigma \geq 0$ *such that* $\forall i \in \{1,\ldots,n\}$ *and* $\forall t \in \mathbb{R}$, $\mathbf{E}_{y_i} e^{t(y_i - \mathbf{E}y_i)} \leq e^{\sigma^2 t^2/2}$.

Both Gaussian and bounded random variables are sub-Gaussian using the above definition. For example, if a random variable $\xi \in [a,b]$, then $\mathbf{E}_\xi e^{t(\xi - \mathbf{E}\xi)} \leq e^{(b-a)^2 t^2/8}$. If a random variable is Gaussian: $\xi \sim N(0,\sigma^2)$, then $\mathbf{E}_\xi e^{t\xi} \leq e^{\sigma^2 t^2/2}$.

The following theorem gives conditions under which the forward greedy algorithm can identify the correct set of features.

Theorem 1 *Consider the greedy least squares algorithm in Figure 1, where Assumption 1 holds. Given any* $\eta \in (0,0.5)$*, with probability larger than* $1-2\eta$*, if the stopping criterion satisfies*

$$\epsilon > \frac{1}{1-\mu_X(\bar{F})}\sigma\sqrt{2\ln(2d/\eta)}, \qquad \min_{j\in\bar{F}} |\bar{\beta}_j| \geq 3\epsilon\rho_X(\bar{F})^{-1}/\sqrt{n},$$

then when the procedure stops, we have $F^{(k-1)} = \bar{F}$ *and*

$$\|\beta^{(k-1)} - \bar{\beta}\|_\infty \leq \sigma\sqrt{(2\ln(2|\bar{F}|/\eta))/(n\rho_X(\bar{F}))}.$$

The result is a simple consequence of the following slightly more general theorem (its proof is left to Appendix B).

Theorem 2 *Consider the greedy least squares algorithm in Figure 1, where Assumption 1 holds. Given any* $\eta \in (0,0.5)$*, with probability larger than* $1-2\eta$*, if the stopping criterion satisfies*

$$\epsilon > \frac{1}{1-\mu_X(\bar{F})}\sigma\sqrt{2\ln(2d/\eta)},$$

then when the procedure stops, the following claims are true:

- $F^{(k-1)} \subset \bar{F}$.
- $|\bar{F} - F^{(k-1)}| \leq 2|\{j \in \bar{F} : |\bar{\beta}_j| < 3\epsilon\rho_X(\bar{F})^{-1}/\sqrt{n}\}|$
- $\|\beta^{(k-1)} - \hat{\beta}_X(\bar{F},\mathbf{y})\|_2 \leq \epsilon\rho(\bar{F})^{-1}\sqrt{|\bar{F} - F^{(k-1)}|/n}$.
- $\|\hat{\beta}_X(\bar{F},\mathbf{y}) - \bar{\beta}\|_\infty \leq \sigma\sqrt{(2\ln(2|\bar{F}|/\eta))/(n\rho_X(\bar{F}))}$.

In the following, we discuss some consequences of Theorem 1 and Theorem 2, and compare them with those of Lasso. Let $k(\epsilon)$ be the number of $j \in \bar{F}$ such that $|\bar{\beta}_j| < 3\epsilon\rho_X(\bar{F})^{-1}/\sqrt{n}$. Theorem 2 implies that $|\bar{F} - F^{(k-1)}| \leq 2k(\epsilon)$; that is, $|\bar{F} - F^{(k-1)}|$ is small when $k(\epsilon)$ is small. In such case, the feature set $F^{(k-1)}$ selected by the greedy least squares algorithm is approximately correct. Moreover, we have $\beta^{(k-1)} \approx \bar{\beta}$. In fact, one can show (e.g., see Zhang, 2008) that with probability larger than $1-\eta$:

$$\|\hat{\beta}_X(\bar{F},\mathbf{y}) - \bar{\beta}\|_2 \leq \sigma\sqrt{|\bar{F}|/(\rho(\bar{F})n)}[1 + \sqrt{20\ln(1/\eta)}].$$

By combining this estimate with Theorem 2, we have

$$\|\beta^{(k-1)} - \bar{\beta}\|_2 \leq \sigma\sqrt{|\bar{F}|/(\rho(\bar{F})n)}[1 + \sqrt{20\ln(1/\eta)}] + \epsilon\rho(\bar{F})^{-1}\sqrt{2k(\epsilon)/n}.$$

That is, $\|\beta^{(k-1)}-\bar{\beta}\|_2 = O(\sqrt{|\bar{F}|/n}+\epsilon\sqrt{k(\epsilon)/n})$. This implies that when $\mu_X(\bar{F})<1$, greedy least squares regression leads to a good estimation of the true parameter $\bar{\beta}$. By choosing $\epsilon = O(\sigma\sqrt{\ln(2d/\eta)})$, we obtain $\|\beta^{(k-1)}-\bar{\beta}\|_2 = O(\sqrt{|\bar{F}|/n}+\sqrt{k(\epsilon)\ln d/n})$. The corresponding result for the Lasso estimator $\hat{\beta}$ in (2) is $\|\hat{\beta}-\bar{\beta}\|_2 = O(\lambda\sqrt{|\bar{F}|/n})$, where we require λ to be of the order $\sigma\sqrt{\ln(d/\eta)/n}$ or larger. Therefore, if $k(\epsilon)$ is small, then Lasso is inferior due to the extra $\ln d$ factor. This factor is inherent to the L_1 regularization in Lasso, which introduces a bias that cannot be removed.

In this paper, we are mainly interested in the situation $k(\epsilon)=0$, which implies that with the stopping criterion ϵ, greedy least squares regression can correctly identify all features with large probability. Note that in order to correctly identify all features ($F^{(k-1)}=\bar{F}$), the requirement $\min_{j\in\bar{F}}|\bar{\beta}_j| \geq 3\epsilon\rho_X(\bar{F})^{-1}/\sqrt{n}$ in Theorem 1 is natural. Observe that we may take $\epsilon = \sigma\sqrt{3\ln(d/\eta)}/(1-\mu_X(\bar{F}))$. This means that under the assumption of Theorem 1, it is possible to identify all features correctly using the greedy least squares algorithm as long as the target coefficients $\bar{\beta}_j$ ($j\in\bar{F}$) are larger than the order $\sigma\sqrt{\ln(d/\eta)/n}$.

In fact, since $\sigma\sqrt{\ln(d/\eta)/n}$ is the noise level, if there exists a target coefficient $\bar{\beta}_j$ that is smaller than $O(\sigma\sqrt{\ln(d/\eta)/n})$ in absolute value, then we cannot distinguish such a small coefficient from zero (or noise) with large probability. Therefore when the condition $\mu_X(\bar{F})<1$ holds, it is not possible to do much better than greedy least squares regression except for the constant hidden in $O(\cdot)$ and its dependency on $\rho(\bar{F})$ and $\mu_X(\bar{F})$.

In comparison, for Lasso, the condition required of $\min_{j\in\bar{F}}|\bar{\beta}_j|$ depends not only on $\rho(\bar{F})^{-1}$ and $(1-\mu_X(\bar{F}))^{-1}$, but also on the quantity $\|(X_{\bar{F}}^T X_{\bar{F}})^{-1}\|_{\infty,\infty}$ (see Wainwright, 2006), where

$$\|(X_{\bar{F}}^T X_{\bar{F}})^{-1}\|_{\infty,\infty} = \sup_{\mathbf{u}\in\mathbb{R}^{|\bar{F}|}} \frac{\|(X_{\bar{F}}^T X_{\bar{F}})^{-1}\mathbf{u}\|_\infty}{\|\mathbf{u}\|_\infty}.$$

Consider the matrix $(X_{\bar{F}}^T X_{\bar{F}})^{-1} = I + 0.5B/\sqrt{|\bar{F}|}$, where $B_{i,j}=1$ when either $i=1$ or $j=1$ or $i=j$, and $B_{i,j}=0$ otherwise. Then it is not hard to verify that $\rho(\bar{F})^{-1}<2$ and $\|(X_{\bar{F}}^T X_{\bar{F}})^{-1}\|_{\infty,\infty} > 0.5\sqrt{|\bar{F}|}$ (by taking $\mathbf{u}=[1,\ldots,1]$). This means that in the worst case, we can find matrix $X_{\bar{F}}^T X_{\bar{F}}$ such that

$$\|(X_{\bar{F}}^T X_{\bar{F}})^{-1}\|_{\infty,\infty} > 0.25\sqrt{|\bar{F}|}\rho(\bar{F})^{-1}.$$

Therefore, if we only assume that $\rho(\bar{F})$ is bounded away from zero without using the quantity $\|(X_{\bar{F}}^T X_{\bar{F}})^{-1}\|_{\infty,\infty}$, the feature consistency result in Zhao and Yu (2006) and Wainwright (2006) for Lasso requires the condition

$$\min_{j\in\mathrm{supp}(\bar{\beta})} |\bar{\beta}_j| \geq c\sigma\sqrt{|\bar{F}|\ln(d/\eta)/n}$$

for some constant c that is proportional to $\rho(\bar{F})^{-1}(1-\mu_X(\bar{F}))^{-1}$. This is a more restrictive condition than that of greedy least squares regression. Unfortunately, the factor $\sqrt{|\bar{F}|}$ cannot be removed for Lasso, unless we make the additional and stronger assumption that $\|(X_{\bar{F}}^T X_{\bar{F}})^{-1}\|_{\infty,\infty} = O(\rho(\bar{F})^{-1})$.

As we discussed after Theorem 2, the bias of L_1-regularization also leads to suboptimal estimation for Lasso. For example, for the greedy algorithm, we can show $\|\hat{\beta}_X(\bar{F},\mathbf{y})-\bar{\beta}\|_2 = O(\sigma\sqrt{|\bar{F}|/n})$ and $\|\hat{\beta}_X(\bar{F},\mathbf{y})-\bar{\beta}\|_\infty = O(\sigma\sqrt{\ln|\bar{F}|/n})$. Under the conditions of Theorem 1, we have $\beta^{(k-1)} = \hat{\beta}_X(\bar{F},\mathbf{y})$, and thus $\|\beta^{(k-1)}-\bar{\beta}\|_2 = O(\sigma\sqrt{|\bar{F}|/n})$ and $\|\beta^{(k-1)}-\bar{\beta}\|_\infty = O(\sigma\sqrt{\ln|\bar{F}|/n})$. Under the same conditions, for the Lasso estimator $\hat{\beta}$ of (2), we have $\|\hat{\beta} - \bar{\beta}\|_2 = O(\sigma\sqrt{|\bar{F}| \ln d/n})$ and $\|\hat{\beta} - \bar{\beta}\|_\infty = O(\sigma\sqrt{\ln d/n})$. The $\ln d$ factor (bias) is inherent to Lasso, which can be removed with two-stage procedures (e.g., Zhang, 2009). However, such procedures are less robust and more complicated than the simple greedy algorithm.

4. Feature Selection Consistency

As we have mentioned before, the effectiveness of feature selection using Lasso was established by Zhao and Yu (2006), under more refined irrepresentable conditions that depend on the signs of the true target $\mathrm{sgn}(\bar{\beta})$. In comparison, the condition $\mu_X(\bar{F}) < 1$ in Theorem 2 depends only on the design matrix but not the sparse target $\bar{\beta}$. That is, the condition is with respect to the worst case choice of $\bar{\beta}$ with support $\mathrm{supp}(\bar{\beta}) = \bar{F}$. Due to the complexity of greedy procedure, we cannot establish a simple target dependent condition that ensures feature selection consistency. This means for any specific target, the behavior of forward greedy algorithm and Lasso might be different, and one may be preferred over the other under different scenarios. Experiments in Zhang (2008) illustrated this point.

In the following, we introduce the target independent irrepresentable conditions that are equivalent to the irrepresentable conditions of Zhao and Yu (2006) with the worst case choice of $\mathrm{sgn}(\bar{\beta})$ (also see Wainwright, 2006).

Definition 3 *Consider a sequence of problems indexed by n: at each sample size n, let $X^{(n)}$ be an $n \times d^{(n)}$ dimensional data matrix, and we observe $\mathbf{y}^{(n)} \in \mathbb{R}^n$ that is corrupted with noise. Let $\bar{F}^{(n)}$ be the feature set, where $\mathbf{E}\mathbf{y}^{(n)} = X^{(n)}\bar{\beta}^{(n)}$ and $\mathrm{supp}(\bar{\beta}^{(n)}) = \bar{F}^{(n)}$.*

We say that the sequence satisfies the strong target independent irrepresentable *condition if there exists $\delta > 0$ such that $\overline{\lim}_{n\to\infty}\mu_{X^{(n)}}(\bar{F}^{(n)}) \leq 1 - \delta$.*

We say that the sequence satisfies the weak target independent irrepresentable *condition if $\mu_{X^{(n)}}(\bar{F}^{(n)}) \leq 1$ for all sufficiently large n.*

It was shown by Zhao and Yu (2006) that the strong (target independent) irrepresentable condition is sufficient for Lasso to select features consistently for all possible sign combination of $\bar{\beta}^{(n)}$ when $n \to \infty$ (under appropriate assumptions). In addition, the weak (target independent) irrepresentable condition is necessary for Lasso to select features consistently when $n \to \infty$. The target independent irrepresentable conditions are considered by Zhao and Yu (2006) and Wainwright (2006). Similar conditions were also considered by Tropp (2004) without stochastic noise.

Results parallel to that of Lasso can be obtained for Algorithm 1. Specifically, the following two theorems show that the strong target independent irrepresentable condition is sufficient for Algorithm 1 to select features consistently, while the weak target independent irrepresentable condition is necessary.

Theorem 4 *Consider regression problems indexed by the sample size n, and use notations in Definition 3. Let Assumption 1 hold, with noise σ independent of n. Assume that the strong irrepresentable condition holds. For each problem of sample size n, denote by F_n the feature set from Algorithm 1 when it stops with $\varepsilon = n^{s/2}$ for some $s \in (0,1]$. Then for all sufficiently large n, we have*

$$P(F_n \neq \bar{F}^{(n)}) \leq \exp(-n^s/\ln n)$$

if the following conditions hold:

1. $d_n \leq \exp(n^s/\ln n)$

2. $\min_{j \in \bar{F}^{(n)}} |\bar{\beta}_j^{(n)}| \geq 3n^{(s-1)/2}/\rho(\bar{F}^{(n)})$.

Proof When n is sufficiently large, the two conditions of Theorem 1 hold with $\eta = 0.5\exp(-n^s/\ln n)$. Therefore the theorem is a direct consequence. ■

Theorem 5 *Consider regression problems indexed by the sample size n, and use notations in Definition 3. Let Assumption 1 hold, with noise σ independent of n. Assume that the weak irrepresentable condition is violated at sample sizes $n_1 < n_2 < \cdots$. There exist targets $\bar{\beta}^{(n_j)}$ with arbitrarily large $\min_{i \in \bar{F}^{(n_j)}} |\bar{\beta}_i^{(n_j)}|$, such that at each sample size n_j, Algorithm 1 chooses a basis $i^{(1)} \notin \bar{F}$ in the first step with probability larger than* 0.5.

Proof By definition of $\mu_X(\bar{F})$, there exists $\mathbf{v} = (X_{\bar{F}}^T X_{\bar{F}})\mathbf{u} \in \mathbb{R}^{|\bar{F}|}$ such that

$$\begin{aligned}
\mu_X(\bar{F}) &= \max_{j \notin \bar{F}} \|(X_{\bar{F}}^T X_{\bar{F}})^{-1} X_{\bar{F}}^T \mathbf{x}_j\|_1 \\
&= \max_{j \notin \bar{F}} \frac{|\mathbf{v}^T (X_{\bar{F}}^T X_{\bar{F}})^{-1} X_{\bar{F}}^T \mathbf{x}_j|}{\|\mathbf{v}\|_\infty} \\
&= \max_{j \notin \bar{F}} \frac{|\mathbf{u}^T X_{\bar{F}}^T \mathbf{x}_j|}{\|(X_{\bar{F}}^T X_{\bar{F}})\mathbf{u}\|_\infty} \\
&= \frac{\max_{j \notin F} |\mathbf{x}_j^T X_{\bar{F}} \mathbf{u}|}{\max_{i \in \bar{F}} |(\mathbf{x}_i^T X_{\bar{F}})\mathbf{u}|}.
\end{aligned}$$

Therefore if $\mu_X(\bar{F}) > 1$, we can find $\mathbf{u} \in \mathbb{R}^{|\bar{F}|}$ such that $\max_{j \notin F} |\mathbf{x}_j^T X_{\bar{F}} \mathbf{u}| > \max_{i \in \bar{F}} |(\mathbf{x}_i^T X_{\bar{F}})\mathbf{u}|$.

Consider an arbitrary sequence $\delta_n > 0$ ($n = 1, 2, \ldots$). At any sample size $n = n_j$, since $\mu_{X^{(n)}}(\bar{F}^{(n)}) > 1$, we can find a sufficiently large target vector $\bar{\beta}^{(n)}$ such that

$$\max_{i \in \bar{F}^{(n)}} |\mathbf{x}_i^T X^{(n)} \bar{\beta}^{(n)}| < \max_{j \notin \bar{F}^{(n)}} |\mathbf{x}_j^T X^{(n)} \bar{\beta}^{(n)}| - 2\delta_n. \tag{3}$$

Now we may take $\delta_n = \sigma\sqrt{2n\ln(4d_n)}$; then Lemma 8 implies that with probability larger than 0.5, $\max_{j \in \{1,\ldots,d\}} |\mathbf{x}_j^T (\mathbf{y} - X^{(n)}\bar{\beta}^{(n)})| \leq \delta_n$. Therefore (3) implies that

$$\max_{i \in \bar{F}^{(n)}} [|\mathbf{x}_i^T X^{(n)} \bar{\beta}^{(n)}| + |\mathbf{x}_i^T (\mathbf{y} - X^{(n)}\bar{\beta}^{(n)})|] < \max_{j \notin \bar{F}^{(n)}} [|\mathbf{x}_j^T X^{(n)} \bar{\beta}^{(n)}| - |\mathbf{x}_j^T (\mathbf{y} - X^{(n)}\bar{\beta}^{(n)})|].$$

Therefore

$$\max_{i \in \bar{F}^{(n)}} |\mathbf{x}_i^T \mathbf{y}| < \max_{j \notin \bar{F}^{(n)}} |\mathbf{x}_j^T \mathbf{y}|.$$

This means that we pick $i^{(1)} \notin \bar{F}$ in the first step with probability larger than 0.5. ■

5. Conclusion

We have shown that weak and strong target independent irrepresentable conditions are necessary and sufficient conditions for a greedy least squares regression algorithm to select features consistently. These conditions match the target independent versions of the necessary and sufficient conditions for Lasso by Zhao and Yu (2006).

Moreover, if the eigenvalue $\rho(\bar{F})$ is bounded away from zero, then the greedy algorithm can reliably identify features as long as each nonzero coefficient is larger than a constant times the noise level. In comparison, under the same condition, Lasso may require the coefficients to be larger than $O(\sqrt{s})$ times the noise level, where s is the number of nonzero coefficients. This implies that under some conditions, greedy least squares regression may potentially select features more effectively than Lasso in the presence of stochastic noise.

Although the target independent versions of the irrepresentable conditions for greedy least squares regression match those of Lasso, our result does not show which algorithm is better for any specific target. In fact, the target specific behaviors of the two algorithms are different, and one may be preferred over the other under different scenarios.

Acknowledgments

This paper is the outcome of some private discussion with Joel Tropp, which drew the author's attention to the similarities and differences between orthogonal matching pursuit and Lasso for estimating sparse signals.

Appendix A. Auxiliary Lemmas

The following technical lemmas from Zhang (2008) are needed to analyze the behavior of greedy least squares regression under stochastic noise. For completeness, we include them here with proofs.

The following lemma relates the squared error reduction in one greedy step to correlation coefficients.

Lemma 6 *For all* $\mathbf{x}, \mathbf{x}', \mathbf{y} \in \mathbb{R}^n$, *we have*

$$\inf_{\alpha \in \mathbb{R}} \|\mathbf{x} + \alpha \mathbf{x}' - \mathbf{y}\|_2^2 = \|\mathbf{x} - \mathbf{y}\|_2^2 - ((\mathbf{x} - \mathbf{y})^T \mathbf{x}')^2 / \|\mathbf{x}'\|_2^2.$$

Proof The equality follows from simple algebra with the optimal α achieved at $-(\mathbf{x} - \mathbf{y})^T \mathbf{x}' / \|\mathbf{x}'\|_2^2$. ■

The following lemma provides a bound on the squared error reduction of one forward greedy step. Some ingredients of the proof has appeared in Natarajan (1995).

Lemma 7 *Let Assumption 1 hold. Consider* $F \subset \bar{F} \subset \{1, \ldots, d\}$. *Let* $\beta' = \hat{\beta}_X(\bar{F}, \mathbf{y})$, $\beta = \hat{\beta}_X(F, \mathbf{y})$, $\mathbf{x}' = X\beta'$, *and* $\mathbf{x} = X\beta$. *Then*

$$\inf_{\alpha \in \mathbb{R}, j \in \bar{F} - F} \|\mathbf{x} + \alpha \mathbf{x}_j - \mathbf{y}\|_2^2 \leq \|\mathbf{x} - \mathbf{y}\|_2^2 - \frac{\rho(\bar{F})}{|\bar{F} - F|} \|\mathbf{x} - \mathbf{x}'\|_2^2.$$

Proof For all $j \in F$, we have $\|\mathbf{x} + \alpha \mathbf{x}_j - \mathbf{y}\|_2^2$ achieves the minimum at $\alpha = 0$. This implies that $(\mathbf{x}-\mathbf{y})^T\mathbf{x}_j = 0$ for $j \in F$. Therefore we have

$$\begin{aligned}&(\mathbf{x}-\mathbf{y})^T \sum_{j\in\bar{F}-F} (\beta'_j - \beta_j)\mathbf{x}_j \\ =&(\mathbf{x}-\mathbf{y})^T \sum_{j\in\bar{F}\cup F} (\beta'_j - \beta_j)\mathbf{x}_j = (\mathbf{x}-\mathbf{y})^T(\mathbf{x}'-\mathbf{x}) \\ =&-\|\mathbf{x}-\mathbf{x}'\|_2^2 + (\mathbf{x}'-\mathbf{y})^T(\mathbf{x}'-\mathbf{x}) = -\|\mathbf{x}-\mathbf{x}'\|_2^2.\end{aligned}$$

The last quality follows from the definition of $\beta' = \hat{\beta}_X(\bar{F},\mathbf{y})$ and $F \subset \bar{F}$, which implies that $(\mathbf{x}' - \mathbf{y})^T(\mathbf{x}'-\mathbf{x}) = 0$. Now, let $s' = |\bar{F} - F|$, then the above inequality leads to the following derivation $\forall \eta > 0$:

$$\begin{aligned}&s' \inf_{j\in\bar{F}-F} \|\mathbf{x} + \eta(\beta'_j - \beta_j)\mathbf{x}_j - \mathbf{y}\|_2^2 \\ \leq& \sum_{j\in\bar{F}-F} \|\mathbf{x} + \eta(\beta'_j - \beta_j)\mathbf{x}_j - \mathbf{y}\|_2^2 \\ =&s'\|\mathbf{x}-\mathbf{y}\|_2^2 + \eta^2 \sum_{j\in\bar{F}-F} (\beta'_j - \beta_j)^2\|\mathbf{x}_j\|_2^2 + 2\eta(\mathbf{x}-\mathbf{y})^T \sum_{j\in\bar{F}-F} (\beta'_j - \beta_j)\mathbf{x}_j \\ =&s'\|\mathbf{x}-\mathbf{y}\|_2^2 + n\eta^2 \sum_{j\in\bar{F}-F} (\beta'_j - \beta_j)^2 - 2\eta\|\mathbf{x}'-\mathbf{x}\|_2^2.\end{aligned}$$

Note that in the last equation, we have used $\|\mathbf{x}_j\|_2^2 = n$ in Assumption 1. By optimizing over η, we obtain

$$\begin{aligned}&s' \inf_{j\in\bar{F}-F} \|\mathbf{x} + \eta(\beta'_j - \beta_j)\mathbf{x}_j - \mathbf{y}\|_2^2 \\ \leq& s'\|\mathbf{x}-\mathbf{y}\|_2^2 - \frac{\|\mathbf{x}'-\mathbf{x}\|_2^4}{n\sum_{j\in\bar{F}}(\beta'_j - \beta_j)^2} \leq s'\|\mathbf{x}-\mathbf{y}\|_2^2 - \rho(\bar{F})\|\mathbf{x}'-\mathbf{x}\|_2^2.\end{aligned}$$

This leads to the lemma. ■

The following lemma is a standard empirical processes bound for sub-Gaussian random variables. The bound is used to derive probability estimates in our analysis.

Lemma 8 *Consider n independent random variables* $\xi_1, \ldots, \xi_n$ *such that* $\mathbf{E}e^{t(\xi_i - \mathbf{E}\xi_i)} \leq e^{\sigma_i^2 t^2/2}$ *for all t and i. Consider* $g_{i,j}$ *for* $i = 1, \ldots, n$ *and* $j = 1, \ldots, m$, *we have for all* $\eta \in (0,1)$, *with probability larger than* $1 - \eta$*:*

$$\sup_j \left|\sum_{i=1}^n g_{i,j}(\xi_i - \mathbf{E}\xi_i)\right| \leq a\sqrt{2\ln(2m/\eta)},$$

where $a^2 = \sup_j \sum_{i=1}^n g_{i,j}^2 \sigma_i^2$.

Proof For a fixed j, we let $s_j = \sum_{i=1}^n g_{i,j}(\xi_i - \mathbf{E}\xi_i)$; then by assumption, $\mathbf{E}(e^{ts_j} + e^{-ts_j}) \leq 2e^{a^2t^2/2}$, which implies that for all $\varepsilon > 0$: $P(|s_j| \geq \varepsilon)e^{t\varepsilon} \leq 2e^{a^2t^2/2}$. Now let $t = \varepsilon/a^2$, we obtain:

$$P\left(\left|\sum_{i=1}^n g_{i,j}(\xi_i - \mathbf{E}\xi_i)\right| \geq \varepsilon\right) \leq 2e^{-\varepsilon^2/2a^2}.$$

This implies that

$$P\left[\sup_j \left|\sum_{i=1}^n g_{i,j}(\xi_i - \mathbf{E}\xi_i)\right| \geq \varepsilon\right] \leq m \sup_j P\left[\left|\sum_{i=1}^n g_{i,j}(\xi_i - \mathbf{E}\xi_i)\right| \geq \varepsilon\right] \leq 2me^{-\varepsilon^2/(2a^2)}.$$

This implies the lemma. ■

The following lemma gives a bound on the infinity norm of the difference between the estimated parameter $\hat{\beta}_X(\bar{F},\mathbf{y})$ and the true parameter $\bar{\beta}$ when the set of features $\bar{F}$ are known in advance.

Lemma 9 *Let Assumption 1 hold. Consider any fixed* $\bar{F} \subset \{1,\ldots,d\}$. *For all* $\eta \in (0,1)$, *with probability larger than* $1-\eta$, *we have*

$$\|\hat{\beta}_X(\bar{F},\mathbf{y}) - \hat{\beta}_X(\bar{F},\mathbf{E}\mathbf{y})\|_\infty \leq \sigma\sqrt{(2\ln(2|\bar{F}|/\eta))/(n\rho(\bar{F}))}.$$

Proof For simplicity, let $G = X_{\bar{F}}$ and $\bar{k} = |\bar{F}|$. Let $\hat{\beta}' \in \mathbb{R}^{\bar{k}}$ and $\bar{\beta}' \in \mathbb{R}^{\bar{k}}$ be the restrictions of $\hat{\beta}_X(\bar{F},\mathbf{y}) \in \mathbb{R}^d$ and $\hat{\beta}_X(\bar{F},\mathbf{E}\mathbf{y}) \in \mathbb{R}^d$ to $\bar{F}$ respectively. Algebraically, we have $\hat{\beta}' = (G^TG)^{-1}G^T\mathbf{y}$ and $\bar{\beta}' = (G^TG)^{-1}G^T\mathbf{E}\mathbf{y}$. It follows that

$$\hat{\beta}' - \bar{\beta}' = (G^TG)^{-1}G^T(\mathbf{y} - \mathbf{E}\mathbf{y}).$$

Therefore

$$|\hat{\beta}'_j - \bar{\beta}'_j| = |\mathbf{e}_j^T(G^TG)^{-1}G^T(\mathbf{y} - \mathbf{E}\mathbf{y})|.$$

Lemma 8 implies that with probability larger than $1-\eta$, for all j:

$$|\mathbf{e}_j^T(G^TG)^{-1}G^T(\mathbf{y} - \mathbf{E}\mathbf{y})| \leq \sigma\|\mathbf{e}_j^T(G^TG)^{-1}G^T\|_2\sqrt{2\ln(2\bar{k}/\eta)}.$$

Since by definition, $\rho(\bar{F})n$ is no larger than the smallest eigenvalue of G^TG, the desired inequality follows from the estimate

$$\|\mathbf{e}_j^T(G^TG)^{-1}G^T\|_2^2 = \mathbf{e}_j^T(G^TG)^{-1}\mathbf{e}_j \leq 1/(n\rho(\bar{F})).$$

■

The following lemma estimates the correlation coefficient of $\mathbf{x}_j$ with $j \notin \bar{F}$.

Lemma 10 *Let Assumption 1 hold. Assume also that* $\mathbf{E}\mathbf{y} = X\bar{\beta}$ *with* $\operatorname{supp}(\bar{\beta}) \subset \bar{F} \subset \{1,\ldots,d\}$. *For all* $\eta \in (0,1)$, *with probability larger than* $1-\eta$, *we have*

$$\max_{j\notin\bar{F}} |(X\hat{\beta}_X(\bar{F},\mathbf{y}) - \mathbf{y})^T\mathbf{x}_j| \leq \sigma\sqrt{2n\ln(2d/\eta)}.$$

Proof Let P be the projection operator to the subspace spanned by $\{\mathbf{x}_j : j \in \bar{F}\}$ on $\mathbb{R}^n$. Lemma 8 implies that with probability larger than $1-\eta$:

$$\sup_{j=1,\ldots,d} |(\mathbf{y} - \mathbf{E}\mathbf{y})^T(I-P)\mathbf{x}_j| \leq \sigma\sqrt{2n\ln(2d/\eta)},$$

where we have used $\|\mathbf{x}_j\|_2 = \sqrt{n}$. Let $\hat{\mathbf{x}} = X\hat{\beta}_X(\bar{F},\mathbf{y})$. Since $(\hat{\mathbf{x}} - \mathbf{y})^T\mathbf{x}_j = 0$ for all $j \in \bar{F}$, we have $\hat{\mathbf{x}} = P\mathbf{y}$. Moreover, since $(I-P)\mathbf{E}\mathbf{y} = (I-P)X\bar{\beta} = 0$, we have

$$(\mathbf{y} - \mathbf{E}\mathbf{y})^T(I-P)\mathbf{x}_j = ((I-P)\mathbf{y} - (I-P)\mathbf{E}\mathbf{y})^T\mathbf{x}_j = (\mathbf{y} - \hat{\mathbf{x}})^T\mathbf{x}_j.$$

Combine this equation with the previous inequality, we obtain the desired bound. ■

Appendix B. Proof of Theorem 2

Before the formal proof, we give a brief outline of the main argument. First, Lemma 10 implies that with large probability, $\max_{j\notin\bar{F}} |(X\hat{\beta}_X(\bar{F},\mathbf{y}) - \mathbf{y})^T\mathbf{x}_j|$ is small. Using this fact, and the assumption that $\mu_X(\bar{F}) < 1$, it follows from Lemma 11 below that when $\mathrm{supp}(\beta^{(k-1)}) \subset \bar{F}$, either $\max_i |(X\beta^{(k-1)} - \mathbf{y})^T\mathbf{x}_i|$ is sufficiently small (which implies that the greedy procedure stops and $\beta^{(k-1)} \approx \bar{\beta}$), or $\max_{j\notin\bar{F}} |(X\beta^{(k-1)} - \mathbf{y})^T\mathbf{x}_j| < \max_{i\in\bar{F}} |(X\beta^{(k-1)} - \mathbf{y})^T\mathbf{x}_i|$ (which implies that the greedy procedure chooses a direction $i^{(k)} \in \bar{F}$ in the next iteration). The claims of the theorem then follow by induction.

We start the formal proof by introducing the following critical lemma, which generalizes the essential idea of Tropp (2004). Note that the result there only considered the case $X\hat{\beta}_X(\bar{F},\mathbf{y}) = \mathbf{y}$.

Lemma 11 *Consider* $\beta \in \mathbb{R}^d$ *such that* $\mathrm{supp}(\beta) \subset \bar{F}$. *We have*

$$\max_{j\notin\bar{F}} |(X\beta - \mathbf{y})^T\mathbf{x}_j| \le \max_{j\notin\bar{F}} |(X\hat{\beta}_X(\bar{F},\mathbf{y}) - \mathbf{y})^T\mathbf{x}_j| + \mu_X(\bar{F}) \max_{i\in\bar{F}} |(X\beta - \mathbf{y})^T\mathbf{x}_i|.$$

Proof Let $\beta' = \hat{\beta}_X(\bar{F},\mathbf{y})$. Note that $(X\beta' - \mathbf{y})^T\mathbf{x}_i = 0$ when $i \in \bar{F}$. Therefore

$$\max_{i\in\bar{F}} |(X\beta - \mathbf{y})^T\mathbf{x}_i| = \max_{i\in\bar{F}} |\mathbf{x}_i^T X(\beta - \beta')| = \|X_{\bar{F}}^T X_{\bar{F}}(\beta - \beta')\|_\infty.$$

Let $\mathbf{v} = X_{\bar{F}}^T X_{\bar{F}}(\beta - \beta')$, then the definition of $\mu_X(\bar{F})$ implies that

$$\mu_X(\bar{F}) \ge \max_{j\notin\bar{F}} \frac{|\mathbf{x}_j^T X_{\bar{F}}(X_{\bar{F}}^T X_{\bar{F}})^{-1}\mathbf{v}|}{\|\mathbf{v}\|_\infty} = \frac{\max_{j\notin\bar{F}} |\mathbf{x}_j^T X_{\bar{F}}(\beta - \beta')|}{\|X_{\bar{F}}^T X_{\bar{F}}(\beta - \beta')\|_\infty}.$$

We obtain from the above

$$\max_{j\notin\bar{F}} |\mathbf{x}_j^T X(\beta - \beta')| \le \mu_X(\bar{F})\|X_{\bar{F}}^T X_{\bar{F}}(\beta - \beta')\|_\infty = \mu_X(\bar{F}) \max_{i\in\bar{F}} |(X\beta - \mathbf{y})^T\mathbf{x}_i|.$$

Now, the lemma follows from the simple inequality

$$\max_{j\notin\bar{F}} |(X\beta - \mathbf{y})^T\mathbf{x}_j| \le \max_{j\notin\bar{F}} |\mathbf{x}_j^T X(\beta - \beta')| + \max_{j\notin\bar{F}} |\mathbf{x}_j^T(X\beta' - \mathbf{y})|$$

■

Now we are ready to prove the theorem. From Lemma 9, we obtain with probability larger than $1-\eta$,

$$\|\hat{\beta}_X(\bar{F},\mathbf{y}) - \bar{\beta}\|_\infty \le \sigma\sqrt{(2\ln(2|\bar{F}|/\eta))/(n\rho(\bar{F}))}.$$

From Lemma 10, we obtain with probability larger than $1-\eta$,

$$\max_{j\notin\bar{F}} |(X(\hat{\beta}_X(\bar{F},\mathbf{y}))-\mathbf{y})^T\tilde{\mathbf{x}}_j| \leq \sigma\sqrt{2\ln(2d/\eta)} < (1-\mu_X(\bar{F}))\varepsilon. \qquad (4)$$

With probability larger than $1-2\eta$, both claims hold.

We now proceed by induction on k to show that $F^{(k-1)}\subset\bar{F}$ before the procedure stops. Assume the claim is true after $k-1$ steps for $k\geq 1$. By induction hypothesis, we have $F^{(k-1)}\subset\bar{F}$ at the beginning of step k. Therefore Lemma 7 implies

$$\min_{\alpha,i\in\bar{F}} \|X\beta^{(k-1)}+\alpha\mathbf{x}_i-\mathbf{y}\|_2^2 \leq \|X\beta^{(k-1)}-\mathbf{y}\|_2^2 - \frac{\rho(\bar{F})}{|\bar{F}-F^{(k-1)}|}\|X(\beta^{(k-1)}-\hat{\beta}_X(\bar{F},\mathbf{y}))\|_2^2.$$

Using the above inequality, we obtain from Lemma 6 the following bound on the correlation coefficients:

$$\begin{aligned}\max_{i\in\bar{F}} |(X\beta^{(k-1)}-\mathbf{y})^T\tilde{\mathbf{x}}_i| &= \left(\|X\beta^{(k-1)}-\mathbf{y}\|_2^2 - \min_{\alpha,i\in\bar{F}}\|X\beta^{(k-1)}+\alpha\mathbf{x}_i-\mathbf{y}\|_2^2\right)^{1/2}\\ &\geq \frac{\sqrt{\rho(\bar{F})}}{\sqrt{|\bar{F}-F^{(k-1)}|}}\|X(\beta^{(k-1)}-\hat{\beta}_X(\bar{F},\mathbf{y}))\|_2\\ &\geq \frac{\sqrt{n}\rho(\bar{F})}{\sqrt{|\bar{F}-F^{(k-1)}|}}\|\beta^{(k-1)}-\hat{\beta}_X(\bar{F},\mathbf{y})\|_2.\end{aligned}$$

We only need to consider the following two scenarios:

- $\|\beta^{(k-1)}-\hat{\beta}_X(\bar{F},\mathbf{y})\|_2 > \varepsilon\rho(\bar{F})^{-1}\sqrt{|\bar{F}-F^{(k-1)}|/n}$.

  In this case, we have

  $$\max_{i\in\bar{F}}|(X\beta^{(k-1)}-\mathbf{y})^T\tilde{\mathbf{x}}_i| > \varepsilon > \max_{j\notin\bar{F}}|(X\hat{\beta}_X(\bar{F},\mathbf{y})-\mathbf{y})^T\tilde{\mathbf{x}}_j|/(1-\mu_X(\bar{F})).$$

  The second inequality is due to (4). Now by combining this estimate with Lemma 11, we obtain

  $$\max_{j\notin\bar{F}}|(X\beta^{(k-1)}-\mathbf{y})^T\tilde{\mathbf{x}}_j| < \max_{i\in\bar{F}}|(X\beta^{(k-1)}-\mathbf{y})^T\tilde{\mathbf{x}}_i|.$$

  This implies that $i^{(k)}\in\bar{F}$ and the procedure does not stop.

- $\|\beta^{(k-1)}-\hat{\beta}_X(\bar{F},\mathbf{y})\|_2 \leq \varepsilon\rho(\bar{F})^{-1}\sqrt{|\bar{F}-F^{(k-1)}|/n}$.

  In this case, we have the following scenarios:

  1. $i^{(k)}\notin\bar{F}$: By Lemma 11, we must have:

  $$\begin{aligned}\max_{i\in\bar{F}}|(X\beta-\mathbf{y})^T\tilde{\mathbf{x}}_i| &\leq \max_{j\notin\bar{F}}|(X\beta-\mathbf{y})^T\tilde{\mathbf{x}}_j|\\ &\leq \max_{j\notin\bar{F}}|(X\hat{\beta}_X(\bar{F},\mathbf{y})-\mathbf{y})^T\tilde{\mathbf{x}}_j| + \mu_X(\bar{F})\max_{i\in\bar{F}}|(X\beta-\mathbf{y})^T\tilde{\mathbf{x}}_i|\\ &\leq \max_{j\notin\bar{F}}|(X\hat{\beta}_X(\bar{F},\mathbf{y})-\mathbf{y})^T\tilde{\mathbf{x}}_j| + \mu_X(\bar{F})\max_{j\notin\bar{F}}|(X\beta-\mathbf{y})^T\tilde{\mathbf{x}}_j|.\end{aligned}$$

Therefore

$$\max_{j\notin\bar{F}} |(X\beta-\mathbf{y})^T\tilde{\mathbf{x}}_j| \leq \frac{1}{1-\mu_X(\bar{F})} \max_{j\notin\bar{F}} |(X\hat{\beta}_X(\bar{F},\mathbf{y})-\mathbf{y})^T\tilde{\mathbf{x}}_j| < \varepsilon.$$

The last inequality is due to (4). This implies that the procedure stops.

2. $i^{(k)} \in \bar{F}$ and the procedure does not stop.
3. $i^{(k)} \in \bar{F}$ and the procedure stops.

The above situations imply that if the procedure does not stop, then $i^{(k)} \in \bar{F}$. If the procedure stops, then

$$\|\beta^{(k-1)} - \hat{\beta}_X(\bar{F},\mathbf{y})\|_2 \leq \varepsilon\rho(\bar{F})^{-1}\sqrt{|\bar{F}-F^{(k-1)}|/n}.$$

Therefore by induction, when the procedure stops, the following three claims hold:

- $F^{(k-1)} \subset \bar{F}$.
- $\|\beta^{(k-1)} - \hat{\beta}_X(\bar{F},\mathbf{y})\|_2 \leq \varepsilon\rho_X(\bar{F})^{-1}\sqrt{|\bar{F}-F^{(k-1)}|/n}$.
- $\|\hat{\beta}_X(\bar{F},\mathbf{y}) - \bar{\beta}\|_\infty \leq \sigma\sqrt{(2\ln(2|\bar{F}|/\eta))/(n\rho_X(\bar{F}))} < \varepsilon/\sqrt{n\rho_X(\bar{F})}$.

Now, we can let $\gamma = \sqrt{8}\varepsilon\rho_X(\bar{F})^{-1}/\sqrt{n}$, then the above claims imply that

$$\begin{aligned}
&\gamma\sqrt{|\{j\in\bar{F}-F^{(k-1)}:|\bar{\beta}_j|\geq\gamma\}|}\\
\leq&\left[\sum_{j\in\bar{F}-F^{(k-1)}}|\bar{\beta}_j|^2\right]^{1/2}\\
\leq&\left[\sum_{j\in\bar{F}-F^{(k-1)}}|\bar{\beta}_j-\hat{\beta}_X(\bar{F},\mathbf{y})|^2\right]^{1/2}+\left[\sum_{j\in\bar{F}-F^{(k-1)}}|\hat{\beta}_X(\bar{F},\mathbf{y})|^2\right]^{1/2}\\
\leq&\sqrt{|\bar{F}-F^{(k-1)}|}\|\bar{\beta}-\hat{\beta}_X(\bar{F},\mathbf{y})\|_\infty+\|\beta^{(k-1)}-\hat{\beta}_X(\bar{F},\mathbf{y})\|_2\\
<&\sqrt{|\bar{F}-F^{(k-1)}|/(n\rho_X(\bar{F}))}\varepsilon+\varepsilon\rho_X(\bar{F})^{-1}\sqrt{|\bar{F}-F^{(k-1)}|/n}\\
\leq&2\varepsilon\rho_X(\bar{F})^{-1}\sqrt{|\bar{F}-F^{(k-1)}|/n}=\gamma\sqrt{|\bar{F}-F^{(k-1)}|/2}.
\end{aligned}$$

Therefore

$$2|\{j\in\bar{F}-F^{(k-1)}:|\bar{\beta}_j|\geq\gamma\}| \leq |\bar{F}-F^{(k-1)}|,$$

which implies that

$$\begin{aligned}
|\bar{F}-F^{(k-1)}| =&2|\bar{F}-F^{(k-1)}|-|\bar{F}-F^{(k-1)}|\\
\leq&2|\bar{F}-F^{(k-1)}|-2|\{j\in\bar{F}-F^{(k-1)}:|\bar{\beta}_j|\geq\gamma\}|\\
=&2|\{j\in\bar{F}-F^{(k-1)}:|\bar{\beta}_j|<\gamma\}|\\
\leq&2|\{j\in\bar{F}:|\bar{\beta}_j|<\gamma\}|.
\end{aligned}$$

This proves the theorem.

References

S. Mallat and Z. Zhang. Matching pursuits with time-frequency dictionaries. *IEEE Transactions on Signal Processing*, 41(12):3397–3415, 1993.

Nicolai Meinshausen and Peter Buhlmann. High-dimensional graphs and variable selection with the Lasso. *The Annals of Statistics*, 34:1436–1462, 2006.

Nicolai Meinshausen and Bin Yu. Lasso-type recovery of sparse representations for high-dimensional data. *Annals of Statistics*, 2008. to appear.

B. K. Natarajan. Sparse approximate solutions to linear systems. *SIAM J. Comput.*, 24(2):227–234, 1995. ISSN 0097-5397.

Joel A. Tropp. Greed is good: Algorithmic results for sparse approximation. *IEEE Trans. Info. Theory*, 50(10):2231–2242, 2004.

Martin Wainwright. Sharp thresholds for high-dimensional and noisy recovery of sparsity. Technical report, Department of Statistics, UC. Berkeley, 2006.

Tong Zhang. Some sharp performance bounds for least squares regression with L_1 regularization. *The Annals of Statistics*, 2009. to appear.

Tong Zhang. Forward-backward greedy algorithm for learning sparse representations. Technical report, Rutgers Statistics Department, 2008. A short version is to appear in NIPS 08.

Peng Zhao and Bin Yu. On model selection consistency of Lasso. *Journal of Machine Learning Research*, 7:2541–2567, 2006.

Journal of Machine Learning Research 10 (2009) 569-590 Submitted 7/08; Revised 1/09; Published 3/09

Online Learning with Sample Path Constraints

Shie Mannor[*] SHIE.MANNOR@MCGILL.CA
Department of Electrical and Computer Engineering
McGill University
Québec H3A-2A7

John N. Tsitsiklis JNT@MIT.EDU
Laboratory for Information and Decision Systems
Massachusetts Institute of Technology
Cambridge, MA 02139

Jia Yuan Yu JIA.YU@MCGILL.CA
Department of Electrical and Computer Engineering
McGill University
Québec H3A-2A7

Editor: Gábor Lugosi

Abstract

We study online learning where a decision maker interacts with Nature with the objective of maximizing her long-term average reward subject to some sample path average constraints. We define the reward-in-hindsight as the highest reward the decision maker could have achieved, while satisfying the constraints, had she known Nature's choices in advance. We show that in general the reward-in-hindsight is *not* attainable. The convex hull of the reward-in-hindsight function is, however, attainable. For the important case of a single constraint, the convex hull turns out to be the highest attainable function. Using a calibrated forecasting rule, we provide an explicit strategy that attains this convex hull. We also measure the performance of heuristic methods based on non-calibrated forecasters in experiments involving a CPU power management problem.

Keywords: online learning, calibration, regret minimization, approachability

1. Introduction

We consider a repeated game from the viewpoint of a decision maker (player P1) who plays against Nature (player P2). The opponent (Nature) is "arbitrary" in the sense that player P1 has no prediction, statistical or strategic, of the opponent's choice of actions. This setting was considered by Hannan (1957), in the context of repeated matrix games. Hannan introduced the Bayes utility with respect to the current empirical distribution of the opponent's actions, as a performance goal for adaptive play. This quantity, defined as the highest average reward that player P1 could have achieved, in hindsight, by playing some fixed action against the observed action sequence of player P2. Player P1's *regret* is defined as the difference between the highest average reward-in-hindsight that player P1 could have hypothetically achieved, and the actual average reward obtained by player P1. It was established in Hannan (1957) that there exist strategies whose regret converges to zero as

[*]. Also at Department of Electrical Engineering, Technion, Israel, 32000.

the number of stages increases, even in the absence of any prior knowledge on the strategy of player P2. For recent advances on online learning, see Cesa-Bianchi and Lugosi (2006).

In this paper we consider regret minimization under sample-path constraints. That is, in addition to maximizing the average reward, or more precisely, minimizing the regret, the decision maker has some side constraints that need to be satisfied on the average. In particular, for every joint action of the players, there is an additional penalty vector that is accumulated by the decision maker. The decision maker has a predefined set in the space of penalty vectors, which represents the acceptable tradeoffs between the different components of the penalty vector. An important special case arises when the decision maker wishes to keep some constrained resource below a certain threshold. Consider, for example, a wireless communication system where the decision maker can adjust the transmission power to improve the probability that a message is received successfully. Of course, the decision maker does not know a priori how much power will be needed (this depends on the behavior of other users, the channel conditions, etc.). Still, a decision maker is usually interested in both the rate of successful transmissions, and in the average power consumption. In an often considered variation of this problem, the decision maker wishes to maximize the transmission rate, while keeping the average power consumption below some predefined threshold. We refer the reader to Mannor and Shimkin (2004) and references therein for a discussion of constrained average cost stochastic games and to Altman (1999) for constrained Markov decision problems. We note that the reward and the penalty are not treated the same; otherwise they could have been combined into a single scalar value, resulting in a much simpler problem.

The paper is organized as follows. In Section 2, we present formally the basic model, and provide a result that relates attainability with the value of the game. In Section 3, we provide an example where the reward-in-hindsight cannot be attained. In light of this negative result, in Section 4 we define the closed convex hull of the reward-in-hindsight, and show that it is attainable. Furthermore, in Section 5, we show that when there is a single constraint, this is the maximal attainable objective. In Section 6, we provide a simple strategy, based on calibrated forecasting, that attains the closed convex hull. Section 7 presents heuristic algorithms derived from an online forecaster, while incorporating strictly enforced constraints. The application of the algorithms of Section 7 to a power management domain is presented in Section 8. We finally conclude in Section 9 with some open questions and directions for future research.

2. Problem Definition

We consider a repeated game against Nature, in which a decision maker tries to maximize her reward, while satisfying some constraints on certain time-averages. The underlying stage game is a game with two players: P1 (the decision maker of interest) and P2 (who represents Nature and is assumed arbitrary). For our purposes, we only need to define rewards and constraints for P1.

A constrained game with respect to a set T is defined by a tuple (A,B,R,C,T) where:

1. A is the set of actions of P1; we will assume $A = \{1,2,\ldots,|A|\}$.

2. B is the set of actions of P2; we will assume $B = \{1,2,\ldots,|B|\}$.

3. R is an $|A| \times |B|$ matrix where the entry $R(a,b)$ denotes the expected reward obtained by P1, when P1 plays action $a \in A$ and P2 action $b \in B$. The actual rewards obtained at each play of actions a and b are assumed to be IID random variables, with finite second moments,

distributed according to a probability law $\Pr_R(\cdot\,|\,a,b)$. Furthermore, the reward streams for different pairs (a,b) are statistically independent.

4. C is an $|A| \times |B|$ matrix, where the entry $C(a,b)$ denotes the expected d-dimensional penalty vector incurred by P1, when P1 plays action $a \in A$ and P2 action $b \in B$. The actual penalty vectors obtained at each play of actions a and b are assumed to be IID random variables, with finite second moments, distributed according to a probability law $\Pr_C(\cdot\,|\,a,b)$. Furthermore, the penalty vector streams for different pairs (a,b) are statistically independent.

5. T is a set in $\mathbb{R}^d$ within which we wish the average of the penalty vectors to lie. We assume that T is convex and closed. Since the entries of C are bounded, we will also assume, without loss of generality, that T is bounded.

The game is played in stages. At each stage t, P1 and P2 simultaneously choose actions $a_t \in A$ and $b_t \in B$, respectively. Player P1 obtains a reward r_t, distributed according to $\Pr_R(\cdot\,|\,a_t,b_t)$, and a penalty c_t, distributed according to $\Pr_C(\cdot\,|\,a_t,b_t)$. We define P1's average reward by time t to be

$$\hat{r}_t = \frac{1}{t}\sum_{\tau=1}^{t} r_\tau, \tag{1}$$

and P1's average penalty vector by time t to be

$$\hat{c}_t = \frac{1}{t}\sum_{\tau=1}^{t} c_\tau. \tag{2}$$

A *strategy* for P1 (resp. P2) is a mapping from the set of all possible past histories to the set of mixed actions on A (resp. B), which prescribes the (mixed) action of that player at each time t, as a function of the history in the first $t-1$ stages. Loosely, P1's goal is to maximize the average reward while having the average penalty vector converge to T, pathwise:

$$\limsup_{t\to\infty} \operatorname{dist}(\hat{c}_t, T) \to 0, \qquad \text{a.s.}, \tag{3}$$

where $\operatorname{dist}(\cdot)$ is the point-to-set Euclidean distance, that is, $\operatorname{dist}(x,T) = \inf_{y\in T}\|y-x\|_2$, and the probability measure is the one induced by the policy of P1, the policy of P2, and the randomness in the rewards and penalties.

We will often consider the important special case where $T = \{c \in \mathbb{R}^d : c \le c_0\}$, for some given $c_0 \in \mathbb{R}^d$, with the inequality interpreted component-wise. We simply call such a game a constrained game with respect to (a vector) c_0. For that special case, the requirement (3) is equivalent to:

$$\limsup_{t\to\infty} \hat{c}_t \le c_0, \quad \text{a.s.}.$$

For a set D, we will use the notation $\Delta(D)$ to denote the set of all probability measures on D. If D is finite, we will identify $\Delta(D)$ with the set of probability vectors of the same size as D. If D is a subset of Euclidean space, we will assume that it is endowed with the Borel σ-field.

2.1 Reward-in-hindsight

We define $\hat{q}_t \in \Delta(B)$ as the empirical distribution of P2's actions by time t, that is,

$$\hat{q}_t(b) = \frac{1}{t}\sum_{\tau=1}^{t} 1_{\{b_t = b\}}, \quad b \in B. \tag{4}$$

If P1 knew in advance that $\hat{q}_t$ will equal q, and if P1 were restricted to using a fixed action, then P1 would pick an optimal response (generally a mixed action) to the mixed action q, subject to the constraints specified by T. In particular, P1 would solve the convex program[1]

$$\begin{aligned} \max_{p \in \Delta(A)} \quad & \sum_{a,b} p(a)q(b)R(a,b), \\ \text{s.t.} \quad & \sum_{a,b} p(a)q(b)C(a,b) \in T. \end{aligned} \tag{5}$$

By playing a p that solves this convex program, P1 would meet the constraints (up to small fluctuations that are a result of the randomness and the finiteness of t), and would obtain the maximal average reward. We are thus led to define P1's reward-in-hindsight, which we denote by $r^* : \Delta(B) \mapsto \mathbb{R}$, as the optimal objective value in the program (5), as a function of q. The function r^* is often referred to as the *Bayes envelope*.

For the special case of a constrained game with respect to a vector c_0, the convex constraint $\sum_{a,b} p(a)q(b)C(a,b) \in T$ is replaced by $\sum_{a,b} p(a)q(b)C(a,b) \leq c_0$ (the inequality is to be interpreted component-wise).

The following examples show some of the properties of the Bayes envelope. Consider a 2×2 constrained game with respect to a scalar c_0 specified by:

$$\begin{pmatrix} (1,0) & (0,1) \\ (0,1) & (1,0) \end{pmatrix},$$

where each entry (pair) corresponds to $(R(a,b), C(a,b))$ for a pair of actions a and b. (Here a and b correspond to a choice of row and column, respectively.) Suppose first that $c_0 = 1$. In that case the constraint does not play a part in the problem, and we are dealing with a version of the matching pennies game. So, if we identify q with the frequency of the first action, we have that $r^*(q) = \max(q, 1-q)$. Suppose now that $c_0 = 1/2$. In this case, it is not difficult to show that $r^*(q) = 1/2$, since P1 cannot take advantage of any deviation from $q = 1/2$ while satisfying the constraint.

The next example involves a game where P2's action does not affect the constraints; such games are further discussed in Section 4.1. Consider a 2×2 constrained game with respect to a scalar c_0, specified by:

$$\begin{pmatrix} (1,1) & (0,1) \\ (0,0) & (1,0) \end{pmatrix},$$

1. If T is a polyhedron (specified by finitely many linear inequalities), then the optimization problem is a linear program.

where each entry (pair) corresponds to $(R(a,b),C(a,b))$ for a pair of actions a and b. We identify q with the frequency of the second action of P2 as before. Suppose first that $c_0 = 1$. As before, the constraint has no effect and $r^*(q) = \max(q, 1-q)$. Suppose now that $c_0 = 1/2$. It is not hard to show that in this case $r^*(q) = \max(q, 1/2)$. Finally, if $c_0 = 0$, P1 is forced to choose the second action; in this case, $r^*(q) = q$. The monotonicity of $r^*(q)$ in c_0 is to be expected since the lower c_0 is, the more stringent the constraint in Eq. (5).

2.2 The Objective

Formally, our goal is to attain a function r in the sense of the following definition. Naturally, the higher the function r, the better.

Definition 1 *A function $r : \Delta(B) \mapsto \mathbb{R}$ is* attainable *by P1 in a constrained game with respect to a set T if there exists a strategy σ of P1 such that for every strategy ρ of P2:*

(i) $\liminf_{t\to\infty}(\hat{r}_t - r(\hat{q}_t)) \geq 0$, *a.s., and*

(ii) $\limsup_{t\to\infty} \operatorname{dist}(\hat{c}_t, T) \to 0$, *a.s.,*

where the almost sure convergence is with respect to the probability measure induced by σ and ρ.

In constrained games with respect to a vector c_0 we can replace (ii) in the definition with

$$\limsup_{t\to\infty} \hat{c}_t \leq c_0, \quad \text{a.s.}$$

2.3 The Value of the Game

In this section, we consider the attainability of a constant function $r : \Delta(B) \mapsto \mathbb{R}$, that is, $r(q) = \alpha$, for all q. We will establish that attainability is equivalent to having $\alpha \leq \mathrm{v}$, where v is a naturally defined "value of the constrained game."

We first introduce the assumption that P1 is always able to satisfy the constraint.

Assumption 1 *For every mixed action $q \in \Delta(B)$ of P2, there exists a mixed action $p \in \Delta(A)$ of P1, such that:*

$$\sum_{a,b} p(a)q(b)C(a,b) \in T. \tag{6}$$

For constrained games with respect to a vector c_0, the condition (6) reduces to the inequality $\sum_{a,b} p(a)q(b)C(a,b) \leq c_0$.

If Assumption 1 is not satisfied, then P2 can choose a q such that for every (mixed) action of P1, the constraint is violated in expectation. By repeatedly playing this q, P1's average penalty vector will be outside T, and the objectives of P1 will be impossible to meet.

The following result deals with the attainability of the value, v, of an average reward repeated constrained game, defined by

$$\mathrm{v} = \inf_{q\in\Delta(B)} \sup_{p\in\Delta(A):\sum_{a,b} p(a)q(b)C(a,b)\in T} \sum_{a,b} p(a)q(b)R(a,b). \tag{7}$$

The existence of a strategy for P1 that attains the value was proven in Shimkin (1994) in the broader context of stochastic games.

Proposition 2 *Suppose that Assumption 1 holds. Then,*

(i) P1 has a strategy that guarantees that the constant function $r(q) \equiv \mathrm{v}$ is attained with respect to T.

(ii) For every number $\mathrm{v}' > \mathrm{v}$ there exists $\delta > 0$ such that P2 has a strategy that guarantees that either $\liminf_{t\to\infty} \hat{r}_t < \mathrm{v}' - \delta$ or $\limsup_{t\to\infty} \mathrm{dist}(\hat{c}_t, T) > \delta$, almost surely. (In particular, the constant function v' is not attainable.)

Proof The proof relies on Blackwell's approachability theory (Blackwell, 1956a). We construct a nested family of convex sets in $\mathbb{R}^{d+1}$ defined by $S_\alpha = \{(r,c) \in \mathbb{R} \times \mathbb{R}^d : r \geq \alpha, c \in T\}$. Obviously, $S_\alpha \subset S_\beta$ for $\alpha > \beta$. Consider the vector-valued game in $\mathbb{R}^{d+1}$ associated with the constrained game. In this game, P1's vector-valued payoff at time t is the $d+1$ dimensional vector $m_t = (r_t, c_t)$ and P1's average vector-valued payoff is $\hat{m}_t = (\hat{r}_t, \hat{c}_t)$. Since S_α is convex, it follows from approachability theory for convex sets (Blackwell, 1956a) that each S_α is either approachable[2] or excludable.[3] If S_α is approachable, then S_β is approachable for every $\beta < \alpha$. We define $\mathrm{v}_0 = \sup\{\beta \mid S_\beta \text{ is approachable}\}$. It follows that S_{v_0} is approachable (as the limit of approachable sets; see Spinat, 2002). By Blackwell's theorem, for every $q \in \Delta(B)$, an approachable convex set must intersect the set of feasible payoff vectors when P2 plays q. Using this fact, it is easily shown that v_0 equals v, as defined by Eq. (7), and part (i) follows. Part (ii) follows because a convex set which is not approachable is excludable. ■

Note that part (ii) of the proposition implies that, essentially, v is the highest average reward P1 can attain while satisfying the constraints, if P2 plays an adversarial strategy. By comparing Eq. (7) with Eq. (5), we see that $\mathrm{v} = \inf_q r^*(q)$. On the other hand, if P2 does not play adversarially, P1 may be able to do better, perhaps attaining $r^*(q)$. Our subsequent results address the question whether this is indeed the case.

Remark 3 *In general, the infimum and supremum in (7)* cannot *be interchanged. This is because the set of feasible p in the inner maximization depends on the value of q. Moreover, it can be shown that the set of (p,q) pairs that satisfy the constraint $\sum_{a,b} p(a)q(b)C(a,b) \in T$ is not necessarily convex.*

2. A set X is approachable if there exists a strategy for the agent such that for every $\varepsilon > 0$, there exists an integer N such that, for every opponent strategy:

$$\Pr\left(\mathrm{dist}\left(\frac{1}{n}\sum_{t=1}^{n} m_t, X\right) \geq \varepsilon \text{ for some } n \geq N\right) < \varepsilon.$$

3. A set X is excludable if there exists a strategy for the opponent such that there exists $\delta > 0$ such that for every $\varepsilon > 0$, there exists an integer N such that, for every agent strategy:

$$\Pr\left(\mathrm{dist}\left(\frac{1}{n}\sum_{t=1}^{n} m_t, X\right) \geq \delta \text{ for all } n \geq N\right) > 1 - \varepsilon.$$

2.4 Related Works

Notwithstanding the apparent similarity, the problem that we consider is not an instance of online convex optimization (Zinkevich, 2003; Hazan and Megiddo, 2007). In the latter setting, there is a convex feasible domain $\mathcal{F} \subset \mathbb{R}^n$, and an arbitrary sequence of convex functions $f_t : \mathcal{F} \to \mathbb{R}$. At every step t, the decision maker picks $x_t \in \mathcal{F}$ based on the past history, without knowledge of the future functions f_t, and with the objective of minimizing the regret

$$\sum_{t=1}^{T} f_t(x_t) - \min_{y\in\mathcal{F}} \sum_{t=1}^{T} f_t(y).$$

An analogy with our setting might be possible, by identifying x_t and f_t with a_t and b_t, respectively, and by somehow relating the feasibility constraints described by $\mathcal{F}$ to our constraints. However, this attempt seems to run into some fundamental obstacles. In particular, in our setting, feasibility is affected by the opponent's actions, whereas in online convex optimization, the feasible domain $\mathcal{F}$ is fixed for all time steps. For this reason, we do not see a way to reduce the problem of online learning with constraints to an online convex optimization problem, and given the results below, it is unlikely that such a reduction is possible.

3. Reward-in-Hindsight Is Not Attainable

As it turns out, the reward-in-hindsight cannot be attained in general. This is demonstrated by the following simple 2×2 matrix game, with just a single constraint.

Consider a 2×2 constrained game specified by:

$$\begin{pmatrix} (1,-1) & (1,1) \\ (0,-1) & (-1,-1) \end{pmatrix},$$

where each entry (pair) corresponds to $(R(a,b),C(a,b))$ for a pair of actions a and b. At a typical stage, P1 chooses a row, and P2 chooses a column. We set $c_0 = 0$. Let q denote the frequency with which P2 chooses the second column. The reward of the first row dominates the reward of the second one, so if the constraint can be satisfied, P1 would prefer to choose the first row. This can be done as long as $0 \le q \le 1/2$, in which case $r^*(q) = 1$. For $1/2 \le q \le 1$, player P1 needs to optimize the reward subject to the constraint. Given a specific q, P1 will try to choose a mixed action that satisfies the constraint (on the average) while maximizing the reward. If we let α denote the frequency of choosing the first row, we see that the reward and penalty are:

$$r(\alpha,q) = \alpha - (1-\alpha)q, \quad c(\alpha,q) = 2\alpha q - 1,$$

respectively. We observe that for every q, $r(\alpha)$ and $c(\alpha)$ are monotonically increasing functions of α. As a result, P1 will choose the maximal α that satisfies $c(\alpha) \le 0$, which is $\alpha(q) = 1/2q$, and the optimal reward is $1/2 + 1/2q - q$. We conclude that the reward-in-hindsight is:

$$r^*(q) = \begin{cases} 1, & \text{if } 0 \le q \le 1/2, \\ \dfrac{1}{2} + \dfrac{1}{2q} - q, & \text{if } 1/2 \le q \le 1. \end{cases}$$

The graph of $r^*(q)$ is the solid line in Figure 1.

We now claim that P2 can make sure that P1 does not attain r^*.

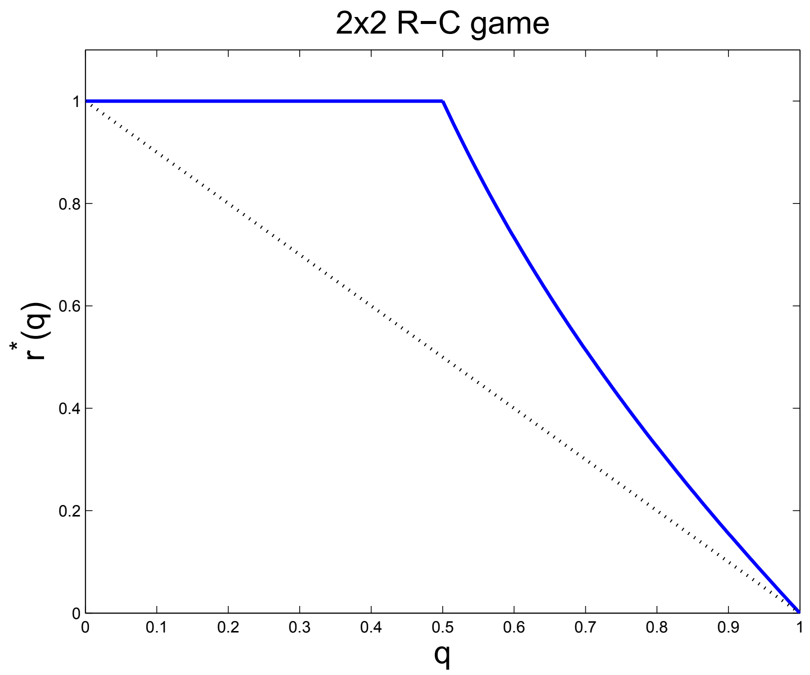


Figure 1: The reward-in-hindsight of the constrained game. Here, $r^*(q)$ is the solid line, and the dotted line connects the two extreme values, for $q = 0$ and $q = 1$.

Proposition 4 *If $c_0 = 0$, then there exists a strategy for P2 such that r^* cannot be attained.*

Proof Suppose that the opponent, P2, plays according to the following strategy. Initialize a counter $k = 1$. Let $\hat{\alpha}_t$ be the empirical frequency with which P1 chooses the *first* row during the first t time steps. Similarly, let $\hat{q}_t$ be the empirical frequency with which P2 chooses the *second* column during the first t time steps.

1. While $k = 1$ or $\hat{\alpha}_{t-1} > 3/4$, P2 chooses the second column, and k is incremented by 1.
2. For the next k times, P2 chooses the first column. Then, reset the counter k to , and go back to Step 1.

We now show that if

$$\limsup_{t\to\infty} \hat{c}_t \leq 0, \quad \text{a.s.},$$

then a strict inequality holds for the regret:

$$\liminf_{t\to\infty} (\hat{r}_t - r^*(\hat{q}_t)) < 0, \quad \text{a.s.}$$

Suppose that Step 2 is entered only a finite number of times. Then, after some finite time, P2 keeps choosing the second column, and $\hat{q}_t$ converges to 1. For P1 to satisfy the constraint $\limsup_{t\to\infty} \hat{c}_t \leq$

0, we must have $\lim \hat{\alpha}_t \leq 1/2$. But then, the condition $\hat{\alpha}_{t-1} > 3/4$ will be eventually violated. This shows that Step 2 is entered an infinite number of times. In particular, there exist infinite sequences t_i and t'_i such that $t_i < t'_i < t_{i+1}$ and (i) if $t_i < t \leq t'_i$, P2 chooses the second column (Step 1); (ii) if $t'_i < t \leq t_{i+1}$, P2 chooses the first column (Step 2).

Note that Steps 1 and 2 last for an equal number of time steps. Thus, we have $\hat{q}_{t_i} = 1/2$, and $r^*(\hat{q}_{t_i}) = 1$, for all i. Furthermore, $t_{i+1} - t'_i \leq t'_i$, or $t'_i \geq t_{i+1}/2$. Note that $\hat{\alpha}_{t'_i} \leq 3/4$, because otherwise P2 would still be in Step 1 at time $t'_i + 1$. Thus, during the first t_{i+1} time steps, P1 has played the first row at most

$$3t'_i/4 + (t_{i+1} - t'_i) = t_{i+1} - t'_i/4 \leq 7t_{i+1}/8$$

times. Due to the values of the reward matrix, we have $\limsup_{t\to\infty} \hat{r}_t < \limsup_{i\to\infty} \hat{r}_{t_i}$. In particular, we have $\hat{r}_{t_{i+1}} \leq 7/8$, and $\liminf_{t\to\infty}(\hat{r}_t - r^*(\hat{q}_t)) \leq 7/8 - 1 < 0$. ■

Intuitively, the strategy that was described above allows P2 to force P1 to move, back and forth, between the extreme points ($q = 0$ and $q = 1$) that are linked by the dotted line in Figure 1. Since $r^*(q)$ is not convex, and since the dotted line is strictly below $r^*(q)$ for $q = 1/2$, this strategy precludes P1 from attaining $r^*(q)$. We note that the choice of c_0 is critical in this example. With other choices of c_0 (for example, $c_0 = -1$), the reward-in-hindsight may be attainable.

4. Attainability of the Convex Hull

Since the reward-in-hindsight is not attainable in general, we have to settle for a more modest objective. More specifically, we are interested in functions $f : \Delta(B) \to \mathbb{R}$ that are attainable with respect to a given constraint set T. As a target we suggest the closed convex hull of the reward-in-hindsight, r^*. After defining it, we prove that it is indeed attainable. In the next section, we will also show that it is the highest possible attainable function, when there is a single constraint.

Given a function $f : X \mapsto \mathbb{R}$, over a convex domain X, its *closed convex hull* is the function whose epigraph is

$$\overline{\text{conv}}(\{(x,r) : r \geq f(x)\}),$$

where $\text{conv}(D)$ is the convex hull, and $\overline{D}$ is the closure of a set D. We denote the closed convex hull of r^* by r^c.

We will make use of the following facts. Forming the convex hull and then the closure results in a larger epigraph, hence a smaller function. In particular, $r^c(q) \leq r^*(q)$, for all q. Furthermore, the closed convex hull is guaranteed to be continuous on $\Delta(B)$. (This would not be true if we had considered the convex hull, without forming its closure.) Finally, for every q in the interior of $\Delta(B)$, we have:

$$\begin{aligned} r^c(q) \quad = \quad & \inf_{q_1,q_2,\ldots,q_k\in\Delta(B),\alpha_1,\ldots,\alpha_k} \sum_{i=1}^{k} \alpha_i r^*(q_i) \\ & \text{s.t.} \quad \sum_{i=1}^{k} \alpha_i q_i(b) = q(b), \quad \forall b \in B, \\ & \qquad \alpha_i \geq 0, \quad i = 1,2,\ldots,k, \\ & \qquad \sum_{i=1}^{k} \alpha_i = 1, \end{aligned} \tag{8}$$

where k can be taken equal to $|B|+2$ by Caratheodory's Theorem.

The following result is proved using Blackwell's approachability theory. The technique is similar to that used in other no-regret proofs (e.g., Blackwell, 1956b; Mannor and Shimkin, 2003), and is based on the convexity of a target set in an appropriately defined space.

Theorem 5 *Let Assumption 1 hold for a given convex set $T \subset \mathbb{R}^d$. Then r^c is attainable with respect to T.*

Proof Define the following game with vector-valued payoffs, where the payoffs belong to $\mathbb{R} \times \mathbb{R}^d \times \Delta(B)$ (a $|B|+d+1$ dimensional space, which we denote by $\mathcal{M}$). Suppose that P1 plays a_t, P2 plays b_t, P1 obtains an immediate reward of r_t and an immediate penalty vector of c_t. Then, the vector-valued payoff obtained by P1 is

$$m_t = (r_t, c_t, e(b_t)),$$

where $e(b)$ is a vector of zeroes, except for a 1 in its bth component. It follows that the average vector-valued reward at time t, which we define as $\hat{m}_t = \frac{1}{t}\sum_{\tau=1}^{t} m_\tau$, satisfies: $\hat{m}_t = (\hat{r}_t, \hat{c}_t, \hat{q}_t)$, where $\hat{r}_t$, $\hat{c}_t$, and $\hat{q}_t$ were defined in Eqs. (1), (2), and (4), respectively. Consider the sets:

$$\mathcal{B}_1 = \{(r,c,q) \in \mathcal{M} : r \geq r^c(q)\}, \qquad \mathcal{B}_2 = \{(r,c,q) \in \mathcal{M} : c \in T\},$$

and let $\mathcal{B} = \mathcal{B}_1 \cap \mathcal{B}_2$. Note that $\mathcal{B}$ is a convex set. We claim that $\mathcal{B}$ is approachable. Let $m : \Delta(A) \times \Delta(B) \to \mathcal{M}$ describe the expected payoff in a single stage game, when P1 and P2 choose actions p and q, respectively. That is,

$$m(p,q) = \Big(\sum_{a,b} p(a)q(b)R(a,b),\ \sum_{a,b} p(a)q(b)C(a,b),\ q \Big).$$

Using the sufficient condition for approachability of convex sets (Blackwell, 1956a), it suffices to show that for every q there exists a p such that $m(p,q) \in \mathcal{B}$. Fix $q \in \Delta(B)$. By Assumption 1, the constraint $\sum_{a,b} p(a)q(b)C(a,b) \in T$ is feasible, which implies that the program (5) has an optimal solution p^*. It follows that $m(p^*,q) \in \mathcal{B}$. We now claim that a strategy that approaches $\mathcal{B}$ also attains r^c in the sense of Definition 1. Indeed, since $\mathcal{B} \subseteq \mathcal{B}_2$ we have that $\Pr(d(c_t,T) > \varepsilon \text{ infinitely often}) = 0$ for every $\varepsilon > 0$. Since $\mathcal{B} \subseteq \mathcal{B}_1$ and using the continuity of r^c, we obtain $\liminf(\hat{r}_t - r^c(\hat{q}_t)) \geq 0$. ■

We note that Theorem 5 is not constructive. Indeed, a strategy that approaches $\mathcal{B}$, based on a naive implementation Blackwell's approachability theory, requires an efficient procedure for computing the closest point in $\mathcal{B}$,and therefore a computationally efficient description of $\mathcal{B}$, which may not be available (we do not know whether $\mathcal{B}$ can be described efficiently). This motivates the development of the calibration based scheme in Section 6.

Remark 6 *Convergence rate results also follow from general approachability theory, and are generally of the order of $t^{-1/3}$; see Mertens et al. (1994). It may be possible, perhaps, to improve upon this rate and obtain $t^{-1/2}$, which is the best possible convergence rate for the unconstrained case.*

Remark 7 *For every $q \in \Delta(B)$, we have $r^*(q) \geq \mathrm{v}$, which implies that $r^c(q) \geq \mathrm{v}$. Thus, attaining r^c guarantees an average reward at least as high as the value of the game.*

4.1 Degenerate Cases

In this section, we consider the degenerate cases where the penalty vector is affected by only one of the players. We start with the case where P1 alone affects the penalty vector, and then discuss the case where P2 alone affects the penalty vector.

If P1 alone affects the penalty vector, that is, if $C(a,b) = C(a,b')$ for all $a \in A$ and $b,b' \in B$, then $r^*(q)$ is convex. Indeed, in this case, Eq. (5) becomes (writing $C(a)$ for $C(a,b)$)

$$r^*(q) = \max_{p\in\Delta(A):\sum_a p(a)C(a)\in T} \sum_{a,b} p(a)q(b)R(a,b),$$

which is the maximum of a collection of linear functions of q (one function for each feasible p), and is therefore convex.

If P2 alone affects the penalty vector, that is, if $c(a,b) = c(a',b)$ for all $b \in B$ and $a,a' \in A$, then Assumption 1 implies that the constraint is always satisfied. Therefore,

$$r^*(q) = \max_{p\in\Delta(A)} \sum_{a,b} p(a)q(b)R(a,b),$$

which is again a maximum of linear functions, hence convex.

We conclude that in both degenerate cases, if Assumption 1 holds, then the reward-in-hindsight is attainable.

5. Tightness of the Convex Hull

We now show that r^c is the maximal attainable function, for the case of a single constraint.

Theorem 8 *Suppose that $d = 1$, T is of the form $T = \{c \mid c \le c_0\}$, where c_0 is a given scalar, and that Assumption 1 is satisfied. Let $\tilde{r} : \Delta(B) \mapsto \mathbb{R}$ be a continuous attainable function with respect to the scalar c_0. Then, $r^c(q) \ge \tilde{r}(q)$ for all $q \in \Delta(B)$.*

Proof The proof is constructive, as it provides a concrete strategy for P2 that prevents P1 from attaining $\tilde{r}$, unless $r^c(q) \ge \tilde{r}(q)$ for every q. Assume, in order to derive a contradiction, that there exists some $\tilde{r}$ that violates the theorem. Since $\tilde{r}$ and r^c are continuous, there exists some $q^0 \in \Delta(B)$ and some $\varepsilon > 0$ such that $\tilde{r}(q) > r^c(q) + \varepsilon$ for all q in an open neighborhood of q^0. In particular, q^0 can be taken to lie in the interior of $\Delta(B)$. Using Eq. (8), it follows that there exist $q^1,\ldots,q^k \in \Delta(B)$ and $\alpha_1,\ldots,\alpha_k$ (with $k \le |B|+2$, due to Caratheodory's Theorem) such that

$$\sum_{i=1}^{k} \alpha_i r^*(q^i) \le r^c(q^0) + \frac{\varepsilon}{2} < \tilde{r}(q^0) - \frac{\varepsilon}{2};$$

$$\sum_{i=1}^{k} \alpha_i q^i(b) = q^0(b), \quad \forall\, b \in B; \qquad \sum_{i=1}^{k} \alpha_i = 1; \qquad \alpha_i \ge 0,\ \forall\, i.$$

Let τ be a large positive integer (τ is to be chosen large enough to ensure that the events of interest occur with high probability, etc.). We will show that if P2 plays each q^i for $\lceil \alpha_i \tau \rceil$ time steps, in an appropriate order, then either P1 does not satisfy the constraint along the way or $\hat{r}_\tau \le \tilde{r}(\hat{q}_\tau) - \varepsilon/2$.

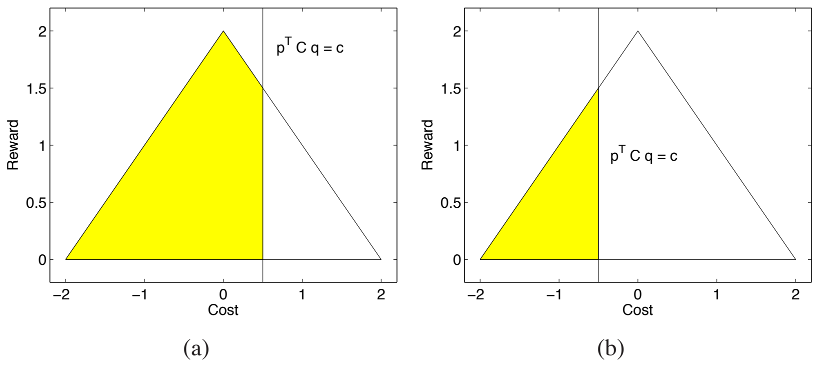


Figure 2: In either part (a) or (b) of the figure, we fix some $q \in \Delta(B)$. The triangle is the set of possible reward-cost pairs, as we vary p over the set $\Delta(A)$. Then, for a given value c in the upper bound on the cost (cf. 10), the shaded region is the set of reward-cost pairs that also satisfy the cost constraint.

We let q^i, $i = 1,\ldots,k$, be fixed, as above, and define a function $f_i : \mathbb{R}^d \to \mathbb{R} \cup \{-\infty\}$ as:

$$f_i(c) = \max_{p \in \Delta(A)} \sum_{a,b} p(a) q^i(b) R(a,b), \tag{9}$$

$$\text{subject to} \qquad \sum_{a,b} p(a) q^i(b) C(a,b) \le c, \tag{10}$$

where the maximum over an empty set is defined to equal $-\infty$. Observe that the feasible set (and hence, optimal value) of the above linear program depends on c. Figure 2 illustrates how the feasible sets to (10) may depend on the value of c. By viewing Eqs. (9)-(10) as a parametric linear program, with a varying right-hand side parameter c, we see that $f_i(c)$ is piecewise linear, concave, and nondecreasing in c (Bertsimas and Tsitsiklis, 1997). Furthermore, $f_i(c_0) = r^*(q^i)$. Let ∂f_i^+ be the right directional derivative of f_i at $c = c_0$, and note that $\partial f_i^+ \ge 0$. From now on, we assume that the q^i have been ordered so that the sequence ∂f_i^+ is nonincreasing (e.g., as in Figure 3). To visualize the ordering that we have introduced, consider the set of possible pairs (r,c), given a fixed q. That is, consider the set $M(q^i) = \{(r,c) : \exists p \in \Delta(A) \text{ s.t. } r = \sum_{a,b} p(a)q^i(b)R(a,b),\ c = \sum_{a,b} p(a)q^i(b)C(a,b)\}$. The set $M(q^i)$ is the image of the simplex under a linear transformation, and is therefore a polytope, as illustrated by the triangular areas in Figure 2. The strategy of P2 is to first play q^i such that the p that maximizes the reward (Eq. 9) satisfies Eq. (10) with equality. (Such a q^i results in a set $M(q^i)$ like the one shown in Figure 2(b).) After all these q^i are played, P2 plays those q^i for which the p that maximizes the reward (Eq. 9) satisfies Eq. (10) with strict inequality, and $\partial f_i^+ = 0$. (Such a q^i results in a set $M(q^i)$ like the one shown in Figure 2(a).)

Suppose that P1 knows the sequence $q^1,\ldots,q^k$ (ordered as above) in advance, and that P2 follows the strategy described earlier. We assume that τ is large enough so that we can ignore the effects of dealing with a finite sample. Let p^i be the average of the mixed actions chosen by P1 while player P2 plays q^i. We introduce the constraints

$$\sum_{i=1}^{\ell} \alpha_i \sum_{a,b} p^i(a) q^i(b) C(a,b) \le c_0 \sum_{i=1}^{\ell} \alpha_i, \quad \ell = 1,2,\ldots,k.$$

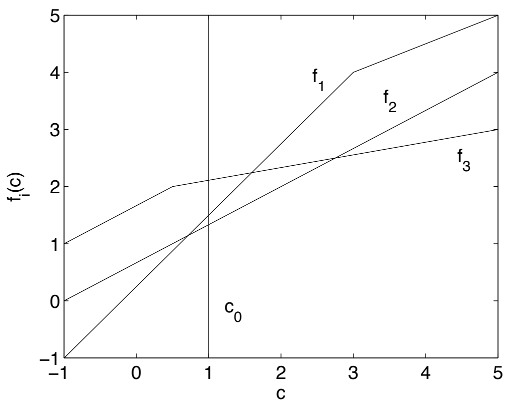


Figure 3: An example of functions f_i ordered according to ∂f_i^+.

These constraints must be satisfied in order to guarantee that $\hat{c}_t$ has negligible probability of substantially exceeding c_0, at the "switching" times from one mixed action to another. If P1 exploits the knowledge of P2's strategy to maximize her average reward at time τ, the resulting expected average reward at time τ will be the optimal value of the objective function in the following linear programming problem:

$$\begin{aligned}&\max_{p^1,p^2,\ldots,p^k} \sum_{i=1}^{k} \alpha_i \sum_{a,b} p^i(a)q^i(b)R(a,b)\\ &\quad \text{s.t.} \sum_{i=1}^{\ell} \alpha_i \sum_{a,b} p^i(a)q^i(b)C(a,b) \le c_0 \sum_{i=1}^{\ell} \alpha_i, \quad \ell = 1,2,\ldots,k,\\ &\quad p^\ell \in \Delta(A), \quad \ell = 1,2,\ldots,k.\end{aligned} \tag{11}$$

Of course, given the value of $\sum_{a,b} p^i(a)q^i(b)C(a,b)$, to be denoted by c_i, player P1 should choose a p^i that maximizes rewards, resulting in $\sum_{a,b} p^i(a)q^i(b)R(a,b) = f_i(c_i)$. Thus, the above problem can be rewritten as

$$\begin{aligned}&\max_{c_1,\ldots,c_k} \sum \alpha_i f_i(c_i)\\ &\quad \text{s.t.} \sum_{i=1}^{\ell} \alpha_i c_i \le c_0 \sum_{i=1}^{\ell} \alpha_i, \quad \ell = 1,2,\ldots,k.\end{aligned} \tag{12}$$

We claim that letting $c_i = c_0$, for all i, is an optimal solution to the problem (12). This will then imply that the optimal value of the objective function for the problem (11) is $\sum_{i=1}^{k} \alpha_i f_i(c_0)$, which equals $\sum_{i=1}^{k} \alpha_i r^*(q^i)$, which in turn, is bounded above by $\tilde{r}(q^0) - \varepsilon/2$. Thus, $\hat{r}_\tau < \tilde{r}(q^0) - \varepsilon/2 + \delta(\tau)$, where the term $\delta(\tau)$ incorporates the effects due to the randomness in the process. By repeating this argument with ever increasing values of τ (so that the stochastic term $\delta(\tau)$ is averaged out and becomes negligible), we obtain that the event $\hat{r}_t < \tilde{r}(q^0) - \varepsilon/2$ will occur infinitely often, and therefore $\tilde{r}$ is not attainable.

It remains to establish the claimed optimality of $(c_0,\ldots,c_0)$. Suppose that $(\bar{c}_1,\ldots,\bar{c}_k) \neq (c_0,\ldots,c_0)$ is an optimal solution of the problem (12). If $\bar{c}_i \le c_0$ for all i, the monotonicity of the f_i implies that $(c_0,\ldots,c_0)$ is also an optimal solution. Otherwise, let j be the smallest index for which $\bar{c}_j > c_0$. If $\partial f_j^+ = 0$ (as in the case shown in Figure 2(b)) we have that $f_i(c)$ is maximized at c_0 for all $i \geq j$ and $(c_0,\ldots,c_0)$ is optimal. Suppose that $\partial f_j^+ > 0$. In order for the constraint (12) to be satisfied, there must exist some index $s < j$ such that $\bar{c}_s < c_0$. Let us perturb this solution by setting $\delta = \min\{\alpha_s(c_0 - \bar{c}_s), \alpha_j(\bar{c}_j - c_0)\}$, increasing $\bar{c}_s$ to $\tilde{c}_s = \bar{c}_s + \delta/\alpha_s$, and decreasing $\bar{c}_j$ to $\tilde{c}_j = \bar{c}_j - \delta/\alpha_j$. This new solution is clearly feasible. Let $\partial f_s^- = \lim_{\varepsilon\downarrow 0}(f_s(c_0) - f_s(c_0 - \varepsilon))/\varepsilon$, which is the left derivative of f_s at c_0. Using the concavity of f_s, and the earlier introduced ordering, we have $\partial f_s^- \geq \partial f_s^+ \geq \partial f_j^+$. Observe that

$$f_s(\tilde{c}_s) = f_s(\bar{c}_s) + \partial f_s^- \delta/\alpha_s,$$
$$f_j(\tilde{c}_j) = f_j(\bar{c}_j) - \partial f_j^+ \delta/\alpha_j,$$

so that $\alpha_s f_s(\tilde{c}_s) + \alpha_j f_j(\tilde{c}_j) \geq \alpha_s f_s(\bar{c}_s) + \alpha_j f_j(\bar{c}_j)$. Therefore, the new solution must also be optimal, but has fewer components that differ from c_0. By repeating this process, we eventually conclude that $(c_0,\ldots,c_0)$ is an optimal solution of (12). ■

To the best of our knowledge, this is the first tightness result for a performance envelope (the reward-in-hindsight) different than the Bayes envelope, for repeated games. On the other hand, we note that our proof relies crucially on the assumption of a single constraint ($d = 1$), which allows us to order the ∂f_i^+.

6. Attaining the Convex Hull Using Calibrated Forecasts

In this section, we consider a specific strategy that attains the convex hull, thus providing a constructive proof for Theorem 5. The strategy is based on forecasting P2's action, and playing a best response (in the sense of Eq. 5) against the forecast. The quality of the resulting strategy depends, of course, on the quality of the forecasts; it is well known that *calibrated* forecasts lead to no-regret strategies in standard repeated matrix games. See Foster and Vohra (1997) and Cesa-Bianchi and Lugosi (2006) for a discussion of calibration and its implications in learning in games. In this section we consider the consequences of calibrated play for repeated games with constraints.

We start with a formal definition of calibrated forecasts and calibrated play, and then show that calibrated play attains r^c in the sense of Definition 1.

A forecasting scheme specifies at each stage k a probabilistic forecast $q_k \in \Delta(B)$ of P2's action b_k. More precisely a (randomized) forecasting scheme is a sequence of maps that associate with each possible history h_{k-1} during the first $k-1$ stages a probability measure μ_k over $\Delta(B)$. The forecast $q_k \in \Delta(B)$ is then selected at random according to the distribution μ_k. Let us clarify that for the purposes of this section, the history is defined to include the realized past forecasts.

We shall use the following definition of calibrated forecasts.

Definition 9 (Calibrated forecasts) *A forecasting scheme is* calibrated *if for every (Borel measurable) set $Q \subset \Delta(B)$ and every strategy of P1 and P2*

$$\lim_{t\to\infty} \frac{1}{t} \sum_{\tau=1}^{t} 1\{q_\tau \in Q\}(e(b_\tau) - q_\tau) = 0, \quad a.s., \tag{13}$$

where $e(b)$ is a vector of zeroes, except for a 1 in its bth component.

Calibrated forecasts, as defined above, have been introduced into game theory in Foster and Vohra (1997), and several algorithms have been devised to achieve them (see Cesa-Bianchi and Lugosi, 2006, and references therein). These algorithms typically start with predictions that are restricted to a finite grid, and gradually increase the number of grid points.

The proposed strategy is to let P1 play a best response against P2's forecasted play while still satisfying the constraints (in expectation, for the single stage game). Formally, we let:

$$\begin{aligned} p^*(q) &= \operatorname*{argmax}_{p\in\Delta(A)} \sum_{a,b} p(a)q(b)R(a,b) \\ &\quad\ \text{s.t.} \sum_{a,b} p(a)q(b)C(a,b) \in T, \end{aligned}$$

where in the case of a non-unique maximum we assume that $p^*(q)$ is uniquely determined by some tie-breaking rule; this is easily done, while keeping $p^*(\cdot)$ a measurable function. The strategy is to play $p_t = p^*(q_t)$, where q_t is a calibrated forecast of P2's actions.[4] We call such a strategy a *calibrated strategy*.

The following theorem states that a calibrated strategy attains the convex hull.

Theorem 10 *Let Assumption 1 hold, and suppose that P1 uses a calibrated strategy. Then, r^c is attained with respect to T.*

Proof Fix $\varepsilon > 0$. We need to show that by playing the calibrated strategy, P1 obtains $\liminf_{t\to\infty}(\hat{r}_t - r^c(\hat{q}_t)) \geq 0$ and $\limsup_{t\to\infty} \operatorname{dist}(\hat{c}_t, T) \leq 0$, almost surely.

Fix some $\varepsilon > 0$. Consider a partition of the simplex $\Delta(B)$ to finitely many measurable sets $Q_1, Q_2, \ldots, Q_\ell$ such that $q, q' \in Q_i$ implies that $\|q - q'\| \leq \varepsilon$ and $\|p^*(q) - p^*(q')\| \leq \varepsilon$. (Such a partition exists by the compactness of $\Delta(B)$ and $\Delta(A)$. The measurability of the sets Q_i can be guaranteed because the mapping $p^*(\cdot)$ is measurable.) For each i, let us fix a representative element $q^i \in Q_i$, and let $p^i = p^*(q^i)$.

Since we have a calibrated forecast, Eq. (13) holds for every Q_i, $1 \leq i \leq \ell$. Define $\Gamma_t(i) = \sum_{\tau=1}^t 1\{q_\tau \in Q_i\}$ and assume without loss of generality that $\Gamma_t(i) > 0$ for large t (otherwise, eliminate those i for which $\Gamma_t(i) = 0$ for all t, and renumber the Q_i). To simplify the presentation, we assume that for every i, and for large enough t, we have $\Gamma_t(i) \geq \varepsilon t$. (If for some i, and t this condition is violated, the contribution of such an i in the expressions that follow will be $O(\varepsilon)$.)

By a law of large numbers for martingales, we have

$$\lim_{t\to\infty}\left(\hat{c}_t - \frac{1}{t}\sum_{\tau=1}^t C(a_\tau, b_\tau)\right) = 0, \qquad \text{a.s.}$$

By definition, we have

$$\frac{1}{t}\sum_{\tau=1}^t C(a_\tau, b_\tau) = \sum_i \frac{\Gamma_t(i)}{t} \sum_{a,b} C(a,b) \frac{1}{\Gamma_t(i)} \sum_{\tau=1}^t 1\{q_\tau \in Q_i\} 1\{a_\tau = a\} 1\{b_\tau = b\}.$$

Observe that whenever $q_\tau \in Q_i$, we have $\left\|p_\tau - p^i\right\| \leq \varepsilon$, where $p_\tau = p^*(q_\tau)$ and $p^i = p^*(q^i)$ because of the way the sets Q_i were constructed. By martingale convergence, the frequency with which a

4. When the forecast μ_t is mixed, q_t is the realization of the mixed rule.

will be selected whenever $q_\tau \in Q_i$ and $b_\tau = b$, will be approximately $p^i(a)$. Hence, for all b,

$$\limsup_{t\to\infty} \left| \frac{1}{\Gamma_t(i)} \sum_{\tau=1}^{t} 1\{q_\tau \in Q_i\} 1\{a_\tau = a\} 1\{b_\tau = b\} - p^i(a) \frac{1}{\Gamma_t(i)} \sum_{\tau=1}^{t} 1\{q_\tau \in Q_i\} 1\{b_\tau = b\} \right| \le \varepsilon,$$

almost surely. By the calibration property (13) for $Q = Q_i$, and the fact that whenever $q, q' \in Q_i$, we have $\|q - q'\| \le \varepsilon$, we obtain

$$\limsup_{t\to\infty} \left| \frac{1}{\Gamma_t(i)} \sum_{\tau=1}^{t} 1\{q_\tau \in Q_i\} 1\{b_\tau = b\} - q^i(b) \right| \le \varepsilon, \qquad \text{a.s.}$$

By combining the above bounds, we obtain

$$\lim_{t\to\infty} \left| \hat{c}_t - \sum_i \frac{\Gamma_t(i)}{t} \sum_{a,b} C(a,b) p^i(a) q^i(b) \right| \le 2\varepsilon, \qquad \text{a.s.} \tag{14}$$

Note that the sum over index i in Eq. (14) is a convex combination (because the coefficients $\Gamma_t(i)/t$ sum to 1) of elements of T (because of the definition of p^i), and is therefore an element of T (because T is convex). This establishes that the constraint is asymptotically satisfied within $O(\varepsilon)$. Note that in this argument, whenever $\Gamma_t(i)/t < \varepsilon$, the summand corresponding to i is indeed of order $O(\varepsilon)$ and can be safely ignored, as stated earlier.

Regarding the average reward, an argument similar to the above yields

$$\liminf_{t\to\infty} \hat{r}_t \ge \liminf_{t\to\infty} \sum_i \frac{\Gamma_t(i)}{t} \sum_{a,b} R(a,b) p^i(a) q^i(b) - 2\varepsilon, \qquad \text{a.s.}$$

Next, observe that

$$\sum_i \frac{\Gamma_t(i)}{t} \sum_{a,b} R(a,b) p^i(a) q^i(b) = \sum_i \frac{\Gamma_t(i)}{t} r^*(q^i) \ge r^c\Big(\sum_i \frac{\Gamma_t(i)}{t} q^i \Big),$$

where the equality is a consequence of the definition of p^i, and the inequality follows by the definition of r^c as the closed convex hull of r^*. Observe also that the calibration property (13), with $Q = \Delta(B)$, implies that

$$\lim_{t\to\infty} \left\| \hat{q}_t - \frac{1}{t} \sum_{\tau=1}^{t} q_\tau \right\| = 0, \qquad \text{a.s.}$$

In turn, since $\|q_\tau - q^i\| \le \varepsilon$ for a fraction $\Gamma_t(i)/t$ of the time,

$$\limsup_{t\to\infty} \left\| \hat{q}_t - \sum_i \frac{\Gamma_t(i)}{t} q^i \right\| = \limsup_{t\to\infty} \left\| \frac{1}{t} \sum_{\tau=1}^{t} q_\tau - \sum_i \frac{\Gamma_t(i)}{t} q^i \right\| \le \varepsilon, \qquad \text{a.s.}$$

Recall that the function r^c is continuous, hence uniformly continuous. Thus, there exists some function g, with $\lim_{\varepsilon\downarrow 0} g(\varepsilon) = 0$, such that when the argument of r^c changes by at most ε, the value of r^c changes by at most $g(\varepsilon)$. By combining the preceding results, we obtain

$$\liminf_{t\to\infty} \hat{r}_t \ge r^c(\hat{q}_t) - 2\varepsilon - g(\varepsilon), \qquad \text{a.s.}$$

The above argument involves a fixed ε, and a fixed number ℓ of sets Q_i, and lets t increase to infinity. As such, it establishes that for any $\varepsilon > 0$ the function $r^c - 2\varepsilon - g(\varepsilon)$ is attainable with respect to the set T^ε defined by $T^\varepsilon = \{x \mid \mathrm{dist}(x,T) \leq 2\varepsilon\}$. Since this is true for every $\varepsilon > 0$, we conclude that the calibrated strategy attains r^c as claimed. ■

7. Algorithms

The results in the previous section motivate us to develop algorithms for online learning with constraints, perhaps based on calibrated forecasts. For practical reasons, we are interested in computationally efficient methods, but there are no known computationally efficient calibrated forecasting algorithms. For this reason, we will consider related heuristics that are similar in spirit, even if they do not have all the desired guarantees.

We first consider a method based on the weighted average predictor. The algorithm in Table 1 keeps track of the performance of the different actions in the set A, updating a corresponding set of weights accordingly at each step. The main idea is to quantify "performance" by a linear combination of the total reward and the magnitude of the constraint violation. The parameter $\lambda > 0$ of the algorithm, which acts similar to a Lagrange multiplier, determines the tradeoff between these two objectives. When the average penalty is higher than c_0 (i.e., there is a violation), the weight of the cost term increases. When the average penalty is lower than c_0, the weight of the cost term decreases. The parameters $\overline{M}$ and $\underline{M}$ are used to bound the magnitude of the weight of the cost term; in the experiments reported in Section 8, they were set to 1000 and 0.001, respectively.

```
1. Set $\lambda, w_0, \overline{M}$, and $\underline{M}$.
2. For $t = 1,2,\ldots$:
    (a) Sample an independent random variable $a_t$ distributed so that
            $a_t = a, \quad \text{with probability } \frac{w_t(a)}{\sum_{a\in A} w_t(a)} \text{ for } a \in A.$
    (b) Compute:
            $w_t(a) = w_{t-1}(a)\, \exp\left(\eta\left(R(a,b_t) - \lambda C(a,b_t)\right)\right), \quad a \in A.$
    (c) For $t = 1,2,\ldots$, update $\lambda$:
            $\lambda := \begin{cases} \min(2\lambda, \overline{M}), & \text{if } \hat{c}_t > c_0, \\ \max(\lambda/2, \underline{M}), & \text{otherwise.} \end{cases}$
```

Table 1: Exponentially weighted average predictor.

The second algorithm uses the tracking forecaster (Mannor et al., 2007) as the forecasting method. This forecaster predicts that the distribution of the next action as a weighted average of previous actions, weighing recent actions more than less recent ones others. For the special case of only two actions, it is calibrated, but *not* calibrated in general. There are, however, some special cases where it is calibrated, in particular if the sequence it tries to calibrate comes from a source with some specific properties; see Mannor et al. (2007) for details. The algorithm is presented in Table 2. If there is a current violation, it selects an action that minimizes the immediate forecasted cost. If the current average penalty does not violate the constraint, it selects a best response to the forecasted action of P2, while satisfying the constraints.

1. Set $\rho \in (0,1)$, c_0, and $f_0 = (1/|B|)\vec{1}$.
2. For $t = 1, 2, \ldots$:
   (a) If $t = 1$ or $\hat{c}_t > c_0$, choose an action that minimizes the worst-case cost:
   $$a_t \in \operatorname*{argmin}_{a \in A} \left(C(a,b) f_{t-1}(b) \right),$$
   (b) Otherwise (if $\hat{c}_t \le c_0$ and $t > 1$), solve
   $$\begin{array}{ll} \displaystyle\max_{p \in \Delta(A)} & \displaystyle\sum_{a,b} p(a) R(a,b) f_{t-1}(b), \\ \text{subject to} & \displaystyle\sum_{a,b} p(a) C(a,b) f_{t-1}(b) \le c_0. \end{array}$$
   and choose a random action distributed according to the solution to the above linear program.
   (c) After observing b_t, update the forecast f_t on the probability distribution of the next opponent action b_{t+1}:
   $$f_t = f_{t-1} + (1/t)^{\rho} \left(e_{b_t} - f_{t-1} \right),$$
   where e_b is a unit vector in $\mathbb{R}^{|B|}$ with the element 1 in the component corresponding to $b \in B$.

Table 2: Tracking forecaster.

8. Experimental Setup

Our experiment addresses the problem of minimizing power consumption in a computer with a human user. The agent is a low-level software controller that decides when to put the central processor (CPU) into a low-power state, thereby reducing power expenditures during periods when the user is idle. The system is driven by a human user, as well as different hardware processes, and can be realistically assumed to be non-stationary. The actions of the system correspond to hardware interrupts (most interrupts are generated by hardware controllers on the motherboard such as direct memory access, hard disk interrupts and networking interrupts) and the ongoing running processes. In the particular application at hand, there is a software interrupt (generated by the Windows operating system) every 16 milliseconds. The times of these interrupts are the decision epochs, at which the software controller can decide if and when to put the CPU to sleep before the next scheduled periodic interrupt.

However, saving energy by putting the processor in the low-power state comes at a cost. In the low-power state, a delay is incurred each time that the processor moves back into the high-power state in response to user-generated interrupts. We wish to limit the delay perceived by the human user. For this purpose, we assign a cost to the event that an interrupt arrives while the processor is in the low-power state, and impose a constraint on the time average of these costs. A similar model was used in Kveton et al. (2008), and we refer the reader to that work for further details.

We formulate the problem as follows. We divide a typical 16 millisecond interval into ten intervals. We let P1's action set be $A = \{0, 0.1, 0.2, \ldots, 1\}$, where action a corresponds to turning off the CPU after $16a$ milliseconds (the action $a = 1$ means the CPU is not turned off during the interval while the action $a = 0$ means it is turned off for the whole interval). Similarly, the action set of P2 is $B = \{0, 0.1, 0.2, \ldots, 0.9\}$, where action b corresponds to an interrupt after $16b$ milliseconds. (Note that the action $b = 0$ means there is no interrupt and that there is no point in including an action $b = 1$ in B since it would coincide with the known periodic interrupt.) The assumption is that an interrupt is handled instantaneously so if the CPU chooses a slightly larger than b it maximizes the power savings while incurring no penalty for observed delay (it is assumed for the sake of discussion that only a single interrupt is possible in each 16 millisecond interval). We define the reward at each stage as follows:

$$R(a,b) = \begin{cases} 1-a, & \text{if } b = 0 \text{ or } a > b, \quad \text{that is, if no interrupt occurs or an interrupt occurs} \\ & \text{before the CPU turns off,} \\ b-a, & \text{if } b > 0 \text{ and } a \leq b, \quad \text{that is, if there is an interrupt} \\ & \text{after the CPU is turned off.} \end{cases}$$

The cost is:

$$C(a,b) = \begin{cases} 1, & \text{if } a \leq b \text{ and } b > 0, \\ 0, & \text{otherwise.} \end{cases}$$

In "normal" operation where the CPU is powered throughout, the action is $a = 1$ and in that case there is no reward (no power saving) and no cost (no perceived delay). When $a = 0$ the CPU is turned off immediately and in this case the reward will be proportional to the amount of time until an interrupt (or until the next decision). The cost in the case $a = 0$ is 0 only is there is no interrupt ($b = 0$).

We used the real data trace obtained from what is known as MobileMark 2005 (MM05), a performance benchmark that simulates the activity of an average Microsoft Windows user. This CPU activity trace is 90 minutes long and contains more than 500,000 interrupts, including the periodic scheduled interrupts mentioned earlier. The exponentially weighted algorithm (Table 1) and the tracking forecaster (Table 2) were run on this data set. Figure 4 shows the performance of the two algorithms. The straight line shows the tradeoff between constraint violation and average reward by picking a fixed action over the entire time horizon. The different points for the exponential weighted predictor (Table 1) or the tracking forecaster (Table 2) correspond to different values of c_0. We observe that for the same average cost, the tracking forecast performs better (i.e., gets higher reward).

We selected $c_0 = 0.3$ and used both algorithms for the MM05 trace. Figures 5(a) and 5(b) show the instantaneous cost incurred by the tracking forecaster and the weighted average forecaster over the same short period. It should be observed that the cost of the algorithms is different, reflecting the fact that different policies are employed. Figures 6(a) and 6(b) show the time evolution of the

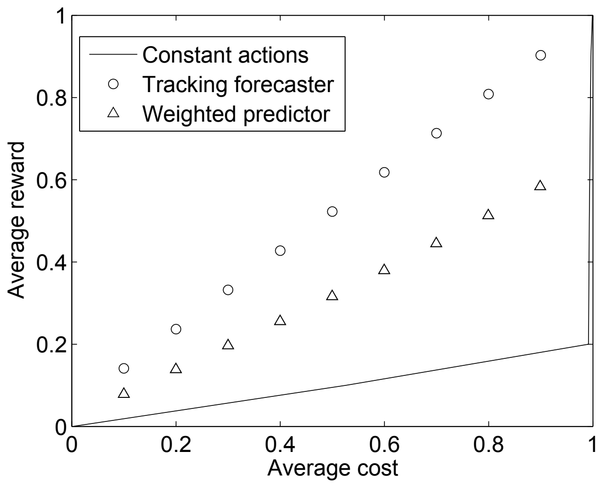
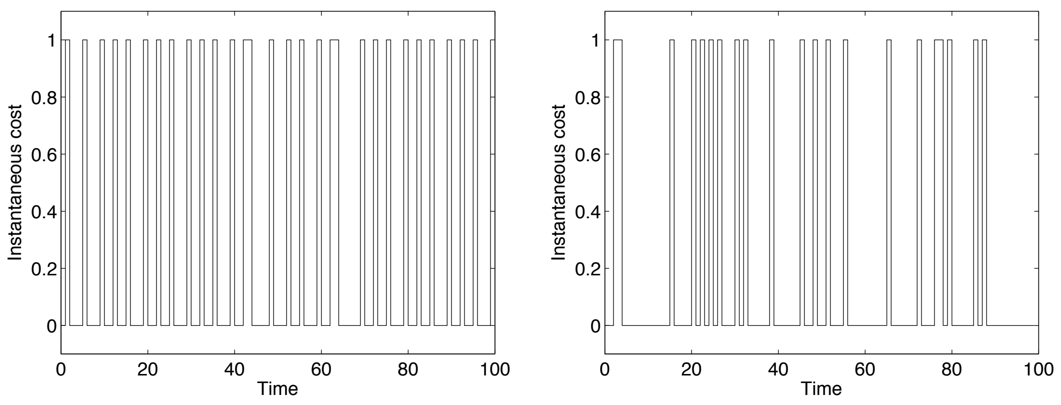


Figure 4: Plot of average reward against constraint violation frequency from experiments in power management for the MM05 data.



(a) Tracking forecaster

(b) Weighted average predictor

Figure 5: Instantaneous cost incurred by the tracking forecaster and weighted average predictor with target constraint $c_0 = 0.3$ for the MM05 data.

average reward and average cost for the same experiment. In spite of not being calibrated, the tracking forecast based algorithm outperforms the exponentially weighted based algorithm.

9. Conclusions

There are several open problems and directions for future research that are worth mentioning. First, the issue of convergence rate is yet to be settled. We noted that there exists an algorithm based on approachability that converges at the rate of $t^{-1/3}$, and that the usual lower bound of $t^{-1/2}$ holds. The other algorithm based on calibration suffers from potentially even worse convergence rate, as

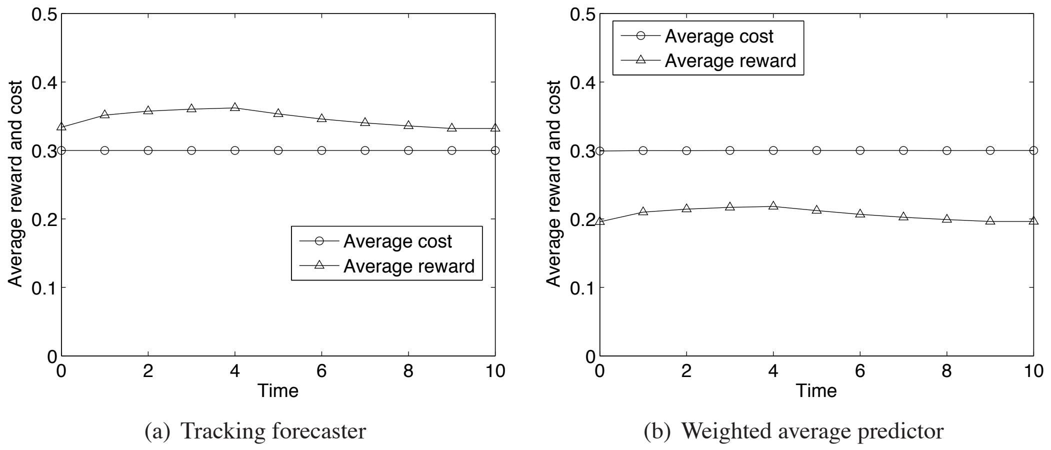


Figure 6: Time evolution of average reward and average cost for the tracking forecaster and weighted average forecaster with $c_0 = 0.3$ for the MM05 data.

we are not aware of any approximate calibration algorithm with comparable convergence rates. Second, the complexity of these two online learning algorithms leaves much to be desired. The complexity of a policy based on approachability theory is left undetermined because we do not have a specific procedure for computing P1's action at each stage. The per stage complexity is unknown for calibrated forecasts, but is exponential for approximately calibrated schemes (Cesa-Bianchi and Lugosi, 2006). Moreover, it is not clear whether online learning with constraints is as hard computationally as finding a calibrated forecast. Third, we only established the tightness of the lower convex hull of the Bayes envelope for the case of a one-dimensional penalty function. This is a remarkable result because it establishes the tightness of an envelope other than the Bayes envelope, and we are not aware of any such results for similar settings. However, it is not clear whether such a result also holds for two-dimensional penalties. In particular, the proof technique of the tightness result does not seem to extend to higher dimensions.

Our formulation of the learning problem (learning with pathwise constraints) is only a first step in considering multi-objective problems in online learning. In particular, other formulations, for example, that consider the number of time-windows where the constraints are violated, are of interest; see Kveton et al. (2008).

Acknowledgments

This research was partially supported by the National Science Foundation under contracts ECS-0312921 and ECCS-0701623, by the Natural Sciences and Engineering Research Council of Canada, by the Canada Research Chairs Program and by the Israel Science Foundation (contract 890015). We are grateful to Georgios Theocharous and Branislav Kveton for supplying us with the CPU power management data and explaining this problem domain to us. We thank two anonymous reviewers for their comments.

References

E. Altman. *Constrained Markov Decision Processes*. Chapman and Hall, 1999.

D. Bertsimas and J. N. Tsitsiklis. *Introduction to Linear Optimization*. Athena Scientific, 1997.

D. Blackwell. An analog of the minimax theorem for vector payoffs. *Pacific J. Math.*, 6(1):1–8, 1956a.

D. Blackwell. Controlled random walks. In *Proc. Int. Congress of Mathematicians 1954*, volume 3, pages 336–338. North Holland, Amsterdam, 1956b.

N. Cesa-Bianchi and G. Lugosi. *Prediction, Learning, and Games*. Cambridge University Press, 2006.

D. P. Foster and R. V. Vohra. Calibrated learning and correlated equilibrium. *Games and Economic Behavior*, 21:40–55, 1997.

J. Hannan. *Approximation to Bayes Risk in Repeated Play*, volume III of *Contribution to The Theory of Games*, pages 97–139. Princeton University Press, 1957.

E. Hazan and N. Megiddo. Online learning with prior information. In *Proceedings of 20th Annual Conference on Learning Theory*, 2007.

B. Kveton, J.Y. Yu, G. Theocharous, and S. Mannor. Online learning with expert advice and finite-horizon constraints. In *AAAI 2008, in press*, 2008.

S. Mannor and N. Shimkin. The empirical Bayes envelope and regret minimization in competitive Markov decision processes. *Mathematics of Operations Research*, 28(2):327–345, 2003.

S. Mannor and N. Shimkin. A geometric approach to multi-criterion reinforcement learning. *Journal of Machine Learning Research*, 5:325–360, 2004.

S. Mannor, J. S. Shamma, and G. Arslan. Online calibrated forecasts: Memory efficiency versus universality for learning in games. *Machine Learning*, 67(1–2):77–115, 2007.

J. F. Mertens, S. Sorin, and S. Zamir. Repeated games. CORE Reprint Dps 9420, 9421 and 9422, Center for Operation Research and Econometrics, Universite Catholique De Louvain, Belgium, 1994.

N. Shimkin. Stochastic games with average cost constraints. In T. Basar and A. Haurie, editors, *Advances in Dynamic Games and Applications*, pages 219–230. Birkhauser, 1994.

X. Spinat. A necessary and sufficient condition for approachability. *Mathematics of Operations Research*, 27(1):31–44, 2002.

M. Zinkevich. Online convex programming and generalized infinitesimal gradient ascent. In *Proceedings of ICML*, 2003.

Journal of Machine Learning Research 10 (2009) 591-622 Submitted 8/06; Revised 9/08; Published 3/09

NEUROSVM: An Architecture to Reduce the Effect of the Choice of Kernel on the Performance of SVM

Pradip Ghanty PRADIP.GHANTY@GMAIL.COM
Praxis Softek Solutions Pvt. Ltd.
Module 616, SDF Building, Sector V, Salt Lake City
Calcutta - 700 091, India

Samrat Paul SAMRAT.PAUL@GMAIL.COM
IBM India Pvt. Ltd.
DLF IT Park, 4th Floor, Tower C, New Town Rajarhut
Calcutta - 700 156, India

Nikhil R. Pal NIKHIL@ISICAL.AC.IN
Electronics and Communication Sciences Unit
Indian Statistical Institute
203, B. T. Road
Calcutta - 700 108, India

Editor: Yoshua Bengio

Abstract

In this paper we propose a new multilayer classifier architecture. The proposed hybrid architecture has two cascaded modules: feature extraction module and classification module. In the feature extraction module we use the multilayered perceptron (MLP) neural networks, although other tools such as radial basis function (RBF) networks can be used. In the classification module we use support vector machines (SVMs)—here also other tool such as MLP or RBF can be used. The feature extraction module has several sub-modules each of which is expected to extract features capturing the discriminating characteristics of different areas of the input space. The classification module classifies the data based on the extracted features. The resultant architecture with MLP in feature extraction module and SVM in classification module is called NEUROSVM. The NEUROSVM is tested on twelve benchmark data sets and the performance of the NEUROSVM is found to be better than both MLP and SVM. We also compare the performance of proposed architecture with that of two ensemble methods: majority voting and averaging. Here also the NEUROSVM is found to perform better than these two ensemble methods. Further we explore the use of MLP and RBF in the classification module of the proposed architecture. The most attractive feature of NEUROSVM is that it practically eliminates the severe dependency of SVM on the choice of kernel. This has been verified with respect to both linear and non-linear kernels. We have also demonstrated that for the feature extraction module, the full training of MLPs is not needed.

Keywords: feature extraction, neural networks (NNs), support vector machines (SVMs), hybrid system, majority voting, averaging

1. Introduction

A classifier designed from a data set $X = \{\mathbf{x}_i | i = 1, 2, \ldots, N, \mathbf{x}_i \in \Re^p\}$, where $\Re^p$ is the p dimensional real space, can be defined as a function $G : \Re^p \rightarrow N_c$. Here $N_c = \{\mathbf{y} \in \Re^c : y_k \in \{0,1\} \forall k, \sum_{k=1}^{c} y_k = 1\}$ is the set of label vectors and c is the number of classes. For any input vector $\mathbf{x} \in \Re^p$, $G(\mathbf{x})$ is a vector in c dimension with only one component as 1 and all others 0. In this paper our primary objective is to find a good G combining neural networks (NNs) and support vector machines (SVMs).

In machine learning literature NN and SVM are two widely used classifiers. NNs have been developed for many years and been used in various applications (Haykin, 1999; Pal et al., 2006). The SVM (Vapnik, 1995) is a classification and regression tool. It is comparatively a new family of learning tools including training algorithms for optimal margin classifiers (Boser et al., 1992) and support vector networks (Cortes and Vapnik, 1995). In SVM the input data are often transformed into a high dimensional space using some kernel functions. A linear separating hyper plane with the maximal margin between the closest positive and closest negative samples in the mapped space is found. The SVM works by solving a quadratic optimization problem that minimizes a sum of two terms. The first term is related with the reciprocal of norm of weight vector associated with the hyper plane and the second term is related to the sum of classification error. The SVM is a very active topic of research (von Luxburg et al., 2004; Adankon and Cheriet, 2007) and it has been successfully applied to many areas including handwritten digit recognition (Vapnik, 1995), object recognition (Pontil and Verri, 1998), protein structure prediction (Nguyen and Rajapakse, 2003) and texture classification (Kim et al., 2002). But there are some computational difficulties associated with using SVM. One of them is the required memory, which grows very quickly with the size of the training data since the SVM algorithm involves solving a large quadratic programming problem where every training data point forms a constraint. This is a constraint on the application of SVM to very large data sets. More importantly, the performance of SVM is significantly dependent on the choice of kernel. Needless to say that for non-linearly separable data, the performance of linear and nonlinear SVM also differs significantly.

Use of an ensemble of classifiers is a popular approach to improve the classification performance. Many ensemble methods are used by researchers to report the improvement in performance over single classifier (Hansen, 1999; Maqsood et al., 2004; Chawla et al., 2004). An ensemble of classifiers can be constructed using both homogeneous and heterogeneous classifiers (Hansen, 1999; Prevost et al., 2003; Garcia-Pedrajas et al., 2005). An ensemble of neural networks is often used for pattern classification problems (Garcia-Pedrajas et al., 2005; Islam et al., 2003) including face recognition (Melin et al., 2005), weather forecasting (Maqsood et al., 2004), protein secondary structure prediction (Guimaraes et al., 2003). Different approaches for constructing ensemble of neural networks have been suggested in the literatures (Wu et al., 2001; Zhou et al., 2002; Windeatt, 2006). In this paper for the purpose of comparison we have considered two ensemble methods for neural networks, one uses the average output of the ensemble of networks while the other one makes the ensemble vote on a classification task.

In this context, the ensemble method of Garcia-Pedrajas et al. (2007) needs a special attention as this method also uses a multilayer perceptron network for feature extraction and hence one may get a false impression that this method and our proposed method are quite similar.

This is an ensemble method where a large number of classifiers are trained and then their outcomes are aggregated using the majority voting rule. This is an interesting method but quite different from our proposed scheme.

Like AdaBoost the first baseline classifier is trained using the original training data while each of the subsequent classifiers is trained using a projected data set created using the hidden output of a trained MLP. The second baseline classifier uses data projected through the hidden layer of a projection network (MLP here). The projection network is an MLP network with number of hidden nodes equal to the number of inputs in the original training data and it is trained using only that *subset* of the training data which are not classified correctly by the first baseline classifier. The projection network (again an MLP with number of hidden nodes equal to the number of inputs in the original data) for the third baseline classifier is trained using the original data points whose projected versions are wrongly classified by the second baseline classifier. The process is repeated to generate a large number of baseline classifiers.

Note that, our proposed method falls in the category of hybrid system. There have been several attempts to combine different machine learning tools to develop efficient hybrid systems for pattern classification problems (Huang and LeCun, 2006; Happel and Murre, 1994; Vincent and Bengio, 2000; Mitra et al., 2006, 2005). To design a hybrid system different combination of classifiers is used including neural network-SVM (Mitra et al., 2005, 2006; Vincent and Bengio, 2000), convolution network-SVM (Huang and LeCun, 2006). Neural networks and support vector machines are used to design a hybrid system for text classification in Mitra et al. (2005) and Lidar detection of underwater objects in Mitra et al. (2006). Mitra et al. (2005) proposed a hybrid system called neuro-SVM which takes the component wise product of the outputs of a cascaded-SVM classifier and a recurrent neural network, and applies a set of heuristic rules to decide on the class. In the work of Mitra et al. (2006), after preprocessing Lidar signal is modeled using a polynomial as well as a linear predictor. The optimal coefficients of the polynomial are used as inputs to train a RBF, while coefficients of the linear predictor are used to train an MLP. The products of the corresponding components of the output vectors from the two networks are used as input to a cascaded-SVM classifier. Huang and LeCun (2006) presented a hybrid system for object recognition that uses the outputs of the last hidden layer of a convolution network to train a SVM with Gaussian kernel. The convolution network is generally used for computer vision problems. A convolution network has several hidden layers alternately consisting of convolution layer and sub-sampling layer. In a convolution network, the successive layers are designed to learn progressively higher-level features until the last layer, which produces categories.

There have been a number of attempts to develop modular networks to solve complex problems efficiently (Ronco and Gawthrop, 1995; Bottou and Gallinari, 1991). The basic philosophy of developing a modular network is to divide the task into a number of, preferably, meaningful subtasks, and then design one module for each subtask. Finally one needs to devise a mechanism to integrate these modules—this will dictate how different modules interact and lead to the final output. Sometimes the knowledge of the problem domain can be used to find the subtasks, but often clustering is used for this purpose. For example in Pal et al. (2003) a self organized map (SOM) is used to find natural clusters (subtasks) in the data and then for each cluster a separate network is trained. A given input is routed to the appropriate MLP module using the SOM. Jenkins and Yuhas (1993) have presented a simple solution to the truck backer-upper problem by decomposing it into subtasks. Then all subtasks are realized in parallel (that is, off line) to obtain the final two-layer feed-forward network, which is used to control the truck. Although our proposed architecture uses several modules, this is not designed following the usual principle of modular network.

In this paper we propose a new classifier architecture called NEUROSVM. The proposed classifier has two modules. In the first module we have used an MLP. We view the first module as a feature extraction module (FM), because outputs of this module can be used as inputs to any other classifier. This new set of features is used in the next module, termed as the classification module (CM). In the classification module we have used SVM with different kernel functions. Instead of SVM, one can use any other classifier also. We also consider the MLP and RBF neural networks in the CM of our proposed architecture. To further demonstrate the effectiveness of NEUROSVM we compare it with two other ensemble methods: majority voting and averaging. We demonstrate the effect of the kernel on SVM and NEUROSVM.

Our proposed method is neither an ensemble method nor has any relation to boosting. There is only one classifier. The classifier uses features extracted from the hidden nodes of several trained networks where typically the number of hidden nodes in a network is smaller than input dimension. Each network used for feature extraction is trained using the same data and each network sees the entire input space as represented through the training data. Thus typically to get improved performance we need fewer feature extraction networks than that would be needed by the ensemble type methods.

2. Methods

The section is arranged as follows. First, we provide a brief description of neural networks for the sake of completeness. Next, we give a brief description of the support vector machine (SVM) classifier and how several binary SVMs can be combined to solve a multiclass problem. Then we explain two popular existing ensemble methods that will be used for comparison. This is followed by a detailed discussion of the proposed method.

2.1 MLP and RBF Neural Networks

The two most widely used neural networks for pattern recognition are multilayer perceptron (MLP) and radial basis function (RBF) networks (Haykin, 1999). We have used the back-propagation algorithm for training MLP networks with single hidden layer.

The RBF network consists of exactly three layers: input layer, basis function layer and output layer. Unlike MLP, the activation functions of the hidden nodes are not of sigmoidal type, rather each hidden node represents a radial basis function. The transformation from the input space to the hidden space is nonlinear but each node in the output layer computes just the weighted sum of the outputs of the previous layer, that is, each output layer node makes a linear transformation. The learning of RBF network is usually performed in two phases. An unsupervised learning method is applied to estimate the basis function parameters. Then a supervised learning method, such as gradient descent or least square error estimate, is applied to tune the network weights between the hidden layer and the output layer. However, the parameters of the basis functions can also be tuned using gradient descent technique. Here we have used the MATLAB implementation of RBF network.

2.2 Support Vector Machines (SVMs)

The basic SVM (Haykin, 1999; Vapnik, 1995) formulation is for two class problems. If the training data are linearly separable, then SVM finds an optimum hyperplane that maximizes the margin of separation between the two classes.

Given a training set (X,Y), $\mathbf{x}_i \in X$, $\mathbf{x}_i \in \Re^p$ and $y_i \in Y$, the class label associated with $\mathbf{x}_i$; $y_i \in \{-1,+1\}$, the learning problem for SVMs is to find the weight vector $\mathbf{w}$ and bias b such that they satisfy the constraints:

$$\mathbf{x}_i.\mathbf{w}+b \geqslant +1 \qquad \text{for } y_i = +1 \tag{1}$$

$$\mathbf{x}_i.\mathbf{w}+b \leqslant -1 \qquad \text{for } y_i = -1 \tag{2}$$

and the weight vector $\mathbf{w}$ minimizes the cost function

$$\Phi(\mathbf{w}) = \frac{1}{2}\mathbf{w}^T\mathbf{w}.$$

The constraints written in Equations (1)-(2) can be combined as

$$y_i(\mathbf{x}_i.\mathbf{w}+b) \geqslant +1\ \forall i.$$

If the training points are not linearly separable, then there is no hyperplane that separates them into positive and negative classes. In this case, the problem is reformulated considering the slack variables $\xi_i \geqslant 0; i = 1,2,\ldots,N$. For most $\mathbf{x}_i$, $\xi_i = 0$. The constraints are now modified as follows:

$$\mathbf{x}_i.\mathbf{w}+b \geqslant +1-\xi_i \qquad \text{for } y_i = +1 \tag{3}$$

$$\mathbf{x}_i.\mathbf{w}+b \leqslant -1+\xi_i \qquad \text{for } y_i = -1 \tag{4}$$

$$\xi_i \geqslant 0,\ \forall i. \tag{5}$$

The SVM then finds $\mathbf{w}$, minimizing

$$\Phi(\mathbf{w},\xi) = \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{N}\xi_i$$

subject to constraints as in Equations (3)-(5). The constant C is termed as a regularization parameter as it controls the trade off between the complexity of the machine and the number of misclassifications.

Typically, when the training points are not linearly separable, a nonlinear mapping φ is used to map the training data from $\Re^p$ to some higher dimensional feature space H, with a hope that the data may be linearly separable in H. The mapping is implicitly realized using a kernel function.

Two kernels that are popular for non-linear SVMs are:

1. Polynomial of degree d: $K(\mathbf{x},\mathbf{x}_i) = (s\mathbf{x}.\mathbf{x}_i+1)^d$, where s is the scaling coefficient of the dot product.

2. Radial Basis Function (RBF): $K(\mathbf{x},\mathbf{x}_i) = e^{-\gamma\|\mathbf{x}-\mathbf{x}_i\|^2}, \gamma > 0$.

In this study, we shall extensively use the RBF kernel with a wide range of γ. We shall also demonstrate the utility of the proposed method with polynomial kernel.

We have used SVM^{light} (Joachims, 2002) software for learning the SVM classifier. Note that, NEUROSVM uses SVM^{light} in the classification module. We also use SVMs on the original data to compare its performance with that of NEUROSVM.

2.3 SVM for Multiclass Problems

The preceding SVM formulation is for two class problems. Multiclass SVMs are generally realized using several two class SVMs. We use the One versus One (OVO) method (Nguyen and Rajapakse, 2003; Weston and Watkins, 1999). Let us assume that we have a c class problem. In this method we construct one binary classifier for every pair of distinct classes. So we get $c \times (c-1)/2$ binary classifiers for a c class problem. In the training data, suppose k_i samples are from class $i, N = \sum_{i=1}^{c} k_i$. For the class pair (i, j), a binary classifier C_{ij} is trained using k_i and k_j data points from class i and j. An unknown sample $\mathbf{x}$ is then classified by each of the $c \times (c-1)/2$ different classifiers. If classifier C_{ij} classifies $\mathbf{x}$ as class i then the vote for class i for sample $\mathbf{x}$ is increased by one. Otherwise, vote for class j for sample $\mathbf{x}$ is increased by one. In this way for sample $\mathbf{x}$, the votes for all c classes are calculated using the output of all $c \times (c-1)/2$ classifiers. After that we assign $\mathbf{x}$ to class l, if class l has the largest number of votes for $\mathbf{x}$. Ties are randomly resolved.

2.4 Ensemble Methods: Majority Voting and Averaging

Different methods of classifier fusion are available in the literature (Maqsood et al., 2004; Ko et al., 2007; Brown et al., 2005; Tang et al., 2006; Kuncheva and Whitaker, 2003; Windeatt, 2006; Islam et al., 2003), of which the majority voting scheme is probably the most popular method (Stepenosky et al., 2006). In this method, the final class is determined by the maximum number of votes counted among all the classifiers fused. Let us consider a c class problem and let m be the number of classifiers to be fused. For an unknown sample $\mathbf{x}$, vote for class j, $v_j, (j = 1, 2, \ldots, c)$ is computed from the ensemble of classifiers $C_i, (i = 1, 2, ..., m)$. If $C_i, (i = 1, 2, ..., m)$ assigns sample $\mathbf{x}$ to class j then v_j is incremented by 1. Note that, initially vote for every class is initialized to 0; that is, $v_j = 0, (j = 1, 2, ..., c)$. The final class determination by the ensemble for sample $\mathbf{x}$ is k, if $v_k = \max_{j=1}^{c}\{v_j\}$.

Averaging also is a simple but effective method and is used in many classification problems (Guimaraes et al., 2003; Naftaly et al., 1997). In this method, the final class is determined by the average of continuous outputs of all classifiers (here MLPs) fused. For an unknown sample $\mathbf{x}$, let the output for class j $(j = 1, 2, ..., c)$ from classifier $C_i, (i = 1, 2, ..., m)$ be o_{ij}. Then the output from the ensemble classifier is obtained as $O_j = \frac{1}{m} \sum_{i=1}^{m} o_{ij}, \; j = 1, 2, \ldots, c$. The final class assignment by the ensemble to $\mathbf{x}$ is k, if $O_k = \max_{j=1}^{c}\{O_j\}$.

2.5 Proposed NEUROSVM Classifier

The proposed multilayer architecture can be thought of as a combination of two types of modules: feature extraction module (FM) and classification module (CM). The FM consists of a number of sub-modules SFM$_i$, $i = 1, 2, \ldots, m$. Each sub-module SFM$_i$ takes the same p dimensional data $\mathbf{x} = (x_1, x_2, \ldots, x_p)^T$ as input and produces n_i dimensional output vectors $\mathbf{v}_i = (v_{i1}, v_{i2}, \ldots, v_{in_i})^T$. Thus $n = \sum_{i=1}^{m} n_i$ output values together as shown in Equations (6) and (7) constitute an n dimensional input to the classification module.

$$\mathbf{z} = \begin{pmatrix} \mathbf{v}_1 \\ \mathbf{v}_2 \\ \vdots \\ \mathbf{v}_m \end{pmatrix} \in \Re^{n_1+n_2+\ldots+n_m} \tag{6}$$

and

$$\begin{aligned} \mathbf{v}_1 &= (v_{11}, v_{12}, \ldots, v_{1n_1})^T, \\ \mathbf{v}_2 &= (v_{21}, v_{22}, \ldots, v_{2n_2})^T, \\ &\vdots \\ \mathbf{v}_m &= (v_{m1}, v_{m2}, \ldots, v_{mn_m})^T. \end{aligned} \tag{7}$$

In general, different SFM$_i$ can use different methods of feature extractions or they can use the same principle for feature extraction. Similarly, the classification module can use any principle like neurocomputing, support vector machines and so on.

In this investigation, the sub-modules SFM$_i$s are derived from multilayer perceptron networks, while the classification module consists of support vector machines. And, hence, we call the resulting architecture NEUROSVM.

In order to constitute the i^{th} sub-module SFM$_i$, we consider an MLP with just one hidden layer, with architecture (p, n_i, c), where p is the input dimension, n_i is the number of nodes in the hidden layer and c is the number of classes. Note that, although the number of input and output nodes in each MLP remains the same, the number of nodes in the hidden layer could be different for different MLPs. Each MLP is then trained using the training data $X = \{\mathbf{x}_i; i = 1, 2, \ldots, N\} \subset \Re^p$, $Y = \{\mathbf{y}_i; i = 1, 2, \ldots, N\} \subset \Re^c$ where $\mathbf{y}_i$ is the target output corresponding to $\mathbf{x}_i$.

Once each network is trained, the output of the hidden layer can be taken as the extracted features. These features capture characteristics of the data that can discriminate between classes; hence using these features we can do the classification job using just a single layer network.

Note that, instead of MLP, we can use RBF also in the feature extraction module. In Figure 1, the top panel has m different trained MLPs labeled as $\text{MLP}_1, \text{MLP}_2, \ldots, \text{MLP}_m$. After the training, we remove the output layer and its associated connections from each of the MLPs and then the truncated two-layer sub-networks are taken as feature extraction sub-modules. The subnets $\text{SFM}_1, \text{SFM}_2, \ldots, \text{SFM}_m$ in the lower panel of the NEUROSVM are constructed from the trained MLPs in the upper panel. The first two layers of MLP$_i$ constitute $\text{SFM}_i, i = 1, 2, \ldots, m$.

As depicted in the lower panel of Figure 1, the output from the m sub-modules, taken together constitutes the input to the classification module. Here we consider SVMs for classification, but other classifiers such as neural networks (MLP or RBF network) can also be used. Note that, each sub-module receives the same input $\mathbf{x} = (x_1, x_2, \ldots, x_p)^T$.

Given the training data X and Y, in order to train the CM we use the following data set. For each $\mathbf{x}_i \in X$, the FM produces an output $\mathbf{z}_i \in \Re^n$ as in Equation (6). Like an MLP, every node in the second layer of NEUROSVM computes the weighted sum of its input and applies a sigmoidal activation function to produce its output. Thus $Z = \{\mathbf{z}_i; i = 1, 2, \ldots, N\}, \mathbf{z}_i \in \Re^n$, as in Equation (6), is used as the input data and corresponding to each $\mathbf{z}_i \in Z$, the associated $\mathbf{y}_i \in \Re^c$, $\mathbf{y}_i \in Y$ is taken as the target output. The CM is trained using (Z, Y).

In the present case the CM has two layers. The first layer, as shown in Figure 1, is the SVM kernel layer where each node is associated to a mapped training sample $\mathbf{z}_i$ (it is the output from an FM that represents a support vector) and it computes the kernel output $K(\mathbf{z}, \mathbf{z}_i)$ on a mapped input $\mathbf{z}$, while the other layer is the output layer.

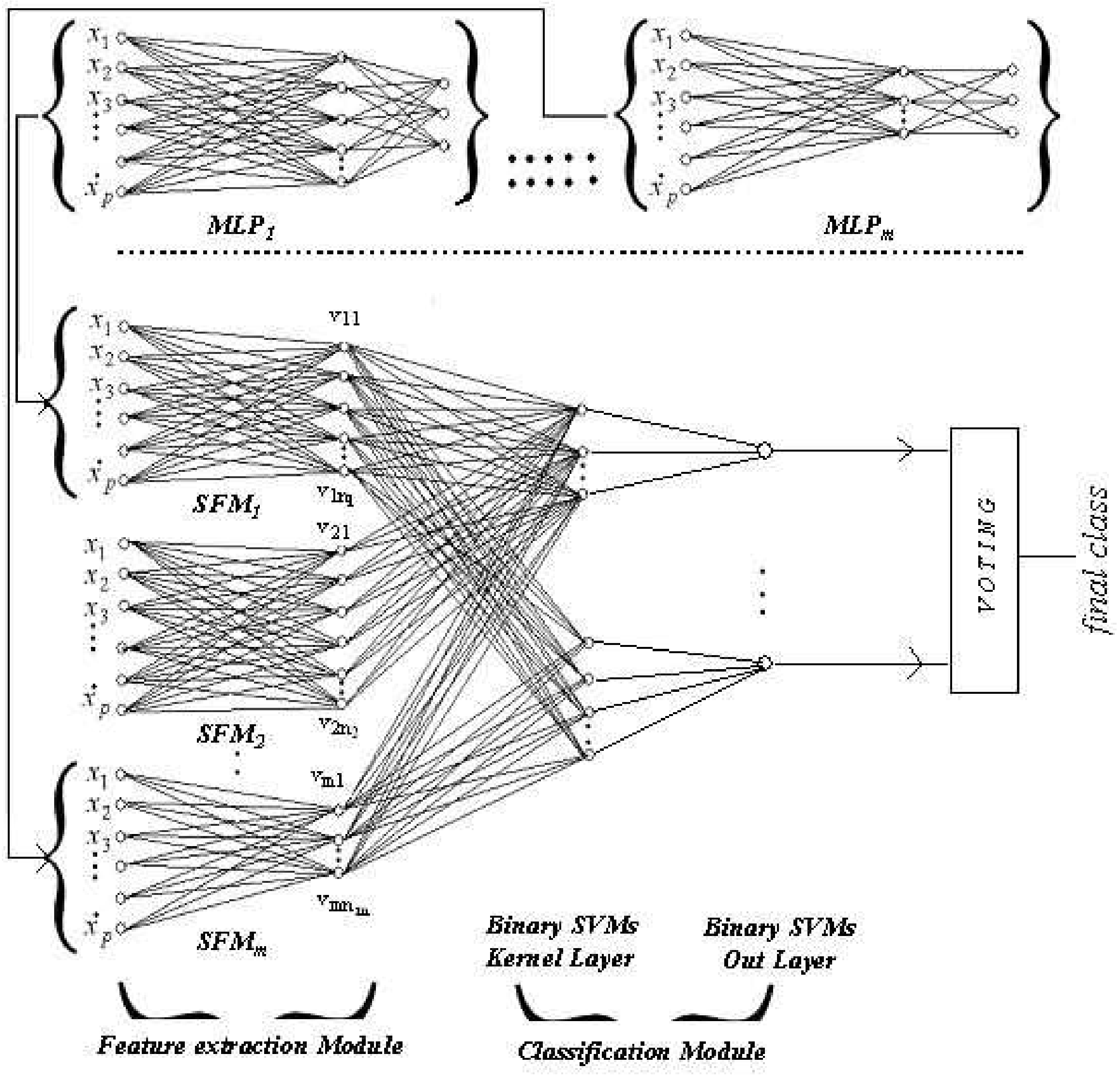


Figure 1: The proposed NEUROSVM classifier

2.6 Advantages of the Proposed Method

A natural question comes, why such an architecture (NEUROSVM) will be better or more useful than the usual SVM or MLP? There are number of reasons behind this. Note that, we are not considering very simple data sets where most classifiers will lead to zero training-test errors.

1. Typically, due to the local minima problem of MLP training and its dependence on initialization, different MLPs may learn different areas of the input space better. Hence when we combine the output of the hidden layer of different networks to generate new features, the learning task becomes simpler to the CM. This is true irrespective of whether the CM is a neural network or SVM.

2. The extracted features result in simpler classification boundaries because a single layer network can classify the new data (consider a two layer network consisting of the hidden and output layers of an MLP). This also makes the learning task of the CM simpler.

3. For high dimensional data, typically the number of nodes in the hidden layer is much smaller than the number of the input nodes and one does not need many feature extraction submodules (SFMs). Hence, the dimensionality of the input for the CM can be reduced compared to the original dimension of the input. This makes simpler error surface, faster learning and allows us to do more experiments, if CM is a neural network.

4. This is not an ensemble method but it makes fusion of salient characteristics of the input space as extracted/learnt by different feature extraction networks. It can at least be viewed as an implicit fusion of multiple classifiers, and hence improvement in performance is expected.

5. For large data sets, it may not be necessary to make full training of the MLPs for constructing the SFM$_i$s, because the objective of the MLPs here is to capture the inherent attributes of the data by the FM.

For low dimensional data sets or simpler data sets this method may not have much advantage because then n (dimension of input to the CM) can be more than p (original dimension of the input) and different SFMs may capture the same attributes of the data resulting in not much of benefits. Note that, the advantages mentioned in 2 and 3 are also applicable to MLPs.

3. Experiments

The section is arranged as follows. First we have listed the selected data sets to validate our proposed method. Then experimental setup is described. Next, the experimental results are presented. Finally, a control experiment to justify one of the advantages of the proposed method is demonstrated.

3.1 Data Sets

To demonstrate the effectiveness of the proposed method, we consider twelve data sets from the UCI Machine Learning Repository (Blake and Merz, 1998). We divide the data sets into two Groups: A and B. The Group A consists of eight data sets: Iris, Vehicle, Breast Cancer (WDBC), Glass, Sonar, Ionosphere, Lymphography Domain (Lymph) and Pima Indians Diabetes (Pima) data. The Group B contains Pendigits, Image-Segmentation (Img. Seg.), Landsat satellite image (Sat. Img.) and Optdigits data. For Group A data sets some results are available in the literature but the details of the experimental protocols (such as training/test divisions) used are not available. Hence, we report the performance with ten-fold cross-validation experiments. Each data set is divided into ten subsets of almost equal size. One of the subsets is used for testing and the remaining nine subsets are used for training. The procedure is repeated ten times and the average performance is reported. We report the results in terms of mean test error and its standard error for Group A data sets. For the four data sets in Group B, benchmark results with different classifiers are available along with the training-test partition. Hence we have used the same training-test partition here and report the error on the fixed test set. Table 1 and Table 2 summarize the Group A and Group B data sets respectively.

Data set	No. of classes	No. of features	Size of the data set and class wise distribution
Iris	3	4	150 (= 50 + 50 +50)
Vehicle	4	18	846 (=212+217+218+199)
WDBC	2	30	569 (=212 + 357)
Glass	6	9	214 (=70+76+17+13+9+29)
Sonar	2	60	208 (=97+111)
Ionosphere	2	34	351 (=225+126)
Lymph	4	18	148 (=2+81+61+4)
Pima	2	8	768 (=500+268)

Table 1: Group A data sets

Data set	No. of classes	No. of features	Training data		Test data	
			Size	Class distribution	Size	Class distribution
Pendigits	10	16	7494	780, 779, 780, 719 780, 720, 720, 778 719, 719	3498	363, 364, 364, 336 364, 335, 336, 364 336, 336
Img. Seg.	7	18	210	30 in each class	2100	300 in each class
Sat. Img.	6	4	500	104, 68, 108, 47 58, 115	5935	1429, 635, 1250, 579 649, 1393
Optdigits	10	64	3823	376, 389, 380, 389 387, 376, 377, 387 380, 382	1797	178, 182, 177, 183 181, 182, 181, 179 174, 180

Table 2: Group B data sets

3.2 Experimental Setup

In this subsection we describe the selection method for hyper parameters of MLP and SVM classifiers. To select the optimal architecture for an MLP, Andersen and Martinezr (1999) used ten-fold cross-validation experiments. Adankon and Cheriet (2007) discussed another scheme for SVM model selection. Here we have used ten-fold cross-validation experiments for MLP architecture selection as well as for selection of SVM kernel parameters. For Group B data sets training-test partitions are fixed and hence we have used ten-fold cross-validation on the training set to select the hyper parameters of classifiers. For Group A data sets, as mentioned earlier, the performances are reported based on ten-fold cross-validation. So, we perform double blind ten-fold cross-validation experiments to select hyper parameters of classifiers for Group A data sets.

Note that, for the FM of NEUROSVM, we need to select $m \geqslant 1$ MLPs. A natural choice would be to select the best m architectures corresponding to the smallest m values of validation errors.

Based on validation error we choose m architectures for each of the ten folds for Group A data sets and m architectures for each of the Group B data sets for NEUROSVM.

In a similar manner the regularization parameter C and spread γ of RBF kernels of SVMs are chosen based on ten-fold cross-validation experiments. We have experimented with n_c choices of C and n_g choices of γ. So, we have used $n_c \times n_g$ sets of choice of parameters. For each choice, the ten-fold cross-validation experiments are conducted. Here we also select the (C,γ) pair that leads to the minimum average validation error. In this investigation $n_c = 12$ and $n_g = 15$ are used resulting in a total of 180 pairs of parameters.

We have also used ten-fold cross-validation to find the sub-modules for NEUROSVM. The hyper parameters of SVMs in the classification module of NEUROSVM are also estimated through ten-fold cross-validation experiments. Note, for Group A data sets we have used double blind ten-fold cross-validation. We have selected m (=5) SFMs. Hence using the m SFMs we can generate $2^m - 1$ different feature subsets combinations. In our case it is $2^5 - 1 = 31$. Then for each of the 31 combinations with all 180 pairs of (C,γ) we have conducted the ten-fold cross-validation experiments on training set(s). We have obtained the best (C,γ) for each of the 31 combinations. Then the best combination is chosen based on the minimum average validation error over all 31 combinations. Finally using the best combination and corresponding (C,γ) pair the performance of NEUROSVM is reported.

We have performed statistical tests (Dietterich, 1998) to compare the proposed algorithms with that of standard algorithms. For Group A data sets where cross-validation is performed, we have applied the ten-fold cross-validation paired t-test with 9 degrees of freedom and 95% significance level. For the four data sets of Group B where a single test set is employed, we have constructed McNemar test with 1 degree of freedom and 95% significance level. The formulations of these tests are as follows.

3.2.1 K-FOLD CROSS-VALIDATION PAIRED T-TEST (DIETTERICH, 1998)

Consider two classifier models, D_1 and D_2, and a data set X. The data set is split into K parts of approximately equal sizes, and each part is used in turn for testing of a classifier built on the pooled remaining $K-1$ parts. Classifiers D_1 and D_2 are trained on the training set and tested on the test set. Denote the observed test accuracies as P_1 and P_2, respectively. This process is repeated K times and test accuracies are tagged with superscript $(i), i = 1,2,\ldots,K$. Thus a set of K differences is obtained, $P^{(1)} = P_1^{(1)} - P_2^{(1)}$ to $P^{(K)} = P_1^{(K)} - P_2^{(K)}$. Under the null hypothesis (H_0: equal accuracies), the following statistic has a t-distribution with $K-1$ degrees of freedom

$$t = \frac{\overline{P}\sqrt{K}}{\sqrt{\sum_{i=1}^{K}(P^{(i)} - \overline{P})^2/(K-1)}},$$

where $\overline{P} = (1/K)\sum_{i=1}^{K} P^{(i)}$. If the calculated t is greater than the tabulated value for chosen level of significance (here 0.05) and $K-1$ (here 9) degrees of freedom, we reject null hypothesis H_0 and accept that there are significant differences in the two compared classifier models.

3.2.2 McNemar Test (Dietterich, 1998)

As done before consider two classifiers D_1 and D_2. Let us define the following: N_{00} = number of samples which both D_1 and D_2 classify incorrectly, N_{01} = number of samples which D_1 classifies incorrectly but D_2 classifies correctly, N_{10} = number of samples which D_1 classifies correctly but D_2 classifies incorrectly and N_{11}= number of samples which both D_1 and D_2 classify correctly. Let, $N = N_{00} + N_{01} + N_{10} + N_{11}$ be the total number of samples in the test set. The null hypothesis, H_0, is that there is no difference between the accuracies of the two classifiers. If the null hypothesis is correct, then the expected counts for N_{01} and N_{10} are $\frac{1}{2}(N_{01} + N_{10})$. The discrepancy between the expected and the observed counts is measured by the following statistic

$$\chi^2 = \frac{(|N_{01} - N_{10}| - 1)}{N_{01} + N_{10}},$$

which is approximately distributed as χ^2 with 1 degree of freedom. To carry out the test we simply calculate χ^2 and compare it with the tabulated χ^2 value for a given level of significance, say, 0.05 (in our case).

We have performed all experiments using two Sun Blade 2500 with dual processors. The svm_learn and svm_classify modules for binary SVMs training and classification are used from SVM^{light} (Joachims, 2002) software. For the RBF neural network MATLAB toolbox is used. All other programs are written in c.

3.3 Experimental Results

In this subsection first we list the selected hyper parameters of MLP and SVM by cross-validation experiments. Next selection of sub-modules and hyper parameters of NEUROSVM is discussed. The performance comparison of NEUROSVM with the baseline classifiers and standard ensemble methods is presented. Finally, we present the performance of other variants of NEUROSVM and compare it with the baseline classifiers as well as ensemble methods.

3.3.1 Selection of Hyper Parameters for MLPs to Construct the FM

For Group A data set we use double blind ten-fold cross-validation. The partitioning of data for Group A data sets is explained in Appendix A. For each of the outer level cross validation loop, finding the optimal number of hidden nodes and computation of the test error are explained in the procedure RunMLP in Appendix B. The initial weights of the MLPs are chosen randomly in [-0.5, +0.5] and the learning rate used to train the MLPs is 0.60. The number of iterations used to train the networks for different data sets are chosen based on a few trial experiments. For each data set, a set of choices on the number of hidden nodes is used to train the MLPs. In Table 3, number of iterations and number of hidden nodes that are used to train the MLPs for the twelve data sets are listed. We have decided to use $m = 5$ neural networks for feature extraction modules and hence for each fold, we have to select a set of five hidden nodes to train five MLPs.

First we display the variation of the average validation error of cross-validation experiments as a function of the number of hidden nodes for both Group A and Group B data sets. Since for each data set in Group A 10 panels are required for the 10 folds, we include the figure for only one data set, Vehicle, in Figure 2. In Figure 3, four panels are included, one for each of the four data sets in Group B. In both Figures 2 and 3 we also include the average training errors. As mentioned earlier, for the FM of NEUROSVM, we want to use $m = 5$ networks (SFMs). Consider a data set in Group

A. Suppose, we have trained MLPs with M different architectures, that is, with M different choices of hidden nodes. Then for each of the outer level fold, we shall have M different hidden nodes each associated with an average validation error. Now we order these M hidden nodes in ascending order of the associated validation error. Then select the top five hidden nodes from this ordered list. These five different choices of hidden nodes will be used to train five MLPs for feature extraction for that particular fold. For each data set, in Table 4, we depict the list of selected hidden nodes for each fold (outer level). As an example, for the IRIS data for the first fold (outer level), the selected hidden nodes are (7, 2, 5, 6, 8). This means that for the first fold (outer level) we got the least validation error with 7 hidden nodes; the next smaller validation error is obtained with 2 hidden nodes and so on.

Since the first element of this set of five resulted in the smallest validation error, we use this choice of hidden nodes to train MLPs when we report the performance of the MLP networks as classifiers. For each data set in Group B, since the training and test partitions are fixed, we have only one outer loop and hence only one set with five choices of hidden nodes as shown in Table 4. We follow the same protocol as that of Group A data sets to choose the number of hidden nodes for computing the performance of MLP networks.

Data set	Training iterations	Hidden nodes explore
Iris	1500	2-10
Vehicle	2000	3-16
WDBC	1500	3, 5-10, 12, 15, 20
Glass	1500	2-15
Sonar	2000	3, 5, 7, 10, 12, 15, 20, 25, 30, 35, 40
Ionosphere	1500	5-10, 12, 15, 20, 25, 30
Lymph	1500	4-10, 12, 15, 20
Pima	1500	2-10
Pendigits	1500	5-10, 12, 15, 18, 20, 25
Img. Seg.	1500	3-10, 12, 15, 20
Sat. Img.	5000	2-10
Optdigits	1500	5, 8, 10, 12, 15, 18, 20, 25, 30, 35, 40, 50

Table 3: List of explore hidden nodes and number of iterations for MLP for the twelve data sets

3.3.2 SELECTION OF HYPER PARAMETERS FOR SVMS

In this section we consider the problem of selecting hyper parameters for a regular SVM that we shall use as benchmark in experiments for the purpose of comparison with NEUROSVM. To select the regularization parameter C and spread γ for RBF kernel of SVM classifiers we have tried a wide range of C and γ. In this experiment we have used 12 different values of C and 15 different values of γ resulting in a total of 180 pairs of (C,γ). The 12 different values of C are 0.001, 0.01, 0.10, 0.20, 0.50, 1.00, 2.00, 5.00, 10.00, 20.00, 100.00 and 1000.00. The 15 different values of γ that we have used are 0.0001, 0.001, 0.01, 0.10, 0.20, 0.40, 0.80, 1.00, 2.00, 5.00, 10.00, 20.00, 100.00, 1000.00 and 10000.00. In a manner similar to the way the optimal number of hidden node is chosen for each fold (outer level), the optimal (C,γ) is chosen using ten-fold cross-validation experiments. This is further explained by Procedure RunSVM included in Appendix C. For each of the twelve data sets,

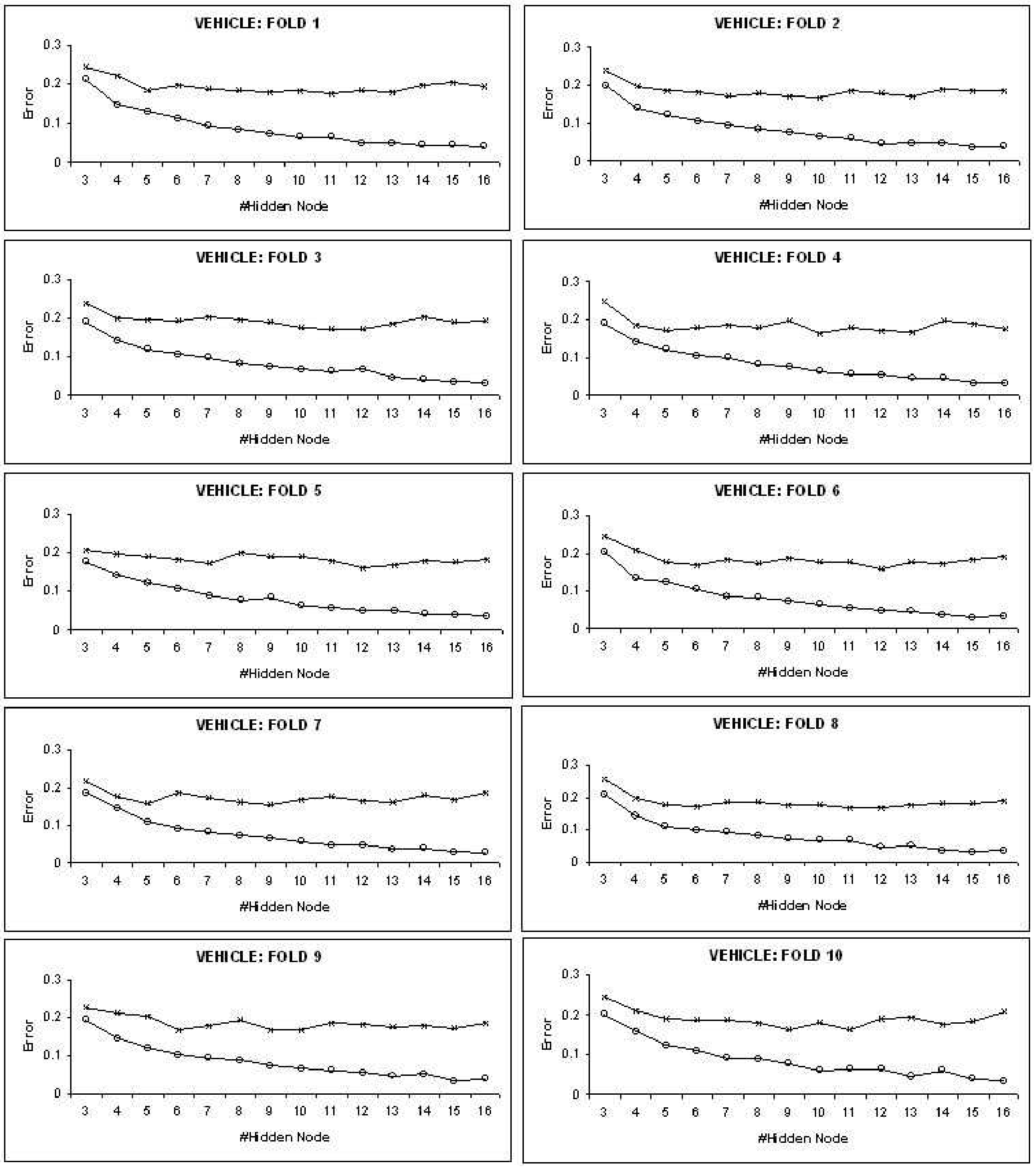


Figure 2: For each of the ten-folds the variation of cross-validation error with different choices of number of hidden nodes for MLPs on the Vehicle data set. The lines with cross-mark denote the validation error while the lines with circles denote the training error.

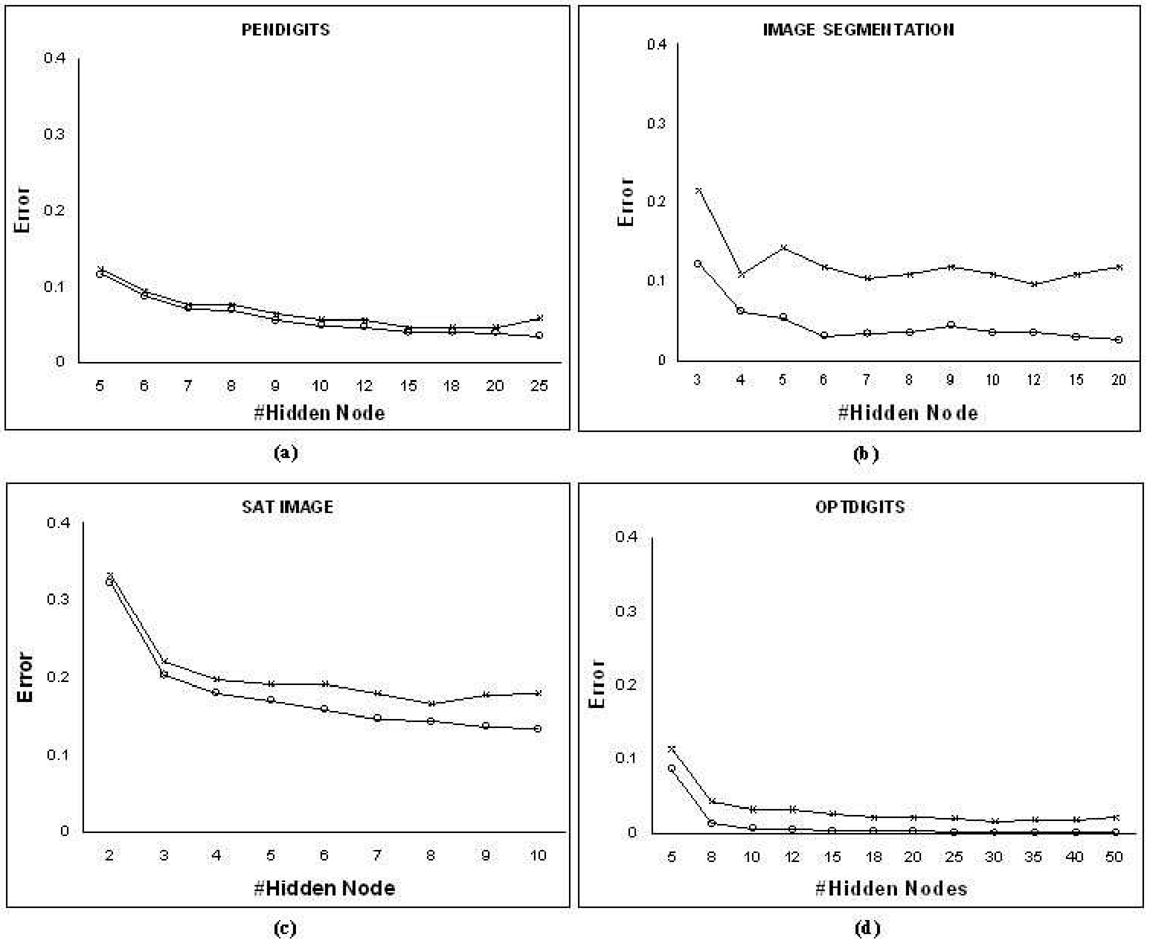


Figure 3: Variation of cross-validation error with different choices of number of hidden nodes for MLPs on four data sets in Group B: (a) Pendigits (b) Img. Seg. (c) Sat. Img. and (d) Optdigits. The lines with cross-mark denote the validation error while the lines with circles denote the training error.

the obtained optimal set of parameters is in Table 5. The results reported for SVMs correspond to these choices.

Data set	Hidden nodes
Iris	(7, 2, 5, 6, 8), (7, 9, 3, 6, 5), (7, 6, 5, 8, 9), (5, 6, 7, 8, 3), (7, 9, 10, 8, 6), (5, 6, 7, 8, 9), (7, 8, 9, 5, 6), (5, 6, 7, 8, 9), (9, 8, 7, 10, 6), (2, 5, 6, 7, 8)
Vehicle	(11, 9, 13, 10, 5), (10, 13, 9, 7, 8), (12, 11, 10, 13, 9), (10, 13, 12, 5, 16), (12, 13, 7, 15, 14), (12, 6, 14, 8, 5), (9, 5, 8, 13, 12), (12, 13, 6, 11, 9), (11, 6, 10, 15, 13), (9, 11, 14, 8, 10)
WDBC	(12, 10, 9, 15, 7), (10, 6, 8, 9, 7), (8, 15, 12, 9, 10), (12, 7, 8, 15, 10), (15, 8, 12, 10, 9), (8, 20, 15, 10, 7), (12, 10, 15, 7, 9), (7, 15, 8, 9, 10), (12, 15, 6, 9, 10), (9, 7, 20, 10, 12)
Glass	(11, 15, 13, 4, 9), (12, 9, 13, 7, 8), (13, 12, 7, 10, 14), (11, 14, 12, 15, 10), (10, 15, 9, 13, 6), (10, 12, 9, 14, 8), (14, 5, 13, 4, 8), (11, 13, 12, 10, 14), (12, 15, 8, 6, 9), (11, 13, 14, 15, 12)
Sonar	(25, 15, 12, 35, 30), (25, 35, 20, 30, 5), (30, 35, 7, 40, 10), (30, 5, 20, 25, 7), (20, 35, 15, 30, 25), (25, 30, 10, 12, 7), (30, 15, 25, 10, 20), (25, 35, 7, 10, 30), (20, 25, 7, 12, 15), (10, 12, 15, 7, 20)
Ionosphere	(7, 25, 10, 12, 15), (15, 9, 7, 20, 8), (15, 9, 8, 12, 20), (9, 12, 15, 8, 20), (8, 25, 12, 9, 10), (7, 9, 8, 20, 15), (10, 8, 7, 25, 15), (8, 20, 15, 12, 10), (15, 25, 20, 6, 5), (6, 15, 12, 20, 7)
Lymph	(9, 6, 15, 5, 10), (10, 8, 15, 6, 7), (9, 10, 8, 12, 20), (15, 10, 7, 20, 12), (9, 10, 12, 6, 20), (9, 15, 10, 8, 6), (7, 6, 10, 8, 9), (7, 10, 15, 12, 8), (12, 8, 9, 10, 6), (10, 12, 8, 15, 9)
Pima	(6, 7, 8, 9, 5), (7, 9, 8, 5, 6), (6, 7, 9, 8, 10), (7, 5, 9, 6, 8), (7, 9, 10, 6, 8), (7, 6, 9, 5, 8), (7, 5, 6, 8, 9), (8, 7, 9, 10, 5), (5, 9, 8, 7, 10), (6, 7, 8, 9, 5)
Pendigits	(15, 18, 20, 12, 10)
Img. Seg.	(12, 7, 8, 10, 15)
Sat. Img.	(8, 9, 7, 10, 6)
Optdigits	(30, 35, 40, 25, 20)

Table 4: List of selected hidden nodes for SFMs of the NEUROSVM for the twelve data sets selected by cross-validation experiments

3.3.3 SELECTION OF SFMS AND HYPER PARAMETERS FOR NEUROSVM

We have already explained, for each fold how to choose the number of hidden nodes for the five MLPs that will be required in the feature extraction module of NEUROSVM. These choices for different data sets are listed in Table 4. In order to use these data extraction MLPs, two issues need to be addressed. First do we need all five feature extraction MLPs, or for different folds, different subsets of the five would be more appropriate. In other words, for each fold, using five feature extraction MLPs we can have 31 possible combinations of feature sets. And we have to use the

Data set	(C,γ)
Iris	$(2.00,0.20),(2.00,0.20),(100.00,0.01),(1.00,0.20),$ $(20.00,0.01),(20.00,0.01),(0.10,1.00),(0.50,0.40),$ $(5.00,0.10),(1.00,0.40)$
Vehicle	$(100.00,0.0001),(1000.00,0.0001),(100.00,0.0001),(100.00,0.0001),$ $(100.00,0.0001),(100.00,0.0001),(100.00,0.0001),(1000.00,0.0001),$ $(100.00,0.0001),(100.00,0.0001)$
WDBC	$(5.00,0.0001),(20.00,0.0001),(0.20,0.0001),(2.00,0.0001),$ $(10.00,0.0001),(20.00,0.0001),(20.00,0.0001),(2.00,0.0001),$ $(20.00,0.0001),(2.00,0.0001)$
Glass	$(20.00,0.20),(5.00,0.80),(10.00,0.80),(5.00,0.80),$ $(10.00,0.80),(20.00,0.20),(5.00,0.10),(10.00,0.40),$ $(5.00,1.00),(10.00,1.00)$
Sonar	$(2.00,1.00),(2.00,1.00),(20.00,0.40),(20.00,0.20),$ $(2.00,0.80),(5.00,0.40),(10.00,0.40),(2.00,1.00),$ $(5.00,2.00),(5.00,0.40)$
Ionosphere	$(5.00,0.10),(20.00,0.20),(20.00,0.20),(20.00,0.40),$ $(100.00,0.01),(100.00,0.40),(2.00,0.10),(2.00,0.20),$ $(20.00,0.40),(5.00,0.40)$
Lymph	$(1000.00,0.0001),(1000.00,0.0001),(2.00,0.20),(100.00,0.001),$ $(1000.00,0.0001),(5.00,0.10),(1000.00,0.0001),(100.00,0.001),$ $(100.00,0.01),(1000.00,0.001)$
Pima	$(0.50,0.0001),(1.00,0.0001),(0.50,0.0001),(10.00,0.0001),$ $(5.00,0.0001),(1.00,0.0001),(5.00,0.0001),(10.00,0.0001),$ $(2.00,0.0001),(1.00,0.0001)$
Pendigits	$(10.00,0.0001)$
Img. Seg.	$(100.00,0.0001)$
Sat. Img.	$(20.00,0.01)$
Optdigits	$(20.00,0.0001)$

Table 5: List of regularization parameter and spread of the RBF kernel for SVMs selected by cross-validation experiments

most appropriate combination for each fold. The second issue is to find the optimal hyper parameter for each combination of feature sets. Thus for each fold, to obtain the best choice of combination of feature subsets and the associated optimal hyper parameter, for each of the 31 combinations, as we did for SVM (in Procedure RunSVM) we use ten-fold cross-validation. This is summarized in Appendix D by Procedure RunNEUROSVM. The selected combinations of SFMs along with the hyper parameters for the twelve data sets are listed in Table 6. The set within braces in the second column of Table 6 shows the best combination of feature extraction MLPs selected by cross-validation experiments. As an illustration, for fold 1 of Iris, a set of 7 features is used in the classification module, which is generated by two selected SFMs each with 2 and 5 hidden nodes. The (C,γ) pair within parenthesis followed by the combination shows the regularization parameter

(C) and spread (γ) of the RBF kernel for SVM classifiers in the classification module that are selected by the cross-validation. From Table 6 we see that NEUROSVM with single SFM is not selected for any data sets. Hence using just one SFM we shall not gain anything. The selected combination of SFMs and corresponding (C,γ) are used to report the results of NEUROSVM. From Table 5 and Table 6 we observe that the values of γ chosen for SVM are usually smaller than those for NEUROSVM. It is probably because the hidden layers of neural networks are more suited for linear classification than the original inputs, so a higher γ (less non-linearity) is more appropriate.

Four of the twelve data sets have dimensionality 30 or more. For these four data sets dimensionality is reduced in the classification module of NEUROSVM. The dimensions of the four data sets, WDBC, Sonar, Ionosphere and Optdigits, in the classification module of NEUROSVM are reduced by 16.67-56.67% (average 41.33%), 16.67-80.00% (average 48.33%), 47.06-67.65% (average 54.41%) and 14.06% respectively. Hence for high dimensional data the dimensionality of input for the CM can be reduced compared to original dimension of the input.

3.3.4 PERFORMANCE COMPARISON OF NEUROSVM WITH THE BASELINE CLASSIFIERS AND STANDARD ENSEMBLE METHODS

We compare the performance of NEUROSVM with MLP, SVM as well as two existing neural ensemble methods. The majority voting and averaging are simple yet effective ensemble methods. In Table 7, test error results of NEUROSVM, MLP, SVM, majority voting and averaging are shown. In Table 7, majority voting ensemble method is denoted by MVOTING while the average ensemble method is denoted by AVERAGING. The results in Table 7 show that based on the paired t-test for Group A data sets and McNemar test for Group B data sets NEUROSVM is significantly better than the baseline classifiers for 11 data sets when compared with MLP and for 6 data sets when compared with SVM.

Note that the results of MVOTING and AVERAGING in Table 7 are obtained using the same combinations of networks (SFMs) that are used in the FM of NEUROSVM. From Table 7 we see that NEUROSVM performs significantly better than the standard ensemble methods for 11 data sets when compared with majority voting and for 10 data sets when compared with averaging.

As a summary, NEUROSVM is superior to MLP, SVM as well as two ensemble methods for 6 data sets. These data sets are Vehicle, WDBC, Ionosphere, Lymph, Pima and Img. Seg. For four out of remaining six data sets NEUROSVM performs significantly better than MLP, MVOTING and AVERAGING. The performance of NEUROSVM and SVM for these four data sets, Iris, Sonar, Pendigits and Sat. Img, is not significantly different. For the Glass data, the performance of NEUROSVM is significantly better than MLP and MVOTING but is not significantly different from that of SVM and AVERAGING. For the Optdigits data set all algorithms perform equally well. No data set is found where two baseline classifiers (MLP and SVM) or two ensemble methods perform better than NEUROSVM.

For the results in Table 7, for each data set, the combination of SFMs used is selected by cross-validation for NEUROSVM. So, a natural question arises, will other combinations perform better with majority voting or averaging than NEUROSVM? To investigate this we have compared the performance of NEUROSVM using the combinations selected by cross-validation separately for each of MVOTING and AVERAGING. Following the same protocol as used for NEUROSVM, the SFMs for MVOTING and AVERAGING are selected using the double ten-fold cross-validation for Group A data sets and ten-fold cross-validation for Group B data sets. The selected combinations of

Data set		Selected combinations and (C,γ) pair of these combinations
Group A	Iris	$\{2,5\}(0.10,2.00)$, $\{3,5\}(0.10,2.00)$, $\{6,5\}(0.10,0.10)$, $\{5,3\}(0.10,2.00)$, $\{7,6\}(0.10,5.00)$, $\{5,6\}(0.10,1.00)$, $\{5,6\}(0.001,0.0001)$, $\{5,6\}(0.50,10.00)$, $\{7,6\}(1000.00,0.80)$, $\{2,5\}(0.10,2.00)$
	Vehicle	$\{11,13\}(1.00,0.20)$, $\{9,7\}(1000.00,2.00)$, $\{12,13\}(20.00,0.80)$, $\{10,5\}(0.50,0.10)$, $\{12,13\}(1000.00,0.40)$, $\{12,14\}(1000.00,2.00)$, $\{8,13\}(1000.00,0.001)$, $\{12,13\}(0.20,1.00)$, $\{11,6,10,13\}(1.00,0.40)$, $\{11,8\}(20.00,0.01)$
	WDBC	$\{10,15\}(0.50,0.20)$, $\{6,7\}(0.001,0.0001)$, $\{8,12\}(20.00,0.20)$, $\{7,10\}(0.10,5.00)$, $\{8,10\}(0.20,20.00)$, $\{8,7\}(0.50,0.40)$, $\{15,7\}(5.00,0.40)$, $\{7,8\}(0.01,0.40)$, $\{6,9\}(0.01,0.40)$, $\{9,7\}(0.50,0.01)$
	Glass	$\{11,15,9\}(1000.00,0.01)$, $\{7,8\}(1.00,2.00)$, $\{13,7\}(1.00,5.00)$, $\{11,10\}(2.00,10.00)$, $\{9,6\}(1000.00,0.01)$, $\{12,14,8\}(0.50.\ 0.20)$, $\{5,4\}(5.00,0.40)$, $\{11,14\}(0.10,0.20)$, $\{12,8,6\}(10.00,0.10)$, $\{14,15\}(1000.00,0.01)$
	Sonar	$\{12,30\}(0.20,2.00)$, $\{35,5\}(100.00,0.10)$, $\{30,7,10\}(0.20,2.00)$, $\{5,7\}(0.50,5.00)$, $\{20,30\}(1.00,2.00)$, $\{7,9\}(2.00,0.01)$, $\{25,20\}(1.00,5.00)$, $\{7,10\}(0.10,0.10)$, $\{7,12\}(0.10,0.20)$, $\{15,7\}(2.00,0.01)$
	Ionosphere	$\{7,10\}(0.001,0.0001)$, $\{7,8\}(0.001,0.0001)$, $\{9,8\}(0.001,0.0001)$, $\{9,8\}(0.001,0.0001)$, $\{8,9\}(0.001,0.0001)$, $\{7,8\}(0.001,0.0001)$, $\{8,7\}(0.001,0.0001)$, $\{8,10\}(0.001,0.0001)$, $\{6,5\}(0.001,0.0001)$, $\{6,7\}(0.001,0.0001)$
	Lymph	$\{6,5\}(0.10,2.00)$, $\{6,7\}(20.00,0.40)$, $\{9,8\}(5.00,0.10)$, $\{10,7\}(0.10,0.20)$, $\{9,12\}(0.10,2.00)$, $\{8,6\}(0.10,0.10)$, $\{7,9\}(2.00,5.00)$, $\{7,8\}(0.10,0.10)$, $\{9,6\}(0.50,0.20)$, $\{10,12,8,9\}(0.20,0.01)$
	Pima	$\{6,5\}(5.00,100.00)$, $\{5,6\}(0.10,100.00)$, $\{6,7\}(5.00,0.40)$, $\{5,6\}(0.50,1.00)$, $\{7,6\}(1000,0.40)$, $\{6,5\}(0.20,100.00)$, $\{5,6\}(1.00,10000.00)$, $\{7,5\}(10.00,20.00)$, $\{5,7\}(20.00,0.40)$, $\{6,5\}(100.00,0.10)$
Group B	Pendigits	$\{15,20,10\}$ $(20.00,0.10)$
	Img. Seg.	$\{12,15\}$ $(10.00,1.00)$
	Sat. Img.	$\{7,6\}$ $(100.00,0.40)$
	Optdigits	$\{30,25\}$ $(2.00,0.10)$

Table 6: The combination of SFMs and (C,γ) pair for NEUROSVM selected by cross-validation for each of the twelve data sets

SFMs for MVOTING and AVERAGING are listed in Table 8. Table 9 reports the test error statistics using the combinations shown in Table 8. Based on the paired t-test for Group A data sets and McNemar test for Group B data sets, Table 9 reveals that NEUROSVM is significantly better than

	Data set	NEUROSVM	Baseline classifiers		Standard ensemble methods	
			MLP	SVM	MVOTING	AVERAGING
Gr. A	Iris	0.013±0.008	**0.040±0.017**	0.033±0.014	**0.040±0.017**	**0.040±0.017**
	Vehicle	0.101±0.013	**0.166±0.016**	**0.190±0.011**	**0.159±0.022**	**0.142±0.013**
	WDBC	0.019±0.009	**0.042±0.007**	**0.070±0.008**	**0.040±0.007**	**0.037±0.008**
	Glass	0.251±0.034	**0.309±0.024**	0.290±0.026	**0.308±0.025**	0.290±0.030
	Sonar	0.067±0.017	**0.168±0.025**	0.138±0.038	**0.148±0.029**	**0.148±0.027**
	Ionosphere	0.011±0.002	**0.074±0.017**	**0.060±0.014**	**0.088±0.016**	**0.080±0.018**
	Lymph	0.094±0.019	**0.168±0.030**	**0.202±0.034**	**0.148±0.029**	**0.168±0.030**
	Pima	0.211±0.013	**0.252±0.014**	**0.249±0.013**	**0.246±0.010**	**0.249±0.012**
Gr. B	Pendigits	0.022	**0.077**	0.016	**0.074**	**0.074**
	Img. Seg.	0.059	**0.075**	**0.075**	**0.074**	**0.075**
	Sat. Img.	0.154	**0.178**	0.158	**0.178**	**0.179**
	Optdigits	0.027	0.033	0.026	0.034	0.032
	Win/Loss		11/0	6/0	11/0	10/0

bold/*Italic* significantly worse/better than NEUROSVM using ten-fold cross-validation paired t-test for Group A data sets and McNemar test for Group B data sets.

Table 7: Performance comparison of NEUROSVM with baseline classifiers and standard ensemble methods

the standard ensemble methods for 8 data sets when compared with majority voting and for 7 data sets when compared with averaging. Here also no data set is found where two ensemble methods perform better than NEUROSVM. Hence we can conclude that NEUROSVM performs consistently better than majority voting and averaging.

3.3.5 PERFORMANCE COMPARISON OF OTHER VARIANTS OF NEUROSVM WITH BASELINE CLASSIFIERS AND STANDARD ENSEMBLE METHODS

As stated earlier, for the proposed architecture, in the classification module we can use other tools also. Here we demonstrate the effect of using MLP and RBF neural networks in the classification module instead of SVM. We termed these two architectures as NMLP and NRBF respectively. The combination of SFMs and the number of hidden nodes for MLP and RBF networks in the classification module are selected using double ten-fold cross-validation for Group A data sets and using ten-fold cross-validation for Group B data sets. The performance comparison of these variants of NEUROSVM with the original NEUROSVM is shown in Table 10. From Table 10, we see that original NEUROSVM is significantly better than other variants for 4 data sets when compared with NMLP and better than 2 data sets when compared with NRBF. The performance of NEUROSVM and NMLP is equally well for 8 data sets. There is no significant difference in performance between NEUROSVM and NRBF for 10 data sets. For six out of twelve data sets, all three variants of the proposed architecture perform equally well. So, proposed architecture can be considered a general one.

Now we compare the performance of baseline classifiers and two ensemble methods with the two new variants of NEUROSVM, that is, NMLP and NRBF, by statistical test. These results are summarized in Table 11 for NMLP and in Table 12 for NRBF. The results of MVOTING and AVERAGING are obtained in Table 11 and Table 12 using the same combinations of SFMs that are

Data set		Selected combinations	
		MVOTING	AVERAGING
Group A	Iris	$\{2,5\},\{3,5\},\{6,5\},\{6,3\},\{7,6\},\{5,6\},\{5,6\},\{5,6\},\{7,6\},\{2,5\}$	$\{2,5\},\{3,5\},\{6,5\},\{6,8\},\{7,6\},\{5,6\},\{5,6\},\{5,6\},\{7,6\},\{2,5\}$
	Vehicle	$\{9,13,10,5\},\{13,7,8\},\{10,13,9\},\{13,12\},\{13,7\},\{12,14,8\},\{9,13,12\},\{11,9\},\{11,6,13\},\{14,10\}$	$\{11,10\},\{13,9\},\{12,13,9\},\{10,5\},\{13,7\},\{12,14,8\},\{9,8,13\},\{13,11\},\{11,6,10\},\{11,14\}$
	WDBC	$\{10,7\},\{6,8,9\},\{8,10\},\{12,7,15\},\{8,9\},\{8,7\},\{15,7\},\{8,10\},\{6,9\},\{9,7\}$	$\{10,9\},\{6,7\},\{15,12\},\{12,8\},\{8,9\},\{8,7\},\{15,7\},\{7,8\},\{6,9\},\{9,7\}$
	Glass	$\{15,4\},\{12,13\},\{12,7,14\},\{12,10\},\{15,6\},\{10,8\},\{14,8\},\{11,10\},\{12,8\},\{11,12\}$	$\{15,9\},\{13,7\},\{7,14\},\{15,10\},\{10,6\},\{12,9\},\{4,8\},\{11,13\},\{8,9\},\{11,12\}$
	Sonar	$\{25,12\},\{20,5\},\{7,10\},\{5,7\},\{20,15\},\{25,7\},\{15,10\},\{7,10\},\{7,12,15\},\{15,7\}$	$\{25,12\},\{20,5\},\{7,10\},\{5,7\},\{20,15\},\{25,7\},\{15,10\},\{7,10\},\{7,12\},\{10,7\}$
	Ionosphere	$\{7,10\},\{9,8\},\{8,12\},\{9,15\},\{8,10\},\{7,8\},\{8,7\},\{8,20\},\{6,5\},\{12,20\}$	$\{7,10\},\{9,8\},\{9,12\},\{15,8\},\{8,12\},\{7,8\},\{8,7\},\{8,15\},\{6,5\},\{15,12\}$
	Lymph	$\{6,5\},\{6,7\},\{9,8\},\{10,7\},\{9,12\},\{8,6\},\{10,8,9\},\{7,8\},\{9,6\},\{8,9\}$	$\{9,5\},\{6,7\},\{9,8\},\{10,7\},\{6,20\},\{10,6\},\{10,9\},\{7,8\},\{8,6\},\{8,9\}$
	Pima	$\{8,9\},\{9,8\},\{6,7\},\{6,8\},\{10,8\},\{7,8\},\{7,5\},\{8,10\},\{8,10\},\{6,7\}$	$\{9,5\},\{9,6\},\{6,10\},\{5,8\},\{6,8\},\{5,8\},\{6,8\},\{8,9\},\{5,8\},\{6,7\}$
Group B	Pendigits	$\{15,18,12\}$	$\{15,18,20\}$
	Img. Seg.	$\{8,15\}$	$\{7,8,10,15\}$
	Sat. Img.	$\{9,7,10,6\}$	$\{8,9,7,10,6\}$
	Optdigits	$\{35,40,20\}$	$\{35,40\}$

Table 8: The combination of SFMs for MVOTING and AVERAGING selected by cross-validation for each of the twelve data sets

used in the FM of NMLP and NRBF respectively. From Table 11 we can see that NMLP performs significantly better than baseline classifiers for 8 data sets when compared with MLP and better than 5 data sets when compared with SVM. The NMLP performs significantly better than MVOTING on 8 data sets. It also performs significantly better than AVERAGING for 7 data sets. From Table 12, it is observed that NRBF performs significantly better than MLP on 8 data sets and it is better than SVM for 4 data sets. The SVM performs significantly better than NRBF *only* with one data set. When compared with the standard ensemble methods the NRBF is found to perform significantly better for majority of the data sets. For example, NRBF performs significantly better than majority voting for 8 data sets and better than averaging for 7 data sets.

Now we compare the results of NMLP and NRBF with the results of MVOTING and AVERAGING in Table 9 by statistical test. We find that the NMLP is significantly better than the standard ensemble methods for 3 data sets (Pima, Pendigits and Sat. Img.) when compared with majority voting and for 2 data sets (Pima and Pendigits) when compared with averaging. The NRBF performs

			Standard ensemble methods	
	Data set	NEUROSVM	MVOTING	AVERAGING
	Iris	0.013±0.008	0.033±0.017	0.033±0.017
	Vehicle	0.101±0.013	**0.123±0.014**	**0.125±0.016**
	WDBC	0.019±0.009	**0.033±0.008**	**0.032±0.009**
Group A	Glass	0.251±0.034	0.279±0.031	0.274±0.030
	Sonar	0.067±0.017	**0.124±0.024**	**0.134±0.027**
	Ionosphere	0.011±0.002	**0.074±0.016**	**0.065±0.017**
	Lymph	0.094±0.019	**0.134±0.027**	**0.141±0.022**
	Pima	0.211±0.013	0.228±0.009	0.229±0.011
	Pendigits	0.022	**0.074**	**0.072**
Group B	Img. Seg.	0.059	**0.071**	**0.074**
	Sat. Img.	0.154	**0.172**	0.164
	Optdigits	0.027	0.032	0.032
	Win/Loss		8/0	7/0

bold/*Italic* significantly worse/better than NEUROSVM using ten-fold cross-validation paired t-test for Group A data sets and McNemar test for Group B data sets.

Table 9: Performance comparison of NEUROSVM, MVOTING and AVERAGING for selected combinations with corresponding algorithm

	Data set	NEUROSVM	NMLP	NRBF
	Iris	0.013±0.008	0.027±0.014	0.013±0.013
	Vehicle	0.101±0.013	0.116±0.013	**0.114±0.014**
	WDBC	0.019±0.009	**0.030±0.008**	0.019±0.009
Group A	Glass	0.251±0.034	0.255±0.026	0.253±0.024
	Sonar	0.067±0.017	**0.119±0.029**	0.057±0.015
	Ionosphere	0.011±0.002	**0.062±0.016**	0.045±0.013
	Lymph	0.094±0.019	**0.135±0.025**	0.101±0.020
	Pima	0.211±0.013	0.199±0.009	0.194±0.012
	Pendigits	0.022	0.023	0.032
Group B	Img. Seg.	0.059	0.063	0.067
	Sat. Img.	0.154	0.159	**0.171**
	Optdigits	0.027	0.029	0.029
	Win/Loss		4/0	2/0

bold/*Italic* significantly worse/better than NEUROSVM using ten-fold cross-validation paired t-test for Group A data sets and McNemar test for Group B data sets.

Table 10: Performance comparison of three variants of proposed algorithm

significantly better than both of MVOTING and AVERAGING on 5 data sets. These 5 data sets are WDBC, Sonar, Lymph, Pima and Pendigits. It is worth noticing here that for no data set the proposed methods are worse than standard ensemble methods. So, the proposed method consistently works better than baseline classifiers and standard ensemble methods.

			Baseline classifiers		Standard ensemble methods	
Data set		NMLP	MLP	SVM	MVOTING	AVERAGING
	Iris	0.027±0.014	0.040±0.017	0.033±0.014	0.040±0.017	0.040±0.017
	Vehicle	0.116±0.013	**0.166±0.016**	**0.190±0.011**	**0.153±0.013**	**0.151±0.015**
	WDBC	0.030±0.008	**0.042±0.007**	**0.070±0.008**	**0.044±0.008**	0.035±0.008
Gr. A	Glass	0.255±0.026	**0.309±0.024**	0.290±0.026	**0.294±0.030**	**0.294±0.030**
	Sonar	0.119±0.029	**0.168±0.025**	0.138±0.038	0.144±0.021	**0.139±0.027**
	Ionosphere	0.062±0.016	0.074±0.017	0.060±0.014	**0.093±0.019**	0.068±0.016
	Lymph	0.135±0.025	0.168±0.030	**0.202±0.034**	0.155±0.026	0.161±0.028
	Pima	0.199±0.009	**0.252±0.014**	**0.249±0.013**	**0.245±0.010**	**0.238±0.011**
	Pendigits	0.023	**0.077**	0.016	**0.074**	**0.072**
Gr. B	Img. Seg.	0.063	**0.075**	**0.075**	**0.083**	**0.078**
	Sat. Img.	0.159	**0.178**	0.158	**0.176**	**0.168**
	Optdigits	0.029	0.033	0.026	0.033	0.036
	Win/Loss		8/0	5/0	8/0	7/0

bold/*Italic* significantly worse/better than NMLP using ten-fold cross-validation paired t-test for Group A data sets and McNemar test for Group B data sets.

Table 11: Performance comparison of NMLP with baseline classifiers and standard ensemble methods

			Baseline classifiers		Standard ensemble methods	
Data set		NRBF	MLP	SVM	MVOTING	AVERAGING
	Iris	0.013±0.013	0.040±0.017	0.033±0.014	0.040±0.017	0.040±0.017
	Vehicle	0.114±0.014	**0.166±0.016**	**0.190±0.011**	**0.148±0.018**	**0.137±0.013**
	WDBC	0.019±0.009	**0.042±0.007**	**0.070±0.008**	**0.042±0.007**	**0.040±0.007**
Gr. A	Glass	0.253±0.024	**0.309±0.024**	0.290±0.026	**0.309±0.029**	0.295±0.040
	Sonar	0.057±0.015	**0.168±0.025**	0.138±0.038	**0.153±0.027**	**0.153±0.030**
	Ionosphere	0.045±0.013	**0.074±0.017**	0.060±0.014	**0.091±0.017**	**0.077±0.018**
	Lymph	0.101±0.020	**0.168±0.030**	**0.202±0.034**	**0.161±0.031**	**0.189±0.031**
	Pima	0.194±0.012	**0.252±0.014**	**0.249±0.013**	**0.252±0.014**	**0.247±0.011**
	Pendigits	0.032	**0.077**	*0.016*	**0.074**	**0.075**
Gr. B	Img. Seg.	0.067	0.075	0.075	0.074	0.075
	Sat. Img.	0.171	0.178	0.158	0.172	0.167
	Optdigits	0.029	0.033	0.026	0.033	0.033
	Win/Loss		8/0	4/1	8/0	7/0

bold/*Italic* significantly worse/better than NRBF using ten-fold cross-validation paired t-test for Group A data sets and McNemar test for Group B data sets.

Table 12: Performance comparison of NRBF with baseline classifiers and standard ensemble methods

3.4 Controlled Experiments - Avoiding Full Training of Networks for Feature Extraction

We have mentioned in Section 2.6 that for large data sets it may not be necessary to make full training of the MLPs for constructing the SFMs. Now we are going to prove it by experiments. We consider two data sets, one from each group for this experiment. More specifically, we use the Sonar data (in 60 dimension) from Group A and Optdigits (in 64 dimension) from Group B.

We have conducted the experiments as before for NEUROSVM except we stop the training of MLPs to construct SFMs only after 100 iterations. In this case, we have obtained the test error of 0.135 ± 0.087 for Sonar and 0.023 for Optdigits data sets. By statistical test we observe that these errors are not significantly different from the previous NEUROSVM errors when the MLPs were fully trained.

4. The Kernel Independence of NEUROSVM

In this section we shall illustrate an attractive feature of NEUROSVM, its kernel independence. In order to perform such study we choose three kernels for SVM: linear, RBF and polynomial. We choose these kernels also for SVMs in the classification module of NEUROSVM. The different kernels are tried with a set of parameters. We perform ten-fold (double ten-fold for Group A data sets) cross-validation to select the best parameter set for each kernel of SVM. We also conducted cross-validation experiments to select the best combination of SFMs and hyper parameters of NEUROSVM for each of three choices of kernel. For the RBF kernel we choose 12 different C and 15 different γ resulting 180 pairs. Similarly, for the polynomial kernel we choose 12 different C, 5 different degrees d and 7 different scaling coefficients of dot products s resulting 420 triplets and 12 different C are used in linear kernel. The values of C and γ are presented in Section 3.3.2. The five values of d for the polynomial kernel are $2, 3, 4, 5$ and 6. The seven different choices of s are 0.001, 0.01, 0.10, 1.00, 10.00, 100.00 and 1000.00.

In Figure 4, the test errors of SVM and NEUROSVM with three choices of kernel for the twelve data sets are shown. It is clear from Figures 4(b)-4(h) that the performance of SVM significantly depends on the choice of kernels for Vehicle, WDBC, Glass, Sonar, Ionosphere, Lymph and Pima data sets respectively. Also the kernel dependency of SVM is noticeable for the data sets Iris, Pendigits, Img. Seg. and Optdigits (Figure 4(a), 4(i), 4(j) and 4(l)). Only for the Sat. Img. the SVM produces almost the same test errors for all three choices of kernel. On the other hand, from Figures 4(a)-4(l) we see that the performance of NEUROSVM for eight (out of the twelve) data sets practically does not depend on the choice of kernels. To observe it more closely for each data set we find out the difference of percentage errors between the maximum and minimum errors produce by the three kernels for SVM and NEUROSVM (Table 13). To explain the entries in Table 13, consider the WDBC data set. The test errors produced by SVM on WDBC data set with the three kernels are 0.049, 0.070 and 0.095 respectively. Hence the minimum and maximum errors are 0.049 and 0.095 respectively. So, the difference in error rates and hence the percentage are 0.046 and 4.60% respectively. Whereas the test error rates for NEUROSVM on WDBC data set with the three kernels are $0.023, 0.019$ and 0.023 respectively. Here the minimum, maximum and percentage of difference of these two errors are $0.019, 0.023$ and 0.40% respectively. From Table 13, it is clear that for eight data sets the performance of NEUROSVM using the three kernels remains almost the same (with error less than 1%). On the other hand, with SVM only for Sat. Img. the difference is less than 1%. Thus NEUROSVM is found to perform equally well with different choices of kernels of the SVM in the classification module.

5. Conclusions

We have proposed a multilayer classifier architecture consisting of two modules. The first module is the feature extraction module (FM), while the second module is the classification module (CM). In

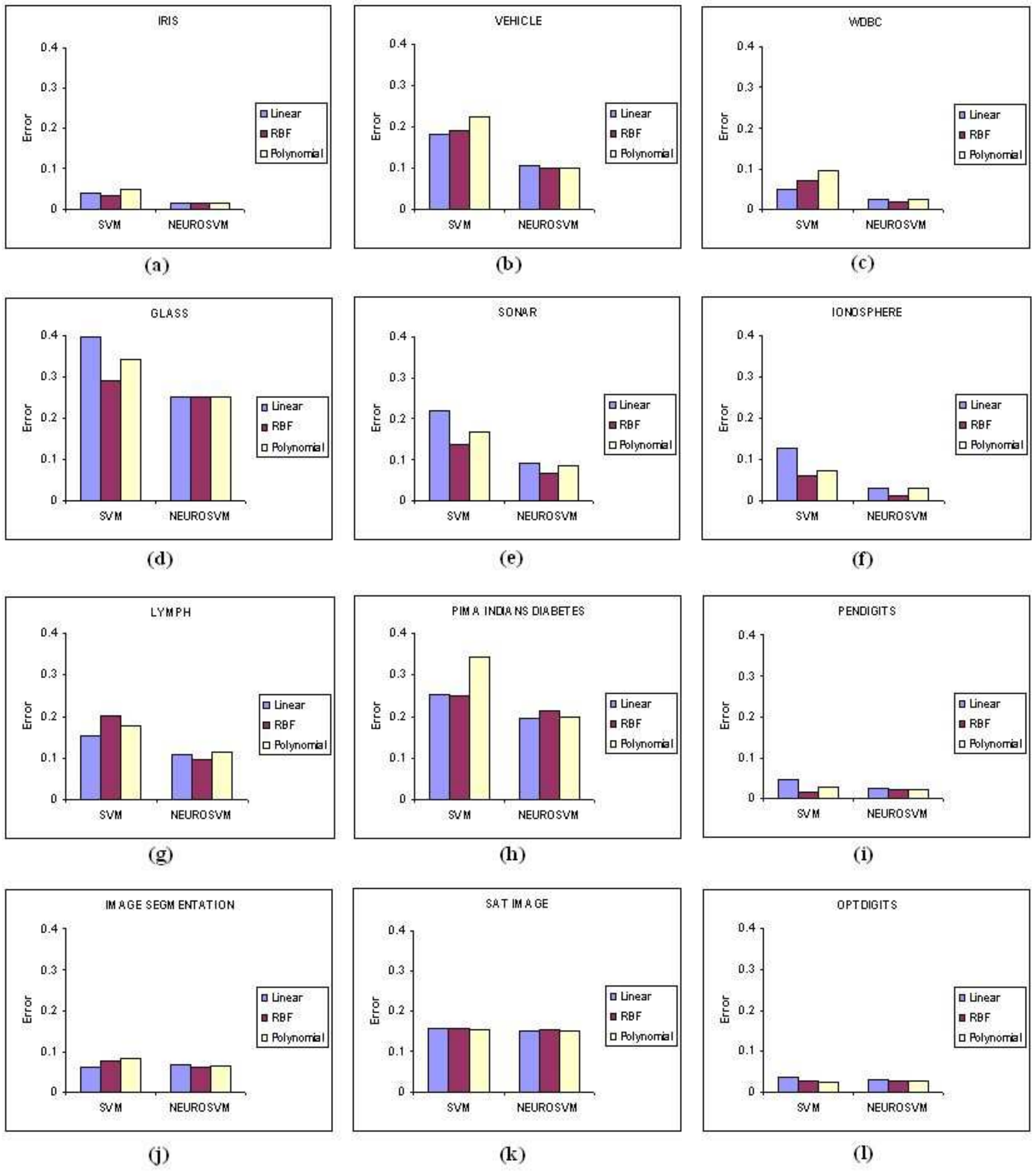


Figure 4: Comparison of the test errors of SVM and NEUROSVM for twelve data sets using Linear, RBF, and Polynomial kernels.

	Differences of the percentage error	
Data set	SVM	NEUROSVM
Iris	1.40%	0.00%
Vehicle	4.10%	0.60%
WDBC	4.60%	0.40%
Glass	10.80%	0.00%
Sonar	8.40%	2.30%
Ionosphere	6.50%	2.10%
Lymph	4.80%	2.00%
Pima	9.30%	1.40%
Pendigits	2.70%	0.10%
Img. Seg.	1.90%	0.90%
Sat. Img.	0.20%	0.20%
Optdigits	1.40%	0.10%

Table 13: Differences of the percentage errors between the maximum and minimum errors produced by linear, RBF and polynomial kernels for SVM and NEUROSVM

the FM, we have used MLP, while for the CM we have used SVM resulting in the classifier, called NEUROSVM. The architecture is general in nature and both for FM and CM other tools can be used. We have experimented using RBF and MLP in the CM. We have tested the performance of the proposed system on twelve benchmark data sets and NEUROSVM is found to perform consistently better than MLP and SVM. The performance of NEUROSVM is also better than the ensemble methods based on majority voting and averaging. A noticeable feature of NEUROSVM is that nonlinear NEUROSVM and linear NEUROSVM perform equally well on all data sets tried.

Other advantages of NEUROSVM are as follows:

- For large data sets, it may not be necessary to make a full training of the MLPs in the FM because in an MLP, the extraction of the salient feature of the data is done at the beginning of the training.

- Typically the number of nodes in the hidden layer of MLPs is much smaller than the number of the input nodes, and one does not need many feature extraction sub-modules. Hence, the dimensionality of the input for the SVMs (or MLP/RBF) in the classification module can be reduced compared to the original dimension of the input. So, for solving bioinformatics problems such as protein secondary structure prediction or protein fold recognition such an architecture may be very useful.

- It may be viewed as an implicit fusion of multiple classifiers and hence the improvement in performance is expected.

We have demonstrated the advantages of the proposed architecture by good experimental results. In our experimental results we have noticed that most of the time out of the 5 SFMs, 2 or 3 are selected for NEUROSVM by the cross-validation method. This limited use of the architectures could be due to the fact that all networks are trained using the same data and some of the networks may be extracting similar information from the data. We are currently working on developing a more theoretical view of our proposed method that may help further explain the results reported here.

Appendix A. Procedure DataPreparation

```
Input:   A data set $X$
Output: Training and test data sets
Algorithm:
        1.  if $X$ belongs to Group A then
        2.    Set $no\_of\_fold = 10$.
        3.    $X$ is randomly partitioned into 10 subsets $X_i; i = 1, 2, \ldots, 10$
              such that, $X = \bigcup_{i=1}^{10} X_i, X_i \cap X_j = \phi, i \neq j$.
        4.    Get the training set for fold $i$ of $X$ as $XT_i = \bigcup_{j \neq i} X_j$ and the test data set is
              $XTe_i = X_i$. So we get 10 training-test set $(XT_i, XTe_i), i = 1, 2, , 10$.
        5.  else /* for Group B data sets */
        6.    Set $no\_of\_fold = 1$.
        7.    Let the training set be $XT_1$ and the test set be $XTe_1$ .
        8.  end if
End DataPreparation
```

NB: For a given data set (in Group A), the Procedure DataPreparation returns the same outer level ten-folds to RunMLP, RunSVM and RunNEUROSVM.

Appendix B. Procedure RunMLP

```
Input:   A data set X.
           A set of hidden nodes $H = \{h_1, h_2, \ldots, h_m\}$.
Output: Test error of MLP on X
Algorithm:
        1.  Perform DataPreparation
        2.  for $i = 1$ to $no\_of\_fold$
          /* To choose the optimal network size for $XT_i$, we use ten-fold cross-validation
          experiment on $XT_i$ */
        3.    $XT_i$ is divided into 10 equal (or almost equal) parts $Z_j; j = 1, 2, \ldots, 10$
              such that, $XT_i = \bigcup_{j=1}^{10} Z_j, Z_j \cap Z_k = \phi, j \neq k$ .
        4.    Get the training set for fold $j$ of $XT_i$ as $ZT_j = \bigcup_{k \neq j} Z_k$ and the validation
              set as $ZV_j = Z_j$. So we get 10 training-validation set $(ZT_j, ZV_j)$,
              $j = 1, 2, \ldots, 10$ for fold $i$ of $X$.
        5.    for each $a$ in $\{h_1, h_2, \ldots, h_m\}$
        6.      for $j = 1$ to 10
        7.          Train a network (MLP) for architecture $a$ with training data set $ZT_j$
                    and find validation error on $ZV_j$. Let the validation error with fold
                    $(ZT_j, ZV_j)$ be $e_j^a$.
```

```
 8.        end for /* end for $j$ */
 9.        The average validation error for an architecture $a$ related to $XT_i$ of the
           original fold $(XT_i, XTe_i)$ is $\bar{e}_i^a = \frac{1}{10}\sum_{j=1}^{10} e_j^a$.
10.      end for /* end of for $a$ */
11.      Let $\bar{e}_i^k = \min_a\{\bar{e}_i^a\}$, then we choose $k$ as the optimal architecture for fold $XT_i$.
12.      Train a network (MLP) for architecture $k$ with training data $XT_i$ and find
         test error on $XTe_i$. Let the test error with fold $(XT_i, XTe_i)$ be $E_i$.
13.    end for /* end of for $i$ */
14.    Find average test error $\bar{E} = \frac{1}{no\_of\_fold}\sum_{i=1}^{no\_of\_fold} E_i$.
```

End RunMLP

Appendix C. Procedure RunSVM

Input: A data set X.
A set of 12 choices of C and 15 choices of γ for RBF kernel resulting in a total of 180 pairs of (C,γ).
Output: Test error of SVM on X
Algorithm:

```
  1.   **Perform DataPreparation**
  2.   for $i = 1$ to $no\_of\_fold$
         /* To choose the best $(C,\gamma)$ pair for $XT_i$, we use ten-fold cross-validation
          experiment on $XT_i$ */
3-4.     Same as steps 3-4 of RunMLP
  5.     for each $(C_k,\gamma_k)$ pair on 180 pairs
  6.       for $j = 1$ to $10$
  7.         Train SVM with parameters $(C_k,\gamma_k)$ of RBF kernel for training
             data $ZT_j$ and find validation error on $ZV_j$.
             Let the validation error with fold $(ZT_j, ZV_j)$ be $e_j^k$.
  8.       end for /* end of for $j$ */
  9.       The average validation error for $(C_k,\gamma_k)$ pair related to $XT_i$ of the
           original fold $(XT_i, XTe_i)$ is $\bar{e}_i^k = \sum_{j=1}^{10} e_j^k$.
 10.     end for /* end of for $(C_k,\gamma_k)$ */
 11.     Let $\bar{e}_i^m = \min_k\{\bar{e}_i^k\}$, then we choose $(C_m,\gamma_m)$ pair as the best hyper
         parameters for fold $XT_i$.
 12.     Train SVM with RBF kernel and $(C_m,\gamma_m)$ pair with training data $XT_i$ and
         find test error on $XTe_i$. Let the test error with fold $(XT_i, XTe_i)$ be $E_i$.
 13.   end for /* end of for $i$ */
 14.   Find average test error $\bar{E} = \frac{1}{no\_of\_fold}\sum_{i=1}^{no\_of\_fold} E_i$.
```

End RunSVM

Appendix D. Procedure RunNEUROSVM

```
Input:  A data set $X$.
          A set of hidden nodes $H = \{h_1, h_2, \ldots, h_m\}$.
          A set of 12 choices of $C$ and 15 choices of $\gamma$ for RBF kernel resulting in a total
          of 180 pairs of $(C,\gamma)$.
Output: Test error of NEUROSVM on $X$
Algorithm:
        1.  **Perform DataPreparation**
        2.  for $i = 1$ to $no\_of\_fold$
              /* To choose 5 MLPs for 5 SFMs for $XT_i$, we use ten-fold cross-validation
              experiment on $XT_i$ */
     3-10.      Same as steps 3-10 of RunMLP
       11.      We need to select 5 MLP architectures for 5 SFMs to construct NEUROSVM.
                The best 5 architectures corresponding to the smallest 5 values of $\bar{e}_i^a$.
                In other word, we select 5 architecture $(a_{i1}, a_{i2}, \ldots, a_{i5})$ where
                $\bar{e}_i^{a_{i1}}, \bar{e}_i^{a_{i2}}, \ldots, \bar{e}_i^{a_{i5}}, \ldots$ is the sequence of $\bar{e}_i^a$'s sorted in ascending order.
       12.      Train 5 networks with above 5 selected architectures for training data $XT_i$.
       13.      Find projected data of $(XT_i, XTe_i)$ from hidden layer of above 5 MLPs
                and hence we get 5 SFMs.
       14.      Construct $2^5 - 1 = 31$ combinations of projected data using 5 SFMs, that is,
                31 sets of training-test data using 5 SFMs. So, we get 31 sets of training-test
                data $(\tilde{Z}T_p^i, \tilde{Z}Te_p^i), p = 1, 2, \ldots, 31$ for NEUROSVM.
                /* To select the best combination among 31 combinations and $(C,\gamma)$ pair
                of RBF kernel of SVM in the classification module we perform ten-fold
                cross-validation experiment */
       15.      for $p = 1$ to $31$
       16.          Perform ten-fold cross-validation as steps 3-10 of RunSVM on $\tilde{Z}T_p^i$.
       17.          Choose $(C,\gamma)$ for combination $p$ with minimum average validation error
                    say $\bar{e}_i^p$.
       18.      end for /* end of for $p$ */
       19.      Finally choose $k^{th}$ combination and corresponding $(C,\gamma)$ pair (say $(C_k,\gamma_k)$)
                where $\bar{e}_i^k = \min_p \{\bar{e}_i^p\}$ .
       20.      Train SVM with training data $\tilde{Z}T_k^i$ and $(C_k,\gamma_k)$ pair of RBF kernel.
                Find test error on $\tilde{Z}Te_k^i$ , say test error is $E_i$.
       21.  end for /* end of for $i$ */
       22.  Find average test error $\bar{E} = \frac{1}{no\_of\_fold} \sum_{i=1}^{no\_of\_fold} E_i$.
**End RunNEUROSVM**
```

References

M. M. Adankon and M. Cheriet. Optimizing resources in model selection for support vector machine. *Pattern Recognition*, 40(3):953–963, 2007.

T. Andersen and T. Martinezr. Cross validation and mlp architecture selection. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN '99)*, volume 3, pages 1614–1619, 1999.

C. L. Blake and C. J. Merz. *UCI Repository of Machine Learning Databases: Univ. of California*. Dept. of Inform. and Comput. Sci, 1998.

B. E. Boser, I. M. Guyon, and V. N. Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the 5th Annual ACM Workshop on Computational Learning Theory*, pages 144–152, 1992.

L. Bottou and P. Gallinari. A framework for the cooperation of learning algorithms. *Advances in Neural Information Processing Systems*, 3:781–788, 1991.

G. Brown, J. L. Wyatt, and P. Tino. Managing diversity in regression ensembles. *Journal of Machine Learning Research*, 6:1621–1650, 2005.

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.

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

T. G. Dietterich. Approximation statistical tests for comparing supervised classification learning algorithms. *Neural Computation*, 10(7):1895–1923, 1998.

N. Garcia-Pedrajas, C. Hervas-Martinez, and D. Ortiz-Boyer. Cooperative coevolution of artificial neural network ensembles for pattern classification. *IEEE Trans. on Evolutionary Computation*, 9(3):271–302, 2005.

N. Garcia-Pedrajas, C. Garcia-Osorio, and C. Fyfe. Nonlinear boosting projections for ensemble construction. *Journal of Machine Learning Research*, 8:1–33, 2007.

K. S. Guimaraes, J. C. B. Melo, and G. D. C. Cavalcanti. Combining few neural networks for effective secondary structure prediction. In *Proceedings of the IEEE Symposium on Bioinformatics and BioEngineering (BIBE'03)*, pages 415–420, 2003.

J. V. Hansen. Combining predictors: comparison of five meta machine learning methods. *Information Sciences*, 119(1-2):91–105, 1999.

B. Happel and J. Murre. Design and evolution of modular neural network architectures. *Neural Networks*, 7(6-7):985–1004, 1994.

S. Haykin. *Neural Networks: A Comprehensive Foundation*. Englewood Cliffs, NJ: Prentice-Hall, 1999.

F. J. Huang and Y. LeCun. Large-scale learning with svm and convolutional for generic object categorization. In *Proceedings of the Computer Vision and Pattern Recognition Conference (CVPR06)*, volume 1, pages 284–291, 2006.

Md. M. Islam, X. Yao, and K. Murase. A constructive algorithm for training cooperative neural network ensembles. *IEEE Trans. on Neural Networks*, 14(4):820–834, 2003.

R. E. Jenkins and B. P. Yuhas. A simplified neural network solution through problem decomposition: The case of the truck backer-upper. *IEEE Trans. on Neural Networks*, 4(4):718–720, 1993.

T. Joachims. *SVM^{light}: Support Vector Machine*. http://svmlight.joachims.org/., 2002.

K. I. Kim, K. Jung, S. H. Park, and H. J. Kim. Support vector machines for texture classification. *IEEE Trans. Pattern Anal. Machine Intell.*, 24(11):1542–1550, 2002.

A. H. R. Ko, R. Sabourin, A. de Souza Britto Jr., and L. Oliveira. Pairwise fusion matrix for combining classifiers. *Pattern Recognition*, 40(8):2198–2210, 2007.

L. I. Kuncheva and C. J. Whitaker. Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy. *Machine Learning*, 51(2):181–207, 2003.

I. Maqsood, M. R. Khan, and A. Abraham. An ensemble of neural networks for weather forecasting. *Neural Computing and Applications*, 13(2):112–122, 2004.

P. Melin, C. Felix, and O. Castillo. Face recognition using modular neural networks and the fuzzy sugeno integral for response integration. *International Journal of Intelligent Systems*, 20(2):275–291, 2005.

V. Mitra, C-J. Wang, and S. Banerjee. A neuro-svm model for text classification using latent semantic indexing. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN '05)*, volume 1, pages 564–569, 2005.

V. Mitra, C-J. Wang, and S. Banerjee. Lidar detection of underwater objects using a neuro-svm-based architecture. *IEEE Trans. on Neural Networks*, 17(3):717–731, 2006.

U. Naftaly, N. Intrator, and D. Horn. Optimal ensemble averaging of neural networks. *Network: Computation in Neural Systems*, 8(3):283–296, 1997.

M. N. Nguyen and J. C. Rajapakse. Multi-class support vector machines for protein secondary structure prediction. *Genome Informatics*, 14:218–227, 2003.

N. R. Pal, S. Pal, J. Das, and K. Majumder. Sofm-mlp: A hybrid neural network for atmospheric temperature prediction. *IEEE Trans. Geoscience and Remote Sensing*, 41(12):2783–2791, 2003.

N. R. Pal, A. Sharma, S. K. Sanadhya, and Karmeshu. On identifying marker genes from gene expression data in a neural framework through online feature analysis. *International Journal of Intelligent Systems*, 21(4):453–467, 2006.

M. Pontil and A. Verri. Support vector machines for 3-d object recognition. *IEEE Trans. Pattern Anal. Machine Intell.*, 20(6):637–646, 1998.

L. Prevost, C. Michel-Sendis, L. Oudot A. Moises, and M. Milgram. Combining model-based and discriminative classifiers: application to handwritten character recognition. In *Proceedings of the Seventh International Conference on Document Analysis and Recognition (ICDAR'03)*, pages 31–35, 2003.

E. Ronco and P. Gawthrop. *Modular Neural Networks: A State of the Art*. Technical Report CSC-95026. Centre for System and Control. Faculty of Mechanical Engineering, University of Glasgow, UK, 1995.

N. Stepenosky, D. Green, J. Kounios, C. M. Clark, and R. Polikar. Majority vote and decision template based ensemble classifiers trained on event related potentials for early diagnosis of alzheimers's disease. In *Proceedings of the IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, pages 901–904, 2006.

E. K. Tang, P. N. Suganthan, and X. Yao. An analysis of diversity measures. *Machine Learning*, 65 (1):247–271, 2006.

V. Vapnik. *The Nature of Statistical Learning Theory*. New York: Springer-Verlag, 1995.

P. Vincent and Y. Bengio. A neural support vector network architecture with adaptive kernels. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN 2000)*, volume 5, pages 187–192, 2000.

U. von Luxburg, O. Bousquet, and B. Scholkopf. A compression approach to support vector model selection. *Journal of Machine Learning Research*, 5:293–323, 2004.

J. Weston and C. Watkins. Support vector machines for multi-class pattern recognition. In *Proceedings of the Seventh European Symposium On Artificial Neural Networks*, pages 219–224, 1999.

T. Windeatt. Accuracy/diversity and ensemble mlp classifier design. *IEEE Trans. on Neural Networks*, 17(5):1194–1211, 2006.

J-X. Wu, Z-H. Zhou, and Z-Q. Chen. Ensemble of ga based selective neural network ensembles. In *Proceedings of the 8th International Conference on Neural Information Processing*, volume 3, pages 1477–1482, 2001.

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.

Journal of Machine Learning Research 10 (2009) 623-656 Submitted 3/08; Revised 11/08; Published 3/09

Scalable Collaborative Filtering Approaches for Large Recommender Systems

Gábor Takács[*] GTAKACS@SZE.HU
Dept. of Mathematics and Computer Science
Széchenyi István University
Egyetem tér 1.
Győr, Hungary

István Pilászy[*] PILA@MIT.BME.HU
Dept. of Measurement and Information Systems
Budapest University of Technology and Economics
Magyar Tudósok krt. 2.
Budapest, Hungary

Bottyán Németh[*] BOTTYAN@TMIT.BME.HU
Domonkos Tikk[*] TIKK@TMIT.BME.HU
Dept. of Telecom. and Media Informatics
Budapest University of Technology and Economics
Magyar Tudósok krt. 2.
Budapest, Hungary

Editors: Paolo Frasconi, Kristian Kersting, Hannu Toivonen and Koji Tsuda

Abstract

The collaborative filtering (CF) using known user ratings of items has proved to be effective for predicting user preferences in item selection. This thriving subfield of machine learning became popular in the late 1990s with the spread of online services that use recommender systems, such as Amazon, Yahoo! Music, and Netflix. CF approaches are usually designed to work on very large data sets. Therefore the scalability of the methods is crucial. In this work, we propose various scalable solutions that are validated against the Netflix Prize data set, currently the largest publicly available collection. First, we propose various matrix factorization (MF) based techniques. Second, a neighbor correction method for MF is outlined, which alloys the global perspective of MF and the localized property of neighbor based approaches efficiently. In the experimentation section, we first report on some implementation issues, and we suggest on how parameter optimization can be performed efficiently for MFs. We then show that the proposed scalable approaches compare favorably with existing ones in terms of prediction accuracy and/or required training time. Finally, we report on some experiments performed on MovieLens and Jester data sets.

Keywords: collaborative filtering, recommender systems, matrix factorization, neighbor based correction, Netflix prize

[*]. All authors also affiliated with Gravity R&D Ltd., 1092 Budapest, Kinizsi u. 11., Hungary; info@gravityrd.com.

1. Introduction

Recommender systems attempt to profile user preferences over items, and model the relation between users and items. The task of recommender systems is to recommend items that fit a user's tastes, in order to help the user in selecting/purchasing items from an overwhelming set of choices. Such systems have great importance in applications such as e-commerce, subscription based services, information filtering, etc. Recommender systems providing personalized suggestions greatly increase the likelihood of a customer making a purchase compared to unpersonalized ones. Personalized recommendations are especially important in markets where the variety of choices is large, the taste of the customer is important, and last but not least the price of the items is modest. Typical areas of such services are mostly related to art (esp. books, movies, music), fashion, food and restaurants, gaming and humor.

With the burgeoning of web based businesses, an increasing number of web based merchant or rental services use recommender systems. Some of the major participants of e-commerce web, like Amazon and Netflix, successfully apply recommender systems to deliver automatically generated personalized recommendation to their customers. The importance of a good recommender system was recognized by Netflix, which led to the announcement of the Netflix Prize (NP) competition to motivate researchers to improve the accuracy of the recommender system of Netflix (see details in Section 5.1.1).

There are two basic strategies that can be applied when generating recommendations. *Content-based approaches* profile users and items by identifying their characteristic features, such us demographic data for user profiling, and product information/descriptions for item profiling. The profiles are used by algorithms to connect user interests and item descriptions when generating recommendations. However, it is usually laborious to collect the necessary information about items, and similarly it is often difficult to motivate users to share their personal data to help create the database for the basis of profiling.

Therefore, the alternative approach, termed *collaborative filtering* (CF), which makes use of only past user activities (for example, transaction history or user satisfaction expressed in ratings), is usually more feasible. CF approaches can be applied to recommender systems independently of the domain. CF algorithms identify relationships between users and items, and make associations using this information to predict user preferences.

In this paper, we focus on the case when users express their opinion of items by means of ratings. In this framework, the user first provides ratings of some items, titles or artifacts, usually on a discrete numerical scale, and the system then recommends other items based on ratings the *virtual community* has already provided. The virtual community was defined by Hill et al. (1995) as "a group of people who share characteristics and interact in essence or effect only." The underlying assumption is that people who had similar tastes in the past may also agree on their tastes in the future.

1.1 Related Work

The first works on the field of CF were published in the early 1990s. Goldberg et al. (1992) presented the Tapestry system that used collaborative filtering to filter mails simultaneously from several mailing lists, based on the opinion of other users on the readings. Resnick et al. (1994) described the GroupLens system that was one of the pioneer applications of the field where users could rate articles on a 1–5 scale after having read them and were then offered suggestions. Breese et al. (1998)

divided the underlying techniques of predicting user preferences into two main groups. *Memory-based* approaches operate on the entire database of ratings collected by the vendor or service supplier. On the other hand, *model-based* approaches use the database to estimate or learn a model and then apply this model for prediction.

Over the last broad decade many CF algorithms have been proposed that approach the problem by different techniques, including similarity/neighborhood based approaches (Resnick et al., 1994; Sarwar et al., 2001), personality diagnosis (Pennock et al., 2000), Bayesian networks (Breese et al., 1998), restricted Boltzmann machines (RBM) (Salakhutdinov et al., 2007), and various matrix factorization techniques (Canny, 2002; Hofmann, 2004; Sarwar et al., 2000; Srebro et al., 2005). Vozalis and Margaritis (2007) presented an MF approach that incorporates demographic information and ratings to enhance plain CF algorithms. Breese et al. (1998) surveyed in detail the major CF approaches of the early years, Rashid et al. (2006) gave a short description of most recent methods, and Adomavicius and Tuzhilin (2005) also investigated in their survey the possible extensions.

The NP competition boosted the interest in CF, and yielded the publication of a number of new methods. We should also mention here the NP related KDD Cup and Workshop (Bennett et al., 2007), which indicated the possible directions of scalable and accurate CF methods. We feature among them the matrix factorization, neighbor based approaches, and their combinations.

Several matrix factorization techniques have been successfully applied to CF, including singular value decomposition (Sarwar et al., 2000), probabilistic latent semantic analysis (Hofmann, 2004), probabilistic matrix factorization (Salakhutdinov and Mnih, 2008), maximum margin matrix factorization (Srebro et al., 2005), expectation maximization for MF (Kurucz et al., 2007), and alternating least squares (Bell and Koren, 2007a).

Simon Funk (Brandyn Webb) published a detailed implementation of a regularized MF with separate feature update.[1] Paterek (2007) introduced a bunch of novel techniques, including MF with biases, applying kernel ridge regression on the residual of MF, linear model for each item, and asymmetric factor models (NSVD1, NSVD2). Kurucz et al. (2007) showed the application of expectation maximization based MF methods for the NP. Bell and Koren (2007a) presented an improved neighborhood based approach, which removes the global effect from the data, and calculates optimal similarity values by solving regression problems. Salakhutdinov et al. (2007) mention without details a momentum based MF method that uses batch learning.

Our methods are different from the above ones in various aspects. Bell and Koren (2007a) use alternate least squares, but we use incremental gradient descent (also known as stochastic gradient descent) method for weight updates. Their method does not use the chronological order of ratings, while we exploit this information in our approaches. They use only positive and ordinary MFs, while we propose the semi-positive version of the MF algorithm. Paterek (2007) applies a different learning scheme following Simon Funk's path (see the detailed comparison in Section 3.1). We point out that this difference makes our training process somewhat faster and does not deteriorate the accuracy. Other differences are that Paterek uses fewer meta-parameters for his MF methods. Salakhutdinov et al. (2007) apply a momentum based MF with batch learning, but we point out that incremental gradient descent is more appropriate for large scale problems.

1. Details at `http://sifter.org/~simon/journal/20061211.html`.

1.2 Main Results and Organization

The aim of this work is to propose accurate and scalable solutions for a class of collaborative filtering problems. Scalability is crucial since CF systems often have to manage millions of users or items. As it was shown by many researchers (see for example Bell et al., 2007b) combination (also termed blending) of various algorithms typically outperforms single methods in terms of accuracy. Therefore, we would like to emphasize here that a particular method can have beneficial impact on the overall accuracy if (1) it yields a very accurate model on its own or (2) it "blends well", that is it improves the accuracy of the combination of several models. The methods proposed in this paper satisfy one or both of these criteria. We classify the methods presented in this paper into 4 categories:

- crucial in improving accuracy or run-time;
- may negligibly improve accuracy, in a way that is more important for competitions than for real life data;
- does not improve accuracy, but is useful for blending with other techniques;
- important for real life data.

This paper is organized as follows. Section 2 defines the CF setting we focus on in this work. Section 3 describes our proposed MF algorithms. In particular, we present

- a regularized MF and its biased version that use an incremental gradient descent weight updating scheme, termed as RISMF and BRISMF, (biased) regularized incremental simultaneous MF;
- a fast (semi-)positive MF version that approximates the features by using nonnegative values for either users or items or both;
- an accurate momentum based MF approach;
- an incremental variant of MF that efficiently handles new users/ratings (this is crucial in a real-life recommender systems);
- a transductive version of MF that makes use of information from test instances (namely the ratings users have given for certain items) to improve prediction accuracy.

We also introduce a special MF version that supports the visualization of user/item features, which can be used to generate explanation for recommendations. Finally, we illustrate that the learning scheme of some MF algorithms can be directly described in the neural network framework.

Section 4 presents a neighbor based correction approach for MF, which alloys the global perspective of MF and the localized property of neighbor based approaches efficiently. We propose here two similarity functions for items.

Section 5 contains the experiments. First, we introduce the Netflix Prize data set against which our proposed algorithms are validated. In addition, we also describe the MovieLens and the Jester data sets, on which we also run some experiments. Next, we mention some important implementation issues: we comment on the efficient storage of very large rating databases, we consider the proper ordering of training examples, and we outline our parameter optimization heuristic. We then report on our comprehensive experiments run on the NP data set. In order to illustrate the applicability of MF methods to other data sets, we also report on tests executed with a selected MF method, BRISMF, on the MovieLens and Jester data sets and compare the results with baseline predictors. For the NP data set, we report on the results of each proposed method and the most accurate combinations of methods. We briefly analyze the accuracy-time complexity trade-off of MF methods. We finally compare our results to other published ones to show that the proposed methods are comparable to the existing one in terms of root mean squared error (RMSE). We also discuss the time complexity of some selected methods, and we point out that it compares favorably to similar type methods published so far. We further mention here that all the presented methods are part of the blended solution of our team Gravity in the NP contest.

2. Problem Definition

We define the problem of *collaborative filtering* (CF) in the following setting. The problem can be modeled by the random triplet (U,I,R), where

- U taking values from $\{1,\dots,N\}$ is the *user identifier* (N is the number of users),
- I taking values from $\{1,\dots,M\}$ is the *item identifier* (M is the number of items), and
- R taking values from $\mathcal{X} \subset \mathbb{R}$ is the rating value. Typical rating values can be binary ($\mathcal{X} = \{0,1\}$), integers from a given range (for example, $\mathcal{X} = \{1,2,3,4,5\}$), or real numbers of a closed interval (for example, $\mathcal{X} = [-10,10]$).

A realization of (U,I,R) denoted by (u,i,r) means that user u rated item i with value r.

The goal is to estimate R from (U,I) such that the root mean squared error of the estimate,

$$\mathrm{RMSE} = \sqrt{\mathbf{E}\{(\hat{R}-R)^2\}}, \tag{1}$$

is minimal, where $\hat{R}$ is the estimate[2] of R. We briefly discuss other possible evaluation metrics in CF in Section 5.

In practice, the distribution of (U,I,R) is not known: we are only given a finite sample, $\mathcal{T}' = \{(u_1,i_1,r_1),(u_2,i_2,r_2),\dots,(u_t,i_t,r_t)\}$, generated by it. The sample $\mathcal{T}'$ can be used for training predictors. We assume *sampling without replacement* in the sense that (user ID, item ID) pairs are unique in the sample, which means that users do not rate items more than once. Let us introduce the notation $\mathcal{T} = \{(u,i) : \exists r : (u,i,r) \in \mathcal{T}'\}$ for the set of (user ID, item ID) pairs. Note that $|\mathcal{T}'| = |\mathcal{T}|$, and typically $|\mathcal{T}| \ll N \cdot M$, because most of the users rate only a small subset of the entire set of items. The sample can be represented as a partially specified matrix denoted by $\mathbf{R} \in \mathbb{R}^{N\times M}$, where the matrix elements are known in positions $(u,i) \in \mathcal{T}$, and unknown in positions $(u,i) \notin \mathcal{T}$. The value of the matrix $\mathbf{R}$ at position $(u,i) \in \mathcal{T}$, denoted by r_{ui}, stores the rating of user u for item i. For clarity, we use the term (u,i)-th rating in general for r_{ui}, and (u,i)-th training example if $r_{ui} : (u,i) \in \mathcal{T}$.

2. In general, superscript "hat" denotes the prediction of the given quantity, so $\hat{x}$ is the prediction of x.

The goal of this CF setup is to create such predictors that aim at minimizing the error (1). In practice, we cannot measure the error because the distribution of (U,I,R) is unknown, but we can estimate the error on a validation set. Let us denote the validation set by $\mathcal{V}' \subset [1,\dots,N] \times [1,\dots,M] \times \mathcal{X}$, assuming sampling without replacement as defined above, and we further assume the uniqueness of (user ID, item ID) pairs across $\mathcal{T}'$ and $\mathcal{V}'$. We define $\mathcal{V} = \{(u,i) : \exists r : (u,i,r) \in \mathcal{V}'\}$. The assumptions ensure that $\mathcal{T} \cap \mathcal{V} = \emptyset$. If both the training set $\mathcal{T}'$ and validation set $\mathcal{V}'$ are generated from the same distribution the estimate of RMSE can be calculated as

$$\widehat{\text{RMSE}} = \sqrt{\frac{1}{|\mathcal{V}|} \sum_{(u,i)\in\mathcal{V}} (\hat{r}_{ui} - r_{ui})^2}.$$

For better readability, from now on we omit the "hat" from the RMSE, recalling that we always calculate the estimate of the error.

When we predict a given rating r_{ui} by $\hat{r}_{ui}$ we refer to the user u as *active user* and to the item i as *active item*. The (u,i) pair of active user and active item is termed *query*.

3. Matrix Factorization

Matrix factorization is one of the most often applied techniques for CF problems. Numerous different MF variants have been already published and were validated against the NP data set as well. We should credit here again Simon Funk, who published the first detailed implementation notes on this problem. He applied a regularized MF with gradient descent learning scheme. His model trained factors one after another, which can be considered as a series of 1 factor MF training processes performed on the residual of the previous one. Other efficient MF variants were published by Bell and Koren (2007a), where they used alternating least squares for weight updates. Biased MF was applied by Paterek (2007), whose technique was also proposed at the same time in our previous work (Takács et al., 2007) as "constant values in matrices".

In this section we give an overview of our MF variants that proved to be effective in tackling the large scale practical problem of NP, and hence may be considered as a useful collection of tools for practitioners. Here we also propose several modifications for already known MF variants, which are effective in improving the accuracy of the generated models. We also give details concerning the time requirement of certain methods and propose efficient implementation solutions.

The idea behind MF techniques is very simple. Suppose we want to approximate the matrix $\mathbf{R}$ as the product of two matrices:

$$\mathbf{R} \approx \mathbf{PQ},$$

where $\mathbf{P}$ is an $N \times K$ and $\mathbf{Q}$ is a $K \times M$ matrix. We call $\mathbf{P}$ the user feature matrix and $\mathbf{Q}$ the item feature matrix, and K is the number of features in the given factorization. If we consider the matrices as linear transformations, the approximation can be interpreted as follows: matrix $\mathbf{Q}$ is a transform from $\mathcal{S}_1 = \mathbb{R}^M$ into $\mathcal{S}_2 = \mathbb{R}^K$, and matrix $\mathbf{P}$ is a transform from $\mathcal{S}_2$ into $\mathcal{S}_3 = \mathbb{R}^N$. Typically, $K \ll N$ and $K \ll M$, therefore the vector space $\mathcal{S}_2$ acts as a bottleneck when predicting $\mathbf{v}_3 \in \mathcal{S}_3$ from $\mathbf{v}_1 \in \mathcal{S}_1$. In other words, the number of parameters to describe $\mathbf{R}$ can be reduced from $|\mathcal{T}|$ to $NK + KM$. Note that $\mathbf{Q}$ and $\mathbf{P}$ typically contain real numbers, even when $\mathbf{R}$ contains only integers.

In the case of the given problem, the unknown ratings of $\mathbf{R}$ cannot be represented by zero. For this case, the approximation task can be defined as follows. Let p_{uk} denote the elements of

$\mathbf{P} \in \mathbb{R}^{N\times K}$, and q_{ki} the elements of $\mathbf{Q} \in \mathbb{R}^{K\times M}$. Further, let $\mathbf{p}_u$ denote a row (vector) of $\mathbf{P}$, and $\mathbf{q}_i$ a column (vector) of $\mathbf{Q}$. Then:

$$\hat{r}_{ui} = \sum_{k=1}^{K} p_{uk} q_{ki} = \mathbf{p}_u \mathbf{q}_i, \tag{2}$$

$$e_{ui} = r_{ui} - \hat{r}_{ui} \quad \text{for } (u,i) \in \mathcal{T},$$

$$e'_{ui} = \frac{1}{2} e_{ui}^2, \tag{3}$$

$$\mathrm{SSE} = \sum_{(u,i)\in\mathcal{T}} e_{ui}^2 = \sum_{(u,i)\in\mathcal{T}} \left(r_{ui} - \sum_{k=1}^{K} p_{uk} q_{ki} \right)^2,$$

$$\mathrm{SSE}' = \frac{1}{2}\mathrm{SSE} = \sum_{(u,i)\in\mathcal{T}} e'_{ui},$$

$$\mathrm{RMSE} = \sqrt{\mathrm{SSE}/|\mathcal{T}|},$$

$$(\mathbf{P}^*, \mathbf{Q}^*) = \arg\min_{(\mathbf{P},\mathbf{Q})} \mathrm{SSE}' = \arg\min_{(\mathbf{P},\mathbf{Q})} \mathrm{SSE} = \arg\min_{(\mathbf{P},\mathbf{Q})} \mathrm{RMSE}. \tag{4}$$

Here $\hat{r}_{ui}$ denotes how the u-th user would rate the i-th item, according to the model, e_{ui} denotes the training error measured at the (u,i)-th rating, and SSE denotes the sum of squared training errors. Eq. (4) states that the optimal $\mathbf{P}$ and $\mathbf{Q}$ minimize the sum of squared errors only on the known elements of $\mathbf{R}$.

In order to minimize RMSE, which is in this case equivalent to minimizing SSE$'$, we apply a simple incremental gradient descent method to find a local minimum of SSE$'$, where one gradient step intends to decrease the square of prediction error of only one rating, or equivalently, either e'_{ui} or e_{ui}^2.

Minimizing RMSE can be seen as a weighted low-rank approximation of $\mathbf{R}$. Weighted low-rank approximations try to minimize the objective function $\mathrm{SSE_W} = \sum_{u=1}^{N}\sum_{i=1}^{M} w_{ui} \cdot e_{ui}^2$, where w_{ui}-s are predefined non-negative weights. For collaborative filtering problems, w_{ui} is 1 for known ratings, and 0 for unknown ratings. Srebro and Jaakkola (2003) showed that when the rank of $((w_{ui}))$ is 1, all local minima are global minima. However, when it is greater than 1—as in the case of collaborative filtering—this statement does not hold any more, which was shown by the authors via counterexamples.

For the incremental gradient descent method, suppose we are at the (u,i)-th training example, r_{ui}, and its approximation $\hat{r}_{ui}$ is given.

We compute the gradient of e'_{ui}:

$$\frac{\partial}{\partial p_{uk}} e'_{ui} = -e_{ui} \cdot q_{ki}, \qquad \frac{\partial}{\partial q_{ki}} e'_{ui} = -e_{ui} \cdot p_{uk}.$$

We update the weights in the direction opposite to the gradient:

$$p'_{uk} = p_{uk} + \eta \cdot e_{ui} \cdot q_{ki},$$
$$q'_{ki} = q_{ki} + \eta \cdot e_{ui} \cdot p_{uk}.$$

That is, we change the weights in $\mathbf{P}$ and $\mathbf{Q}$ to decrease the square of actual error, thus better approximating r_{ui}. Here η is the learning rate. This basic MF method is referred to as ISMF, that is incremental simultaneous MF, due to its distinctive incremental and simultaneous weight updating method to other MF methods.

When the training has been finished, each value of $\mathbf{R}$ can be computed easily using Eq. (2), even at unknown positions. In other words, the model ($\mathbf{P}^*$ and $\mathbf{Q}^*$) provides a description of how an arbitrary user would rate any item.

3.1 RISMF

The matrix factorization presented in the previous section can overfit for users with few (no more than K) ratings: assuming that the feature vectors of the items rated by the user are linearly independent and $\mathbf{Q}$ does not change, there exists a user feature vector with zero training error. Thus, there is a potential for overfitting, if η and the extent of the change in $\mathbf{Q}$ are both small. A common way to avoid overfitting is to apply regularization by penalizing the square of the Euclidean norm of weights. This is often used in machine learning methods, for example Support Vector Machines and ridge regression apply that. It is common also in neural networks, where it is termed as weight decay (Duda et al.). Penalizing the weights results in a new optimization problem:

$$\begin{aligned} e'_{ui} &= (e_{ui}^2 + \lambda \cdot \mathbf{p}_u \cdot \mathbf{p}_u^T + \lambda \cdot \mathbf{q}_i^T \cdot \mathbf{q}_i)/2, \\ \mathrm{SSE}' &= \sum_{(u,i)\in\mathcal{T}} e'_{ui}, \\ (\mathbf{P}^*, \mathbf{Q}^*) &= \operatorname*{arg\,min}_{(\mathbf{P},\mathbf{Q})} \mathrm{SSE}'. \end{aligned} \tag{5}$$

Here $\lambda \geq 0$ is the regularization factor. Note that minimizing SSE' is no longer equivalent to minimizing SSE, unless $\lambda = 0$, in which case we get back the ISMF. We call this MF variant RISMF that stands for regularized incremental simultaneous MF.

Similar to the ISMF approach, we compute the gradient of e'_{ui}:

$$\frac{\partial}{\partial p_{uk}} e'_{ui} = -e_{ui} \cdot q_{ki} + \lambda \cdot p_{uk}, \qquad \frac{\partial}{\partial q_{ki}} e'_{ui} = -e_{ui} \cdot p_{uk} + \lambda \cdot q_{ki}. \tag{6}$$

We update the weights in the direction opposite to the gradient:

$$\begin{aligned} p'_{uk} &= p_{uk} + \eta \cdot (e_{ui} \cdot q_{ki} - \lambda \cdot p_{uk}), \\ q'_{ki} &= q_{ki} + \eta \cdot (e_{ui} \cdot p_{uk} - \lambda \cdot q_{ki}). \end{aligned} \tag{7}$$

For the training algorithm used in RISMF, see Algorithm 1. Note that we use early stopping in Algorithm 1, thus $\mathbf{P}^*$ and $\mathbf{Q}^*$ differs from Eq. (5), because we optimize for the validation set. Note that the matrices are initialized randomly. If both $\mathbf{P}$ and $\mathbf{Q}$ are initialized with a constant value, that is, both are rank 1, the weight update will not increase the rank, which is equivalent to the $K = 1$ case. Random initialization is a simple way to avoid this. Typically, we uniformly choose random values from $[-0.01, 0.01]$ or $[-0.02, 0.02]$.

We point out that RISMF differs from Funk's MF in a few important aspects. We update each feature simultaneously and initialize the matrix randomly. Simon Funk's approach learns each feature separately during a certain number of epochs, but there are no specification as to when the learning procedure has to step to the next feature. His approach converges slower than ours, because it iterates over $\mathbf{R}$ more times. Both methods use regularization and early stopping to prevent overfitting.

```
   Input: $\mathcal{T}'$: training set, $\eta$: learning rate, $\lambda$: regularization factor
   Output: $\mathbf{P}^*$,$\mathbf{Q}^*$: the user and item feature matrices
1  Partition $\mathcal{T}'$ into two sets: $\mathcal{T}'_{\mathrm{I}}$, $\mathcal{T}'_{\mathrm{II}}$ (validation set)
2  Initialize $\mathbf{P}$ and $\mathbf{Q}$ with small random numbers.
3  loop until the terminal condition is met. One epoch:
4      iterate over each $(u,i,r_{ui})$ element of $\mathcal{T}'_{\mathrm{I}}$:
5          compute $e'_{ui}$;
6          compute the gradient of $e'_{ui}$, according to Eq. (6);
7          for each $k$
8              update $\mathbf{p}_u$, the $u$-th row of $\mathbf{P}$,
9              and $\mathbf{q}_i$, the $i$-th column of $\mathbf{Q}$ according to Eq. (7);
10     calculate the RMSE on $\mathcal{T}'_{\mathrm{II}}$;
11     if the RMSE on $\mathcal{T}'_{\mathrm{II}}$ was better than in any previous epoch:
12         Let $\mathbf{P}^* = \mathbf{P}$ and $\mathbf{Q}^* = \mathbf{Q}$.
13     terminal condition: RMSE on $\mathcal{T}'_{\mathrm{II}}$ does not decrease during two epochs.
14 end
```

Algorithm 1: Training algorithm for RISMF

We observed (Takács et al., 2007), that the learning curve of epochs (RMSE on the validation set as a function of the number of epochs) is always convex, regardless of the value of λ, that is why we use not only regularization but also early stopping.

3.2 BRISMF: Constant Values in Matrices

One may argue that some users tend to rate all items higher or lower than the average. The same may hold for items: some items can be very popular. Although MF can reconstruct the original matrix exactly when K is large enough and $\lambda = 0$, this is not the case when overfitting is to be avoided. There is a straightforward way to extend RISMF to be able to directly model this phenomenon, by extending MF with biases for users and items.

The bias feature idea was mentioned by Paterek (2007). He termed his version RSVD2, which appeared at the same time in our work in a generalized form by incorporating constant values in the MF (Takács et al., 2007). Paterek's and our variants share some common features, but he used Simon Funk's approach to update feature weights.

We incorporate bias features into RISMF by fixing the first column of **P** and the second row of **Q** to the constant value of 1. By "fixing to a constant value" we mean initialize $\mathbf{p}_{\bullet 1}$ and $\mathbf{q}_{2\bullet}$ with a fixed constant value instead of random values and drop the application of (7) when updating $\mathbf{p}_{\bullet 1}$ and $\mathbf{q}_{2\bullet}$. In this way, we get back exactly Paterek's RSVD2, except that we update features simultanously. The pair of these features ($\mathbf{q}_{1\bullet}$ and $\mathbf{p}_{\bullet 2}$) can serve as bias features. Our $\mathbf{p}_{u2}$ and $\mathbf{q}_{1i}$ corresponds to Paterek's c_i and d_j resp.

We refer to this method as BRISMF that stands for biased regularized incremental simultaneous MF. This simple extension speeds up the training phase and yields a more accurate model with better generalization performance. Since BRISMF is always superior to RISMF in terms of both the accuracy and the running time, it is our recommended basic MF method.

3.3 Semipositive and Positive MF

The RISMF algorithm can generate not only positive but also negative feature values. Nonnegative matrix factorization (Lee and Seung, 1999) generates models with only nonnegative features, which enables additive part-based representation of the data. Positive (Bell and Koren, 2007a) and semipositive (Hofmann, 2004) matrix factorization techniques have been successfully applied in the field of CF. We present a simple modification of RISMF that can give positive and semipositive factorizations. Although, these modifications do not yield more accurate models, they are important, because the models are still accurate (Section 5.4.6), they blend well with other methods (Section 5.4.8), and the running time (Section 5.4.10), simplicity and accuracy compares favorably with Bell and Koren's positive MF method.

We talk about semipositive MF when exactly one of **P** and **Q** contains both nonnegative and negative values, and positive MF, when both contains only nonnegative values.

We apply a simple thresholding method to ensure the nonnegativeness of features: for the (u,i)-th training example in a given epoch, if p_{uk} or q_{ki} would become negative when applying (7), we reset their value to 0. We describe the modified equations for the case when both user and item features are required to be nonnegative:

$$\begin{aligned} p'_{uk} &= \max\{0, p_{uk} + \eta \cdot e_{ui} \cdot q_{ki} - \lambda \cdot p_{uk}\}, \\ q'_{ki} &= \max\{0, q_{ki} + \eta \cdot e_{ui} \cdot p_{uk} - \lambda \cdot q_{ki}\}. \end{aligned} \tag{8}$$

If we allow, for instance, user features to be negative, we can simply use Eq. (7) instead of Eq. (8) for p'_{uk}. Allowing only nonnegative item features can be treated similarly.

3.4 Applying Momentum Method

This method modifies the learning rule of RISMF slightly. In each learning step the weight updates are calculated from the actual gradient and from the last weight changes. With the modification of (7) we get the following equations:

$$\begin{aligned} p_{uk}(t+1) &= p_{uk}(t) + \Delta p_{uk}(t+1), \\ \Delta p_{uk}(t+1) &= \eta \cdot (e_{ui} \cdot q_{ki}(t) - \lambda \cdot p_{uk}(t)) + \sigma \cdot \Delta p_{uk}(t), \\ q_{ki}(t+1) &= q_{ki}(t) + \Delta q_{ki}(t+1), \\ \Delta q_{ki}(t+1) &= \eta \cdot (e_{ui} \cdot p_{uk}(t) - \lambda \cdot q_{ki}(t)) + \sigma \cdot \Delta q_{ki}(t). \end{aligned}$$

Here σ is the momentum factor and $p_{uk}(t+1)$ and $p_{uk}(t)$ stands for the new and old k-th feature values of user u, respectively. Analogously, $\Delta p_{uk}(t+1)$ and $\Delta p_{uk}(t)$ denote the current and last change of the given feature value. The notations are similar for the item features.

Without a detailed description, the momentum method was mentioned by Salakhutdinov et al. (2007), but they used batch learning, which makes the training slower. In our experiments, momentum MF does not yield more accurate models; however, it blends well with other methods (Section 5.4.8).

3.5 Retraining User Features

The incremental gradient descent weight update of RISMF has a serious drawback: item features change while we iterate through users. If the change is significant, user features updated in the beginning of an epoch may be inappropriate at the end of the epoch. We found two ways to solve this.

1. We can evaluate the test ratings of a user immediately after iterating through its ratings in the training set, and before starting to iterate through the next user's ratings.
2. We can completely recompute the user features after the learning procedure. This method can serve as an efficient incremental training method for recommender systems. Algorithm 2 summarizes the method.

```
  Input: $\mathcal{T}'$: training set, $\eta$: learning rate, $\lambda$: regularization factor
  Output: $\mathbf{P}^*$,$\mathbf{Q}^*$: the user and item feature matrices
1 Partition $\mathcal{T}'$ into two sets: $\mathcal{T}'_{\mathrm{I}}$, $\mathcal{T}'_{\mathrm{II}}$ (tuning set)
2 First training step: call Algorithm 1, store the result in $(\mathbf{P}_1, \mathbf{Q}_1)$
3 Let $\mathbf{Q} = \mathbf{Q}_1$, initialize $\mathbf{P}$ randomly.
4 Second training step: call Algorithm 1, with the following restrictions:
5     skip the weight initialization step, use $\mathbf{P}$ and $\mathbf{Q}$ from here.
6     do not change weights in $\mathbf{Q}$, that is, do not apply (7) or its variants for $q_{ki}$.
7     store the optimal number of epochs, denote it $n^*$.
8 Return the result of Algorithm 1 called in line 4
9 end
```

Algorithm 2: Algorithm for retraining user features

Note that this method can efficiently incorporate into the model new users or new ratings of existing users without the necessity of retraining the entire model, which is very important for recommender systems. Then we do not apply the whole training procedure, just reset the user feature weights of the active user u and apply the second training procedure for n^* epochs for u. The second training procedure needs to iterate through the entire database—which requires slow (but sequential) disk access operation—only once (not n^* times), as the ratings of user u can be kept in memory and can be immediately re-used in the next epoch.

We remark that the presented algorithm cannot handle the addition of new items, and after the addition of many new ratings to the database, $\mathbf{Q}$ will be obsolete, thus the first training step should be re-run. Retraining user features mostly yields a slightly more accurate model (Section 5.4.7), which means that it is useful for real life recommender systems to handle the addition of new users or ratings.

3.6 Transductive MF

Transductive learning involves the use of validation examples but not their labels. In the case of CF, this means that the algorithm "knows" what validation examples the model will be applied for, the $(u,i) \in \mathcal{V}$ pairs, but not the corresponding r_{ui} values. The first transductive model for CF was the conditional restricted Boltzmann machine (Salakhutdinov et al., 2007). In this RBM variant, the distribution over the visible and hidden units is defined conditional on which items the user has rated. The authors noted that the model performance is significantly improved by using conditional distribution instead of unconditional one.

We give a possible practical example when transductive learning model is useful. Let us suppose that, when buying items, users can optionally rate them. Via the proposed technique, the information that a user purchased but did not rate an item can be incorporated in such cases into the prediction model.

We propose a transductive MF that can exploit this information after the learning process. The idea behind the method is the following: suppose that user u (with feature vector $\mathbf{p}_u$) has the following v queries in the validation set: $r_{u1}, \dots, r_{uv}$. When we are at the i-th validation example of user u, we can predict another feature vector based only on what other items $(1 \leq i' \leq v, i' \neq i)$ are to be predicted. A proper linear combination—not depending on u or i—of the original user feature vector and this new feature vector can yield a third feature vector for the prediction of r_{ui} that produces a better predictor than the original one (see Table 7 for the performance gain obtained by transductive MF). Formally:

$$\mathbf{p}'_u(j) = \frac{1}{\sqrt{|i' : (u,i') \in \mathcal{V}| + 1}} \sum_{\substack{i''=1 \\ i'' \neq i}}^{v} \mathbf{q}_{i''}^T. \tag{9}$$

$$\hat{r}'_{ui} = \hat{r}_{ui} + \nu \cdot \mathbf{p}'_u(j) \cdot \mathbf{q}_i = \mathbf{p}_u \cdot \mathbf{q}_i + \nu \cdot \mathbf{p}'_u(j) \cdot \mathbf{q}_i.$$

The attenuation factor in Eq. (9) ensures that the more ratings a user has in the training set, the less the prediction relies on the information the validation set provides, thus $\hat{r}'_{ui}$ will differ less from $\hat{r}_{ui}$.

In practice, ν need not be determined: we can use $\hat{r}'_{ui}$ and $\mathbf{p}'_u(j) \cdot \mathbf{q}_i$ as two predictions for r_{ui}, and apply linear regression to get an improved RMSE. Transductive MF is a post-processing method for MF, which can exploit the information provided by the existence of ratings even if the values of the ratings are unknown. This can be the case in some real life problems, for example, when a user buys many items but rates only a few of them.

3.7 2D Variant of the Matrix Factorization Algorithm

We propose here an MF variant that provides visual information about MF features, which can be used to generate an explanation for recommendations. The idea is that we store user and item features in a two dimensional (2D) grid instead of a one dimensional row vector. Accordingly, we replace the p_{uk} and q_{ki} notation of feature values with p_{ukl} and q_{imn}. Furthermore we define a simple neighborhood relation between the features based on their distance in the grid. If the difference of the horizontal and vertical positions of two features is small, then the "meaning" of those features should also be close. To achieve this, we modify our error function in (3) to penalize the difference between neighbor features:

$$e_{ui}'' = e'_{ui} + \sum_{k,l} \sum_{m \neq k, n \neq l} s(k,l,m,n)(p_{ukl} - p_{umn})^2 + \sum_{k,l} \sum_{m \neq k, n \neq l} s(k,l,m,n)(q_{ikl} - q_{imn})^2.$$

Here (k,l), (m,n) are the indices in the 2D grid, and $s(k,l,m,n)$ is the similarity between the (k,l)-th and (m,n)-th positions of the grid. For example we can use the inverse of the squared Euclidean distance as similarity:

$$s(k,l,m,n) = \frac{\rho}{(k-m)^2 + (l-n)^2}.$$

With this function the gradient used for learning p_{ukl} and q_{ikl} will be:

$$\frac{\partial}{\partial p_{ukl}} e_{ui}'' = \frac{\partial}{\partial p_{ukl}} e'_{ui} + 2 \sum_{m \neq k, n \neq l} s(k,l,m,n)(p_{ukl} - p_{umn}),$$

$$\frac{\partial}{\partial q_{ikl}} e_{ui}'' = \frac{\partial}{\partial q_{ikl}} e'_{ui} + 2 \sum_{m \neq k, n \neq l} s(k,l,m,n)(q_{ikl} - q_{imn}).$$

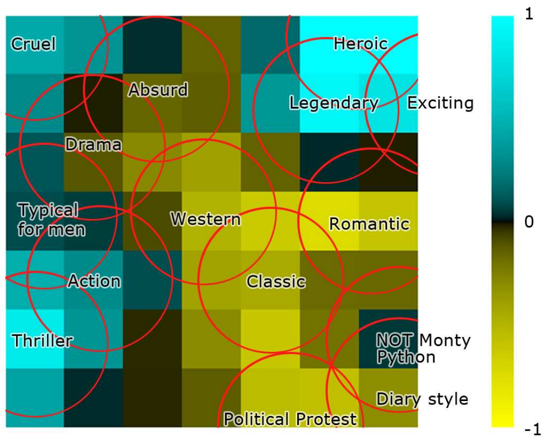


Figure 1: Features of movie *Constantine*

The 2D features of items (or users) can be visualized on a so-called *feature map*. In our next examples, items are movies from the Netflix Prize data set. Meanings can be associated to the regions of the feature map. Note that the labels on Figure 1 are assigned to *groups of features* and not to single features. The labels have been manually determined based on such movies that have extreme values at the given feature group.

The labeling process is performed as follows: we select a cell of the feature map. Then we list the movies with the highest and the lowest feature values in the selected cell; we term them *representative movies*. Finally, we select an expression which describes the best the common properties of the movies with high feature values in contrast to the movies with low feature values. In the case when neighboring features have similar representative movies, we assign a common label to the corresponding cells. As a result of the label unification, we can obtain larger areas of the feature map with the same label; see for instance labels *Classic*, *Western* or *Action* on Figure 1. Since the described labeling process is subjective, it requires human interaction. The size of the areas the labels are assigned to depends on multiple factors like the neighborhood strength ρ, the size of the feature map, and how general the label term assigned is. In the example (Figure 1), the labels principally characterize the features directly under themselves and immediate neighbors. The further a feature is from the label, the less the label reflects its meaning.

Such feature maps are useful for detecting main differences between movies or episodes of the same movie. Figure 2 represents the three episodes of The Matrix movie. One can observe that the main characteristic of the feature maps are the same, but there are noticeable changes between the first and the other episodes. In the first episode the feature values are higher in the area of *Political protest* and *Absurd* and lower around *Legendary*. This may indicate that the first episode presents a unique view of the world, but the other two are rather based on the success of the first episode. Visual feature maps can also be used to demonstrate to the users why they are provided the current recommendations: showing similar feature map of items formerly ranked high by the user can justify the recommendations. This kind visual of explanation may be a preferred by people who can capture the meaning of visual information faster than textual one. We, therefore, recommend this technique for real life recommender systems, although it does not improve the blending of the methods.

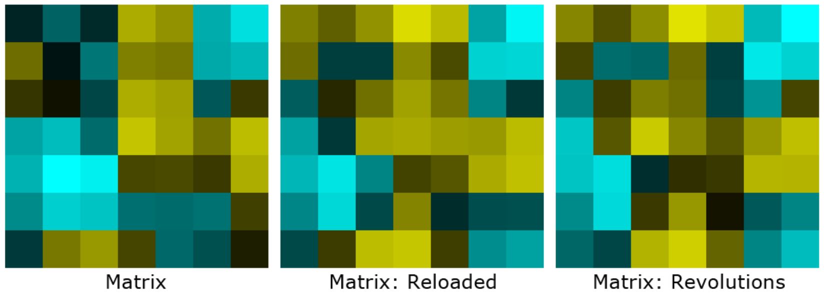


Figure 2: Features of *The Matrix* episodes

3.8 Connections with Neural Networks

In this section we point out that the learning scheme of some MF variants can be directly represented in the framework of neural networks (NN). We show that this correspondence can be exploited by applying the methodology of NN learning for CF problems.

The learning of the ISMF and RISMF models can be paired with the multi-layer perceptron depicted on Figure 3. The network has N inputs, M outputs and K hidden neurons and an identity activation function in each neuron. The weight between the u-th input and the k-th hidden neuron corresponds to p_{uk}, and the weight between the k-th hidden neuron and the i-th output neuron corresponds to q_{ki}.

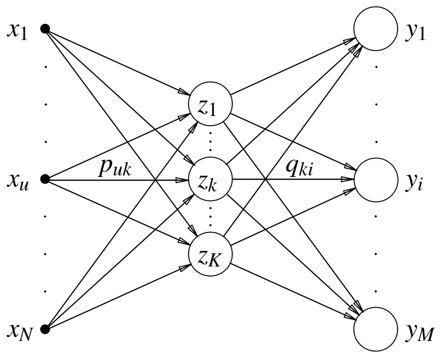


Figure 3: The multilayer perceptron equivalent for the general MF scheme

In the learning phase, an incremental learning method is used. For the (u,i)-th rating, we set the input $\mathbf{x}$ so that x_u is 1 and $x_{u'\neq u} = 0$, thus $z_k = p_{uk}$ holds. Let $\hat{r}_{ui} = y_j$ denote the i-th output of the network. We compute the error: $e_{ui} = r_{ui} - \hat{r}_{ui}$. This error is back-propagated from the i-th output neuron towards the input layer. This neural network (NN) with this special learning is equivalent to the ISMF.

If we apply regularization (weight decay), we get the RISMF approach.

We can extend this NN to be equivalent to the BRISMF: item biases can be handled by adding a bias (constant 1) input to the output neurons; user biases can be handled by setting $\mathbf{q}_{2\bullet}$ weights to 1 and keeping them constant. In the evaluation phase, we set the input as in the learning phase, and y_i $(i = 1, \ldots, M)$ predicts the active user's rating on the i-th item.

It was shown that feed-forward neural networks with linear activation function can find the principal components of a data set (Baldi and Hornik, 1989), since all local minima are global minima there. However, our problem setting differs in two minor and one major issues: First, our goal is to factorize a non-squared matrix (the number of inputs of the neural network is different from the number of outputs), second, we penalize the weights of the neural network, and principally, not all output are defined, since users rate only a small subset of the items. This latter point infers that we cannot expect local minima to be global minima, as it has been reflected on page 629.

4. Neighbor Based Correction of MF

Neighbor based (NB) approaches exploit the observation that similar users rate similar items similarly. In the NB scheme a set of similar users is selected for each query from among those who rated the active item. Or analogously, a set of similar items is selected from among those that have been rated by the active user. The answer of the predictor is then obtained by combining the ratings of similar users (items) for the active item (user). The first variant is termed the user neighbor based, and the second is termed the item neighbor based approach.

MF and NB approaches complement each other well:

- The MF approach views the data from a high level perspective. MF can identify the major structural patterns in the ratings matrix. An appealing property of MF is that it is able to detect the similarity between 2 items, even if no user rated both of them.

- The NB approach is more localized. It is typically good at modeling pairs of users/items and not so good at modeling interdependency within larger sets of users/items. NB methods are memory based, therefore they do not require any training.[3]

It is known that the combination of MF and NB can lead to very accurate predictions (Bell and Koren, 2007a; Bell et al., 2007b). However, the price of additional accuracy is paid by the decreased scalability. Here we propose a scalable solution for unifying the MF and NB approaches. We will point out that it is computationally less expensive than Bell et al's approaches.

The idea is that we try to improve an existing MF model ($\mathbf{P}$,$\mathbf{Q}$) by adding a neighbor based correction term to its answer in the prediction phase. Assuming the item neighbor based scheme, the corrected answer for query (u,i) is the following:

$$\hat{r}_{ui} = \mathbf{p}_u^T \mathbf{q}_i + \gamma \frac{\sum_{j \in \mathcal{T}_u \setminus \{i\}} s_{ij} \left(\mathbf{p}_u^T \mathbf{q}_j - r_{uj} \right)}{\sum_{j \in \mathcal{T}_u \setminus \{i\}} s_{ij}},$$

where s_{ij} is the similarity between $\mathbf{q}_i$ and $\mathbf{q}_j$, and $\mathcal{T}_u$ is set of the items rated by user u. The weight of the correction term γ can be optimized via cross-validation. This model can be seen as a unification of the MF and NB approaches.

3. However, it can be useful to precompute the similarity values to speed up the prediction phase.

The similarity s_{ij} can be defined in many different ways. Here we propose two variants.

- (S1): Normalized scalar product based similarity.

$$s_{ij} = \left(\frac{\max\{0, \sum_{k=1}^{K} q_{ki} q_{kj}\}}{\sqrt{\sum_{k=1}^{K} q_{ki}^2} \cdot \sqrt{\sum_{k=1}^{K} q_{kj}^2}} \right)^{\alpha},$$

  where α is an amplification parameter.

- (S2): Normalized Euclidean distance based similarity.

$$s_{ij} = \left(\frac{\sum_{k=1}^{K} (q_{ki} - q_{kj})^2}{\sqrt{\sum_{k=1}^{K} q_{ki}^2} \cdot \sqrt{\sum_{k=1}^{K} q_{kj}^2}} \right)^{-\alpha}.$$

In both cases, the value s_{jk} can be calculated in $\mathrm{O}(K)$ time, thus $\hat{r}_{ui}$ can be calculated in $\mathrm{O}(K \cdot |\mathcal{T}_u|)$. We remark that one can restrict to use only the top S neighbors of the queried item (Bell and Koren, 2007a), however, it does not affect the time requirement, if we use the same function for s_{ij} and neighbor selection.

Now let us comment in more details on the time and memory requirements of our NB corrected MF in comparison with the improved neighborhood based approach of Bell and Koren (2007a), which can also be applied to further improve the results of an MF. For a given query, the running time of our method is $\mathrm{O}(K \cdot S)$, while their method requires to solve a separate linear least squares problem with S variables, thus it is $\mathrm{O}(S^3)$. Memory requirements: for the u-th user, our method requires the storing $\mathbf{q}_u$ and $\mathbf{Q}$ in the memory, that is $\mathrm{O}(KN)$, while their approach must store the item-by-item matrix and the ratings of the user, which is $\mathrm{O}(M^2 + |\mathcal{T}_u|)$. For all users, our method requires $\mathrm{O}(NK + KM)$ while their approach requires $\mathrm{O}(M^2 + |\mathcal{T}|)$ memory.

Despite the simplicity of our method its effectiveness is comparable with that of Bell and Koren's method, see Section 5.4.9. We highly recommend this method as it improves accuracy significantly, as we show above and in Section 5.4.8.

This model can be seen as simple, scalable, and accurate unification of the MF and NB approaches. The training is identical to the regular MF training. The prediction consists of an MF and a NB term. The similarities used in the NB term need not to be precomputed and stored, because they can be calculated very efficiently from the MF model.

5. Experiments

Currently, the largest publicly available ratings data set is provided by Netflix, a popular online DVD rental company. Netflix initiated the Netflix Prize contest in order to improve their recommender system—called Cinematch—that provides movie recommendations to customers. The data set released for the competition was substantially larger than former benchmark data sets and contained about 100 million ratings from over 480k users on nearly 18k movies (see details in Subsection 5.1.1). For comparison, the well-known EachMovie data set[4] only consists of 2,811,983 ratings of

4. It used to be available upon request from Compaq, but in 2004 the proprietary retired the data set, and since then it has no longer been available for download.

72,916 users and 1,628 movies. Rashid et al. (2006) used a 3 millions ratings subset of the GroupLens project, which entirely contains about 13 million ratings from 105k users on 9k movies. We evaluated all of our algorithms against the Netflix data set, since currently this is the most challenging problem for the collaborative filtering community, and our work was greatly motivated by it. In addition, to illustrate the applicability of the presented methods on other data sets, we also performed experiments with a selected MF, BRISMF, on the 1M MovieLens and the Jester data set (see details in Subsection 5.1.2–5.1.3).

The evaluation metrics of recommender systems can greatly vary depending on the characteristics of the data set (size, rating density, rating scale), the goal of recommendation, the purpose of evaluation (Herlocker et al., 2004). In the current CF setting the goal is to evaluate the predictive accuracy, namely, how closely the recommender system can predict the true ratings of the users, measured in terms of root mean squared error.

In case of MovieLens and Jester data sets, we also provide the mean absolute error (MAE) since this is the most common performance measure for these sets:

$$\mathrm{MAE} = \frac{1}{|\mathcal{V}|} \sum_{(u,i)\in\mathcal{V}} |\hat{r}_{ui} - r_{ui}|.$$

5.1 Data Sets

In this section we describe in details the above mentioned three data sets.

5.1.1 THE NETFLIX PRIZE DATA SET

The data set provided generously by Netflix for the NP competition contains (u, i, r_{ui}, d_{ui}) rating quadruples, representing that user u rated item i as r_{ui} on date d_{ui}, where $d_{ui} \in \mathcal{D}$ the ordered set of possible dates. The ratings r_{ui} are integers from 1 to 5, where 1 is the worst, and 5 is the best. The data were collected between October, 1998 and December, 2005 and reflect the distribution of all ratings received by Netflix during this period (Bennett and Lanning, 2007). The collected data was released in a train-test setting in the following manner (see also Figure 4).

Netflix selected a random subset of users from their entire customer base with at least 20 ratings in the given period. A *Hold-out set* was created from the 9 most recent ratings of the users, consisting of about 4.2 million ratings. The remaining data formed the Training set. The ratings of the Hold-out set were split randomly with equal probability into three subsets of equal size: Quiz, Test and Probe. The *Probe set* was added to the Training set and was released with ratings. The ratings of the *Quiz and Test sets* were withheld as a *Qualifying set* to evaluate competitors. The Quiz/Test split of the Qualifying set is unknown to the public. We remark that the date based partition of the entire NP data set into train-test sets reflects the original aim of recommender systems, which is the prediction of future interest of users from their past ratings/activities.

As the aim of the competition is to improve the prediction accuracy of user ratings, Netflix adopted RMSE as evaluation measure. The goal of the competition is to reduce by at least 10 percent the RMSE on the Test set, relative to the RMSE achieved by Cinematch.[5] The contestants have to submit predictions for the Qualifying set. The organizers return the RMSE of the submissions on

5. The first team achieving the 10 percent improvement is promised to be awarded by a grand prize of $1 million by Netflix. Not surprisingly, this prospective award drawn much interest towards the competition. So far, more than 3 000 teams submitted entries for the competition.

the Quiz set, which is also reported on a public leaderboard.[6] Note that the RMSE on the Test set is withheld by Netflix.

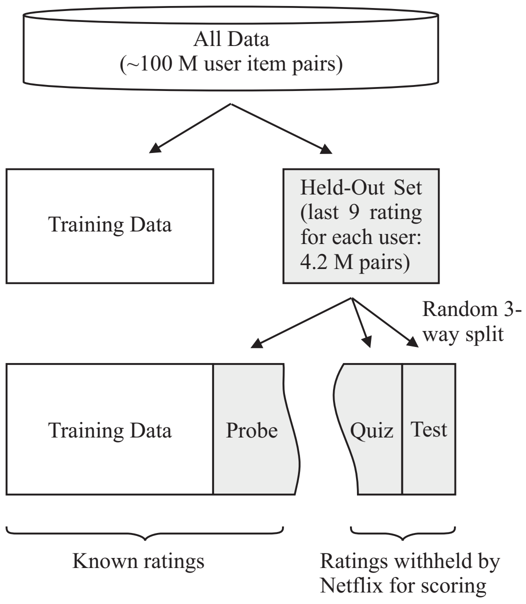


Figure 4: The train-test split and the naming convention of Netflix Prize data set, after Bell et al. (2007a)

There are some interesting characteristics of the data and the set-up of the competition that pose a difficult challenge for prediction:

- The distribution over the time of the ratings of the Hold-out set is quite different from the Training set. As a consequence of the selection method, the Hold-out set does not reflect the skewness of the movie-per-user, observed in the much larger Training set. Therefore the Qualifying set contains approximately equal number of queries for often and rarely rating users.

- The designated aim of the release of the Probe set is to facilitate unbiased estimation of RMSE for the Quiz/Test sets despite of the different distributions of the Training and the Hold-out sets. In addition, it permits off-line comparison of predictors before submission.

- We already mentioned that users' activity at rating is skewed. To put this into numbers, ten percent of users rated 16 or fewer movies and one quarter rated 36 or fewer. The median is 93. Some very active users rated more than 10,000 movies. A similar biased property can be observed for movies: The most-rated movie, *Miss Congeniality* was rated by almost every second user, but a quarter of titles were rated fewer than 190 times, and a handful were rated fewer than 10 times (Bell et al., 2007a).

6. Found at `http://www.netflixprize.com/leaderboard`.

- The variance of movie ratings is also very different. Some movies are rated approximately equally by the user base (typically well), and some partition the users. The latter ones may be more informative in predicting the taste of individual users.

In this experimentation section we evaluate the presented methods on a randomly selected 10% subset of the Probe set, which we term as Probe10.[7] Unless we explicitly mention, from now on the RMSE values refer to the Probe10 RMSE. We have decided to report on RMSE values measured on the Probe10 set, since in our experiments the Probe10 RMSE are significantly closer to the Quiz RMSE than Probe RMSE, and Quiz RMSE tell us more about the accuracy of the predictor, since it excludes the impact of overtraining. On the other hand the rules of NP competition allows only 1 submission daily, which limits the number of the Quiz RMSE calculation drastically. We remark that we measured typically 0.0003 difference between the Probe10 and the Quiz RMSE values (sometimes 0.0010), while this was of an order of magnitude larger for the Probe set. This advantageous property nominates the Probe10 set for being a standard and Netflix-independent evaluation set for predictors trained on the NP data set.

We performed a thorough analysis to check how reliable the Probe10 results are. For this, we will show that if a given method has better RMSE compared to another method for a particular subset of the Probe set, then it has the same performance gain on other subsets of the Probe set. To do this, we partitioned the Probe set into 10 subsets, and we ran 10 different methods using them. Therefore, in total we had 100 runs. We denote the test RMSE of the x-th method on the y-th test set by m_{xy}. To summarize the results, we calculated the average test RMSE for each method, denoted by $(\overline{m}_{1\bullet}, \dots, \overline{m}_{10\bullet})$, and a *difficulty offset* for each test set, denoted by $(o_1, \dots, o_{10})$, defined as

$$o_y = \frac{1}{10} \sum_{x=1}^{10} m_{xy} - \overline{m}_{x\bullet}.$$

The test RMSE of the 100 runs are approximated as $m_{xy} = \overline{m}_{x\bullet} + o_y$. The standard deviation of this approximation is 0.000224 RMSE score, and the maximal deviation is less than 0.0007 RMSE score, which means that m_{xy} is well approximated, thus we can assign a difficulty offset to each test set. Consequently, our initial hypothesis is verified.

When Quiz RMSE values are reported we also mention the percentage of improvement over Cinematch (IoC). We performed all tests on an average single processor laptop (a 2 GHz Intel Pentium M (Dothan) with 1 GB RAM), on which reported training times were measured.

5.1.2 MOVIELENS DATA SET

The 1M MovieLens data set[8] contains cca. 1 million ratings from 6,040 users on 3,900 movies. As in the case of NP, ratings are made on a 5 star scale, and the rating records are also quadruples containing the timestamp of the rating. Demographic data provided with the ratings are not used in our setting. Since there is no standard train-test split of the data set, we applied a simple random split to generate a 90%–10% train-test setting.[9]

7. A Perl script can be downloaded from our homepage, `http://gravityrd.com`, which selects the Probe10 from the original Netflix Probe set to ensure repeatability and comparability.
8. Available at: `http://www.grouplens.org/node/73`.
9. This split can be downloaded from our website: `http://gravityrd.com`.

5.1.3 JESTER DATA SET

The Jester data set[10] (Goldberg et al., 2001) contains 4,136,360 ratings from 73,421 users on 100 jokes. Users rated jokes on the continuous $[-10, +10]$ range. Ten percent of the jokes (called the gauge set, which users were asked to rate) are densely rated, others, more sparsely. Two thirds of the users have rated at least 36 jokes, and the remaining ones have rated between 15 and 35 jokes. The average number of ratings per user is 46, so it is a particularly dense data set compared to NP and MovieLens. Goldberg et al. (2001) created their train-test split by a random division of a subset of 18,000 users into two disjoint sets. For our experiments this split is obviously inappropriate since it does not enable us to integrate user preferences into the model. Therefore, here we also applied a random split to generate a 90%–10% train-test setting.[11]

5.2 Implementation Issues

Because the data set is huge, its storage is an important issue. This was thoroughly investigated in our previous work (see Takács et al., 2007). We have shown there that the entire data set can be stored in 300 MB without storing the dates but keeping the chronological order of the ratings, and in 200 MB without even keeping the order. This enables to perform the algorithms on an average PC (our platform details are given at the end of Section 5.1.1). As we will point out here, MF methods are sensitive to the order of training examples and the selection of their proper order exploits date information. On the other hand, pure NB approaches are not sensitive to the order of training examples.

The users' tastes change in time, and when providing them recommendations, only their current taste matters. This phenomenon is modeled in the NP data set as the value of d_{ui} is greater for test examples than for training examples. We can condition MF methods to exploit the date information by properly ordering training examples. We found the following order to be very effective: iterate over users in an arbitrary order, and for each user, take the ratings in an increasing chronological order, that is, starting from the oldest and ending with the newest. Unless explicitly mentioned otherwise, in our experiments we use this training example order for MF methods. The impact of the order on the accuracy of MFs is investigated in Subsection 5.4.4.

5.3 Parameter Optimization

All of the presented methods have many pre-defined parameters that greatly influence the accuracy. Sometimes a few experiments are enough to set them well, sometimes we need to apply parameter optimization to find the best settings.

We recall that a parameter setting can be advantageous because (1) it produces a very accurate model (low validation RMSE) or (2) it "blends well", that is it improves the accuracy of the blended model. The more parameters a method has, the harder to set them well, but the more chance to get a better RMSE.

We used random search and Algorithm 3 to optimize parameters. The typical value of n is 2. In case of MF, we have experimented with many parameters, namely:

- the number of features: K;
- different learning rate and regularization factor

10. Available at: `http://goldberg.berkeley.edu/jester-data/`.
11. This split can be downloaded from our website: `http://gravityrd.com`.

```
   Input: $L, p_1, \dots, p_L, n$
   Output: $v_1, \dots, v_l$
 1 Randomly initialize parameters $p_1, \dots, p_L$
 2 Iterate forever (iteration is stopped manually).
 3    Randomly choose one parameter: $p_l$;
 4    Randomly generate $n$ different values
 5       to that parameter: $u_1, \dots, u_n$;
 6    Let $u_0$ be the current value of the $p_l$.
 7    For each of the $u_0, \dots, u_n$ values, run a training
 8       algorithm, temporarily setting $p_l$ to that value, and
 9       evaluating the model on the validation set.
10    Assign the best value to $p_l$.
11 end
```

Algorithm 3: Simple parameter optimization algorithm

  - for users and movies ($\eta^{(p)}, \eta^{(q)}, \lambda^{(p)}, \lambda^{(q)}$);
  - for the corresponding variables of bias features ($\eta^{(pb)}, \eta^{(qb)}, \lambda^{(pb)}, \lambda^{(qb)}$);
- minimum and maximum weights in the uniform random initialization of **P** and **Q**: $w_{\underline{p}}, w_{\overline{p}}, w_{\underline{q}}, w_{\overline{q}}$;
- G: the offset to subtract from **R** before learning (can be, for example, set to the global average of ratings);

We subsampled the matrix **R** for faster evaluation of parameter settings. We have experienced that movie-subsampling substantially increased the error, in contrast to user-subsampling, thus we do not perform the former. The reason for this is that in the evaluation data set movies have much more ratings than users. Consequently, if we do user-subsampling for example with 100 instead of the original 200 ratings we lose more information than at movie-subsampling when we have for example 10000 ratings instead of 20000. Interestingly, the larger the subsample is, the fewer iterations are required to achieve the optimal model. This can be explained by the existing redundancy in the data set. This implies also that the time-complexity of MF is sublinear in the number of ratings.

5.4 Results of Matrix Factorization

We recall that we applied linear combinations of methods for blending,[12] and we ordered the training examples user-wise and then by date, as specified in Section 5.2.

5.4.1 COMPARING REGULARIZED AND BIASED MF VARIANTS

We compare:

- an instance of the regularized RISMF, termed as RISMF#0, with the following parameter settings: $K = 40, \eta = 0.01, \lambda = 0.01, w_{\underline{p}} = -w_{\overline{p}} = w_{\underline{q}} = -w_{\overline{q}} = -0.01$
- an instance of the biased BRISMF, named briefly as BRISMF#0, with the following parameter settings: $K = 40, \eta = 0.01, \lambda = 0.01, w_{\underline{p}} = -w_{\overline{p}} = w_{\underline{q}} = -w_{\overline{q}} = -0.01$

12. The source code of our combination algorithm can be downloaded from our web site: `http://gravityrd.com`.

RISMF#0 reaches its optimal RMSE in the 13th epoch: 0.9214, while these numbers for BRISMF#0 are: 10th and 0.9113, which is a 0.0101 improvement.

5.4.2 CHANGING THE PARAMETERS OF THE BRISMF

Table 1 presents the influence of η and λ on the Probe10 RMSE and the optimal number of epochs. Other parameters are the same as in BRISMF#0. The best result is RMSE $= 0.9056$ when $\eta = 0.007, \lambda = 0.005$, which is a 0.0057 improvement. We refer to this MF in the following as BRISMF#1. The running time for this MF is only 14 minutes! Note that running time depends only on K, and on the optimal number of epochs.

Note that decreasing η or increasing λ increases the optimal number of epochs, (except for $\eta = \lambda = 0.020$).

η \ λ	0.005	0.007	0.010	0.015	0.020
0.005	0.9061 / 13	0.9079 / 15	0.9117 / 19	0.9168 / 28	0.9168 / 44
0.007	**0.9056 / 10**	0.9074 / 11	0.9112 / 13	0.9168 / 19	0.9169 / 31
0.010	0.9064 / 7	0.9077 / 8	0.9113 / 10	0.9174 / 13	0.9186 / 21
0.015	0.9099 / 5	0.9111 / 6	0.9152 / 6	0.9257 / 7	0.9390 / 7
0.020	0.9166 / 4	0.9175 / 4	0.9217 / 4	0.9314 / 4	0.9431 / 3

Table 1: Probe10 RMSE/optimal number of epochs of the BRISMF for various η and λ values ($K = 40$)

Now we show that the usage and proper setting of new parameters can boost the performance: we introduce the parameters $\eta^{(p)}, \eta^{(q)}, \lambda^{(p)}, \lambda^{(q)}$ and $\eta^{(pb)}, \eta^{(qb)}, \lambda^{(pb)}, \lambda^{(qb)}$ (see Section 5.3 for explanation).

Initially $\eta^{(p)} = \eta^{(q)} = \eta^{(qb)} = \lambda^{(pb)} = 0.007$, and $\lambda^{(p)} = \lambda^{(q)} = \lambda^{(pb)} = \lambda^{(qb)} = 0.005$, to yield the RMSE $= 0.9056$ given above. Finding the best setting of these 8 variables is practically impossible and can cause overlearning on Probe10. To demonstrate how the parameter optimization algorithm mentioned in Section 5.3 works, we apply its simplified version: we do not use random numbers, just fix the order of these variables and define the possible values for them. Let the order be: $\eta^{(p)}, \eta^{(q)}, \eta^{(pb)}, \lambda^{(qb)}, \lambda^{(p)}, \lambda^{(q)}, \lambda^{(pb)}, \lambda^{(qb)}$. Let the set of values for η variants be $\{0.005, 0.007, 0.010\}$, and for the λ variants: $\{0.003, 0.005, 0.007\}$. Table 2 shows step-by-step how parameters are optimized one-by-one in 8 iterations. The parameter optimization procedure decreased the RMSE score from 0.9056 to 0.9036.

5.4.3 BRISMF RESULTS ON MOVIELENS AND JESTER DATA SETS

We performed several tests on MovieLens and Jester data sets with the BRISMF method. As mentioned earlier there are no standard train-test split for these data sets; therefore it is difficult to compare our obtained results with already published ones. Consequently, we used three baseline methods for comparison. The *constant* method always predicts the average of the ratings in the training set, the *item average* outputs the average of the training ratings of the active item at querying, while the *item neighbor* (Takács et al., 2008b) is an item neighbor based method with Pearson

	0.003	0.005	0.007	0.010	Decision
$\eta^{(p)}$		0.9057	**0.9056**	0.9058	$\eta^{(p)} := 0.007$ (no change)
$\eta^{(q)}$		0.9057	**0.9056**	0.9061	$\eta^{(q)} := 0.007$ (no change)
$\eta^{(pb)}$		0.9059	0.9056	**0.9052**	$\eta^{(pb)} := 0.010$
$\eta^{(qb)}$		0.9053	**0.9052**	0.9056	$\eta^{(qb)} := 0.007$ (no change)
$\lambda^{(p)}$	0.9053	0.9052	**0.9051**		$\lambda^{(p)} := 0.007$
$\lambda^{(q)}$	0.9057	0.9051	**0.9050**		$\lambda^{(q)} := 0.007$
$\lambda^{(pb)}$	0.9053	0.9050	**0.9047**		$\lambda^{(pb)} := 0.007$
$\lambda^{(qb)}$	**0.9036**	0.9047	0.9066		$\lambda^{(qb)} := 0.003$

Table 2: Effect of parameter optimization on BRISMF#1

correlation based similarity. The MAE results for BRISMFs were obtained by using the same **P** and **Q** that were used to get the RMSE values.

For MovieLens, the main training parameters were set to $\eta^{(p)} = \eta^{(q)} = 0.01, \lambda^{(p)} = \lambda^{(q)} = 0.02$, and the number of features (K) was varied as tabulated in Table 3. The obtained results show that the increase of the number of features K yields better accuracy at a decreasing number of epochs.

Model	Epochs	RMSE w/o S2	RMSE with S2	MAE w/o S2	MAE with S2
constant	–	1.1179	–	0.9348	–
item average	–	0.9793	–	0.7829	–
item neighbor	–	0.8521	–	0.6641	–
BRISMF#5	35	0.8555	0.8537	0.6684	0.6667
BRISMF#10	27	0.8471	0.8426	0.6608	0.6563
BRISMF#20	24	0.8435	0.8363	0.6582	0.6507
BRISMF#50	23	0.8396	0.8319	0.6544	0.6461
BRISMF#100	24	0.8378	0.8299	0.6531	0.6444
BRISMF#200	21	0.8365	0.8285	0.6519	0.6430
BRISMF#500	20	0.8353	0.8275	0.6508	0.6424

Table 3: Test RMSE of various methods on the MovieLens data set

In terms of RMSE, the simplest BRISMF#5 achieves 12.64% improvement over the item average, while this is 14.70% for the largest BRISMF#500.[13] We also included in the table the test RMSE value achieved with neighbor correction using similarity function S2 ($\alpha = 5$). The improvement over the item average is 12.82% and 15.5% for the neighbor corrected BRISMF#5 and BRISMF#500, respectively. The S2 correction improves RMSE value from 0.0018 to 0.0080, which can be as large as almost 1% improvement over the RMSE of the non-corrected version. One can observe that the percentage of improvement increases with the number of features. The improvement of BRISMF#500 with S2 correction over item neighbor is 2.89%, which is similar to the results published in Takács et al. (2008b) and Takács et al. (2008a) for the Netflix data set.

13. We recall that Cinematch produces 9.6% improvement over item average (Quiz RMSE= 1.0540) on the NP data set (Bennett and Lanning, 2007).

In terms of MAE, the tendency of the improvements is almost identical. The simplest and the largest BRISMF achieves 14.63% and 16.87% improvements, which become 14.84% and 17.95% with NB correction. Here the magnitude of improvement is somewhat larger.

The reported results on RMSE and MAE value on the MovieLens data set are not directly comparable with ours because of the use of different train-test splits. The best known results are given by Delannay and Verleysen (2007): RMSE 0.875 and MAE 0.648 that are obtained with an interlaced generalized linear model. Similar MAE $= 0.652$ was reported by DeCoste (2006) achieved with ensembles of maximum margin matrix factorizations.

For Jester, the main training parameters were set to $\eta^{(p)} = \eta^{(q)} = 0.002$, $\lambda^{(p)} = \lambda^{(q)} = 0.02$, and K was varied as tabulated in Table 4. The obtained results show that the increase of K yields better accuracy, while the number of epochs is almost the same. Due the different characteristics of the Jester data set, the magnitude of RMSE scores are larger. It is interesting that on the Jester data set the item neighbor method gives better results than BRISMF. We think that this phenomenon is due to the different characteristics of Jester data set when compared to NP and MovieLens data sets: the rating matrix is almost dense and there are only 100 items.

Model	Epochs	RMSE w/o S2	RMSE with S2	MAE w/o S2	MAE with S2
constant	–	5.2976	–	4.4372	–
item average	–	5.0527	–	4.1827	–
item neighbor	–	4.1123	–	3.1616	–
BRISMF#5	7	4.2080	4.1902	3.2608	3.2352
BRISMF#10	8	4.1707	4.1575	3.2201	3.1967
BRISMF#20	7	4.1565	4.1405	3.2095	3.1820
BRISMF#50	8	4.1399	4.1265	3.1876	3.1616
BRISMF#100	7	4.1395	4.1229	3.1909	3.1606

Table 4: RMSE of various methods on the Jester data set

In terms of RMSE, the simplest BRISMF#5 achieves 16.72% improvement over the item average, while this is 18.07% for the largest BRISMF#100. We also included in the table the test RMSE value achieved with neighbor correction using similarity function S2 ($\alpha = 5$). The improvement over the item average is 17.07% and 18.40% for the neighbor corrected BRISMF#5 and BRISMF#100, respectively. The S2 correction improves the RMSE value from 0.0132 to 0.0178, which can be an over 0.4% improvement over the RMSE of the non-corrected version. In terms of MAE, the improvements are somewhat larger; they reach 23.72% without and 24.43% with S2 correction.

Here we also indicate some of the best published RMSE and MAE scores, keeping in mind that those are not directly comparable due to different test settings. Delannay and Verleysen (2007) achieved RMSE 4.17 and MAE 3.26 with an interlaced generalized linear model; the same MAE score was obtained by Canny (2002) with a sparse factor analysis model.

We can conclude from the experiments performed on the MovieLens and Jester data sets that the applicability of MF based methods and neighbor based correction technique is not restricted to the NP data set. Rather, they are useful CF techniques for different problems as well.

5.4.4 ORDER OF EXAMPLES

We examined how the order of examples influences the result by comparing the RMSE of BRISMF#1 on two different orders: For the proposed order the RMSE is 0.9056, and for a random order—obtained by a random shuffle of the ratings of each user—the RMSE is 0.9104.

5.4.5 SUBSAMPLING USERS

On Figure 5 we demonstrate how the number of users (thus, the number of ratings) influences RMSE and the optimal number of training epochs in case of BRISMF#1. RMSE varies between 0.9056 and 0.9677, and the number of epochs between 10 and 26. The smaller the subset of users used for training and testing, the larger the RMSE and the number of epochs. This means that the time-complexity of MF is sublinear in the number of ratings (see Section 5.3). We remark that the number of ratings is proportional to the number of users; the ratio of them—which is equal to the average number of ratings per user—is 209 in the training set.

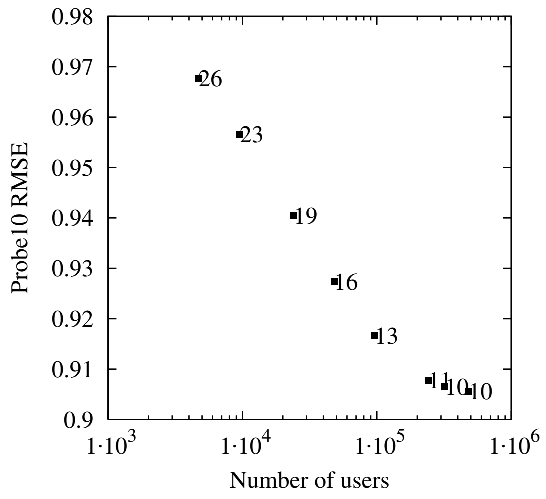


Figure 5: Effect of the number of users on Probe10 RMSE and on the optimal number of training epochs.

5.4.6 SEMIPOSITIVE AND POSITIVE MF

We investigated the accuracy of semipositive and positive variants using the following MFs:

- SemPosMF#800: this is a semipositive MF, where user features are nonnegative and item features are arbitrary. Parameters are set to: $K = 800, w_{\underline{p}} = 0, w_{\overline{p}} = -w_{\underline{q}} = w_{\overline{q}} = 0.005, \eta^{(p)} = \eta^{(pb)} = 0.016, \eta^{(q)} = \eta^{(qb)} = 0.005, \lambda^{(p)} = \lambda^{(q)} = 0.010, \lambda^{(pb)} = \lambda^{(qb)} = 0, G = 3.6043$. After 12 epochs, learning rates are multiplied by 0.01, and the model is trained for another 2 epochs.
- PosMF#400: this is a positive MF. Parameters are the same as in SemPosMF#800, except that $K = 400, w_{\underline{q}} = 0$.
- PosMF#100: like PosMF#400, but $K = 100$.

The results are summarized on Table 5.

Model	Epochs	RMSE
SemPosMF#800	12+2	0.8950
PosMF#400	14	0.9036
PosMF#100	14	0.9078

Table 5: Probe10 RMSE of positive and semipositive MFs

5.4.7 RETRAINING USER FEATURES

We investigated the effectiveness of retraining user features. We tested both solutions proposed in Section 3.5. The experiments were run with three parameter settings: BRISMF#1, and the following two MFs:

- BRISMF#250: $K = 250, w_{\underline{p}} = -0.01, w_{\overline{p}} = -0.006, w_{\underline{q}} = -0.010, w_{\overline{q}} = 0.020, \eta^{(p)} = 0.008, \eta^{(pb)} = 0.016, \eta^{(q)} = 0.015, \eta^{(qb)} = 0.007, \lambda^{(p)} = 0.048, \lambda^{(q)} = 0.008, \lambda^{(pb)} = 0.019, \lambda^{(qb)} = 0, G = 0$.
- BRISMF#1000: the same as BRISMF#250, but $K = 1000$.

First, we investigated the first solution (intra-training user-wise test). For BRISMF#250 the obtained Probe10 RMSE is 0.8959, which is only a slight 0.0002 improvement.

Second, we tested the second solution (see Table 6). Both the simpler case (after the first learning step, reset **P** and retrain only **P**), and the advanced case (after the first learning step, reset **P** and retrain both **P** and **Q**) are analyzed. We append letter "U" to the method name in the simpler case (BRISMF#1 becomes BRISMF#1U, etc.) and letters "UM" in the advanced case (BRISMF#1UM, etc.). We indicated the required number of epochs both in the first and the second training procedure (if available). Note, that in case of BRISMF#250 and BRISMF#1000, the retraining of user features greatly improves their performance. BRISMF#1000UM is currently our best MF: Probe10 RMSE is 0.8921, Quiz RMSE is 0.8918.

Model	Epochs	Probe10	Quiz	IoC
BRISMF#1	10	0.9056		
BRISMF#1U	10+8	0.9072		
BRISMF#1UM	10+6	0.9053		
BRISMF#250	14	0.8961	0.8962	5.80%
BRISMF#250U	14+8	0.8953	0.8954	5.89%
BRISMF#250UM	14+7	0.8937		
BRISMF#1000	14	0.8938	0.8939	6.04%
BRISMF#1000U	14+8	0.8936		
BRISMF#1000UM	14+8	0.8921	0.8918	6.26%

Table 6: Examining the effect of retraining user features

We investigated the effect of retraining only **P**, when BRISMF#250 was learnt on a subset of the database. The question in this case is: how reliable is a **Q** learnt only on a subset of the database.

First, we kept only the 40%, 60% or 80% of users and ran an MF algorithm and fixed the resulting **Q**. Then we reset and learnt **P**, first on the same subset of the database, and then on the entire database. In all 3 cases, the difference between the two Probe10 RMSE results was less than 0.0013. Each Probe10 RMSE was less than 0.8970. Thus, we can conclude that the proposed retraining method can handle the addition of new users.

Second, we discarded the last N_1 ratings of each user and ran the same retraining procedure. In our experiments, N_1 was set to 0, 10, 20, 40. Obviously, the removal of ratings increased Probe10 RMSE significantly; the highest score was 1.0038, whereas Probe10 RMSE on the entire training database went up to only 0.8980, which means that the **Q** calculated on the subset of the data differs slightly from the **Q** calculated on the entire data set. Thus, the proposed retraining method can handle the addition of new ratings as well. These experiments verify the usability of user feature retraining method for handling new users or ratings.

5.4.8 THE EFFECT OF CORRECTION TECHNIQUES

In order to investigate the effect of various correction techniques, we first generated a number of accurate MF models. We will show that correction techniques improve accuracy significantly for all of these models. We applied the parameter optimization method mentioned in section 5.3 to get accurate MFs. Also, we applied the "trial and error" method to get manually parameterized accurate MFs. Here we describe some results of both:

- BRISMF#800: manually parameterized MF, with 800 features. Parameters are set to: $K = 800$, $w_{\underline{p}} = -w_{\overline{p}} = w_{\underline{q}} = -w_{\overline{q}} = -0.005, \eta^{(p)} = \eta^{(pb)} = 0.016, \eta^{(q)} = \eta^{(qb)} = 0.005, \lambda^{(p)} = \lambda^{(q)} = 0.010, \lambda^{(pb)} = \lambda^{(qb)} = 0, G = 3.6043$. After 9 epochs, learning rates are multiplied by 0.01, and the model is trained for another 2 epochs.

- SemPosMF#800: defined in Section 5.4.6.

- MlMF#200: a BRISMF with 200 features. Parameters are found by the parameter optimization algorithm.

- MlMF#80: a BRISMF with 80 features. Parameters are found by the parameter optimization algorithm.

- MomentumMF: a BRISMF with momentum method, manually optimized: $K = 50, \eta = 0.01$, $\sigma = 0.3$ and $\lambda = 0.00005$. Model learnt in 5 epochs.

We refer to a variant of the transductive MF algorithm as Q-correction: in Eq. (9) to improve predictions we use only the ratings in the Qualify set, not in the Probe10 + Qualify set. See Table 7 for the RMSE values of the each method and its blended versions. We applied two NB corrections to the MF models, with similarities S1 and S2.

The results indicated in the Q, S1, S2 and Q + S1 + S2 columns are obtained by using one or more correction techniques; thus those figures refer to linear regression of predictors (columns) on Probe10 data. Each correction technique adds one more column to the combination of the basic method; that is Q, S1, S2 add 1 extra column, Q + S1 + S2 adds 3 extra columns.

One can observe in Table 7 that NB correction significantly improves the result of MF based methods. Starting from an average MF (MlMF#80) the reduction of RMSE can be 0.0179, it reduces the RMSE of the good MomentumMF by 0.0075, and it even improves slightly (0.0026) the very accurate BRISMF#800. We recall that we measured similar accuracy improvement using NB correction (with S2 similarity) in the case of the MovieLens and the Jester data sets (see Sections 5.1.2

#	Model	basic	Q	S1	S2	Q + S1 + S2
1	BRISMF#800	0.8940	0.8930	$0.8916_{(\alpha=8)}$	$0.8914_{(\alpha=7)}$	0.8902
2	SemPosMF#800	0.8950	0.8941	$0.8916_{(\alpha=8)}$	$0.8913_{(\alpha=5)}$	0.8900
3	MlMF#200	0.9112	0.9106	$0.9087_{(\alpha=8)}$	$0.9085_{(\alpha=6)}$	0.9076
4	MlMF#80	0.9251	0.9240	$0.9104_{(\alpha=9)}$	$0.9072_{(\alpha=2)}$	0.9058
5	BRISMF#1000UM	0.8921	0.8918	$0.8905_{(\alpha=7)}$	$0.8907_{(\alpha=5)}$	0.8901
6	MomentumMF	0.9031	0.9020	$0.8979_{(\alpha=6)}$	$0.8956_{(\alpha=3)}$	0.8949
7	$1+2$	0.8923				0.8880
8	$1+2+3$	0.8913				0.8872
9	$1+2+3+4$	0.8909				0.8863
10	$1+2+3+4+5$	0.8895				0.8851
11	$1+2+3+4+5+6$	0.8889				0.8838

Table 7: Probe10 RMSE of accurate MFs without and with applying Q-correction and NB correction (S1 and S2). At columns S1 and S2 we also indicated the optimal value of parameter α.

and 5.1.3). In comparison, BellKor's approach (Bell and Koren, 2007a, Table 2) results in 0.0096 RMSE reduction, starting from MF with 0.9167 RMSE. Here the reduced RMSE score is almost identical with our NB corrected MlMF#80 that has originally only RMSE 0.9251.

If we put in all MFs and all correction techniques, which is a linear combination of 24 methods, then the combination yields RMSE = 0.8838, Quiz RMSE = 0.8839. Using only the first 4 methods with all corrections (combination of 16 methods), it yields RMSE = 0.8863, Quiz RMSE = 0.8862.

It brings only insignificant improvements if one applies Q-correction technique for all MFs. We get RMSE = 0.8839 if we exclude the Q-corrections of all MFs but the first from the combination. Moreover, if we apply neighbor and Q-correction only on BRISMF#800, then the RMSE increases only by 0.0011 to 0.8850. In general, we can state that one "correction technique" brings a major decrease in the RMSE when applied only to a single method in the linear combination. If we apply it multiple times, the improvement becomes less. In other words, Q-correction and NB corrections captures the same aspects of the data, regardless of the MF behind them.

These experiments demonstrate that the Probe10 set containing 140,840 ratings is big enough to evalute not only single methods, but also combinations of many methods.

5.4.9 COMPARISON WITH BELLKOR'S POSTPROCESSING

The neighbor based correction of MF can also be done by running a neighbor based method on the residuals of MF. A very effective known algorithm for postprocessing the residuals of MF is BellKor's neighbor based method (Bell and Koren, 2007b) (BKNB). The comparison of our neighbor correction scheme and BKNB can be seen in Table 8.

In the experiments we applied the techniques on the residuals of 3 models: a BRISMF#100, a SemPosMF#100, and a so called *global effects model* (Bell and Koren, 2007b) with 12 effects. For running S1 and S2 correction on global effects we used the item features of SemPosMF#100.

Model	basic	S1	S2	S1 + S2	BKNB
BRISMF#100	0.8979	0.8940	0.8937	0.8933	0.8948
SemPosMF#100	0.9001	0.8954	0.8951	0.8946	0.8957
GlobalEffects#12	0.9600	0.9196	0.9237	0.9174	0.9145

Table 8: Comparison of NB correction and BKNB in terms of Probe10 RMSE.

For MF models the most accurate postprocessing technique is S2 correction. In the case of global effects BKNB gives the lowest Probe10 RMSE. It is also important to mention that our approach is significantly faster than BKNB (see Section 5.4.10).

5.4.10 SPEED VS. ACCURACY

From the scalability point of view, it is interesting and important to investigate the relationship of speed and accuracy. We ran numerous randomly parameterized MFs with $K = 40$, and collected the best accuracies in each epoch, and then optimized the parameters. Table 9 summarizes the results. One epoch takes 80 seconds ($K = 40$), and the initialization takes an additional 40 seconds (loading the full database into the memory).

An RMSE of 0.9071 can be achieved within 200 seconds (including the time to train with the 100 million available ratings and evaluate on the Probe10)! For a comparison: Netflix's Cinematch algorithm can achieve Quiz RMSE 0.9514, so this fast solution achieves more than 5.6% improvement on Cinematch.

In Table 9, 1.1 epoch means that the model was trained for one epoch and then the ratings of the first 1/10 of users was used for another epoch. The reason is the same as for feature retraining (see Section 3.5): when we train only for 1 epoch, the features of the first trained users will be obsolete at the end of the epoch, since items have nonsense values at the beginning of the training procedure, and item features change significantly by the end of the epoch.

Epoch	Training Time (sec)	RMSE
1	120	0.9179
1.1	128	0.9147
2	200	0.9071
3	280	0.9057
4	360	0.9028
5	440	0.9008
6	520	0.9002

Table 9: Probe10 RMSE and running time of fast and accurate MFs.

To our best knowledge, the running times and accuracies here and in the previous sections are favorable compared to any other method published in the field of Collaborative Filtering. Though this statement might seem somewhat speculative since authors do not tend to publish running times, we can support it with the following arguments:

- we train each feature simultaneously;
- the number of epochs is small;

- we use a gradient descent algorithm, which is the fastest if we can keep the number of required gradient steps low, which is exactly the case.

Note that given n^*, the number of epochs, there are $n^* \cdot |\mathcal{T}| \cdot K$ variable updates in **P** and **Q** during the training. The presented methods can achieve the favorable RMSEs while keeping the number of features (K) and the number of epochs (n^*) low; consequently they are also favorable in terms of time requirement.

Let us compare the time requirement of our MF methods (all major variants) to one of the best published ones. Bell and Koren (2007a) provide a detailed description of their alternating least squares approach proposed to matrix factorization. Briefly, their idea is to initialize **P** and **Q** randomly, recompute one of them using a nonnegative or a regular least squares solver while the other is constant, then recompute the other, and iterate these two alternating steps for a certain number of epochs (n^*). In the case of the **P**-step, one needs to run the solver for each user to determine how the features of the items rated by the user should be combined to best predict the ratings. One run of the solver requires $\Omega(K^3)$ time, which should be run for each user; thus the **P**-step requires $\Omega(|N| \cdot K^3)$ time.[14] Analogously, the **Q**-step requires $\Omega(|M| \cdot K^3)$ time. The K^2 elements of the covariance matrix need to be updated for each rating, thus, in both alternating steps we update K^2 elements $|\mathcal{T}|$ times. Let n^* denote the optimal number of epochs, which is a few dozen according to their paper. In total, their method requires $\Omega((|N| + |M|) \cdot K^3 + |\mathcal{T}| \cdot K^2) \cdot n^*$ time.

Our presented approaches have $O(|\mathcal{T}| \cdot K) \cdot n^*$ computational complexity, where n^* is typically less than 20. We remark that $O(\cdot)$ is an upper bound, while $\Omega(\cdot)$ is a lower bound for the computational complexity.

Here we neglected the cost of parameter optimization. Our MF has 13 parameters. We perform the parameter optimization process (Sec. 5.3) with a subset of users (1/6), and with a small K value (typically K is 20 or 40). The optimization process requires 100–200 MF runs. In case of SemPosMF#800, which is manually parameterized, we performed ~50 runs. One may argue that parameter optimization for alternating least squares type MF is faster, since there are no learning rates, thus it has just 9 parameters. We observed that the more parameters the MF have, the easier it was to tune the parameters to get the same Probe10 RMSE. Consequently, we introduced some additional parameters, for example $\eta^{(p)}, \eta^{(q)}, \eta^{(pb)}, \eta^{(qb)}$ instead of a single η.

5.5 RMSE Values Reported by Other Authors

Finally, let us compare the accuracy of our method (in terms of Probe10 RMSE values that differ from Quiz RMSE values at most by 0.0003) with other RMSE values reported for the Netflix Prize data set. This comparison is difficult since authors often report on RMSE values measured on various custom test sets, different from the Probe and Quiz set. Of the latter two options, Probe RMSE values, which are calculated by leaving out the Probe set from the Train set, can be also misleading, and, consequently, Probe RMSE is often much lower than Quiz RMSE. We remark that Quiz RMSE is often computed by incorporating the Probe data into the training of the predictor. The comparison presented in Table 10 therefore focuses on methods where Quiz RMSE values are available. The table shows that our presented MF method and correction techniques compare favorably with other published ones.

14. There exists somewhat better least squares solvers, but this does not significantly change this comparison.

Source	Method's name	Quiz	IoC	Probe10
Paterek (2007)	Basic + RSVD + RSVD2	0.9070	4.67%	
Salakhutdinov and Mnih (2008)	PMF + PMF with a learnable prior + constrained PMF	0.8970	5.72%	
Bell et al. (2007b)	best stand-alone positive MF	0.9039	4.99%	
	best NB corrected positive MF	0.8953	5.90%	
this paper	stand-alone MF, BRISMF#1000	0.8939	6.04%	0.8938
	stand-alone MF with retrained features, BRISMF#1000UM	0.8918	6.26%	0.8921
	NB corrected MF, BRISMF#1000UM + S1	0.8904	6.41%	0.8905
	stand-alone positive MF, PosMF#400	0.9046	4.92%	0.9036

Table 10: Comparison of Quiz RMSE values of reported MF based methods. We also indicate the Probe10 values of our methods

6. Conclusions

This paper surveyed our approaches for collaborative filtering. We presented several MF methods and a neighbor based correction to the MF. Our methods apply a number of small modifications compared to already published MF variants, but these modifications are together important from the aspects of implementation (time and memory requirements) and accuracy. We performed a comprehensive evaluation of our methods on the Netflix Prize data set, and we showed that the methods can be efficiently applied for other data set (we tested on MovieLens and Jester data sets). We also presented different "correction techniques" to improve prediction accuracy: Q-correction use information from unlabeled examples, while neighbor based correction exploits localized information at prediction. We showed that linear combination of various methods can significantly improve the accuracy of the blended solution. We pointed out that various correction techniques can bring major improvement in accuracy when applied to only one method of the linear combination. We showed that they compare favorably with existing methods in terms of prediction accuracy measured by RMSE and time complexity. The experiments prove that the proposed methods are scalable to large recommender systems having hundreds of millions of ratings.

Acknowledgments

We would like to thank the unknown reviewers for their valuable suggestions and critique. The authors are also grateful for the final thorough proofreading to Lester Mackey. Domonkos Tikk was partly supported by the János Bolyai Research Scholarship of the Hungarian Academy of Science.

References

G. Adomavicius and A. Tuzhilin. Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions. *IEEE Transactions on Knowledge and Data Engineering*, 17:734–749, 2005.

P. Baldi and K. Hornik. Neural networks and principal component analysis: Learning from examples without local minima. *Neural Networks*, 2(1):53–58, 1989.

R. Bell, Y. Koren, and Ch. Volinsky. Chasing $1,000,000: How we won the Netflix Progress Prize. *ASA Statistical and Computing Graphics Newsletter*, 18(2):4–12, December 2007a.

R. M. Bell and Y. Koren. Scalable collaborative filtering with jointly derived neighborhood interpolation weights. In *Proc of. ICDM, 7th IEEE Int. Conf. on Data Mining*, pages 43–52, Omaha, Nebraska, USA, 2007a.

R. M. Bell and Y. Koren. Improved neighborhood-based collaborative filtering. In *Proc. of KDD Cup Workshop at SIGKDD-07, 13th ACM Int. Conf. on Knowledge Discovery and Data Mining*, pages 7–14, San Jose, California, USA, 2007b.

R. M. Bell, Y. Koren, and Ch. Volinsky. The BellKor solution to the Netflix Prize. Technical Report, AT&T Labs Research, 2007b. `http://www.netflixprize.com/assets/ProgressPrize2007_KorBell.pdf`.

J. Bennett and S. Lanning. The Netflix Prize. In *Proc. of KDD Cup Workshop at SIGKDD-07, 13th ACM Int. Conf. on Knowledge Discovery and Data Mining*, pages 3–6, San Jose, California, USA, 2007.

J. Bennett, Ch. Eklan, B. Liu, P. Smyth, and D. Tikk. KDD Cup and Workshop 2007. *ACM SIGKDD Explorations Newsletter*, 9(2):51–52, 2007.

J. S. Breese, D. Heckerman, and C. Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In *Proc. of UAI-98, 14th Conf. on Uncertainty in Artificial Intelligence*, pages 43–52, Madison, Wisconsin, USA, 1998.

J. Canny. Collaborative filtering with privacy via factor analysis. In *Proc. of SIGIR-02, 25th ACM SIGIR Conf. on Research and Development in Information Retrieval*, pages 238–245, Tampere, Finland, 2002.

D. DeCoste. Collaborative prediction using ensembles of maximum margin matrix factorizations. In *Proc. of ICML-06, 23rd Int. Conf. on Machine learning*, pages 249–256, Pittsburgh, Pennsylvania, USA, 2006.

N. Delannay and M. Verleysen. Collaborative filtering with interlaced generalized linear models. In *Proc. of ESANN-07, European Symp. on Artificial Neural Networks*, pages 247–252, Bruges, Belgium, 2007.

O. R. Duda, P. E. Hart., and D. G. Stork. *Pattern Classification*. John Wiley and Sons.

D. Goldberg, D. Nichols, B. M. Oki, and D. Terry. Using collaborative filtering to weave an information tapestry. *Communications of the ACM*, 35(12):61–70, 1992.

K. Goldberg, T. Roeder, D. Gupta, and C. Perkins. Eigentaste: a constant time collaborative filtering algorithm. *Information Retrieval*, 4(2):133–151, 2001.

J. L. Herlocker, J. A. Konstan, L. G. Terveen, and J. T. Riedl. Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems*, 22(1):5–53, 2004.

W. Hill, L. Stead, M. Rosenstein, and G. Furnas. Recommending and evaluating choices in a virtual community of use. In *Proc. of CHI-95, ACM Conf. on Human Factors in Computing Systems*, pages 194–201, Denver, Colorado, USA, 1995. ISBN 0-201-84705-1.

T. Hofmann. Latent semantic models for collaborative filtering. *ACM Trans. Inf. Syst.*, 22(1):89–115, 2004.

M. Kurucz, A. A. Benczúr, and K. Csalogány. Methods for large scale SVD with missing values. In *Proc. of KDD Cup Workshop at SIGKDD'07, 13th ACM Int. Conf. on Knowledge Discovery and Data Mining*, pages 7–14, San Jose, California, USA, 2007.

D. D. Lee and H. S. Seung. Learning the parts of objects by nonnegative matrix factorization. *Nature*, 401(6755):788–791, 1999.

A. Paterek. Improving regularized singular value decomposition for collaborative filtering. In *Proc. of KDD Cup Workshop at SIGKDD-07, 13th ACM Int. Conf. on Knowledge Discovery and Data Mining*, pages 39–42, San Jose, California, USA, 2007.

D. M. Pennock, E. Horvitz, S. Lawrence, and C. L. Giles. Collaborative filtering by personality diagnosis: a hybrid memory and model-based approach. In *Proc. of UAI-00, 16th Conf. on Uncertainty in Artificial Intelligence*, pages 473–480, Stanford, California, USA, 2000.

A. M. Rashid, S. K. Lam, G. Karypis, and J. Riedl. ClustKNN: a highly scalable hybrid model-& memory-based CF algorithm. In *Proc. of WebKDD-06, KDD Workshop on Web Mining and Web Usage Analysis, at 12th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining*, Philadelphia, Pennsylvania, USA, 2006.

P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl. Grouplens: an open architecture for collaborative filtering of netnews. In *Proc. of CSCW-94, 4th ACM Conf. on Computer Supported Cooperative Work*, pages 175–186, Chapel Hill, North Carolina, USA, 1994. ISBN 0-89791-689-1.

R. Salakhutdinov and A. Mnih. Probabilistic matrix factorization. In J. C. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems 20*. MIT Press, Cambridge, Massachusetts, USA, 2008.

R. Salakhutdinov, A. Mnih, and G. Hinton. Restricted Boltzmann machines for collaborative filtering. In *Proc. of ICML-07, 24th Int. Conf. on Machine Learning*, pages 791–798, Corvallis, Oregon, USA, 2007.

B. M. Sarwar, G. Karypis, J. A. Konstan, and J. Riedl. Application of dimensionality reduction in recommender system—a case study. In *Proc. of WebKDD-00, Web Mining for E-Commerce Workshop, at 6th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining*, Boston, Massachusetts, USA, 2000.

B. M. Sarwar, G. Karypis, J. A. Konstan, and J. Riedl. Item-based collaborative filtering recommendation algorithms. In *Proc. of WWW-01, 10th Int. Conf. on World Wide Web*, pages 285–295, Hong Kong, 2001.

N. Srebro and T. Jaakkola. Weighted low-rank approximations. In *Proc. of ICDM-03, 20th Int. Conf. on Machine Learning*, pages 720–727, Melbourne, Florida, USA, 2003.

N. Srebro, J. D. M. Rennie, and T. S. Jaakkola. Maximum-margin matrix factorization. *Advances in Neural Information Processing Systems*, 17, 2005.

G. Takács, I. Pilászy, B. Németh, and D. Tikk. On the Gravity recommendation system. In *Proc. of KDD Cup Workshop at SIGKDD-07, 13th ACM Int. Conf. on Knowledge Discovery and Data Mining*, pages 22–30, San Jose, California, USA, 2007.

G. Takács, I. Pilászy, B. Németh, and D. Tikk. A unified approach of factor models and neighbor based methods for large recommender systems. In *Proc. of ICADIWT-08, 1st IEEE Workshop on Recommender Systems and Personalized Retrieval*, pages 186–191, August 2008a.

G. Takács, I. Pilászy, B. Németh, and D. Tikk. Matrix factorization and neighbor based algoritms for the Netflix Prize problem. In *Proc. of RecSys-08, ACM Conf. on Recomender Systems*, pages 267–274, Lausanne, Switzerland, 2008b.

M. G. Vozalis and K. G. Margaritis. Using SVD and demographic data for the enhancement of generalized collaborative filtering. *Information Sciences*, 177(15):3017–3037, 2007.

Journal of Machine Learning Research 10 (2009) 657-698 Submitted 10/07; Revised 6/08; Published 3/09

Nearest Neighbor Clustering: A Baseline Method for Consistent Clustering with Arbitrary Objective Functions

Sébastien Bubeck SEBASTIEN.BUBECK@INRIA.FR
SequeL Project, INRIA Lille
40 avenue Halley,
59650 Villeneuve d'Ascq, France

Ulrike von Luxburg ULRIKE.LUXBURG@TUEBINGEN.MPG.DE
Max Planck Institute for Biological Cybernetics
Spemannstr. 38,
72076 Tübingen, Germany

Editor: Charles Elkan

Abstract

Clustering is often formulated as a discrete optimization problem. The objective is to find, among all partitions of the data set, the best one according to some quality measure. However, in the statistical setting where we assume that the finite data set has been sampled from some underlying space, the goal is not to find the best partition of the given sample, but to approximate the true partition of the underlying space. We argue that the discrete optimization approach usually does not achieve this goal, and instead can lead to inconsistency. We construct examples which provably have this behavior. As in the case of supervised learning, the cure is to restrict the size of the function classes under consideration. For appropriate "small" function classes we can prove very general consistency theorems for clustering optimization schemes. As one particular algorithm for clustering with a restricted function space we introduce "nearest neighbor clustering". Similar to the k-nearest neighbor classifier in supervised learning, this algorithm can be seen as a general baseline algorithm to minimize arbitrary clustering objective functions. We prove that it is statistically consistent for all commonly used clustering objective functions.

Keywords: clustering, minimizing objective functions, consistency

1. Introduction

Clustering is the problem of discovering "meaningful" groups in given data. In practice, the most common approach to clustering is to define a clustering quality function Q_n, and then construct an algorithm which is able to minimize (or maximize) Q_n. There exists a huge variety of clustering quality functions: the K-means objective function based on the distance of the data points to the cluster centers, graph cut based objective functions such as ratio cut or normalized cut, or various criteria based on some function of the within- and between-cluster similarities. Once a particular clustering quality function Q_n has been selected, the objective of clustering is stated as a discrete optimization problem. Given a data set $\mathcal{X}_n = \{X_1, \dots, X_n\}$ and a clustering quality function Q_n, the ideal clustering algorithm should take into account all possible partitions of the data set and output the one that minimizes Q_n. The implicit understanding is that the "best" clustering can be any partition out of the set of all possible partitions of the data set. The practical challenge is then to construct an algorithm which is able to explicitly compute this "best" clustering by solving an optimization problem. We will call this approach the "discrete optimization approach to clustering".

Now let us look at clustering from the perspective of statistical learning theory. Here we assume that the finite data set has been sampled from an underlying data space $\mathcal{X}$ according to some probability measure $\mathbb{P}$. The ultimate goal in this setting is not to discover the best possible partition of the data set $\mathcal{X}_n$, but to learn the "true clustering" of the underlying space $\mathcal{X}$. While it is not obvious how this "true clustering" should be defined in a general setting (cf. von Luxburg and Ben-David, 2005), in an approach based on quality functions this is straightforward. We choose a clustering quality function Q on the set of partitions of the entire data space $\mathcal{X}$, and define the true clustering f^* to be the partition of $\mathcal{X}$ which minimizes Q. In a finite sample setting, the goal is now to approximate this true clustering as well as possible. To this end, we define an empirical quality function Q_n which can be evaluated based on the finite sample only, and construct the empirical clustering f_n as the minimizer of Q_n. In this setting, a very important property of a clustering algorithm is consistency: we require that $Q(f_n)$ converges to $Q(f^*)$ when $n \to \infty$. This strongly reminds of the standard approach in supervised classification, the empirical risk minimization approach. For this approach, the most important insight of statistical learning theory is that in order to be consistent, learning algorithms have to choose their functions from some "small" function space only. There are many ways how the size of a function space can be quantified. One of the easiest ways is to use shattering coefficients $s(\mathcal{F}, n)$ (see Section 2 for details). A typical result in statistical learning theory is that a necessary condition for consistency is $\mathbb{E} \log s(\mathcal{F}, n)/n \to 0$ (cf. Theorem 2.3 in Vapnik, 1995, Section 12.4 of Devroye et al., 1996). That is, the "number of functions" $s(\mathcal{F}, n)$ in $\mathcal{F}$ must not grow exponentially in n, otherwise one cannot guarantee for consistency.

Stated like this, it becomes apparent that the two viewpoints described above are not compatible with each other. While the discrete optimization approach on any given sample attempts to find the best of all (exponentially many) partitions, statistical learning theory suggests to restrict the set of candidate partitions to have sub-exponential size. So from the statistical learning theory perspective, an algorithm which is considered ideal in the discrete optimization setting will not produce partitions which converge to the true clustering of the data space.

In practice, for most clustering objective functions and many data sets the discrete optimization approach cannot be performed perfectly as the corresponding optimization problem is NP hard. Instead, people resort to heuristics and accept suboptimal solutions. One approach is to use local optimization procedures potentially ending in local minima only. This is what happens in the K-means algorithm: even though the K-means problem for fixed K and fixed dimension is not NP hard, it is still too hard for being solved globally in practice. Another approach is to construct a relaxation of the original problem which can be solved efficiently (spectral clustering is an example for this). For such heuristics, in general one cannot guarantee how close the heuristic solution is to the finite sample optimum. This situation is clearly unsatisfactory: in general, we neither have guarantees on the finite sample behavior of the algorithm, nor on its statistical consistency in the limit.

The following alternative approach looks much more promising. Instead of attempting to solve the discrete optimization problem over the set of all partitions, and then resorting to relaxations due to the hardness of this problem, we turn the tables. Directly from the outset, we only consider candidate partitions in some restricted class $\mathcal{F}_n$ containing only polynomially many functions. Then the discrete optimization problem of minimizing Q_n over $\mathcal{F}_n$ is not NP hard—formally it can be solved in polynomially many steps by trying all candidates in $\mathcal{F}_n$. From a theoretical point of view this approach has the advantage that the resulting clustering algorithm has the potential of being consistent. In addition, this approach also has advantages in practice: rather than dealing with uncontrolled relaxations of the original problem, we restrict the function class to some small subset $\mathcal{F}_n$ of "reasonable" partitions. Within this subset, we then have complete control over the solution of the optimization problem and can find the global optimum. Put another way, one can also interpret this approach as some controlled way to approximate a solution of the NP hard optimization problem on the finite sample, with the positive side effect of obeying the rules of statistical learning theory.

This is the approach we want to describe in this paper. In Section 2 we will first construct an example which demonstrates the inconsistency in the discrete optimization approach. Then we will state a general theorem which gives sufficient conditions for clustering optimization schemes to be consistent. We will see that the key point is to control the size of the function classes the clustering are selected from. In Section 3 we will then introduce an algorithm which is able to work with such a restricted function class. This algorithm is called nearest neighbor clustering, and in some sense it can be seen as a clustering-analogue to the well-known nearest neighbor classifier for classification. We prove that nearest neighbor clustering is consistent under minimal assumptions on the clustering quality functions Q_n and Q. Then we will apply nearest neighbor clustering to a large variety of clustering objective functions, such as the K-means objective function, normalized cut and ratio cut, the modularity objective function, or functions based on within-between cluster similarity ratios. For all these functions we will verify the consistency of nearest neighbor clustering in Section 4. Discussion of our results, also in the context of the related literature, can be found in Sections 5 and 6. The proofs of all our results are deferred to the appendix, as some of them are rather technical.

2. General (In)Consistency Results

In the rest of this paper, we consider a space $\mathcal{X}$ which is endowed with a probability measure $\mathbb{P}$. The task is to construct a clustering $f : \mathcal{X} \to \{1, \dots, K\}$ on this space, where K denotes the number of clusters to construct. We denote the space of all $\mathbb{P}$-measurable functions from $\mathcal{X}$ to $\{1,...,K\}$ by $\mathcal{H}$. Let $Q : \mathcal{H} \to \mathbb{R}^+$ denote a clustering quality function: for each clustering, it tells us "how good" a given clustering is. This quality function will usually depend on the probability measure $\mathbb{P}$. An optimal clustering, according to this objective function, is a clustering f^* which satisfies

$$f^* \in \underset{f \in \mathcal{F}}{\operatorname{argmin}}\, Q(f).$$

where $\mathcal{F} \subseteq \mathcal{H}$ is a fixed set of candidate clusterings. Now assume that $\mathbb{P}$ is unknown, but that we are given a finite sample $X_1, ..., X_n \in \mathcal{X}$ which has been drawn i.i.d according to $\mathbb{P}$. Our goal is to use this sample to construct a clustering f_n which "approximates" an optimal clustering f^*. To this end, assume that $Q_n : \mathcal{H} \to \mathbb{R}^+$ is an estimator of Q which can be computed based on the finite sample only (that is, it does not involve any function evaluations $f(x)$ for $x \notin \{X_1, ..., X_n\}$). We then consider the clustering

$$f_n \in \underset{f \in \mathcal{F}_n}{\operatorname{argmin}}\, Q_n(f).$$

Here, $\mathcal{F}_n$ is a subset of $\mathcal{H}$, which might or might not be different from $\mathcal{F}$. The general question we are concerned with in this paper is the question of consistency: under which conditions do we know that $Q(f_n) \to Q(f^*)$?

Note that to avoid technical overload we will assume throughout this paper that all the minima (as in the definitions of f^* and f_n) exist and can be attained. If this is not the case, one can always go over to statements about functions which are ε-close to the corresponding infimum. We also will not discuss issues of measurability in this paper (readers interested in measurability issues for empirical processes are referred to Section 1 of van der Vaart and Wellner, 1996).

2.1 Inconsistency example

In the introduction we suggested that as in the supervised case, the size of the function class $\mathcal{F}_n$ might be the key to consistency of clustering. In particular, we argued that optimizing over the space of all measurable functions might lead to inconsistency. First of all, we would like to prove this statement by providing a example. This example will show that if we optimize a clustering objective function over a too large class of functions, the resulting clusterings are not consistent.

Example 1 (Inconsistency in general) *As data space we choose $\mathcal{X} = [0,1] \cup [2,3]$, and as probability measure $\mathbb{P}$ we simply use the normalized Lebesgue measure λ on $\mathcal{X}$. We define the following similarity function between points in $\mathcal{X}$:*

$$s(x,y) = \begin{cases} 1 & \text{if } x \in [0,1], y \in [0,1] \\ 1 & \text{if } x \in [2,3],\ y \in [2,3] \\ 0 & \text{otherwise.} \end{cases}$$

For simplicity, we consider the case where we want to construct $K = 2$ clusters called C_1 and C_2. Given a clustering function $f : \mathcal{X} \to \{0,1\}$ we call the clusters $C_1 := \{x \in \mathcal{X} \mid f(x) = 0\}$ and $C_2 := \{x \in \mathcal{X} \mid f(x) = 1\}$. As clustering quality function Q we use the between-cluster similarity (equivalent to cut*, see Section* 4.2 *for details):*

$$Q(f) = \int_{x \in C_1} \int_{y \in C_2} s(X,Y)\, d\mathbb{P}(X)\, d\mathbb{P}(Y).$$

As an estimator of Q we will use the function Q_n where the integrals are replaced by sums over the data points:

$$Q_n(f) = \frac{1}{n(n-1)} \sum_{i \in C_1} \sum_{j \in C_2} s(X_i, X_j).$$

As set $\mathcal{F}$ we choose the set of all measurable partitions on $\mathcal{X}$ (note that the same example also holds true when we only look at the set $\mathcal{F}$ of measurable partitions such that both clusters have a minimal mass ε for some $\varepsilon > 0$). For all $n \in \mathbb{N}$ we set $\mathcal{F}_n = \mathcal{F}$. Let $X_1, ..., X_n \in \mathcal{X}$ be our training data. Now define the functions

$$f^*(x) = \begin{cases} 0 & \text{if } x \in [0,1] \\ 1 & \text{if } x \in [2,3] \end{cases} \qquad \text{and} \qquad f_n(x) = \begin{cases} 0 & \text{if } x \in \{X_1, ..., X_n\} \cap [0,1] \\ 1 & \text{if } x \in [2,3] \\ 0 & \text{if } x \in [0, 0.5] \setminus \{X_1, ..., X_n\} \\ 1 & \text{if } x \in [0.5, 1] \setminus \{X_1, ..., X_n\} \end{cases}.$$

It is obvious that $Q(f^*) = 0$ and $Q_n(f_n) = 0$. As both Q and Q_n are non-negative, we can conclude $f^* \in \operatorname{argmin}_{f \in \mathcal{F}} Q(f)$ and $f_n \in \operatorname{argmin}_{f \in \mathcal{F}} Q_n(f)$. It is also straightforward to compute $Q(f_n) = 1/16$ (independently of n). Hence, we have inconsistency: $1/16 = Q(f_n) \not\to Q(f^*) = 0$.

Note that the example is set up in a rather natural way. The data space contains two perfect clusters ($[0,1]$ and $[2,3]$) which are separated by a large margin. The similarity function is the ideal similarity function for this case, giving similarity 1 to points which are in the same cluster, and similarity 0 to points in different clusters. The function f^* is the correct clustering. The empirical clustering f_n, if restricted to the data points, reflects the correct clustering. It is just the "extension" of the empirical clustering to non-training points which leads to the inconsistency of f_n. Intuitively, the reason why this can happen is clear: the function space $\mathcal{F}$ does not exclude the unsuitable extension chosen in the example, the function overfits. This can happen because the function class is too large.

2.2 Main Result

Now we would like to present our first main theorem. It shows that if f_n is only picked out of a "small" function class $\mathcal{F}_n$, then we can guarantee consistency of clustering. Before stating the theorem we would like to recall the definition of the shattering coefficient in a K-class setting. For a function class $\mathcal{F} : \mathcal{X} \to \{1,\ldots,K\}$ the shattering coefficient of size n is defined as

$$s(\mathcal{F},n) = \max_{x_1,\ldots,x_n \in \mathcal{X}} |\{(f(x_1),\ldots,f(x_n)) \mid f \in \mathcal{F}\}|.$$

To state our theorem, we will also require a pseudo-distance d between functions. A pseudo-distance is a dissimilarity function $d : \mathcal{X} \times \mathcal{X} \to \mathbb{R}^+$ which is symmetric, satisfies the triangle inequality and the condition $f = g \implies d(f,g) = 0$, but not necessarily the condition $d(f,g) = 0 \implies f = g$. For distances between sets of functions we use the standard convention $d(\mathcal{F},\mathcal{G}) = \inf_{f\in\mathcal{F},g\in\mathcal{G}} d(f,g)$. Our theorem is as follows:

Theorem 1 (Consistency of a clustering optimizing scheme) *Let $(X_n)_{n\in\mathbb{N}}$ be a sequence of random variables which have been drawn i.i.d. according to some probability measure $\mathbb{P}$ on some set $\mathcal{X}$. Let $\mathcal{F}_n := \mathcal{F}_n(X_1,\ldots,X_n) \subset \mathcal{H}$ be a sequence of function spaces, and $\mathcal{F} \subset \mathcal{H}$. Let $d : \mathcal{H} \times \mathcal{H} \to \mathbb{R}$ be a pseudo-distance defined on $\mathcal{H}$. Let $Q : \mathcal{H} \to \mathbb{R}^+$ be a clustering quality function, and $Q_n : \mathcal{H} \to \mathbb{R}^+$ an estimator of this function which can be computed based on the finite sample only. Finally let*

$$\widetilde{\mathcal{F}_n} := \bigcup_{X_1,\ldots,X_n \in \mathbb{R}^d} \mathcal{F}_n.$$

Define the true and the empirical clusterings as

$$f^* \in \operatorname*{argmin}_{f\in\mathcal{F}} Q(f),$$
$$f_n \in \operatorname*{argmin}_{f\in\mathcal{F}_n} Q_n(f).$$

Assume that the following conditions are satisfied:

1. *$Q_n(f)$ is a consistent estimator of $Q(f)$ which converges sufficiently fast for all $f \in \widetilde{\mathcal{F}_n}$:*

$$\forall \varepsilon > 0,\ \ s(\widetilde{\mathcal{F}_n},2n) \sup_{f\in\widetilde{\mathcal{F}_n}} \mathbb{P}(|Q_n(f) - Q(f)| > \varepsilon) \to 0,$$

2. *$\mathcal{F}_n$ approximates $\mathcal{F}$ in the following sense:*

$$\begin{aligned}(i)&\quad \forall f \in \mathcal{F}, d(f,\mathcal{F}_n) \to 0 \textit{ in probability,}\\ (ii)&\quad \mathbb{P}(f_n \notin \mathcal{F}) \to 0.\end{aligned}$$

3. *Q is uniformly continuous with respect to the pseudo-distance d between $\mathcal{F}$ and $\widetilde{\mathcal{F}_n}$:*

$$\forall \varepsilon > 0\ \exists \delta(\varepsilon) > 0 \textit{ such that } \forall f \in \mathcal{F}\ \forall g \in \widetilde{\mathcal{F}_n}:\ d(f,g) \le \delta(\varepsilon) \Rightarrow |Q(f) - Q(g)| \le \varepsilon.$$

Then the optimization scheme is weakly consistent, that is $Q(f_n) \to Q(f^*)$ in probability.

This theorem states sufficient conditions for consistent clustering schemes. In the context of the standard statistical learning theory, the three conditions in the theorem are rather natural. The first condition mainly takes care of the estimation error. Implicitly, it restricts the size of the function class $\mathcal{F}_n$ by incorporating the shattering coefficient. We decided to state condition 1 in this rather abstract way to make the theorem as general as possible. We will see later how it can be used in concrete applications. Of course, there are many more ways to specify the size of function classes, and many of them might lead to better bounds in the end. However, in this paper we are not so much concerned with obtaining the sharpest bounds, but we want to demonstrate the general concept (as the reader can see in appendix, the proofs are already long enough using simple shattering numbers). The second condition in the theorem takes care of the approximation error. Intuitively it is clear that if we want to approximate solutions in $\mathcal{F}$, eventually $\mathcal{F}_n$ needs to be "close" to $\mathcal{F}$. The third condition establishes a relation between the quality function Q and the distance function d: if two clusterings f and g are close with respect to d, then their quality values $Q(f)$ and $Q(g)$ are close, too. We need this property to be able to conclude from "closeness" as in Condition 2 to "closeness" of the clustering quality values.

Finally, we would like to point out a few technical treats. First of all, note that the function class $\mathcal{F}_n$ is allowed to be data dependent. Secondly, as opposed to most results in empirical risk minimization we do not assume that Q_n is an unbiased estimator of Q (that is, we allow $\mathbb{E}Q_n \neq Q$), nor does Q need to be "an expectation" (that is, of the form $Q(f) = \mathbb{E}(\Omega(f,X))$ for some Ω). Both facts make the proof more technical, as many of the standard tools (symmetrization, concentration inequalities) become harder to apply. However, this is necessary since in the context of clustering biased estimators pop up all over the place. We will see that many of the popular clustering objective functions lead to biased estimators.

3. Nearest Neighbor Clustering—General Theory

The theorem presented in the last section shows sufficient conditions under which clustering can be performed consistently. Now we want to present a generic algorithm which can be used to minimize arbitrary clustering objective functions. With help of Theorem 1 we can then prove the consistency of its results for a large variety of clustering objective functions.

We have seen that the key to obtain consistent clustering schemes is to work with an appropriate function class. But of course, given quality functions Q and Q_n, the question is how such a function space can be constructed in practice. Essentially, three requirements have to be satisfied:

- The function space $\mathcal{F}_n$ has to be "small". Ideally, it should only contain polynomially many functions.

- The function space $\mathcal{F}_n$ should be "rich enough". In the limit $n \to \infty$, we would like to be able to approximate any (reasonable) measurable function.
- We need to be able to solve the optimization problem $\text{argmin}_{f \in \mathcal{F}_n} Q_n(f)$. This sounds trivial at first glance, but in practice is far from easy.

One rather straightforward way to achieve all requirements is to use a function space of piecewise constant functions. Given a partitioning of the data space in small cells, we only look at clusterings which are constant on each cell (that is, the clustering never splits a cell). If we make sure that the number of cells is only of the order $\log(n)$, then we know that the number of clusterings is at most $K^{\log(n)} = n^{\log(K)}$, which is polynomial in n. In the following we will introduce a data-dependent random partition of the space which turns out to be very convenient.

3.1 Nearest Neighbor Clustering—The Algorithm

We will construct a function class $\mathcal{F}_n$ as follows. Given a finite sample $X_1, \ldots, X_n \in \mathbb{R}^d$, the number K of clusters to construct, and a number $m \in \mathbb{N}$ with $K \leq m \ll n$, randomly pick a subset of m "seed points" $X_{s_1}, \ldots, X_{s_m}$. Assign all other data points to their closest seed points, that is for all $j = 1, \ldots, m$ define the set Z_j as the subset of data points whose nearest seed point is X_{s_j}. In other words, the sets $Z_1, \ldots, Z_m$ are the Voronoi cells induced by the seeds $X_{s_1}, \ldots, X_{s_m}$. Then consider all partitions of $\mathcal{X}_n$ which are constant on all the sets $Z_1, ..., Z_m$. More formally, for given seeds we define the set $\mathcal{F}_n$ as the set of all functions

$$\mathcal{F}_n := \{f : \mathcal{X} \to \{1, \ldots, K\} \mid \forall\, j = 1, \ldots, m : \forall z, z' \in Z_j : f(z) = f(z')\}.$$

Obviously, the function class $\mathcal{F}_n$ contains K^m functions, which is polynomial in n if the number m of seeds satisfies $m \in O(\log n)$. Given $\mathcal{F}_n$, the most simple polynomial-time optimization algorithm is then to evaluate $Q_n(f)$ for all $f \in \mathcal{F}_n$ and choose the solution $f_n = \text{argmin}_{f \in \mathcal{F}_n} Q_n(f)$. We call the resulting clustering the *nearest neighbor clustering* and denote it by $\text{NNC}(Q_n)$. The entire algorithm is summarized in Figure 1. We have already published results on the empirical performance

```
**Nearest Neighbor Clustering NNC($Q_n$), naive implementation**

**Parameters:** number $K$ of clusters to construct, number $m \in \mathbb{N}$ of seed points
to use (with $K \leq m \ll n$), clustering quality function $Q_n$

**Input:** data set $\mathcal{X}_n = \{X_1, \ldots, X_n\}$, distances $d_{ij} = d(X_i, X_j)$

    • Subsample $m$ seed points from the data points, without replacement.
    • Build the Voronoi decomposition $Z_1, \ldots, Z_m$ of $\mathcal{X}_n$ based on the distances
      $d_{ij}$ using the seed points as centers
    • Define $\mathcal{F}_n := \{f : \mathcal{X}_n \to \{1, \ldots, K\} \mid f$ constant on all cells $Z_j\}$
    • For all $f \in \mathcal{F}_n$ evaluate $Q_n(f)$.

**Output:** $f_n := \text{argmin}_{f \in \mathcal{F}_n} Q_n(f)$
```

Figure 1: Nearest neighbor clustering for a general clustering objective function Q_n.

of the algorithm in von Luxburg et al. (2008), and more results can be found in Section 3 of Jegelka (2007). We have found that on finite samples, the algorithm performs surprisingly well in terms of quality function: using $m = \log n$ seed points, the objective function values obtained at the solutions are comparable to these of K-means or spectral clustering, respectively. Moreover, there exist efficient ways to compute f_n using branch and bound methods. Using these methods, the running time of nearest neighbor clustering using $m = \log n$ seeds is roughly comparable to the one of the other clustering algorithms. See von Luxburg et al. (2008) and Jegelka (2007) for details on the experimental results.

3.2 Consistency of Nearest Neighbor Clustering (General Statement)

Now we want to prove that nearest neighbor clustering is consistent. We will see that even though we can rely on Theorem 1, the consistency proof for nearest neighbor clustering does not come for free. Let $f : \mathcal{X} \to \{1,\dots,K\}$ be a clustering function. In the following, we will often use the notation f_k for the indicator function of the k-th cluster:

$$f_k(x) := \mathbb{1}_{f(x)=k}.$$

This is a slight abuse of notation, as we already reserved the notation f_n for the minimizer of the empirical quality function. However, from the context it will always be clear whether we will refer to f_n or f_k, respectively, as we will not mix up the letters n (for the sample size) and k (a cluster index).

As distance function between two clusterings we use the 0-1-loss

$$d(f,g) := \mathbb{P}(f(X) \neq g(X) | X_1,\dots,X_n).$$

Here the conditioning is needed for the cases where the functions f or g are data dependent. Note that in clustering, people often consider a variant of this distance which is independent with respect to the choice of labels, that is they choose $\tilde{d}(f,g) := \min_\pi P(f(X) \neq \pi(g(X)) | X_1,\dots,X_n)$, where π runs over all permutations of the set $\{1,\dots,K\}$. However, we will see that for our purposes it does not hurt to use the overly sensitive 0-1 distance instead. The main reason is that at the end of the day, we only want to compare functions based on their quality values, which do not change under label permutations. In general, the theorems and proofs could also be written in terms of $\tilde{d}$. For better readability, we decided to stick to the standard 0-1 distance, though.

We will see below that in many cases, even in the limit case one would like to use a function space $\mathcal{F}$ which is a proper subset of $\mathcal{H}$. For example, one could only be interested in clusterings where all clusters have a certain minimal size, or where the functions satisfy certain regularity constraints. In order to be able to deal with such general function spaces, we will introduce a tool to restrict function classes to functions satisfying certain conditions. To this end, let

$$\Phi : \mathcal{H} \to \mathbb{R}^+$$

be a functional which quantifies certain aspects of a clustering. In most cases, we will use functionals Φ which operate on the individual cluster indicator functions f_k. For example, $\Phi(f_k)$ could measure the size of cluster k, or the smoothness of the cluster boundary. The function class $\mathcal{F}$ will then be defined as

$$\mathcal{F} = \{f \in \mathcal{H} \mid \Phi(f_k) > a \text{ for all } k = 1,\dots,K\},$$

where $a \geq 0$ is a constant. In general, the functional Φ can be used to encode our intuition about "what a cluster is". Note that this setup also includes the general case of $\mathcal{F} = \mathcal{H}$, that is the case where we do not want to make any further restrictions on $\mathcal{F}$, for example by setting $\Phi(f_k) \equiv 1$, $a \equiv 0$. As it is the case for Q, we will usually not be able to compute Φ on a finite sample only. Hence we also introduce an empirical counterpart Φ_n which will be used in the finite sample case.

The following theorem will state sufficient conditions for the consistency of nearest neighbor clustering. For simplicity we state the theorem for the case $\mathcal{X} = \mathbb{R}^d$, but the proofs can also be carried over to more general spaces. Also, note that we only state the theorem for the case $d \geq 2$; in case $d = 1$ the theorem holds as well, but the formulas look a bit different.

Theorem 2 (Consistency of nearest neighbor clustering) *Let $\mathcal{X} = \mathbb{R}^d$, $d \geq 2$, $Q : \mathcal{H} \to \mathbb{R}^+$ be a clustering quality function, and $Q_n : \mathcal{H} \to \mathbb{R}^+$ an estimator of this function which can be computed based on the finite sample only. Similarly, let $\Phi : \mathcal{H} \to \mathbb{R}^+$, and $\Phi_n : \mathcal{H} \to \mathbb{R}^+$ an estimator of this function. Let $a > 0$ and $(a_n)_{n \in \mathbb{N}}$ be such that $a_n > a$ and $a_n \to a$. Let $m = m(n) \leq n \in \mathbb{N}$. Finally, denote $d(f,g)$ the 0-1-loss, and let $NN_m(x)$ be the nearest neighbor of x among $X_1, \ldots, X_m$ according to the Euclidean distance. Define the function spaces*

$$\mathcal{F} := \{f : \mathbb{R}^d \to \{1, \ldots, K\} \mid f \text{ continuous a.e. and } \forall k \in \{1, \ldots, K\} \ \Phi(f_k) > a\}$$
$$\mathcal{F}_n := \{f : \mathbb{R}^d \to \{1, \ldots, K\} \mid f(x) = f(NN_m(x)) \text{ and } \forall k \in \{1, \ldots, K\} \ \Phi_n(f_k) > a_n\}$$
$$\widetilde{\mathcal{F}_n} := \bigcup_{X_1, \ldots, X_n \in \mathbb{R}^d} \mathcal{F}_n$$
$$\widehat{\mathcal{F}_n} := \{f : \mathbb{R}^d \to \{1, \ldots, K\} \mid \exists \text{ Voronoi partition of } m \text{ cells: } f \text{ constant on all cells}\}.$$

Assume that the following conditions are satisfied:

1. *$Q_n(f)$ is a consistent estimator of $Q(f)$ which converges sufficiently fast for all $f \in \widetilde{\mathcal{F}_n}$:*
$$\forall \varepsilon > 0, K^m (2n)^{(d+1)m^2} \sup_{f \in \widetilde{\mathcal{F}_n}} \mathbb{P}(|Q_n(f) - Q(f)| > \varepsilon) \to 0,$$
2. *$\Phi_n(f_k)$ is a consistent estimator of $\Phi(f_k)$ which converges sufficiently fast for all $f \in \widehat{\mathcal{F}_n}$:*
$$\forall \varepsilon > 0, K^m (2n)^{(d+1)m^2} \sup_{f \in \widehat{\mathcal{F}_n}} \mathbb{P}(|\Phi_n(f_k) - \Phi(f_k)| > \varepsilon) \to 0,$$
3. *Q is uniformly continuous with respect to the pseudo-distance $d(f,g)$ between $\mathcal{F}$ and $\widetilde{\mathcal{F}_n}$, as defined in Condition (3) of Theorem 1,*
4. *$\Phi_k(f) := \Phi(f_k)$ is uniformly continuous with respect to the pseudo-distance $d(f,g)$ between $\mathcal{F}$ and $\widehat{\mathcal{F}_n}$, as defined in Condition (3) of Theorem 1,*
5. *a_n decreases slowly enough to a:*
$$K^m (2n)^{(d+1)m^2} \sup_{g \in \widehat{\mathcal{F}_n}, k} \mathbb{P}(\Phi_n(g_k) - \Phi(g_k) \geq a_n - a) \to 0,$$

6. $m \to \infty$.

Then nearest neighbor clustering based on m seed points using quality function Q_n is weakly consistent, that is for $f_n \in \operatorname{argmin}_{f \in \mathcal{F}_n} Q_n(f)$ and $f^* \in \operatorname{argmin}_{f \in \mathcal{F}} Q(f)$ we have $Q(f_n) \to Q(f^*)$ in probability.

This theorem is still rather abstract, but pretty powerful. In the following we will demonstrate this by applying it to many concrete clustering objective functions. To define our objective functions, we will from now on adopt the convention $0/0 = 0$.

4. Nearest Neighbor Clustering with Popular Clustering Objective Functions

In this section we want to study the consistency of nearest neighbor clustering when applied to particular objective functions. For simplicity we assume in this section that $\mathcal{X} = \mathbb{R}^d$.

4.1 NNC Using the K-means Objective Function

The K-means objective function is the within-cluster sum of squared distances, called WSS for short. To define it properly, for a given clustering function $f : \mathbb{R}^d \to \{1, \dots, K\}$ we introduce the following quantities:

$$\mathrm{WSS}_n(f) := \frac{1}{n} \sum_{i=1}^{n} \sum_{k=1}^{K} f_k(X_i) \|X_i - c_{k,n}\|^2 \qquad \text{where}$$

$$c_{k,n} := \frac{1}{n_k} \frac{1}{n} \sum_{i=1}^{n} f_k(X_i) X_i \qquad \text{and} \qquad n_k := \frac{1}{n} \sum_{i=1}^{k} f_k(X_i)$$

$$\mathrm{WSS}(f) := \mathbb{E} \sum_{k=1}^{K} f_k(X) \|X - c_k\|^2 \qquad \text{where} \qquad c_k := \frac{\mathbb{E} f_k(X) X}{\mathbb{E} f_k(X)}.$$

Here, WSS_n plays the role of Q_n and WSS the role of Q. Let us point out some important facts. First the empirical quality function is not an unbiased estimator of the true one, that is $\mathbb{E}\mathrm{WSS}_n \neq \mathrm{WSS}$ and $\mathbb{E} c_{k,n} \neq c_k$ (note that in the standard treatment of K-means this can be achieved, but not on arbitrary function classes, see below for some discussion). However, at least we have $\mathbb{E} n_k = \mathbb{E} f_k(X)$ and $\mathbb{E} \frac{1}{n} \sum_{i=1}^{n} f_k(X_i) X_i = \mathbb{E} f_k(X) X$. Moreover, one should remark that if we define $\mathrm{WSS}(\cdot, \mathbb{P}) := \mathrm{WSS}$ then $\mathrm{WSS}_n = \mathrm{WSS}(\cdot, \mathbb{P}_n)$ where $\mathbb{P}_n$ is the empirical distribution.

Secondly, our setup for proving the consistency of nearest neighbor clustering with the WSS objective function is considerably more complicated than proving the consistency of the global minimizer of the K-means algorithm (e.g., Pollard, 1981). The reason is that for the K-means algorithm one can use a very helpful equivalence which does not hold for nearest neighbor clustering. Namely, if one considers the minimizer of WSS_n in the space of *all possible partitions*, then one can see that the clustering constructed by this minimizer always builds a Voronoi partition with K cells; the same holds in the limit case. In particular, given the cluster centers $c_{k,n}$ one can reconstruct the whole clustering by assigning each data point to the closest cluster center. As a consequence, to prove the convergence of K-means algorithms one usually studies the convergence of the empirical cluster centers $c_{k,n}$ to the true centers c_k. However, in our case this whole chain of arguments breaks down. The reason is that the clusters chosen by nearest neighbor clustering *from the set $\mathcal{F}_n$* are not necessarily Voronoi cells, they do not even need to be convex (all clusters are composed by small Voronoi cells, but the union of "small" Voronoi cells is not a "large" Voronoi cell). Also, it is not the case that each data point is assigned to the cluster corresponding to the closest cluster center. It may very well happen that a point x belongs to cluster C_i, but is closer to the center of another cluster C_j than to the center of its own cluster C_i. Consequently, we cannot reconstruct the nearest neighbor clustering from the centers of the clusters. This means that we cannot go over to the convergence of centers, which makes our proof considerably more involved than the one of the standard K-means case.

Due to these technical problems, it will be of advantage to only consider clusters which have a certain minimal size (otherwise, the cluster quality function WSS is not uniformly continuous). To achieve this, we use the functionals

$$\Phi_{\text{WSS}}(f_k) := \mathbb{E} f_k(X), \qquad\qquad \Phi_{\text{WSS}_n}(f_k) := n_k(f).$$

and will only consider clusterings where $\Phi(f_k) \geq a > 0$. In practice, this can be interpreted as a simple means to avoid empty clusters. The constant a can be chosen so small that its only effect is to make sure that each cluster contains at least one data point. The corresponding function spaces are

$$\mathcal{F} := \{f : \mathbb{R}^d \to \{1,\dots,K\} \mid f \text{ continuous a.e. and } \forall k \in \{1,\dots,K\}\ \ \Phi_{\text{WSS}}(f_k) > a\}$$
$$\mathcal{F}_n := \{f : \mathbb{R}^d \to \{1,\dots,K\} \mid f(x) = f(NN_m(x)) \text{ and } \forall k \in \{1,\dots,K\}\ \ \Phi_{\text{WSS}_n}(f_k) > a_n\}$$

Moreover, for technical convenience we restrict our attention to probability measures which have a bounded support inside some large ball, that is which satisfy $\operatorname{supp}\mathbb{P} \subset B(0,A)$ for some constant $A > 0$. It is likely that our results also hold in the general case, but the proof would get even more complicated. With the notation of Theorem 2 we have:

Theorem 3 (Consistency of NNC(WSS)**)** *Assume that $a_n > a, a_n \to a, m \to \infty$ and*

$$\frac{m^2 \log n}{n(a-a_n)^2} \to 0.$$

Then for all probability measures on $\mathbb{R}^d$ with bounded support, nearest neighbor clustering with WSS *is consistent, that is if $n \to \infty$ then* $\text{WSS}(f_n) \to \text{WSS}(f^*)$ *in probability.*

This theorem looks very nice and simple. The conditions on a_n and m are easily satisfied as soon as these quantities do not converge too fast. For example, if we define

$$a_n = a + \frac{1}{\log n} \quad \text{and} \quad m = \log n$$

then

$$\frac{m^2 \log n}{n(a_n-a)^2} = \frac{(\log n)^5}{n} \to 0.$$

Moreover, it is straightforward to see from the proofs that this theorem is still valid if we consider the objective functions WSS_n and WSS with $\|\cdot\|$ instead of $\|\cdot\|^2$. It also holds for any other norm, such as the p-norms $\|\cdot\|_p$. However, it does not necessarily hold for powers of norms (in this sense, the squared Euclidean norm is an exception). The proof shows that the most crucial property is

$$\|X_i - c_{k,n}\| - \|X_i - c_k\| \leq \mathrm{const} \cdot \|c_{k,n} - c_k\|.$$

This is straightforward if the triangle inequality holds, but might not be possible for general powers of norms.

By looking more carefully at our proofs one can state the following rate of convergence:

Theorem 4 (Convergence Rate for $\mathrm{NNC}(\mathrm{WSS})$**)** *Assume that* $\operatorname{supp}\mathbb{P} \subset B(0,A)$ *for some constant* $A > 0$ *and that* $n(a_n - a)^2 \to \infty$*. Let* $\varepsilon \leq 1$ *and* $a^* := \inf_k \mathbb{E} f_k^*(X) - a > 0$*. Then there exists :*

$$\begin{aligned} &N = N((a_n), a^*) \in \mathbb{N}, \\ &C_1 = C_1(a, a^*, \varepsilon, K, A) > 0, && C_2 = C_2(a, a^*, \varepsilon, A, f^*, \mathbb{P}) > 0, \\ &C_3 = C_3(a, d, \varepsilon, K, A) > 0, && C_4 = C_4(a, d, A) > 0 \end{aligned}$$

such that for $n \geq N$ *the following holds true:*

$$\begin{aligned} &\mathbb{P}(|\mathrm{WSS}(f_n) - \mathrm{WSS}(f^*)| \geq \varepsilon) \\ &\leq C_1 e^{-C_2 m} + K^{m+1} (2n)^{(d+1)m^2} \left(C_3 e^{-C_4 \varepsilon^2 n} + 8K e^{-\frac{n(a_n - a)^2}{8}} \right). \end{aligned}$$

At first glance, it seems very tempting to try to use the Borel-Cantelli lemma to transform the weak consistency into strong consistency. However, we do not have an explicit functional form of dependency of C_2 on ε. The main reason is that in Lemma 11 (Appendix) the constant $b(\varepsilon)$ will be defined only implicitly. If one would like to prove strong consistency of nearest neighbor clustering with WSS one would have to get an explicit form of $b(\varepsilon)$ in Lemma 11.

For a general discussion relating the consistency result of $\mathrm{NNC}(\mathrm{WSS})$ in to the consistency results by Pollard (1981) and others see Section 5.

4.2 NNC Using Standard Graph-cut Based Objective Functions

In this section we want to look into the consistency of nearest neighbor clustering for graph based objective functions as they are used in spectral clustering (see von Luxburg, 2007 for details). Let $s : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}^+$ be a similarity function which is upper bounded by a constant C. The two main quantities we need to define graph-cut based objective functions are the cut and the volume. For a given cluster described by the cluster indicator function $f_k : \mathbb{R}^d \to \{0,1\}$, we set

$$\mathrm{cut}(f_k) := \mathrm{cut}(f_k, \mathbb{P}) := \mathbb{E} f_k(X_1)(1 - f_k(X_2)) s(X_1, X_2),$$

$$\mathrm{vol}(f_k) := \mathrm{vol}(f_k, \mathbb{P}) := \mathbb{E} f_k(X_1) s(X_1, X_2).$$

For $f \in \mathcal{H}$ we can then define the normalized cut and the ratio cut by

$$\mathrm{Ncut}(f) := \mathrm{Ncut}(f,\mathbb{P}) := \sum_{k=1}^{K} \frac{\mathrm{cut}(f_k)}{\mathrm{vol}(f_k)},$$
$$\mathrm{RatioCut}(f) := \mathrm{RatioCut}(f,\mathbb{P}) := \sum_{k=1}^{K} \frac{\mathrm{cut}(f_k)}{\mathbb{E} f_k(X)}.$$

The empirical estimators of these objective functions will be $\mathrm{Ncut}(f,\mathbb{P}_n)$ and $\mathrm{RatioCut}(f,\mathbb{P}_n)$, in explicit formulas:

$$\mathrm{cut}_n(f_k) := \frac{1}{n(n-1)} \sum_{i,j=1}^{n} f_k(X_i)(1-f(X_j))s(X_i,X_j),$$

$$\mathrm{vol}_n(f_k) := \frac{1}{n(n-1)} \sum_{i,j=1}^{n} f_k(X_i)s(X_i,X_j), \qquad n_k := \frac{1}{n}\sum_{i=1}^{k} f_k(X_i),$$

$$\mathrm{Ncut}_n(f) := \sum_{k=1}^{K} \frac{\mathrm{cut}_n(f_k)}{\mathrm{vol}_n(f_k)}, \qquad \mathrm{RatioCut}_n(f) := \sum_{k=1}^{K} \frac{\mathrm{cut}_n(f_k)}{n_k}.$$

Again we need to define how we will measure the size of the clusters. We will use

$$\Phi_{\mathrm{cut}}(f_k) := \mathrm{vol}(f_k), \qquad \Phi_{\mathrm{Ncut}}(f_k) := \mathrm{vol}(f_k), \qquad \Phi_{\mathrm{RatioCut}}(f_k) := \mathbb{E} f_k(X).$$

with the corresponding empirical quantities Φ_{cut_n}, Φ_{Ncut_n} and $\Phi_{\mathrm{RatioCut}_n}$. Then, with the notations of Theorem 2, we have:

Theorem 5 (Consistency of NNC(cut), NNC(Ncut) **and** NNC(RatioCut)**)** *Assume that the similarity function s is bounded by a constant $C > 0$, let $a_n > a$, $a_n \to a$, $m \to \infty$ and*

$$\frac{m^2 \log n}{n(a-a_n)^2} \to 0.$$

Then nearest neighbor clustering with cut, Ncut *and* RatioCut *is universally weakly consistent, that is for all probability measures, if $n \to \infty$ we have* $\mathrm{cut}(f_n) \to \mathrm{cut}(f^*)$, $\mathrm{Ncut}(f_n) \to \mathrm{Ncut}(f^*)$ *and* $\mathrm{RatioCut}(f_n) \to \mathrm{RatioCut}(f^*)$ *in probability.*

For these objective functions one can also state a rate of convergence. For sake of shortness we only state it for the normalized cut:

Theorem 6 (Convergence Rate for NNC(Ncut)**)** *Assume that the similarity function s is bounded by $C > 0$ and that $n(a_n - a)^2 \to \infty$. Let $\varepsilon \leq 1$ and $a^* := \inf_k vol(f_k^*) - a > 0$. Then there exist*

$$N = N((a_n),a^*) \in \mathbb{N},$$
$$C_1 = C_1(a,a^*,\varepsilon,K,C) > 0, \qquad C_2 = C_2(a,a^*,\varepsilon,C,K,f^*,\mathbb{P}) > 0,$$
$$C_3 = C_3(a,\varepsilon,K,C) > 0, \qquad C_4 = C_4(a,K,C) > 0.$$

such that for $n \geq N$ the following holds true:

$$\mathbb{P}(|\mathrm{Ncut}(f_n) - \mathrm{Ncut}(f^*)| \geq \varepsilon)$$
$$\leq C_1 e^{-C_2 m} + K^{m+1}(2n)^{(d+1)m^2}\left(C_3 e^{-C_4 \varepsilon^2 n} + 8K e^{-\frac{n(a_n-a)^2}{8}}\right).$$

4.3 NNC Using the Modularity Objective Function

A slightly different objective functions for graph clustering is the “modularity”, which has been put forward by Newman (2006) for detecting communities in networks. In this paper, the modularity is formulated as an objective function to find communities in a finite graph. However, as it is the case for Ncut or RatioCut, the modularity cannot be directly minimized. Instead, a spectral relaxation has been developed to minimize the modularity, see Newman (2006) for details. Of course, the nearest neighbor clustering algorithm can also be used to minimize this objective function directly, without using a relaxation step. Using our own notation we define:

$$\begin{aligned}
&\mathrm{Mod}_n(f) = \\
&\quad \sum_{k=1}^{n} \frac{1}{n(n-1)} \sum_{i \neq j} f_k(X_i) f_k(X_j) \left(\frac{1}{(n-1)^2} \sum_{l, l \neq i} s(X_i, X_l) \sum_{l, l \neq j} s(X_j, X_l) - s(X_i, X_j) \right), \\
&\mathrm{Mod}(f) = \\
&\quad \sum_{k=1}^{n} \int \int f_k(X) f_k(Y) \left(\int s(X,Z) d\mathbb{P}(Z) \int s(Y,Z) d\mathbb{P}(Z) - s(X,Y) \right) d(\mathbb{P} \times \mathbb{P})(X,Y).
\end{aligned}$$

In the proof we will see that as the limit function $\mathrm{Mod}(\cdot)$ is uniformly continuous on $\mathcal{H}$, we do not need to quantify any function Φ or Φ_n to measure the volume of the clusters. The function classes are thus

$$\begin{aligned}
\mathcal{F} &:= \{ f : \mathbb{R}^d \to \{1, \ldots, K\} \mid f \text{ continuous a.e.} \}, \\
\mathcal{F}_n &:= \{ f : \mathbb{R}^d \to \{1, \ldots, K\} \mid f(x) = f(NN_m(x)) \}.
\end{aligned}$$

Theorem 7 (Consistency of $\mathrm{NNC}(\mathrm{Mod})$**)** *Assume that $m \to \infty$ and*

$$\frac{m^2 \log n}{n} \to 0.$$

Then nearest neighbor clustering with Mod *is universally weakly consistent: for all probability measures, if $n \to \infty$ then* $\mathrm{Mod}(f_n) \to \mathrm{Mod}(f^*)$ *in probability.*

4.4 NNC Using Objective Function Based on the Ratio of Within-cluster and Between-cluster Similarity

Often, clustering algorithms try to minimize joint functions of the within-cluster similarity and the between cluster similarity. The most popular choice is the ratio of those two quantities, which is closely related to the criterion used in Fisher linear discriminant analysis. Formally, the between-cluster similarity corresponds to the cut, and the within similarity of cluster k is given by

$$\mathrm{WS} := \mathbb{E} f(X_1) f(X_2) s(X_1, X_2).$$

Thus the ratio of between- and within-cluster similarity is given as

$$\mathrm{BWR}(f) := \sum_{k=1}^{K} \frac{\mathrm{cut}(f_k)}{\mathrm{WS}(f_k)}.$$

Again we use their empirical estimations:

$$\mathrm{WS}_n(f_k) := \frac{1}{n(n-1)} \sum_{i,j=1}^{n} f_k(X_i) f_k(X_j) s(X_i, X_j),$$
$$\mathrm{BWR}_n(f) := \sum_{k=1}^{K} \frac{\mathrm{cut}_n(f_k)}{\mathrm{WS}_n(f_k)}.$$

To measure the size of the cluster we use

$$\Phi_{\mathrm{BWR}}(f_k) := \mathrm{WS}(f_k)$$

and its natural empirical counterpart. This leads to function spaces

$$\mathcal{F} := \{f : \mathbb{R}^d \to \{1, \dots, K\} \mid f \text{ continuous a.e. and } \forall k \in \{1, \dots, K\} \;\; \Phi_{\mathrm{BWR}}(f_k) > a\},$$
$$\mathcal{F}_n := \{f : \mathbb{R}^d \to \{1, \dots, K\} \mid f(x) = f(NN_m(x)) \text{ and } \forall k \in \{1, \dots, K\} \;\; \Phi_{\mathrm{BWR}_n}(f_k) > a_n\}.$$

Theorem 8 (Consistency of NNC(BWR)**)** *Assume that the similarity function s is bounded by a constant $C > 0$, let $a_n > a, a_n \to a, m \to \infty$ and*

$$\frac{m^2 \log n}{n(a - a_n)^2} \to 0.$$

Then nearest neighbor clustering with BWR *is universally weakly consistent, that is for all probability measure if $n \to \infty$ then* $\mathrm{BWR}(f_n) \to \mathrm{BWR}(f^*)$ *in probability.*

5. Relation to Previous Work

In this section we want to discuss our results in the light of the existing literature on consistent clusterings.

5.1 Standard Consistency Results for Center-based Algorithms

For a few clustering algorithms, consistency results are already known. The most well-known among them is the K-means algorithm. For this algorithm it has been first proved by Pollard (1981) that the global minimizer of the K-means objective function on a finite sample converges to the global minimizer on the underlying space.

First of all, we would like to point out that the consistency result by Pollard (1981) can easily be recovered using our theorems. Let us briefly recall the standard K-means setting. The objective function which K-means attempts to optimize is the function WSS, which we already encountered in the last sections. In the standard K-means setting the optimization problem is stated over the space of all measurable functions $\mathcal{H}$:

$$f^* = \operatorname*{argmin}_{f \in \mathcal{H}} \mathrm{WSS}(f).$$

It is not difficult to prove that the solution f^* of this optimization problem always has a particular form. Namely, the solution f^* forms a Voronoi decomposition of the space, where the cluster centers c_k are the centers of the Voronoi cells. Thus, we can rewrite the optimization problem above equivalently as

$$f^* = \operatorname*{argmin}_{f \in \mathcal{G}_K} \mathrm{WSS}(f)$$

where $\mathcal{G}_K$ denotes the set of all clusterings for which the clusters are Voronoi cells. The optimization problem for the finite sample case can be stated analogously:

$$f_n = \operatorname*{argmin}_{f \in \mathcal{G}_K} \mathrm{WSS}_n(f).$$

So in this particular case we can set $\mathcal{F}_n = \mathcal{F} = \mathcal{G}_K$. This means that even though the original optimization problem has been set up to optimize over the huge set $\mathcal{H}$, the optimization only needs to run over the small set $\mathcal{G}_K$. It is well known that the shattering coefficient of $\mathcal{G}_K$ is polynomial in n, namely it is bounded by $K^K n^{(d+1)K^2}$ (cf. Lemma 10). Moreover, the uniform continuity of WSS on $\mathcal{G}_K$ (Condition (3) of Theorem 2) can easily be verified if we assume that the probability distribution has compact support. As a consequence, using similar techniques as in the proofs of Theorem 3 we can prove that the global minimizer of the empirical K-means objective function WSS_n converges to the global minimizer of the true K-means objective function WSS. By this we recover the well-known result by Pollard (1981), under slightly different assumptions. In this sense, our Theorem 1 can be seen as a blueprint for obtaining Pollard-like results for more general objective functions and function spaces.

Are there any more advantages of Theorem 3 in the K-means setting? At first glance, our result in Theorem 3 looks similar to Pollard's result: the global minimizers of both objective functions converge to the true global minimizer. However, in practice there is one important difference. Note that as opposed to many vector quantization problems (cf. Garey et al., 1982), minimizing the K-means objective function is not NP-hard in n: the solution is always a Voronoi partition, there exist polynomially many Voronoi partitions of n points, and they can be enumerated in polynomial time (cf. Inaba et al., 1994). However, the size of the function class $\mathcal{G}_K$ is still so large that it would take too long to simply enumerate all its functions and select the best one. Namely, we will see in Lemma 10 that the number of Voronoi partitions of n points in $\mathbb{R}^d$ using K cells is bounded by $n^{(d+1)K}$, which is huge even for moderate d and K. As a work-around in practice one uses the well-known K-means *algorithm*, which is only able to find a *local* minimum of $\mathrm{WSS}_n(f)$. In contrast, nearest neighbor clustering works with a different function class which is much smaller than $\mathcal{G}_K$: it has only size $n^{\log K}$. On this smaller class we are still able to compute the *global* minimum of $\mathrm{WSS}_n(f)$. Consequently, our result in Theorem 3 is not only a theoretical statement about some abstract quantity as it is the case for Pollard's result, but it applies to the algorithm used in practice. While Pollard's result abstractly states that the global minimum (which cannot be computed efficiently) converges, our result implies that the result of nearest neighbor clustering does converge.

5.2 Consistency of Spectral Clustering

In the previous section we have seen in Theorems 5 and 6 that NNC is consistent for all the standard graph cut objective functions. Now we want to discuss these results in connection with the graph cut literature. It is well known that the discrete optimization problem of minimizing Ncut_n or $\mathrm{RatioCut}_n$ is an NP-hard problem, see Wagner and Wagner (1993). However, approximate solutions of relaxed problems can be obtained by spectral clustering, see von Luxburg (2007) for a tutorial. Consistency results for spectral clustering algorithms have been proved in von Luxburg et al. (2008). These results show that under certain conditions, the solutions computed by spectral clustering on finite samples converge to some kind of "limit solutions" based on the underlying distribution. In the light of the previous discussions, this sounds plausible, as the space of solutions of spectral clustering is rather restricted: we only allow solutions which are eigenfunctions of certain integral operators. Thus, spectral clustering implicitly works with a small function class.

However, it is important to note that the convergence results of spectral clustering do not make any statement about the minimizers of Ncut (a similar discussion also holds for RatioCut). The problem is that on any finite sample, spectral clustering only solves a relaxation of the original problem of minimizing Ncut_n. The Ncut_n-value of this solution can be arbitrarily far away from the minimal Ncut_n-value on this sample (Guattery and Miller, 1998), unless one makes certain assumptions which are not necessarily satisfied in a standard statistical setting (cf. Spielman and Teng, 1996, or Kannan et al., 2004). Thus the convergence statements for the results computed by the spectral clustering algorithm cannot be carried over to consistency results for the minimizers of Ncut. One knows that spectral clustering converges, but one does not have any guarantee about the Ncut-value of the solution. Here our results for nearest neighbor clustering present an improvement, as they directly refer to the minimizer of Ncut. While it is known that spectral clustering converges to "something", for the solutions computed by nearest neighbor clustering we know that they converge to the global minimizer of Ncut (or RatioCut, respectively).

5.3 Consistency of Other Clustering Schemes

To the best of our knowledge, apart from results on center-based algorithms and spectral clustering, there are very few non-parametric clustering algorithms for which statistical consistency has been proved so far. The only other major class of algorithms for which consistency has been investigated is the class of linkage algorithms. While single linkage can be proved to be "fractionally consistent", that is it can at least discover sufficiently distinct high-density regions, both complete and average linkage are not consistent and can be misleading (cf. Hartigan, 1981, 1985). A more general method for hierarchical clustering used in Wong and Lane (1983) is statistically consistent, but essentially first estimates the density and then constructs density level sets based on this estimator.

Concerning parametric clustering algorithms, the standard setting is a model-based approach. One assumes that the underlying probability distribution has a certain parametric form (for example a mixture of Gaussians), and the goal is to estimate the parameters of the distribution from the sample. Estimating parameters in parametric models has been intensively investigated in statistics, in particular in the maximum likelihood framework and the Bayesian framework (for an overview how this can be done for clustering see Fraley and Raftery, 1998, or the book McLachlan and Peel, 2004). Numerous consistency results are known, but typically they require that the true underlying distribution indeed comes from the model class under consideration. For example, in a Bayesian setting one can show that in the large sample limit, the posterior distribution will concentrate around the true mixture parameters. However, if the model assumptions are not satisfied, counter-examples to consistency can be constructed. Moreover, the consistency results mentioned above are theoretic in the sense that the algorithm used in practice does not necessarily achieve them. Standard approaches to estimate mixture parameters are the EM algorithm (in a frequentist of MAP setting), or for example Markov Chain Monte Carlo sampling in a fully Bayesian approach. However, as it is the case for the K-means algorithm, these methods can get stuck in local optima, and no convergence towards the global optimum can be guaranteed. Another way to tackle model-based clustering problems is based on the minimum message length or minimum description length principle. The standard reference for MML approaches to learn mixtures is Figueiredo and Jain (2002), for a general overview on MDL see Grünwald (2007). Consistency results for MML are quite similar to the ones for the Bayesian approach: if the true distribution indeed comes from the mixture class and the number of components is known, then consistency can be achieved. For general results on consistency of MDL see Sections 16 and 17.11 in Grünwald (2007). Often, MML/MDL approaches are interpreted as a particular way to work with small function classes, consisting of functions which can be described in a "compact" way. In this sense, this method can also be seen as a way of achieving "small" function classes.

5.4 Sublinear Time Algorithms Using Subsampling

Some algorithms related to our approach have been published in the theoretical computer science community, such as Indyk (1999), Mishra et al. (2001), or Czumaj and Sohler (2007). The general idea is to use subsampling approaches to approximate clustering solutions, and to prove that these approximations are quite accurate. Given a sample of n points, one draws a subsample of $m \ll n$ points, applies some (approximate) clustering algorithm to the subsample, and then extends this clustering to the remaining points. Using techniques such as concentration inequalities, Chernoff bounds or Hoeffding bounds, one can then prove that the resulting clustering approximates the best clustering on the original point set.

While at first glance, this approach sounds very similar to our nearest neighbor clustering, note that the focus in these papers is quite a different one than ours. The authors do not aim for consistent clustering solutions (that is, solutions which are close to the "true clustering solution" of the underlying space), but they want to find algorithms to approximate the optimal clustering on a given finite sample in sublinear time. The sublinearity is achieved by the fact that already a very small subsample (say, $m = \log n$) is enough to achieve good approximation guarantees. However, our main point that it is important to control the size of the underlying function class, is not revealed in these papers. As the authors mainly deal with K-means type settings, they automatically work with polynomial function classes of center-based clusterings, and the issue of inconsistency does not arise. Moreover, subsampling is just one way of reducing the function class to a smaller size, there can be many others. In this sense, we believe that our "small function class" approach is more general than the subsampling approach.

Finally, one difference between our approach and the subsampling approach is the kind of results of interest. We are mainly concerned with asymptotic results, and on our way achieve approximation guarantees which are good for large sample size n. The focus of the subsampling papers is non-asymptotic, dealing with a small or moderate sample size n, and to prove approximation guarantees in this regime.

5.5 Other Statistical Learning Theory Approaches to Clustering

In the last years there have been several papers which started to look at clustering from a statistical learning theory perspective. A general statistical learning theory approach to clustering, based on a very similar intuition as ours, has already been presented in Buhmann (1998). Here the authors put forward an "empirical risk approximation" approach for unsupervised learning, along the lines of empirical risk minimization for the supervised case. The setting under consideration is that the clustering quality function is an expectation with respect to the true underlying probability distribution, and the empirical quality function is the corresponding empirical expectation. Then, similar to the statistical learning theory for supervised learning, generalization bounds can be derived, for example using VC dimensions. Additionally, the authors discuss regularization approaches and relate them to annealing schemes for center-based clusterings.

A different approach has been investigated in Ben-David (2007). Here the author formalizes the notion of a "cluster description scheme". Intuitively, a clustering problem can be described by a cluster description scheme of size $l \in \mathbb{N}$ if each clustering can be described using l points from the space (and perhaps some additional parameter). For instance, this is the case for center-based clusterings, where the clustering can be described by the centroids only. Ben-David then proves generalization bounds for clustering description schemes which show that the global minimizer of the empirical quality function converges to the global minimizer of the true quality function. The proof techniques used in this paper are very close to the ones used in standard minimum description length results.

Another class of results about K-means algorithms has been proved in Rakhlin and Caponnetto (2007). After computing covering numbers for the underlying classes, the authors study the stability behavior of K-means. This leads to statements about the set of "almost-minimizers" (that is the set of all functions whose quality is ε close to the one of the global optimal solutions). As opposed to our results and all the other results discussed above, the main feature of this approach is that at the end of the day, one is able to make statements about the clustering functions themselves, rather than only about their quality values. In this sense, the approach in Rakhlin and Caponnetto (2007) has more powerful results, but its application is restricted to K-means type algorithms.

All approaches outlined above implicitly or explicitly rely on the same intuition as our approach: the function class needs to be "small" in order to lead to consistent clusterings. However, all previous results have some restrictions we could overcome in our approach. First of all, in the papers discussed above the quality function needs to be an expectation, and the empirical quality function is simply the empirical expectation. Here our results are more general: we neither require the quality functions to be expectations (for example, Ncut cannot be expressed as an expectation, it is a ratio of two expectations) nor do we require unbiasedness of the empirical quality function. Second, the papers discussed above make statements about global optimizers, but do not really deal with the question how such a global optimizer can be computed. The case of standard K-means shows that this is by no means simple, and in practice one has to use heuristics which discover local optima only. In contrast, we suggest a concrete algorithm (NNC) which computes the global optimum over the current function class, and hence our results not only concern abstract global minimizers which are hard to obtain, but refer to exactly the quantities which are computed by the algorithm. Finally, our algorithm has the advantage that it provides a framework for dealing with more general clustering objective functions than just center-based ones. This is not the case in the papers above.

Finally, we would like to mention that a rather general but vague discussion of some of the open issues in statistical approaches to clustering has been led in von Luxburg and Ben-David (2005). Our current paper partly solves some of the open issues raised there.

6. Discussion

Our paper is concerned with clustering algorithms which minimize certain quality functions. Our main point is that as soon as we require statistical consistency we have to work with function classes $\mathcal{F}_n$ which are "small". Our results have a similar taste as the well-known corresponding results for supervised classification. While in the domain of supervised classification practitioners are well aware of the effect of overfitting, it seems like this effect has been completely overlooked in the clustering domain.

We would like to highlight a convenient side-effect of working with small function classes. In clustering, for many objective functions the problem of finding the best partition of the discrete data set is an NP-hard problem (for example, this is the case for all balanced graph-cut objective functions). On the other side, if we restrict the function class $\mathcal{F}_n$ to have polynomial size (in n), then the trivial algorithm of evaluating all functions in $\mathcal{F}_n$ and selecting the best one is inherently polynomial. Moreover, if the small function class is "close" to the large function class, then the solution found in the small function class approximates the best solution in the unrestricted space of all clusterings.

We believe that the approach of using restricted function classes can be very promising, also from a practical point of view. It can be seen as a more controlled way of constructing approximate solutions of NP hard optimization problems than the standard approaches of local optimization or relaxation. While the effects of the latter cannot be controlled in general, we are able to control the effects of optimizing over smaller function classes by carefully selecting $\mathcal{F}_n$. This strategy circumvents the problem that solutions of local optimization or relaxation heuristics can be arbitrarily far away from the optimal solution.

The generic clustering algorithm we studied in this article is nearest neighbor clustering, which produces clusterings that are constant on small local neighborhoods. We have proved that this algorithm is statistically consistent for a large variety of popular clustering objective functions. Thus, as opposed to other clustering algorithms such as the K-means algorithm or spectral clustering, nearest neighbor clustering is guaranteed to converge to a minimizer of the true global optimum on the underlying space. This statement is much stronger than the results already known for K-means or spectral clustering. For K-means it has been proved that the global minimizer of the WSS objective function on the sample converges to a global minimizer on the underlying space (e.g., Pollard, 1981). However, as the standard K-means algorithm only discovers a local optimum on the discrete sample, this result does not apply to the algorithm used in practice. A related effect happens for spectral clustering, which is a relaxation attempting to minimize Ncut or RatioCut. For this class of algorithms, it has been shown that under certain conditions the solution of the relaxed problem on the finite sample converges to some limit clustering. However, this limit clustering is not necessarily the optimizer of the Ncut or RatioCut objective function.

It is interesting to note that the problems about the existing consistency results for K-means and spectral clustering are "reverse" to each other: while for K-means we know that the global minimizer converges, but this result does not apply to the algorithm used in practice, for spectral clustering there exist consistency results for the algorithm used in practice, but these results do not relate to the global minimizer. For both cases, our consistency results represent an improvement: we have constructed an algorithm which provably converges to the true limit minimizer of WSS or Ncut, respectively. The same result also holds for a large number of alternative objective functions used for clustering.

We believe that a big advantage of our approach is that both the algorithm and the statistical analysis is not restricted to center-based algorithms only, as it has been the case for most approaches in the literature (Buhmann, 1998; Ben-David, 2007; Rakhlin and Caponnetto, 2007). Instead, nearest neighbor clustering can be used as a baseline method to construct clusterings for any objective function. In von Luxburg et al. (2008) we have shown how nearest neighbor clustering can be implemented efficiently using branch and bound, and that in terms of quality, its results can compete with algorithms of spectral clustering (for the Ncut objective function) or K-means (for the WSS objective function). We believe that in particular for unusual objective functions for which no state of the art optimizer exists yet, nearest neighbor clustering is a promising baseline to start with. We have seen that for many commonly used objective functions, statistical guarantees for nearest neighbor clustering can be obtained, and we expect the same to be true for many more clustering objective functions.

Finally, it is a fair question how statistical consistency helps in practical applications. Is it any help in solving the big open issues in clustering, such as the question of selecting clustering algorithms for a particular data set, or selecting the number of clusters? In this generality, the answer is no. In our opinion, consistency is a *necessary* requirement which any clustering algorithm should satisfy. If an algorithm is not consistent, even with a high amount of data one cannot rely on a clustering constructed on a finite amount of data—and this is not due to computational problems, but to inherent statistical problems. Such an algorithm cannot be trusted when constructing results on a finite sample; given another sample, it might just come up with a completely different clustering. Or, the more samples one gets, the more "trivial" the solution might become (unnormalized spectral clustering is an example for such an algorithm). In this sense, consistency is just one piece of evidence to discard unreliable clustering algorithms. In our opinion, it is very hard to come up with *sufficient* conditions about "what a good clustering algorithm is". The applications of clustering are just too diverse, and 50 years of clustering literature show that people will not agree on a unique definition of what a good clustering algorithm is. This is the reason why we believe that it is very fruitful to start by studying necessary conditions first. Our current paper is meant as a contribution to this effort.

Appendix A. All Proofs

In this section we concentrate all the proofs.

A.1 The Proof of Theorem 1

The following lemma will be central in our analysis. It allows to take a supremum out of a probability.

Lemma 9 *With the notation in Theorem* 1 *we have:*

$$\mathbb{P}(\sup_{f\in\mathcal{F}_n}|Q_n(f)-Q(f)|\geq\varepsilon)\leq 2s(\widetilde{\mathcal{F}_n},2n)\frac{\sup_{f\in\widetilde{\mathcal{F}_n}}\mathbb{P}(|Q_n(f)-Q(f)|\geq\varepsilon/4)}{\inf_{f\in\widetilde{\mathcal{F}_n}}\mathbb{P}(|Q_n(f)-Q(f)|\leq\varepsilon/2)}.$$

The proof technique is similar to the one in Devroye et al. (1996), Section 12.3. The unusual term in the denominator originates in the symmetrization step. In a more standard setting where we have $\mathbb{E}Q_n = Q$, this term usually "disappears" as it can be lower bounded by $1/2$, for example using

Chebyshev's inequality (e.g., Section 12.3 of Devroye et al., 1996). Unfortunately, this does not work in our more general case, as we do not assume unbiasedness and instead also allow $\mathbb{E}Q_n \neq Q$. However, note that the ratio in Lemma 9 essentially has the form $u_n/(1-u_n)$. Thus, as soon as the term u_n in the numerator becomes non-trivial (i.e., $u_n < 1$ or say, $u_n < 3/4$), then the denominator will only play the role of a small constant (it is lower bounded by 1/4). This means that in the regime where the numerator is non-trivial, the whole bound will essentially behave like the numerator.

Proof First note that we can replace the data-dependent function class $\mathcal{F}_n$ by the class $\widetilde{\mathcal{F}_n}$ which does not depend on the data:

$$\mathbb{P}(\sup_{f\in\mathcal{F}_n}|Q_n(f)-Q(f)| \geq \varepsilon) \leq \mathbb{P}(\sup_{f\in\widetilde{\mathcal{F}_n}}|Q_n(f)-Q(f)| \geq \varepsilon).$$

Now we want to use a symmetrization argument. To this end, let $X_1',\dots,X_n'$ be a ghost sample (that is a sample drawn i.i.d. according to $\mathbb{P}$ which is independent of our first sample $X_1,\dots,X_n$), and denote by Q_n' the empirical quality function based on the ghost sample.

Let $\widehat{f}\in\widetilde{\mathcal{F}_n}$ be such that $|Q_n(\widehat{f})-Q(\widehat{f})|\geq\varepsilon$; if such an $\widehat{f}$ does not exist then just choose $\widehat{f}$ as some other fixed function in $\widetilde{\mathcal{F}_n}$. Note that $\widehat{f}$ is a data-dependent function depending on the sample $X_1,...,X_n$. We have the following inequalities:

$$\begin{aligned}
&\mathbb{P}(\sup_{f\in\widetilde{\mathcal{F}_n}}|Q_n(f)-Q_n'(f)|\geq\varepsilon/2)\\
&\geq \mathbb{P}(|Q_n(\widehat{f})-Q_n'(\widehat{f})|\geq\varepsilon/2)\\
&\geq \mathbb{P}(|Q_n(\widehat{f})-Q(\widehat{f})|\geq\varepsilon, |Q_n'(\widehat{f})-Q(\widehat{f})|\leq\varepsilon/2)\\
&= \mathbb{E}\Big(\mathbb{P}(|Q_n(\widehat{f})-Q(\widehat{f})|\geq\varepsilon, |Q_n'(\widehat{f})-Q(\widehat{f})|\leq\varepsilon/2|X_1,\dots,X_n)\Big)\\
&= \mathbb{E}\Big(\mathbb{P}(|Q_n(\widehat{f})-Q(\widehat{f})|\geq\varepsilon|X_1,\dots,X_n)\mathbb{P}(|Q_n'(\widehat{f})-Q(\widehat{f})|\leq\varepsilon/2|X_1,\dots,X_n)\Big)\\
&= \mathbb{E}\Big(1_{|Q_n(\widehat{f})-Q(\widehat{f})|\geq\varepsilon}\mathbb{P}(|Q_n'(\widehat{f})-Q(\widehat{f})|\leq\varepsilon/2|X_1,\dots,X_n)\Big)\\
&\geq \mathbb{E}\left(1_{|Q_n(\widehat{f})-Q(\widehat{f})|\geq\varepsilon}\inf_{f\in\widetilde{\mathcal{F}_n}}\mathbb{P}(|Q_n'(f)-Q(f)|\leq\varepsilon/2|X_1,\dots,X_n)\right)\\
&= \mathbb{E}(1_{|Q_n(\widehat{f})-Q(\widehat{f})|\geq\varepsilon})\mathbb{E}\left(\inf_{f\in\widetilde{\mathcal{F}_n}}\mathbb{P}(|Q_n'(f)-Q(f)|\leq\varepsilon/2|X_1,\dots,X_n)\right)\\
&= \mathbb{P}(|Q_n(\widehat{f})-Q(\widehat{f})|\geq\varepsilon)\inf_{f\in\widetilde{\mathcal{F}_n}}\mathbb{P}(|Q_n(f)-Q(f)|\leq\varepsilon/2)\\
&= \mathbb{P}(\sup_{f\in\widetilde{\mathcal{F}_n}}|Q_n(f)-Q(f)|\geq\varepsilon)\inf_{f\in\widetilde{\mathcal{F}_n}}\mathbb{P}(|Q_n(f)-Q(f)|\leq\varepsilon/2).
\end{aligned}$$

The last step is true because of the definition of $\widehat{f}$: note that due to the definition of $\widehat{f}$ the event $|Q_n(\widehat{f})-Q(\widehat{f})|\geq\varepsilon$ is true iff there exists some $f\in\widetilde{\mathcal{F}_n}$ such that $|Q_n(f)-Q(f)|\geq\varepsilon$, which is true iff $\sup_{f\in\widetilde{\mathcal{F}_n}}|Q_n(f)-Q(f)|\geq\varepsilon$ (recall that we assumed for ease of notations that all supremum are attained). Rearranging the inequality above leads to

$$\mathbb{P}(\sup_{f\in\widetilde{\mathcal{F}_n}}|Q_n(f)-Q(f)|\geq\varepsilon) \leq \frac{\mathbb{P}(\sup_{f\in\widetilde{\mathcal{F}_n}}|Q_n(f)-Q_n'(f)|\geq\varepsilon/2)}{\inf_{f\in\widetilde{\mathcal{F}_n}}\mathbb{P}(|Q_n(f)-Q(f)|\leq\varepsilon/2)}.$$

Due to the symmetrization we got rid of the quantity $Q(f)$ in the numerator. Furthermore, using the assumption of the theorem that $Q_n(f)$ does not involve any function evaluations $f(x)$ for $x \notin \{X_1, \dots, X_n\}$ we can apply a union bound argument to move the supremum in the numerator out of the probability:

$$\begin{aligned}
&\mathbb{P}(\sup_{f \in \widetilde{\mathcal{F}_n}} |Q_n(f) - Q'_n(f)| \geq \varepsilon/2) \\
&\leq s(\widetilde{\mathcal{F}_n}, 2n) \sup_{f \in \widetilde{\mathcal{F}_n}} \mathbb{P}(|Q_n(f) - Q'_n(f)| \geq \varepsilon/2) \\
&\leq s(\widetilde{\mathcal{F}_n}, 2n) \sup_{f \in \widetilde{\mathcal{F}_n}} \mathbb{P}(|Q_n(f) - Q(f)| + |Q(f) - Q'_n(f)| \geq \varepsilon/2) \\
&\leq 2s(\widetilde{\mathcal{F}_n}, 2n) \sup_{f \in \widetilde{\mathcal{F}_n}} \mathbb{P}(|Q_n(f) - Q(f)| \geq \varepsilon/4).
\end{aligned}$$

This completes the proof of the lemma. ■

Now we are ready to prove our first main theorem.

A.2 Proof of Theorem 1

Additionally to the functions f_n and f^*, we will define

$$\begin{aligned}
f_n^* &\in \operatorname*{argmin}_{f \in \mathcal{F}_n} Q(f), \\
\widetilde{f}^* &\in \operatorname*{argmin}_{f \in \mathcal{F}_n} d(f, f^*).
\end{aligned}$$

To prove the theorem we have to show that under the conditions stated, for any fixed $\varepsilon > 0$ the term $\mathbb{P}(|Q(f_n) - Q(f^*)| \geq \varepsilon)$ converges to 0. We can study each "side" of this convergence independently:

$$\mathbb{P}(|Q(f_n) - Q(f^*)| \geq \varepsilon) = \mathbb{P}(Q(f_n) - Q(f^*) \leq -\varepsilon) + \mathbb{P}(Q(f_n) - Q(f^*) \geq \varepsilon).$$

To treat the "first side" observe that if $f_n \in \mathcal{F}$ then $Q(f_n) - Q(f^*) > 0$ by the definition of f^*. This leads to

$$\mathbb{P}(Q(f_n) - Q(f^*) \leq -\varepsilon) \leq \mathbb{P}(f_n \notin \mathcal{F}).$$

Under Assumption (2) of Theorem 1 this term tends to 0.

The main work of the proof is to take care of the second side. To this end we split $Q(f_n) - Q(f^*)$ in two terms, the estimation error and the approximation error:

$$Q(f_n) - Q(f^*) = Q(f_n) - Q(f_n^*) + Q(f_n^*) - Q(f^*).$$

For a fixed $\varepsilon > 0$ we have

$$\mathbb{P}(Q(f_n) - Q(f^*) \geq \varepsilon) \leq \mathbb{P}(Q(f_n) - Q(f_n^*) \geq \varepsilon/2) + \mathbb{P}(Q(f_n^*) - Q(f^*) \geq \varepsilon/2).$$

In the following sections we will treat both parts separately.

A.2.1 ESTIMATION ERROR

The first step is to see that

$$Q(f_n) - Q(f_n^*) \leq 2 \sup_{f \in \mathcal{F}_n} |Q_n(f) - Q(f)|.$$

Indeed, since $Q_n(f_n) \leq Q_n(f_n^*)$ by the definition of f_n we have

$$\begin{aligned} Q(f_n) - Q(f_n^*) &= Q(f_n) - Q_n(f_n) + Q_n(f_n) - Q_n(f_n^*) + Q_n(f_n^*) - Q(f_n^*) \\ &\leq Q(f_n) - Q_n(f_n) + Q_n(f_n^*) - Q(f_n^*) \\ &\leq 2 \sup_{f \in \mathcal{F}_n} |Q_n(f) - Q(f)|. \end{aligned}$$

Using Lemma 9 we obtain

$$\mathbb{P}(Q(f_n) - Q(f_n^*) \geq \varepsilon/2) \leq 2s(\widetilde{\mathcal{F}_n}, 2n) \frac{\sup_{f \in \widetilde{\mathcal{F}_n}} \mathbb{P}(|Q_n(f) - Q(f)| \geq \varepsilon/16)}{\inf_{f \in \widetilde{\mathcal{F}_n}} \mathbb{P}(|Q_n(f) - Q(f)| \leq \varepsilon/8)}.$$

Now observe that under Assumption (1) the numerator of the expression in the proposition tends to 0 and the denominator tends to 1, so the whole term tends to 0.

A.3 Approximation Error

By definition of f_n^* it is clear that

$$Q(f_n^*) - Q(f^*) \leq Q(\widetilde{f}^*) - Q(f^*).$$

Using Assumption (3) this leads to

$$\begin{aligned} \mathbb{P}(Q(f_n^*) - Q(f^*) \geq \varepsilon/2) &\leq \mathbb{P}(Q(\widetilde{f}^*) - Q(f^*) \geq \varepsilon/2) \\ &\leq \mathbb{P}(d(f, \widetilde{f}^*) \geq \delta(\varepsilon/2)). \end{aligned}$$

The right hand side clearly tends to 0 by Assumption (2).

■

A.4 The Proof of Theorem 2

Before proving Theorem 2, we again need to prove a few technical lemmas. The first one is a simple relation between the shattering coefficients of the nearest neighbor function classes.

Lemma 10 *Let $u \in \mathbb{N}$ and $\widetilde{\mathcal{F}_n}$ and $\widehat{\mathcal{F}_n}$ be the function sets defined in Theorem* 2. *Then*

$$s(\widetilde{\mathcal{F}_n}, u) \leq s(\widehat{\mathcal{F}_n}, u) \leq K^m u^{(d+1)m^2}.$$

Proof The first inequality is obvious as we have $\widetilde{\mathcal{F}_n} \subset \widehat{\mathcal{F}_n}$. For the second inequality observe that

$$s(\widehat{\mathcal{F}_n}, u) \leq K^m s^*(\widehat{\mathcal{F}_n}, u)$$

where $s^*(\widehat{\mathcal{F}_n},u)$ is the maximal number of different ways u points can be partitioned by cells of a Voronoi partition of m points. It is well known (e.g., Section 21.5 of Devroye et al., 1996) that $s^*(\widehat{\mathcal{F}_n},u) \le u^{(d+1)m^2}$ for $d > 1$. Note that for $d = 1$ a similar inequality holds, we do not consider this case any further. ■

The second lemma relates a function evaluated at a point x to the same function, evaluated at the nearest neighbor of x in the training points. This lemma builds on ideas of Fritz (1975).

Lemma 11 *Let $f : \mathcal{X} \to \{1,\dots,K\}$ be continuous almost everywhere and*

$$L_n := \mathbb{P}(f(X) \neq f(NN_m(X))|X_1,\dots,X_n).$$

Then for every $\varepsilon > 0$ there exists a constant $b_f(\varepsilon) > 0$ independent of n such that

$$\mathbb{P}(L_n \ge \varepsilon) \le \frac{2}{\varepsilon} e^{-mb_f(\varepsilon)}.$$

Proof By $B(x,\delta)$ we denote the Euclidean ball of center x and radius δ. The first step of the proof consists in constructing a certain set D (depending on ε) which satisfies the following statement:

For all $\varepsilon > 0$ there exists some $\delta(\varepsilon) > 0$, a measurable set $D \subset \mathbb{R}^d$ and a constant $1 > u > 0$ such that

(a) $\mathbb{P}(D) \ge 1 - \varepsilon/2$

(b) $\forall x \in D : \mathbb{P}(B(x,\delta)) > u$

(c) $\forall x \in D$ the function f is constant on $B(x,\delta)$.

Assume we have such a set D. Then using Properties (c) and (a) we can see that

$$\begin{aligned} L_n &= \mathbb{P}(f(X) \neq f(NN_m(X))|X_1,\dots,X_n) \\ &\le \mathbb{P}(X \notin D|X_1,\dots,X_n) + \mathbb{P}(X \in D, |X - NN_m(X)| > \delta|X_1,\dots,X_n) \\ &\le \tfrac{\varepsilon}{2} + \mathbb{P}(X \in D, |X - NN_m(X)| > \delta|X_1,\dots,X_n). \end{aligned}$$

Using the Markov inequality we can then see that

$$\begin{aligned} \mathbb{P}(L_n > \varepsilon) &\le \mathbb{P}(\mathbb{P}(X \in D, |X - NN_m(X)| > \delta|X_1,\dots,X_n) \ge \tfrac{\varepsilon}{2}) \\ &\le \tfrac{2}{\varepsilon}\mathbb{E}(\mathbb{P}(X \in D, |X - NN_m(X)| > \delta|X_1,\dots,X_n)) \\ &= \tfrac{2}{\varepsilon}\mathbb{P}(X \in D, |X - NN_m(X)| > \delta) \\ &= \tfrac{2}{\varepsilon}\int_D \mathbb{P}(|x - NN_m(x)| > \delta)\, d\mathbb{P}(x). \end{aligned}$$

Due to Property (b) we know that for all $x \in D$,

$$\begin{aligned} \mathbb{P}(|x - NN_m(x)| > \delta) &= \mathbb{P}(\forall i \in \{1,\dots,m\}, x \notin B(X_i,\delta)) \\ &= (1 - \mathbb{P}(B(x,\delta)))^m \\ &\le (1-u)^m. \end{aligned}$$

Setting $b(\varepsilon) := -log(1-u) > 0$ then leads to

$$P(L_n > \varepsilon) \leq \frac{2}{\varepsilon}\mathbb{P}(D)(1-u)^m \leq \frac{2}{\varepsilon}e^{-mb(\delta(\varepsilon))}.$$

Note that this constant $b(\varepsilon)$ will also be used in several of the following lemmas. To finish the proof of the lemma we have to show how the set D can be constructed. By the assumption of the lemma we know that f is continuous a.e., and that f only takes finitely many values $1,...,K$. This implies that the set

$$C = \{x \in \mathbb{R}^d : \exists \delta > 0 : d(x,y) \leq \delta \Rightarrow f(x) = f(y)\}$$

satisfies $\mathbb{P}(C) = 1$. Furthermore, for any $\delta > 0$ we define the set

$$A_\delta = \{x \in C : d(x,y) \leq \delta \Rightarrow f(x) = f(y)\}.$$

We have $\cup_\delta A_\delta = C$, and for $\sigma > \delta$ we have $A_\sigma \subset A_\delta$. This implies that given some $\varepsilon > 0$ there exists some $\delta(\varepsilon) > 0$ such that $\mathbb{P}(A_{\delta(\varepsilon)}) \geq 1 - \varepsilon/4$. By construction, all points in $A_{\delta(\varepsilon)}$ satisfy Property (c).

As the next step, we can see that for every $\delta > 0$ one has $\mathbb{P}(B(x,\delta)) > 0$ almost surely (with respect to x). Indeed, the set $U = \{x : \exists \delta > 0 : \mathbb{P}(B(x,\delta)) = 0\}$ is a union of sets of probability zero. So using the fact that $\mathbb{R}^d$ is separable we see that $\mathbb{P}(U) = 0$. Thus, $\mathbb{P}(\mathbb{P}(B(X,\delta)|X) > 0) = 1$, which implies $\mathbb{P}(\mathbb{P}(B(X,\delta)|X) > \frac{1}{n}) \to 1$. This means that given $\varepsilon > 0$ and $\delta > 0$ there exists a set A and a constant $u > 0$ such that $\mathbb{P}(A) \geq 1 - \varepsilon/4$ and $\forall x \in A, \mathbb{P}(B(x,\delta)) > u$. So all points in A satisfy Property (b).

Now finally define the set $D = A \bigcap A_{\delta(\varepsilon)}$. By construction, this set has probability $P(D) \geq \varepsilon/2$, so it satisfies Property (a). It satisfies Properties (b) and (c) by construction of A and $A_{\delta(\varepsilon)}$, respectively. ■

A.5 Proof of Theorem 2

To prove this theorem we will verify that the conditions (1) - (3) of Theorem 1 are satisfied for the function classes studied in Theorem 2.

Lemma 10 proves that Condition (1) of Theorem 2 implies Condition (1) of Theorem 1. Moreover, it is obvious that Condition (3) of Theorem 2 implies Condition (3) of Theorem 1.

Thus we only have to prove Condition (2) of Theorem 1. We begin by proving that $\mathbb{P}(f_n \notin \mathcal{F}) \to 0$. As $f_n \in \mathcal{F}_n$ by definition we have that $\Phi_n(f_{n,k}) > a_n$ for all $k = 1,\dots,K$. A union bound argument shows that

$$\mathbb{P}(f_n \notin \mathcal{F}) \leq K \sup_k \mathbb{P}(\Phi(f_{n,k}) \leq a).$$

Using the same techniques as in the proof of Lemma 9 we can see that

$$\begin{aligned}\mathbb{P}(\Phi(f_{n,k}) \leq a) &\leq \mathbb{P}(\Phi_n(f_{n,k}) - \Phi(f_{n,k}) \geq a_n - a)\\ &\leq \mathbb{P}(\sup_{g \in \mathcal{F}_n} \Phi_n(g_k) - \Phi(g_k) \geq a_n - a)\\ &\leq 2s(\widehat{\mathcal{F}_n}, 2n)\frac{\sup_{g \in \widehat{\mathcal{F}_n}} \mathbb{P}(\Phi_n(g_k) - \Phi(g_k) \geq (a_n - a)/4)}{\inf_{g \in \widehat{\mathcal{F}_n}} \mathbb{P}(\Phi_n(g_k) - \Phi(g_k) \leq (a_n - a)/2)}.\end{aligned}$$

Moreover, we already proved in Lemma 10 that $s(\widehat{\mathcal{F}_n}, 2n) \leq K^m (2n)^{(d+1)m^2}$. Condition (5) of Theorem 2 then implies that $\mathbb{P}(\Phi(f_{n,k}) \leq a)$ tends to 0.

Now we have to prove that for $f \in \mathcal{F}$ the term $d(f, \mathcal{F}_n) := \min_{g \in \mathcal{F}_n} d(f,g)$ tends to 0 in probability. Let $\widetilde{f}(x) = f(NN_m(x))$. If $\widetilde{f} \in \mathcal{F}_n$ then $d(f, \mathcal{F}_n) \leq d(f, \widetilde{f})$, so the following holds true:

$$\mathbb{P}(d(f, \mathcal{F}_n) \geq \varepsilon) \leq \mathbb{P}(\widetilde{f} \notin \mathcal{F}_n) + \mathbb{P}(d(f, \widetilde{f}) \geq \varepsilon).$$

The second term on the right hand side tends to 0 because of Lemma 11. To deal with the first term on the right hand side, observe that

$$\mathbb{P}(\widetilde{f} \notin \mathcal{F}_n) \leq K \sup_k \mathbb{P}(\Phi_n(\widetilde{f_k}) \leq a_n).$$

Because of Condition (4), for all $\varepsilon > 0, f \in \mathcal{F}$ and $g \in \widehat{\mathcal{F}_n}$ there exists $\delta(\varepsilon) > 0$ such that

$$d(f,g) \leq \delta(\varepsilon) \Rightarrow \Phi(f_k) - \Phi(g_k) \leq \varepsilon.$$

Define $a_n^f := \inf_k \Phi(f_k) - a_n$. Since $f \in \mathcal{F}$ there exists N such that $n \geq N \Rightarrow a_n^f > 0$. For $n \geq N$ we have the following inequalities:

$$\begin{aligned}
&\mathbb{P}(\Phi_n(\widetilde{f_k}) \leq a_n)\\
&= \mathbb{P}(\Phi(f_k) - \Phi_n(\widetilde{f_k}) \geq \Phi(f_k) - a_n)\\
&= \mathbb{P}(\Phi(f_k) - \Phi(\widetilde{f_k}) + \Phi(\widetilde{f_k}) - \Phi_n(\widetilde{f_k}) \geq \Phi(f_k) - a_n)\\
&\leq \mathbb{P}(\Phi(f_k) - \Phi(\widetilde{f_k}) \geq (\Phi(f_k) - a_n)/2) + \mathbb{P}(\Phi(\widetilde{f_k}) - \Phi_n(\widetilde{f_k}) \geq (\Phi(f_k) - a_n)/2)\\
&\leq \mathbb{P}(\Phi(f_k) - \Phi(\widetilde{f_k}) \geq a_n^f/2) + \mathbb{P}(\Phi(\widetilde{f_k}) - \Phi_n(\widetilde{f_k}) \geq a_n^f/2)\\
&\leq \mathbb{P}(d(f, \widetilde{f}) > \delta(a_n^f/2)) + \mathbb{P}(\sup_{g \in \widehat{\mathcal{F}_n}} \Phi(g_k) - \Phi_n(g_k) \geq a_n^f/2)\\
&\leq \frac{2}{\delta(a_n^f/2)} e^{-mb(\delta(a_n^f/2))} + \mathbb{P}(\sup_{g \in \widehat{\mathcal{F}_n}} \Phi(g_k) - \Phi_n(g_k) \geq a_n^f/2).
\end{aligned}$$

If $m \to \infty$ then the first term goes to 0. Indeed, $\delta(a_n^f/2)$ and $b(\delta(a_n^f/2))$ tend to positive constants since $f \in \mathcal{F}$ and thus $a_n^f \to \inf_k \Phi(f_k) - a > 0$. For the second term, the key step is to see that by the techniques used in the proof of Lemma 9 we get

$$\begin{aligned}
&\mathbb{P}(\sup_{g \in \widehat{\mathcal{F}_n}} \Phi(g_k) - \Phi_n(g_k) \geq a_n^f/2)\\
&\leq 2K^m (2n)^{(d+1)m^2} \frac{\sup_{g \in \widehat{\mathcal{F}_n}} \mathbb{P}(\Phi(g_k) - \Phi_n(g_k) \geq a_n^f/8)}{\inf_{g \in \widehat{\mathcal{F}_n}} \mathbb{P}(\Phi(g_k) - \Phi_n(g_k) \leq a_n^f/4)}.
\end{aligned}$$

Under Condition (2) this term tends to 0. ■

A.6 The Proofs of the Consistency Theorems 3, 5, 7 and 8

All these theorems are applications of Theorem 2 to specific objective functions Q_n and Q and to specific functions Φ_n and Φ. For all of them, we individually have to check whether the conditions in Theorem 2 are satisfied. In this section, we do not follow the order of the Theorems in the paper. This is only due to better readability of the proofs.

In most of these proves, we will use the McDiarmid inequality (McDiarmid, 1989), which we recall for the convenience of the reader:

Theorem 12 (McDiarmid inequality) *Let $(X_n)_{n\in\mathbb{N}}$ be a sequence of independent random variables. Let $g : (\mathbb{R}^d)^n \to \mathbb{R}$ be measurable and $c > 0$ a constant such that for all $1 \le i \le n$ we have*

$$\sup_{x_1,\ldots,x_n,x'\in\mathbb{R}^d} g(x_1,\ldots,x_n) - g(x_1,\ldots,x_{i-1},x',x_{i+1},\ldots,x_n) \le c.$$

Then

$$\mathbb{P}(|g(X_1,\ldots,X_n) - \mathbb{E}g(X_1,\ldots,X_n)| \ge \varepsilon) \le 2e^{-\frac{2\varepsilon^2}{nc^2}}.$$

Moreover, several times we will use the fact that $a_n \to a, m \to \infty$ and $\frac{m^2\log n}{n(a-a_n)^2} \to 0$ implies that $n(a-a_n)^2 \to \infty$ and $\frac{m^2\log n}{n} \to 0$.

Before we look at the "combined" objective functions such as Ncut, RatioCut, WSS, we will prove some technical conditions about their "ingredients" cut, vol, $Ef_k(X)$ and WS.

Lemma 13 (Conditions (2), (4), and (5) for cut, vol, $\mathbb{E}f_k(X)$, and WS) *Assume that*

$$\frac{m^2\log n}{n(a-a_n)^2} \to 0$$

then vol, cut, $\mathbb{E}f_k(X)$ *and* WS *satisfy Conditions (2), (4) and (5) of Theorem 2.*

Proof To prove Conditions (2) and (5) we are going to use the McDiarmid inequality. Observe that if one replaces one variable X_i by a new one X_i', then vol_n changes by at most $2C/n$, cut_n changes by at most $2C/n$, $\mathrm{WS}(f_k)$ changes by at most $2C/n$, and $n_k(f)$ changes by at most $1/n$. Using the McDiarmid inequality, this implies that for all $g \in \widehat{\mathcal{F}_n}$ and $\varepsilon > 0$

$$\mathbb{P}(|\mathrm{vol}_n(g_k) - \mathrm{vol}(g_k)| \ge \varepsilon) \le 2e^{-\frac{n\varepsilon^2}{2C^2}},$$

$$\mathbb{P}(|\mathrm{cut}_n(g_k) - \mathrm{cut}(g_k)| \ge \varepsilon) \le 2e^{-\frac{n\varepsilon^2}{2C^2}},$$

$$\mathbb{P}(|\mathrm{WS}_n(g_k) - \mathrm{WS}(g_k)| \ge \varepsilon) \le 2e^{-\frac{n\varepsilon^2}{2C^2}},$$

$$\mathbb{P}(|n_k(g) - \mathbb{E}g_k(X)| \ge \varepsilon) \le 2e^{-2n\varepsilon^2}.$$

So to prove Condition (2) we have to show that

$$\forall\varepsilon > 0, K^m(2n)^{(d+1)m^2}e^{-n\varepsilon} \to 0.$$

This follows clearly from $K^m(2n)^{(d+1)m^2}e^{-n\varepsilon} = e^{-n\left(\frac{m\log K+(d+1)m^2\log(2n)}{-n}+\varepsilon\right)}$ and $\frac{m^2\log n}{n} \to 0$. Moreover, since $n(a-a_n)^2 \to \infty$ Condition (5) is also true.

To prove (4) for each of the objective functions, let $f, g \in \mathcal{H}$ and f_k and g_k be the corresponding cluster indicator functions for cluster k. Then we can see that

$$\begin{aligned}
|\operatorname{vol}(g_k) - \operatorname{vol}(f_k)| &= |\int\int (f_k(X) - g_k(X))s(X,Y)\, dP(X)dP(Y)| \\
&\leq C\int_{\{f_k=g_k\}} 0\, dP(X)dP(Y) + C\int_{\{f_k=g_k\}^c} 1\, dP(X)dP(Y) \\
&= C\mathbb{P}(f_k \neq g_k) \\
&\leq Cd(f,g),
\end{aligned}$$

$$\begin{aligned}
|\operatorname{cut}(g_k) - \operatorname{cut}(f_k)| &= |\int\int f_k(X)(1-f_k(Y))s(X,Y) - g_k(X)(1-g_k(Y))s(X,Y)\, dP(X)dP(Y)\, | \\
&\leq C\int\int_{\{f=g\}^2} 0\, dP(X)dP(Y) + C\int\int_{(\{f=g\}^2)^c} 1\, dP(X)dP(Y) \\
&= C(1-\mathbb{P}(f(X)=g(X))^2) \\
&= C(1-(1-d(f,g))^2) \\
&\leq 2Cd(f,g),
\end{aligned}$$

$$|\mathbb{E}f_k(X) - \mathbb{E}g_k(X)| \leq d(f,g),$$

$$\begin{aligned}
|\operatorname{WS}(f_k) - \operatorname{WS}(g_k)| &= |\int\int (f_k(X)f_k(Y) - g_k(X)g_k(Y))s(X,Y)\, dP(X)dP(Y)\, | \\
&\leq \int_{\{f=g\}^2} 0\, dP(X)dP(Y) + C\int_{(\{f=g\}^2)^c} 1\, dP(X)dP(Y) \\
&= C(1-\mathbb{P}(f=g)^2) \\
&= C(1-(1-d(f,g))^2) \\
&\leq 2Cd(f,g).
\end{aligned}$$

■

Now we are going to check that the "combined" objective functions Ncut, RatioCut, Mod, WSS, BWR satisfy the conditions of Theorem 2. For many of the objective functions, one important step in the proof is to separate the convergence of the whole term into the convergence of the numerator and the denominator.

Lemma 14 (Condition (1) for Ncut) *Assume that*

$$\frac{m^2 \log n}{n} \to 0$$

then Ncut *satisfies Condition (1) of Theorem 2.*

Proof We first want to split the deviations of Ncut into the ones of cut and vol, respectively. To this end we want to show that for any $f \in \widetilde{\mathcal{F}_n}$

$$\{|\operatorname{cut}_n(f_k) - \operatorname{cut}(f_k)| \leq \tfrac{a}{2}\varepsilon\} \bigcap \{|\operatorname{vol}_n(f_k) - \operatorname{vol}(f_k)| \leq \tfrac{a}{2}\varepsilon\}$$

$$\subset \{|\tfrac{\operatorname{cut}_n(f_k)}{\operatorname{vol}_n(f_k)} - \tfrac{\operatorname{cut}(f_k)}{\operatorname{vol}(f_k)}| \leq \varepsilon\}.$$

This can be seen as follows. Assume that $|\operatorname{cut}_n(f_k) - \operatorname{cut}(f_k)| \leq \varepsilon$ and $|\operatorname{vol}_n(f_k) - \operatorname{vol}(f_k)| \leq \varepsilon$. If $\operatorname{vol}(f_k) \neq 0$ then we have (using the facts that $\operatorname{cut}(f_k) \leq \operatorname{vol}(f_k)$ and that $\operatorname{vol}_n(f_k) > a_n > a$ by definition of $\widetilde{\mathcal{F}_n}$):

$$\begin{aligned}
\tfrac{\operatorname{cut}_n(f_k)}{\operatorname{vol}_n(f_k)} - \tfrac{\operatorname{cut}(f_k)}{\operatorname{vol}(f_k)} &= \tfrac{\operatorname{cut}_n(f_k)\operatorname{vol}(f_k) - \operatorname{cut}(f_k)\operatorname{vol}_n(f_k)}{\operatorname{vol}_n(f_k)\operatorname{vol}(f_k)} \\
&\leq \tfrac{(\operatorname{cut}(f_k)+\varepsilon)\operatorname{vol}(f_k) - \operatorname{cut}(f_k)(\operatorname{vol}(f_k)-\varepsilon)}{\operatorname{vol}_n(f_k)\operatorname{vol}(f_k)} \\
&= \tfrac{\varepsilon}{\operatorname{vol}_n(f_k)} \tfrac{\operatorname{cut}(f_k)+\operatorname{vol}(f_k)}{\operatorname{vol}(f_k)} \\
&\leq \tfrac{2\varepsilon}{a}.
\end{aligned}$$

On the other hand, if $\operatorname{vol}(f_k) = 0$ then we have $\operatorname{cut}(f_k) = 0$, which implies $\operatorname{cut}_n(f_k) \leq \varepsilon$ by the assumption above. Thus the following statement holds true:

$$\frac{\operatorname{cut}_n(f)}{\operatorname{vol}_n(f)} - \frac{\operatorname{cut}(f)}{\operatorname{vol}(f)} = \frac{\operatorname{cut}_n(f)}{\operatorname{vol}_n(f)} \leq \frac{\varepsilon}{a} \leq \frac{2\varepsilon}{a}.$$

Using the same technique we have the same bound for $\frac{\operatorname{cut}(f_k)}{\operatorname{vol}(f_k)} - \frac{\operatorname{cut}_n(f_k)}{\operatorname{vol}_n(f_k)}$, which proves our set inclusion.

Now we apply a union bound and the McDiarmid inequality. For the latter, note that if one changes one X_i then $\operatorname{cut}_n(f)$ and $\operatorname{vol}_n(f)$ will change at most by $2C/n$. Together all this leads to

$$\begin{aligned}
&\mathbb{P}(|\operatorname{Ncut}(f) - \operatorname{Ncut}_n(f)| > \varepsilon) \\
&\leq K \sup_k \mathbb{P}(|\frac{\operatorname{cut}_n(f_k)}{\operatorname{vol}_n(f_k)} - \frac{\operatorname{cut}(f_k)}{\operatorname{vol}(f_k)}| > \varepsilon/K) \\
&\leq K \sup_k \Big(\mathbb{P}(|\operatorname{cut}_n(f_k) - \operatorname{cut}(f_k)| > \frac{a}{2K}\varepsilon) + \mathbb{P}(|\operatorname{vol}_n(f_k) - \operatorname{vol}(f_k)| > \frac{a}{2K}\varepsilon)\Big) \\
&\leq 4Ke^{-\frac{na^2\varepsilon^2}{8C^2K^2}}.
\end{aligned}$$

To finish we have to prove that

$$\forall \varepsilon > 0, K^{m+1}(2n)^{(d+1)m^2} e^{-n\varepsilon} \to 0.$$

This follows clearly from $K^{m+1}(2n)^{(d+1)m^2} e^{-n\varepsilon} = e^{-n\left(\frac{(m+1)\log K + (d+1)m^2 \log(2n)}{-n} + \varepsilon\right)}$ and $\frac{m^2 \log n}{n} \to 0$. ■

Lemma 15 (Condition (3) for Ncut**)** Ncut *satisfies Condition (3) of Theorem* 2.

Proof Let $f \in \mathcal{F}, g \in \widetilde{\mathcal{F}_n}$. In the proof of Lemma 13 we have already seen that

$$\begin{aligned} |\,\mathrm{cut}(f_k) - \mathrm{cut}(g_k)| &\leq 2Cd(f,g), \\ |\,\mathrm{vol}(f_k) - \mathrm{vol}(g_k)| &\leq 2Cd(f,g). \end{aligned}$$

If $\mathrm{vol}(g) \neq 0$ then we have (using the fact that we always have $\mathrm{cut}(f) \leq \mathrm{vol}(f)$):

$$\begin{aligned} \tfrac{\mathrm{cut}(f_k)}{\mathrm{vol}(f_k)} - \tfrac{\mathrm{cut}(g_k)}{\mathrm{vol}(g_k)} &= \tfrac{\mathrm{cut}(f_k)\,\mathrm{vol}(g_k) - \mathrm{cut}(g_k)\,\mathrm{vol}(f_k)}{\mathrm{vol}(f_k)\,\mathrm{vol}(g_k)} \\ &\leq \tfrac{(\mathrm{cut}(g_k)+2Cd(f,g))\,\mathrm{vol}(\widetilde{f}) - \mathrm{cut}(g_k)(\mathrm{vol}(g_k)-2Cd(f,g))}{\mathrm{vol}(f_k)\,\mathrm{vol}(g_k)} \\ &= \tfrac{2Cd(f,g)}{\mathrm{vol}(f_k)}\,\tfrac{\mathrm{vol}(g_k)+\mathrm{cut}(g_k)}{\mathrm{vol}(g_k)} \\ &\leq \tfrac{4C}{a} d(f,g)). \end{aligned}$$

On the other hand if $\mathrm{vol}(g_k) = 0$ then we have $|\,\mathrm{cut}(f_k)| \leq |\,\mathrm{vol}(f_k)| \leq 2Cd(f,g)$, in which case the following holds true:

$$\frac{\mathrm{cut}(f_k)}{\mathrm{vol}(f_k)} - \frac{\mathrm{cut}(g_k)}{\mathrm{vol}(\widetilde{f}_k)} = \frac{\mathrm{cut}(f_k)}{\mathrm{vol}(f_k)} \leq \frac{2Cd(f,g)}{a} \leq \frac{4C}{a} d(f,g).$$

So all in all we have

$$\mathrm{Ncut}(f) - \mathrm{Ncut}(g) \leq \frac{4CK}{a} d(f,g).$$

We can use the same technique to bound $\mathrm{Ncut}(g) - \mathrm{Ncut}(f)$. This proves that Ncut is Lipschitz and thus uniformly continuous. ■

Lemma 16 (Condition (1) for RatioCut**)** *Assume that*

$$\frac{m^2 \log n}{n} \to 0$$

then RatioCut *satisfies Condition (1) of Theorem* 2.

Proof Using exactly the same proof as for Lemma 14 (just changing $\mathrm{vol}_n(f_k)$ to n_k and $\mathrm{vol}(f_k)$ to $\mathbb{E} f_k(X)$ and using the fact that $\mathrm{cut}(f_k) \leq C\mathbb{E} f_k(X)$) we get

$$\begin{aligned} &\mathbb{P}(|\,\mathrm{RatioCut}_n(f) - \mathrm{RatioCut}(f)| > \varepsilon) \\ &\leq K \sup_k \left(\mathbb{P}(|\,\mathrm{cut}_n(f_k) - \mathrm{cut}(f_k)| > \frac{a}{(S+1)K}\varepsilon) + \mathbb{P}(|n_k(f) - \mathbb{E} f_k(X)| > \frac{a}{(S+1)K}\varepsilon) \right). \end{aligned}$$

Now a simple McDiarmid argument (using again the fact that changing one X_i changes cut_n by at most $2S/n$) gives

$$\mathbb{P}(|\,\mathrm{RatioCut}_n(f) - \mathrm{RatioCut}(f)| > \varepsilon) \leq 2Ke^{-\frac{na^2\varepsilon}{8C^2K^2}} + 2Ke^{-\frac{na^2\varepsilon^2}{2K^2}}.$$

We conclude the proof with the same argument as in Lemma 14. ■

Lemma 17 (Condition (3) for RatioCut) RatioCut *satisfies Condition (3) of Theorem* 2.

Proof This follows by the same proof as Lemma 14, just changing $\mathrm{vol}_n(f_k)$ to n_k, $\mathrm{vol}(f_k)$ to $\mathbb{E}f_k(X)$ and using the fact that $\mathrm{cut}(f_k) \leq C\mathbb{E}f_k(X)$. ■

Lemma 18 (Condition (1) for BWR) *If* $m^2 \log n/n \to 0$, *then* BWR *satisfies Condition (1) of Theorem* 2.

Proof Let $f \in \widetilde{\mathcal{F}_n}$. Let $\varepsilon \leq a/2$. If $|\mathrm{WS}_n(f_k) - \mathrm{WS}(f_k)| \leq \varepsilon$ and $|\mathrm{cut}_n(f_k) - \mathrm{cut}(f_k)| \leq \varepsilon$ then $\mathrm{WS}(f_k) \geq a/2 > 0$ (because $\mathrm{WS}_n(f_k) > a_n > a$ since $f \in \widetilde{\mathcal{F}_n}$). This implies

$$\begin{aligned}
\frac{\mathrm{cut}_n(f_k)}{\mathrm{WS}_n(f_k)} - \frac{\mathrm{cut}(f_k)}{\mathrm{WS}(f_k)} &= \frac{\mathrm{WS}(f_k)\,\mathrm{cut}_n(f_k) - \mathrm{WS}_n(f_k)\,\mathrm{cut}(f_k)}{\mathrm{WS}_n(f_k)\,\mathrm{WS}(f_k)} \\
&\leq \frac{\mathrm{WS}(f_k)(\mathrm{cut}(f_k)+\varepsilon) - (\mathrm{WS}(f_k)-\varepsilon)\,\mathrm{cut}(f_k)}{\mathrm{WS}_n(f_k)\,\mathrm{WS}(f_k)} \\
&= \frac{\varepsilon}{\mathrm{WS}_n(f_k)} \frac{\mathrm{WS}(f_k)+\mathrm{cut}(f_k)}{\mathrm{WS}(f_k)} \\
&\leq \frac{2C\varepsilon}{a^2}.
\end{aligned}$$

The analogous statement holds for $\frac{\mathrm{cut}(f_k)}{\mathrm{WS}(f_k)} - \frac{\mathrm{cut}_n(f_k)}{\mathrm{WS}_n(f_k)}$. Thus, if $\varepsilon \leq C/a$ then

$$\begin{aligned}
&\{|\mathrm{WS}_n(f_k) - \mathrm{WS}(f_k)| \leq a^2\varepsilon/(2C)\} \cap \{|\mathrm{cut}_n(f_k) - \mathrm{cut}(f_k)| \leq a^2\varepsilon/(2C)\} \\
&\subset \left\{\left|\frac{\mathrm{cut}_n(f_k)}{\mathrm{WS}_n(f_k)} - \frac{\mathrm{cut}(f_k)}{\mathrm{WS}(f_k)}\right| \leq \varepsilon\right\}.
\end{aligned}$$

As a consequence, if $\varepsilon \leq CK/a$ we have

$$\begin{aligned}
&\mathbb{P}(|\mathrm{BWR}_n(f) - \mathrm{BWR}(f)| > \varepsilon) \leq K \sup_k \mathbb{P}\left(\left|\frac{\mathrm{cut}_n(f_k)}{\mathrm{WS}_n(f_k)} - \frac{\mathrm{cut}(f_k)}{\mathrm{WS}(f_k)}\right|| > \varepsilon/K\right) \\
&\quad \leq K \sup_k \left(\mathbb{P}(|\mathrm{WS}_n(f_k) - \mathrm{WS}(f_k)| > a^2\varepsilon/(2CK)) + \mathbb{P}(|\mathrm{cut}_n(f_k) - \mathrm{cut}(f_k)| > a^2\varepsilon/(2CK))\right).
\end{aligned}$$

Using the McDiarmid inequality together with the fact that changing one point changes cut_n and WS_n by at most $C/(2n)$, we get for $\varepsilon \leq CK/a$:

$$\mathbb{P}(|\mathrm{BWR}_n(f) - \mathrm{BWR}(f)| > \varepsilon) \leq 4Ke^{-\frac{na^4\varepsilon^2}{8C^4K^2}}.$$

On the other hand, for $\varepsilon > CK/a$ we have

$$\begin{aligned}
&\mathbb{P}(|\mathrm{BWR}_n(f) - \mathrm{BWR}(f)| > \varepsilon) \\
&\leq \mathbb{P}(|\mathrm{BWR}_n(f) - \mathrm{BWR}(f)| > SK/a) \\
&\leq 4Ke^{-\frac{na^4(SK/a)^2}{8C^4K^2}}.
\end{aligned}$$

So all in all we have proved that

$$\mathbb{P}(|\,\mathrm{BWR}_n(f)-\mathrm{BWR}(f)|>\varepsilon)\leq 2Ke^{-\frac{na^4(\min(\varepsilon,CK/a))^2}{8C^4K^2}}.$$

We conclude the proof with the same argument as in Lemma 14. ■

Lemma 19 (Condition (3) for BWR) BWR *satisfies Condition (3) of Theorem 2.*

Proof Let $\varepsilon>0$, $f\in\mathcal{F}$ and $g\in\widetilde{\mathcal{F}_n}$. We have already proved the two following inequalities (in the proofs of Lemmas 13 and 15):

$$|\,\mathrm{cut}(f_k)-\mathrm{cut}(g_k)|\leq 2Cd(f,g),$$

$$|\,\mathrm{WS}(f_k)-\mathrm{WS}(g_k)|\leq 2Cd(f,g).$$

If $2Cd(f,g)\leq a/2$, then using that $\mathrm{WS}(f_k)>a$ we get $\mathrm{WS}(g_k)\geq a/2>0$. By the same technique as at the beginning of Lemma 18 we get

$$|\,\mathrm{BWR}(f)-\mathrm{BWR}(g)|\leq\frac{2CK}{a^2}2Cd(f,g).$$

Written a bit differently,

$$d(f,g)\leq\frac{a}{4C}\Rightarrow|\,\mathrm{BWR}(f)-\mathrm{BWR}(g)|\leq\frac{4C^2K}{a^2}d(f,g).$$

Now recall that we want to prove that there exists $\delta>0$ such that $d(f,g)\leq\delta\Rightarrow|\,\mathrm{BWR}(f)-\mathrm{BWR}(g)|\leq\varepsilon$.

If $\varepsilon\leq CK/a$ then we have:

$$d(f,g)\leq\frac{a^2}{4C^2K}\varepsilon\leq\frac{a}{4C}\Rightarrow|\,\mathrm{BWR}(f)-\mathrm{BWR}(g)|\leq\frac{4C^2K}{a^2}d(f,g)\leq\varepsilon.$$

On the other hand, if $\varepsilon>CK/a$ then

$$d(f,g)\leq\frac{a}{4C}\Rightarrow|\,\mathrm{BWR}(f)-\mathrm{BWR}(g)|\leq\frac{4C^2K}{a^2}d(f,g)\leq CK/a\leq\varepsilon$$

so we have proved the lemma. ■

Lemma 20 (Condition (1) for WSS) *If* $\frac{m^2\log n}{n}\to 0$ *and that* $\operatorname{supp}\mathbb{P}\subset B(0,A)$, *then* WSS *satisfies Condition (1) of Theorem 2.*

Proof Let $f\in\widetilde{\mathcal{F}_n}$. First note that

$$\begin{aligned}|\,\mathrm{WSS}_n(f)-\mathrm{WSS}(f)|&=\left|\frac{1}{n}\sum_{i=1}^n\sum_{k=1}^K f_k(X_i)\|X_i-c_{k,n}\|^2-\mathbb{E}\sum_{k=1}^K f_k(X)\|X-c_k\|^2\right|\\&\leq\left|\frac{1}{n}\sum_{i=1}^n\sum_{k=1}^K f_k(X_i)\|X_i-c_{k,n}\|^2-\frac{1}{n}\sum_{i=1}^n\sum_{k=1}^K f_k(X_i)\|X_i-c_k\|^2\right|\\&\quad+\left|\frac{1}{n}\sum_{i=1}^n\sum_{k=1}^K f_k(X_i)\|X_i-c_k\|^2-\mathbb{E}\sum_{k=1}^K f_k(X)\|X-c_k\|^2\right|.\end{aligned}$$

Now we will bound the probability for each of the terms on the right hand side. For the second term we can simply apply McDiarmid's inequality. Due to the assumption that $\operatorname{supp}\mathbb{P} \subset B(0,A)$ we know that for any two points $x, y \in \operatorname{supp}\mathbb{P}$ we have $\|x-y\| \leq 2A$. Thus if one changes one variable X_i then the term $\frac{1}{n}\sum_{i=1}^n \sum_{k=1}^K f_k(X_i)\|X_i - c_k\|^2$ will change by at most $A^2/(4n)$. This leads to

$$\mathbb{P}\left(\left| \frac{1}{n}\sum_{i=1}^n \sum_{k=1}^K f_k(X_i)\|X_i - c_k\|^2 - \mathbb{E}\sum_{k=1}^K f_k(X)\|X - c_k\|^2 \right| \geq \varepsilon \right) \leq 2e^{-\frac{2n\varepsilon^2}{A^4}}.$$

Now we have to take care of the first term, which can be written as

$$\frac{1}{n}\sum_{i=1}^n \sum_{k=1}^K f_k(X_i)\left(\|X_i - c_{k,n}\|^2 - \|X_i - c_k\|^2\right).$$

The triangle inequality gives

$$\|X_i - c_{k,n}\|^2 \leq (\|X_i - c_k\| + \|c_{k,n} - c_k\|)^2,$$

and together with the fact that $\operatorname{supp}\mathbb{P} \subset B(0,A)$ this leads to

$$\|X_i - c_{k,n}\|^2 - \|X_i - c_k\|^2 \leq 6A\|c_{k,n} - c_k\|.$$

So at this point we have

$$\left| \frac{1}{n}\sum_{i=1}^n \sum_{k=1}^K f_k(X_i)\|X_i - c_{k,n}\|^2 - \|X_i - c_k\|^2 \right| \leq 6A \sup_k \|c_{k,n} - c_k\|.$$

We will denote the j-th coordinate of a vector X by X^j. Recall that d denotes the dimensionality of our space. Using this notation we have

$$\|c_{k,n} - c_k\|^2 = \sum_{j=1}^d \left(\frac{\mathbb{E}f_k(X)X^j}{\mathbb{E}f_k(X)} - \frac{1}{n_k}\frac{1}{n}\sum_{i=1}^n f_k(X_i)X_i^j \right)^2.$$

Our goal will be to apply the McDiarmid inequality to each coordinate. Before we can do this, we want to show that

$$\{|n_k - \mathbb{E}f_k(X)| \leq \tfrac{a\varepsilon}{A+1}\} \;\cap\; \{|\tfrac{1}{n}\sum_{i=1}^n f_k(X_i)X_i^j - \mathbb{E}f_k(X)X^j| \leq \frac{a\varepsilon}{A+1}\} \;\subset\; \{|c_k^j - c_{k,n}^j| \leq \varepsilon\}.$$

To this end, assume that $|n_k - \mathbb{E}f_k(X)| \leq \varepsilon$ and $|\frac{1}{n}\sum_{i=1}^n f_k(X_i)X_i^j - \mathbb{E}f_k(X)X^j| \leq \varepsilon$.

In case $\mathbb{E}f_k(X) \neq 0$ we have

$$\begin{aligned}
c_k^j - c_{k,n}^j &= \frac{n_k\mathbb{E}f_k(X)X^j - \mathbb{E}f_k(X)\frac{1}{n_k}\frac{1}{n}\sum_{i=1}^n f_k(X_i)X_i^j}{n_k\mathbb{E}f_k(X)} \\
&\leq \frac{(\mathbb{E}f_k(X)+\varepsilon)\mathbb{E}f_k(X)X^j - \mathbb{E}f_k(X)(\mathbb{E}f_k(X)X^j - \varepsilon)}{n_k\mathbb{E}f_k(X)} \\
&= \frac{\varepsilon}{n_k}\frac{\mathbb{E}f_k(X)X^j + \mathbb{E}f_k(X)}{\mathbb{E}f_k(X)} \\
&\leq \frac{(A+1)\varepsilon}{a}
\end{aligned}$$

and similarly for $c^j_{k,n} - c^j_k$.

On the other hand, in case $\mathbb{E} f_k(X) = 0$ we also have $\mathbb{E} f_k(X) X^j = 0$ (as f_k is a non-negative function and $|X|$ is bounded by A). Together with the assumption this means that $\frac{1}{n}\sum_{i=1}^n f_k(X_i) X_i^j \leq \varepsilon$. This implies

$$|c^j_k - c^j_{k,n}| = \frac{1}{n_k}\frac{1}{n}\sum_{i=1}^n f_k(X_i)X_i^j \leq \frac{\varepsilon}{a} \leq \frac{(A+1)\varepsilon}{a}$$

which shows the inclusion stated above. The McDiarmid inequality now yields the two statements

$$\mathbb{P}(|n_k - \mathbb{E} f_k(X)| > \varepsilon) \leq 2e^{-2n\varepsilon^2},$$
$$\mathbb{P}\left(\left|\frac{1}{n}\sum_{i=1}^n f_k(X_i)X_i^j - \mathbb{E} f_k(X)X^j\right| > \varepsilon\right) \leq 2e^{-\frac{2n\varepsilon^2}{A^2}}.$$

Together they show that for the coordinate-wise differences

$$\mathbb{P}(|c^j_k - c^j_{k,n}| > \varepsilon) \leq 2e^{-\frac{2na^2\varepsilon^2}{(A+1)^2}} + 2e^{-\frac{2na^2\varepsilon^2}{A^2(A+1)^2}} \leq 4e^{-\frac{2na^2\varepsilon^2}{\max(1,A^2)(A+1)^2}}.$$

This leads to

$$\begin{aligned}\mathbb{P}(\|c_k - c_{k,n}\| > \varepsilon) &= \mathbb{P}(\sum_{j=1}^d |c^j_k - c^j_{k,n}|^2 > \varepsilon^2) \leq d \sup_j \mathbb{P}(|c^j_k - c^j_{k,n}| > \varepsilon/\sqrt{d}) \\ &\leq 4de^{-\frac{2na^2\varepsilon^2}{d\max(1,A^2)(A+1)^2}}.\end{aligned}$$

Combining all this leads to a bound for the first term of the beginning of the proof:

$$\begin{aligned}&\mathbb{P}\left(\left|\tfrac{1}{n}\textstyle\sum_{i=1}^n \sum_{k=1}^K f_k(X_i)\left(\|X_i - c_{k,n}\|^2 - \|X_i - c_k\|^2\right)\right| \geq \varepsilon\right) \\ &\leq \mathbb{P}\left(\sup_k \|c_{k,n} - c_k\| \geq \varepsilon/(6A)\right) \\ &\leq K \sup_k \mathbb{P}(\|c_{k,n} - c_k\| \geq \varepsilon/(6A)) \\ &\leq 4dKe^{-\frac{na^2\varepsilon^2}{18d\max(1,A^2)A^2(A+1)^2}}.\end{aligned}$$

Now we combine the probabilities for the first and the second term from the beginning of the proof using a union bound to get

$$\mathbb{P}(|\operatorname{WSS}_n(f) - \operatorname{WSS}(f)| > \varepsilon) \leq 4dKe^{-\frac{na^2\varepsilon}{18d\max(1,A^2)A^2(A+1)^2}} + 2e^{-\frac{8n\varepsilon^2}{A^4}}.$$

We conclude the proof with the same argument as in Lemma 14. ■

Lemma 21 (Condition (3) for WSS) *Assume that* $\operatorname{supp}\mathbb{P} \subset B(0,A)$ *then* WSS *satisfies Condition (3) of Theorem 2.*

Proof Let $f \in \mathcal{F}, g \in \widetilde{\mathcal{F}_n}$. We begin with the following inequality, which can be seen by splitting the expectation in the part where $\{f = g\}$ and $\{f \neq g\}$ and using the fact that $\operatorname{supp} \mathbb{P} \subset B(0,A)$:

$$\begin{aligned} |\operatorname{WSS}(f) - \operatorname{WSS}(g)| &= |\mathbb{E}\textstyle\sum_{k=1}^K f_k(X)\|X - c_k(f)\|^2 - g_k(X)\|X - c_k(g)\|^2| \\ &\leq 4A^2 d(f,g) + \textstyle\int_{\{f=g\}} \sum_{k=1}^K f_k(X)\left(\|X - c_k(f)\|^2 - \|X - c_k(g)\|^2\right). \end{aligned}$$

For the second term we have already seen in the proof of the previous lemma that $\|X - c_k(f)\|^2 - \|X - c_k(g)\|^2 \leq 6A\|c_k(f) - c_k(g)\|$. So for the moment we have

$$|\operatorname{WSS}(f) - \operatorname{WSS}(g)| \leq 4A^2 d(f,g) + 6A \sup_k \|c_k(f) - c_k(g)\|.$$

Now we want to bound the expression $\|c_k(f) - c_k(g)\|$. First of all, observe that $|\mathbb{E}f_k(X) - g_k(X)| \leq d(f,g)$ and $\|\mathbb{E}f_k(X)X - g_k(X)X\| \leq Ad(f,g)$.

In case $\mathbb{E}g_k(X) \neq 0$ we have

$$\begin{aligned} \|c_k(f) - c_k(g)\| &= \frac{\|\mathbb{E}g_k(X)\mathbb{E}f_k(X)X - \mathbb{E}f_k(X)\mathbb{E}g_k(X)X\|}{\mathbb{E}f_k(X)\mathbb{E}g_k(X)} \\ &\leq \frac{\|\mathbb{E}g_k(X)\left(\mathbb{E}f_k(X)X - \mathbb{E}g_k(X)X\right)\| + \|\left(\mathbb{E}g_k(X) - \mathbb{E}f_k(X)\right)\mathbb{E}g_k(X)X\|}{\mathbb{E}f_k(X)\mathbb{E}g_k(X)} \\ &\leq \frac{\mathbb{E}g_k(X)\|\mathbb{E}f_k(X)X - g_k(X)X\| + A\mathbb{E}g_k(X)|\mathbb{E}g_k(X) - f_k(X)|}{\mathbb{E}f_k(X)\mathbb{E}g_k(X)} \\ &\leq \frac{2A}{\mathbb{E}f_k(x)} d(f,g) \\ &\leq \frac{2A}{a} d(f,g). \end{aligned}$$

On the other hand, in case $\mathbb{E}g_k(X) = 0$ we also have $\mathbb{E}g_k(X)X = 0$ (as g_k is a non-negative function and $|X|$ is bounded by A). This leads to

$$\|c_k(f) - c_k(g)\| = \|\frac{\mathbb{E}f_k(X)X}{\mathbb{E}f_k(X)} - \frac{\mathbb{E}g_k(X)X}{\mathbb{E}g_k(X)}\| = \|\frac{\mathbb{E}f_k(X)X}{\mathbb{E}f_k(X)}\| \leq \frac{A}{a} d(f,g) \leq \frac{2A}{a} d(f,g).$$

Combining all results leads to

$$|\operatorname{WSS}(f) - \operatorname{WSS}(g)| \leq 4A^2(1 + 3/a)d(f,g)$$

which proves the lemma. ■

Lemma 22 (Condition (1) **for** Mod) *If* $m^2 \log n/n \to 0$, *then* Mod *satisfies Condition* (1) *of Theorem* 2.

Proof Let $f \in \widetilde{f}$. Using McDiarmid inequality one can prove

$$\mathbb{P}(|\sum_{k=1}^K \frac{1}{n(n-1)} \sum_{i \neq j} f_k(X_i) f_k(X_j) s(X_i, X_j) - \sum_{k=1}^K \mathbb{E}f_k(X) f_k(Y) s(X,Y)| \geq \varepsilon) \leq 2e^{-\frac{n\varepsilon^2}{2C^2K^2}}.$$

Now for ease of notation let

$$\begin{aligned}
Q_n(f) &= \frac{1}{n(n-1)^3}\sum_{k=1}^{K}\sum_{i\neq j} f_k(X_i)f_k(X_j)\sum_{l,l\neq i} s(X_i,X_l)\sum_{l,l\neq j} s(X_j,X_l),\\
\widetilde{Q_n}(f) &= \frac{1}{n(n-1)}\sum_{k=1}^{K}\sum_{i\neq j} f_k(X_i)f_k(X_j)\int s(X_i,Z)d\mathbb{P}(Z)\int s(X_j,Z)d\mathbb{P}(Z),\\
Q(f) &= \sum_{k=1}^{K}\int\int f_k(X)f_k(Y)\int s(X,Z)d\mathbb{P}(Z)\int s(Y,Z)d\mathbb{P}(Z)d(\mathbb{P}\times\mathbb{P})(X,Y).
\end{aligned}$$

If we have an exponential bound for $\mathbb{P}(|Q_n(f)-Q(f)|\geq\varepsilon)$ then with the above bound we would have an exponential bound for $\mathbb{P}(|\mathrm{Mod}_n(f)-\mathrm{Mod}(f)|\geq\varepsilon)$. Thus with the same argument than the one at the end of Lemma 14 the current lemma will be proved.
First note that

$$\mathbb{P}(|Q_n(f)-Q(f)|\geq\varepsilon)\leq\mathbb{P}(|Q_n(f)-\widetilde{Q_n}(f)|\geq\varepsilon/2)+\mathbb{P}(|\widetilde{Q_n}(f)-Q(f)|\geq\varepsilon/2).$$

Moreover $\mathbb{E}\widetilde{Q_n}(f)=Q(f)$ and thus with McDiarmid one can prove that

$$\mathbb{P}(|\widetilde{Q_n}(f)-Q(f)|\geq\varepsilon)\leq 2e^{\frac{n\varepsilon^2}{2C^4K^2}}.$$

The next step is to use the fact that for real numbers $a,b,a_n,b_n\in B(0,C)$,

$$|ab-a_nb_n|=|ab-a_nb+a_nb-a_nb_n|\leq C(|a-a_n|+|b-b_n|).$$

This implies the following inequalities:

$$\begin{aligned}
&|Q_n(f)-\widetilde{Q_n}(f)|\\
&\leq\frac{K}{n(n-1)}\sum_{i\neq j}\left|\frac{1}{(n-1)^2}\sum_{l,l\neq i}s(X_i,X_l)\sum_{l,l\neq j}s(X_j,X_l)-\int s(X_i,Z)d\mathbb{P}(Z)\int s(X_j,Z)d\mathbb{P}(Z)\right|\\
&\leq 2CK\sup_i\left|\frac{1}{n-1}\sum_{l,l\neq i}s(X_i,X_l)-\int s(X_i,Z)d\mathbb{P}(Z)\right|.
\end{aligned}$$

Hence the following:

$$\begin{aligned}
\mathbb{P}(|Q_n(f)-\widetilde{Q_n}(f)|\geq\varepsilon)&\leq\mathbb{P}(\sup_i|\frac{1}{n-1}\sum_{l,l\neq i}s(X_i,X_l)-\int s(X_i,Z)d\mathbb{P}(Z)|\geq\varepsilon/(2CK))\\
&\leq n\sup_i\mathbb{P}(|\frac{1}{n-1}\sum_{l,l\neq i}s(X_i,X_l)-\int s(X_i,Z)d\mathbb{P}(Z)|\geq\varepsilon/(2CK)).
\end{aligned}$$

Now to bound the last term we condition on X_i and use the McDiarmid inequality. Then taking the expectation yields the exponential bound:

$$\begin{aligned}
&\mathbb{P}(|\frac{1}{n-1}\sum_{l,l\neq i} s(X_i,X_l) - \int s(X_i,Z)d\mathbb{P}(Z)| \geq \varepsilon/(2CK)) \\
&= \mathbb{E}(\mathbb{P}(|\frac{1}{n-1}\sum_{l,l\neq i} s(X_i,X_l) - \int s(X_i,Z)d\mathbb{P}(Z)| \geq \varepsilon/(2CK)|X_i)) \\
&\leq \mathbb{E}(2e^{-\frac{n\varepsilon^2}{2C^4K^2}}) \\
&= 2e^{-\frac{n\varepsilon^2}{2C^4K^2}}.
\end{aligned}$$

All in all we proved that

$$\mathbb{P}(|\mathrm{Mod}_n(f) - \mathrm{Mod}(f)| \geq \varepsilon) \leq 2e^{-\frac{n\varepsilon^2}{8C^2K^2}} + 2(n+1)e^{-\frac{n\varepsilon^2}{32C^2K^2}}.$$

The n in front of the exponential obviously does not matter for the limit, see end of the proof of Lemma 14. ■

Lemma 23 (Condition (3) **for** Mod**)** Mod *satisfies Condition (*3*) of Theorem* 2.

Proof Let $f \in \mathcal{F}, g \in \widetilde{\mathcal{F}_n}$. Following the proof of Lemma 15 we have:

$$\begin{aligned}
|\mathrm{Mod}(f) - \mathrm{Mod}(g)| &\leq \sum_{k=1}^{K} \int\int_{(\{f=g\}^2)^c} (C + C^2) \\
&= K(C+C^2)(1-(1-d(f,g))^2) \\
&\leq 2K(C+C^2)d(f,g).
\end{aligned}$$

■

A.7 The Proofs of the Convergence Rates in Theorems 4 and 6

The following lemma collects all the bounds given in the previous proofs for WSS. Whenever possible, we used the one-sided McDiarmid inequality.

Lemma 24 *Assume that* $\operatorname{supp}\mathbb{P} \subset B(0,A)$ *for some constant* $A > 0$. *Let* $a_n^* := \inf_k \mathbb{E} f_k^*(X) - a_n$. *Then* $a_n^* \to a^* := \inf_k \mathbb{E} f_k^*(X) - a > 0$. *For all n and* $\varepsilon > 0$ *there exists a constant* $b(a_n^*/2)$ *which tends to a constant* $C' > 0$ *when* $n \to \infty$, *and a constant* $b(\varepsilon/(8A^2(1+3/a)))$ *(see Lemma* 11 *for more details about b) such that the following holds true*

$$\begin{aligned}
&\mathbb{P}(|\mathrm{WSS}(f_n) - \mathrm{WSS}(f^*)| \geq \varepsilon) \\
&\leq 2K^{m+1}(2n)^{(d+1)m^2}\left(\frac{4dKe^{-\frac{na^2\varepsilon}{616d\max(1,A^2)A^2(A+1)^2}} + 2e^{-\frac{n\varepsilon^2}{32A^4}}}{1-4dKe^{-\frac{na^2\varepsilon}{308d\max(1,A^2)A^2(A+1)^2}} - 2e^{-\frac{n\varepsilon^2}{8A^4}}} + \frac{Ke^{-\frac{n(a_n-a)^2}{8}}}{1-e^{-\frac{n(a_n-a)^2}{2}}} + \frac{Ke^{-\frac{na_n^{*2}}{32}}}{1-e^{-\frac{na_n^{*2}}{8}}}\right) \\
&+\frac{4K}{a_n^*}e^{-mb(a_n^*/2)} + (16A^2(1+3/a)/\varepsilon)e^{-mb(\varepsilon/(8A^2(1+3/a)))}.
\end{aligned}$$

A.8 Proof of Theorem 4

First we take care of the last two terms. There exists N' which depends on the rate of convergence of a_n and on a^* such that for $n \geq N'$ we have

$$a_n^* \leq a^*/2.$$

This implies $b(a_n^*/2) \leq b(a^*/4)$ (see Lemma 11 for details). Now let $C_1' := b(\varepsilon/(8A^2(1+3/a)))$ and $C_2' := b(a^*/4)$. Then for $n \geq N'$ we have:

$$\begin{aligned}
&\frac{4K}{a_n^*}e^{-mb(a_n^*/2)} + (16A^2(1+3/a)/\varepsilon)e^{-mb(\varepsilon/(8A^2(1+3/a)))}\\
&\leq 8Ka^*e^{-C_2'm} + (16A^2(1+3/a)/\varepsilon)e^{-C_1'm}\\
&\leq C_1e^{-C_2m}
\end{aligned}$$

with

$$C_1 := \max(8Ka^*; 16A^2(1+3/a)/\varepsilon) \qquad \text{and} \qquad C_2 := \min(C_1'; C_2').$$

C_2 is a positive constant which depends on a, a^*, A, ε and $\mathbb{P}$. C_1 depends on K, a, a^*, ε and A.

Since we assume $n(a_n - a)^2 \to \infty$ there exists N'' which depends on the rate of convergence of a_n and on a^* such that $n \geq N''$ implies:

$$e^{-\frac{n(a_n-a)^2}{8}} \leq 1/2 \qquad \text{and} \qquad e^{-\frac{na_n^*}{32}} \leq e^{-\frac{n(a_n-a)^2}{8}}.$$

This means that for $n \geq N''$:

$$\frac{Ke^{-\frac{n(a_n-a)^2}{8}}}{1-e^{-\frac{n(a_n-a)^2}{2}}} + \frac{Ke^{-\frac{na_n^{*2}}{32}}}{1-e^{-\frac{na_n^{*2}}{8}}} \leq 4Ke^{-\frac{n(a_n-a)^2}{8}}.$$

Finally let $N = \max(N', N'')$ and

$$\begin{aligned}
C_3 &:= \frac{8dK}{1-4dKe^{-\frac{Na^2\varepsilon}{308d\max(1,A^2)A^2(A+1)^2}} - 2e^{-\frac{N\varepsilon^2}{8A^4}}}\\
C_4 &:= \min\left(\frac{a^2}{616d\max(1,A^2)A^2(A+1)^2}; \frac{1}{32A^4}\right).
\end{aligned}$$

Since $\varepsilon \leq 1$ we have with these notations for $n \geq N$:

$$\frac{4dKe^{-\frac{na^2\varepsilon}{616d\max(1,A^2)A^2(A+1)^2}} + 2e^{-\frac{n\varepsilon^2}{32A^4}}}{1-4dKe^{-\frac{na^2\varepsilon}{308d\max(1,A^2)A^2(A+1)^2}} - 2e^{-\frac{n\varepsilon^2}{8A^4}}} \leq (C_3/2)e^{-C_4\varepsilon^2n}.$$

All in all Theorem 4 is proved. ■

The **Proof of Theorem 6** works analogously, we just replace the above lemma by to following one:

Lemma 25 *Assume that the similarity function s is bounded by $C > 0$. Let $a_n^* := \inf_k \operatorname{vol}(f_k^*) - a_n$. Then $a_n^* \to \inf_k \operatorname{vol}(f_k^*) - a > 0$. For all n and $\varepsilon > 0$ there exists a constant $b(a_n^*/(2S))$ which tends to a constant $C' > 0$ when $n \to \infty$, and a constant $b(a\varepsilon/(8SK))$ (see Lemma 11 for more details about b) such that the following holds true*

$$\mathbb{P}(|\operatorname{Ncut}(f_n) - \operatorname{Ncut}(f^*)| \geq \varepsilon)$$

$$\leq 2K^{m+1}(2n)^{(d+1)m^2}\left(\frac{4e^{-\frac{na^2\varepsilon^2}{2048C^2K^2}}}{1-4Ke^{-\frac{na^2\varepsilon^2}{512C^2K^2}}} + \frac{e^{-\frac{n(a_n-a)^2}{32C^2}}}{1-e^{-\frac{n(a_n-a)^2}{8C^2}}} + \frac{e^{-\frac{na_n^{*2}}{128C^2}}}{1-e^{-\frac{na_n^{*2}}{32C^2}}}\right)$$

$$+\frac{4CK}{a_n^*}e^{-mb(a_n^*/(2C))} + \frac{16CK}{a\varepsilon}e^{-mb(a\varepsilon/(8CK))}.$$

References

S. Ben-David. A framework for statistical clustering with constant time approximation algorithms for k-median and k-means clustering. *Machine Learning*, 66:243 – 257, 2007.

J. Buhmann. Empirical risk approximation: An induction principle for unsupervised learning. Technical report, University of Bonn, 1998.

A. Czumaj and C. Sohler. Sublinear-time approximation algorithms for clustering via random sampling. *Random Struct. Algorithms*, 30(1-2):226–256, 2007.

L. Devroye, L. Györfi, and G. Lugosi. *A Probabilistic Theory of Pattern Recognition*. Springer, New York, 1996.

M. Figueiredo and A. Jain. Unsupervised learning of finite mixture models. *PAMI*, 24(3):381–396, 2002.

C. Fraley and A. Raftery. How many clusters? Which clustering method? Answers via model-based cluster analysis. *Comput. J*, 41(8):578–588, 1998.

J. Fritz. Distribution-free exponential error bound for nearest neighbor pattern classification. *IEEE Trans. Inf. Th.*, 21(5):552 – 557, 1975.

M. Garey, D. Johnson, and H. Witsenhausen. The complexity of the generalized Lloyd - max problem (corresp.). *IEEE Trans. Inf. Theory*, 28(2):255–256, 1982.

P. Grünwald. *The Minimum Description Length Principle*. MIT Press, Cambridge, MA, 2007.

S. Guattery and G. Miller. On the quality of spectral separators. *SIAM Journal of Matrix Anal. Appl.*, 19(3):701 – 719, 1998.

J. Hartigan. Consistency of single linkage for high-density clusters. *JASA*, 76(374):388 – 394, 1981.

J. Hartigan. Statistical theory in clustering. *Journal of Classification*, 2:63 – 76, 1985.

M. Inaba, N. Katoh, and H. Imai. Applications of weighted Voronoi diagrams and randomization to variance-based *k*-clustering. In *Proceedings of the 10th Annual Symposium on Computational Geometry*, pages 332–339. ACM Press, Stony Brook, USA, 1994.

P. Indyk. Sublinear time algorithms for metric space problems. In *Proceedings of the Thirty-first Annual ACM Symposium on Theory of Computing (STOC)*, pages 428–434. ACM Press, New York, 1999.

S. Jegelka. Statistical learning theory approaches to clustering. Master's thesis, University of Tübingen, 2007. Available at http://www.kyb.mpg.de/publication.html?user=jegelka.

R. Kannan, S. Vempala, and A. Vetta. On clusterings: Good, bad and spectral. *Journal of the ACM*, 51(3):497–515, 2004.

C. McDiarmid. On the method of bounded differences. *Surveys in Combinatorics*, pages 148 – 188, 1989. Cambridge University Press.

G. McLachlan and D. Peel. *Finite Mixture Models*. John Wiley, New York, 2004.

N. Mishra, D. Oblinger, and L. Pitt. Sublinear time approximate clustering. In *Proceedings of the Twelfth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA-01)*, pages 439–447. ACM Press, New York, 2001.

M. Newman. Finding community structure in networks using the eigenvectors of matrices. *Physical Review E*, 74:036104, 2006.

D. Pollard. Strong consistency of k-means clustering. *Annals of Statistics*, 9(1):135 – 140, 1981.

A. Rakhlin and A. Caponnetto. Stability of k-means clustering. In B. Schölkopf, J. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems 19*. MIT Press, Cambridge, MA, 2007.

D. Spielman and S. Teng. Spectral partitioning works: planar graphs and finite element meshes. In *37th Annual Symposium on Foundations of Computer Science (Burlington, VT, 1996)*, pages 96 – 105. IEEE Comput. Soc. Press, Los Alamitos, CA, 1996. (See also extended technical report version.).

A. W. van der Vaart and J. A. Wellner. *Weak Convergence and Empirical Processes*. Springer, New York, 1996.

V. Vapnik. *The Nature of Statistical Learning Theory*. Springer Verlag, New York, 1995.

U. von Luxburg. A tutorial on spectral clustering. *Statistics and Computing*, 17(4):395 – 416, 2007.

U. von Luxburg and S. Ben-David. Towards a statistical theory of clustering. In *PASCAL workshop on Statistics and Optimization of Clustering, London*, 2005.

U. von Luxburg, M. Belkin, and O. Bousquet. Consistency of spectral clustering. *Annals of Statistics*, 36(2):555 – 586, 2008.

U. von Luxburg, S. Bubeck, S. Jegelka, and M. Kaufmann. Consistent minimization of clustering objective functions. In J.C. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems (NIPS) 21*. MIT Press, Cambridge, MA, 2008.

D. Wagner and F. Wagner. Between min cut and graph bisection. In *Proceedings of the 18th International Symposium on Mathematical Foundations of Computer Science (MFCS)*, pages 744 – 750, London, 1993. Springer.

M. Wong and T. Lane. A kth nearest neighbor clustering procedure. *J.R. Statist.Soc B*, 45(3): 362 – 368, 1983.

Journal of Machine Learning Research 10 (2009) 699-718 Submitted 6/07; Revised 2/08; Published 3/09

Properties of Monotonic Effects on Directed Acyclic Graphs

Tyler J. VanderWeele VANDERWEELE@UCHICAGO.EDU
Department of Health Studies
University of Chicago
Chicago, IL 60615, USA

James M. Robins ROBINS@HSPH.HARVARD.EDU
Departments of Epidemiology and Biostatistics
Harvard School of Public Health
Boston, MA 02115, USA

Editor: Peter Spirtes

Abstract

Various relationships are shown hold between monotonic effects and weak monotonic effects and the monotonicity of certain conditional expectations. Counterexamples are provided to show that the results do not hold under less restrictive conditions. Monotonic effects are furthermore used to relate signed edges on a causal directed acyclic graph to qualitative effect modification. The theory is applied to an example concerning the direct effect of smoking on cardiovascular disease controlling for hypercholesterolemia. Monotonicity assumptions are used to construct a test for whether there is a variable that confounds the relationship between the mediator, hypercholesterolemia, and the outcome, cardiovascular disease.

Keywords: Bayesian networks, conditional expectation, covariance, directed acyclic graphs, effect modification, monotonicity

1. Introduction

Several papers have considered various monotonicity relationships on Bayesian networks or directed acyclic graphs. Wellman (1990) introduced the notion of qualitative causal influence and derived various resulting concerning the propagation of qualitative influences, the preservation of monotonicity under edge reversal, the necessity of first order stochastic dominance for propagating influences and the propagation of sub-additive and super-additive relationships on probabilistic networks. Druzdzel and Henrion (1993) developed a polynomial time algorithm for reasoning in qualitative probabilistic network, based on local sign propagation. More recently, van der Gaag et al. (2004) showed that identifying whether a network exhibits various monotonicity properties is coNP^{PP}- complete. VanderWeele and Robins (2009) introduced the concept of a monotonic effect which is closely related to Wellman's qualitative influence and considered the relationship between monotonicity properties and causal effects, covariance, bias and confounding. In this paper we develop a number of probabilistic properties concerning monotonic effects and weak monotonic effects. Some of these properties give rise to certain inequality constraints that could be used to test for the presence of hidden or unmeasured confounding variables. These inequality constraints which arise from monotonicity relationships provide constraints beyond those already available in the literature (Kang and Tian, 2006). The paper is organized as follows. In Section 2 we describe the notation we will use in this paper and review the definitions concerning directed acyclic graphs. In Section 3 we present a motivating example for the theory that will be developed. In Section 4, we define the concepts of a monotonic effect and a weak monotonic effect in the directed acyclic graph causal framework, the latter essentially being equivalent to Wellman's (1990) qualitative influence. In Section 5, we give a number of results relating weak monotonic effects to the monotonicity in the conditioning argument of certain conditional expectations; we also return to the motivating example and show how the theory developed can be applied to this example. Finally, in Section 6, we give a number of results that relate weak monotonic effects to the existence of qualitative effect modifiers. Section 7 closes with some concluding remarks.

2. Notation and Directed Acyclic Graphs

Following Pearl (1995), a causal directed acyclic graph is a set of nodes $(X_1,...,X_n)$ and directed edges amongst nodes such that the graph has no cycles and such that for each node X_i on the graph the corresponding variable is given by its non-parametric structural equation $X_i = f_i(pa_i, \varepsilon_i)$ where pa_i are the parents of X_i on the graph and the ε_i are mutually independent. We will use Ω to denote the sample space for ε and ω to denote a particular point in the sample space. These non-parametric structural equations can be seen as a generalization of the path analysis and linear structural equation models (Pearl, 1995, 2000) developed by Wright (1921) in the genetics literature and Haavelmo (1943) in the econometrics literature. Directed acyclic graphs can be interpreted as representing causal relationships. The non-parametric structural equations encode counterfactual relationships amongst the variables represented on the graph. The equations themselves represent one-step ahead counterfactuals with other counterfactuals given by recursive substitution. The requirement that the ε_i be mutually independent is essentially a requirement that there is no variable absent from the graph which, if included on the graph, would be a parent of two or more variables (Pearl, 1995, 2000). Further discussion of the causal interpretation of directed acyclic graphs can be found elsewhere (Pearl, 1995, 2000; Spirtes et al., 2000; Dawid, 2002; Robins, 2003).

A path is a sequence of nodes connected by edges regardless of arrowhead direction; a directed path is a path which follows the edges in the direction indicated by the graph's arrows. A node C is said to be a common cause of A and Y if there exists a directed path from C to Y not through A and a directed path from C to A not through Y. We will say that $V_1,...,V_n$ constitutes an ordered list if $i < j$ implies that V_i is not a descendent of V_j. A collider is a particular node on a path such that both the preceding and subsequent nodes on the path have directed edges going into that node, that is, both the edge to and the edge from that node have arrowheads into the node. A path between A and B is said to be blocked given some set of variables Z if either there is a variable in Z on the path that is not a collider or if there is a collider on the path such that neither the collider itself nor any of its descendants are in Z. If all paths between A and B are blocked given Z then A and B are said to be d-separated given Z. It has been shown that if A and B are d-separated given Z then A and B are conditionally independent given Z (Verma and Pearl, 1988; Geiger et al., 1990; Lauritzen et al., 1990). We will use the notation $A \coprod B|Z$ to denote that A is conditionally independent of B given Z; we will use the notation $(A \coprod B|Z)_G$ to denote that A and B are d-separated given Z on graph G. The directed acyclic graph causal framework has proven to be particularly useful in determining whether conditioning on a given set of variables, or none at all, is sufficient to control for confounding. The most important result in this regard is the back-door path criterion (Pearl, 1995). A back-door path from some node A to another node Y is a path which begins with a directed edge into A. Pearl

$$\begin{tikzcd}
& & U \arrow[d, dashed] \arrow[dr] & \\
Q \arrow[r] \arrow[rr, bend right] \arrow[rrr, bend right=50] & A \arrow[r] \arrow[rr, bend left=70, "\oplus"] & R \arrow[r] & Y
\end{tikzcd}$$

Figure 1: Motivating example concerning the estimation of controlled direct effects.

(1995) showed that for intervention variable A and outcome Y, if a set of variables Z is such that no variable in Z is a descendent of A and such that Z blocks all back-door paths from A to Y then conditioning on Z suffices to control for confounding for the estimation of the causal effect of A on Y. The counterfactual value of Y intervening to set $A = a$ we denote by $Y_{A=a}$.

3. Motivating Example

To motivate the theory we develop in this paper consider the following example.

Example 1. Suppose that Figure 1 represents a causal directed acyclic graph. Let A denote smoking; let R hypercholesterolemia; and let Y denote cardiovascular disease. High cholesterol can lead to the narrowing of the arteries resulting in cardiovascular disease; smoking can lead to blood clots through platelet aggregation resulting in cardiovascular disease. Let Q denote some variable that confounds the relationships between smoking and cardiovascular disease and between hypercholesterolemia and cardiovascular disease (e.g., stress). Let U be some unmeasured variable which might confound the relationship between hypercholesterolemia and cardiovascular disease. The researcher is unsure whether the variable U is a cause of R and we therefore represent the edge from U to R as a dashed line. The results of Pearl (2001) imply that it is possible to estimate controlled direct effects of the form $Y_{A=a_1,R=r} - Y_{A=a_0,R=r}$ (i.e., the direct effect of smoking on cardiovascular disease controlling for hypercholesterolemia) on the graph in Figure 1 if that U is not a cause of R. Suppose that although the researcher is unsure about the presence an edge from U to R, it is known that the relationship between A and Y is monotonic in the sense that $P(Y > y|A = a, R = r, Q = q, U = u)$ is non-decreasing in a for all y, r, q and u. In Section 5, we will present theory that will allow us to derive a statistical test for the null hypothesis that there is no unmeasured variable U confounding the relationship between R and Y.

4. On the Definition of a Monotonic Effect

The definition of a monotonic effect is given in terms of a directed acyclic graph's nonparametric structural equations.

Definition 1. *The non-parametric structural equation for some node Y on a causal directed acyclic graph with parent A can be expressed as $Y = f(\widetilde{pa}_Y, A, \varepsilon_Y)$ where $\widetilde{pa}_Y$ are the parents of Y other than A; A is said to have a positive monotonic effect on Y if for all $\widetilde{pa}_Y$ and ε_Y, $f(\widetilde{pa}_Y, A_1, \varepsilon_Y) \geq f(\widetilde{pa}_Y, A_2, \varepsilon_Y)$ whenever $A_1 \geq A_2$. Similarly A is said to have a negative monotonic effect on Y if for all $\widetilde{pa}_Y$ and ε_Y, $f(\widetilde{pa}_Y, A_1, \varepsilon_Y) \leq f(\widetilde{pa}_Y, A_2, \varepsilon_Y)$ whenever $A_1 \geq A_2$.*

As we have defined it above, a causal direct acyclic graph corresponds to a set of non-parametric structural equations and as such the definition of a monotonic effect given above is relative to a particular set of non-parametric structural equations. The presence of a monotonic effect is closely related to the monotonicity of counterfactual variables as is made clear by the following proposition. All proofs of all propositions and lemmas are given in Appendix A.

Proposition 1. *The variable A has a positive monotonic effect on Y if and only if for all ω and all values of $\widetilde{pa}_Y$, $Y_{a_1, \widetilde{pa}_Y}(\omega) \geq Y_{a_0, \widetilde{pa}_Y}(\omega)$ whenever $a_1 \geq a_0$.*

We note that several sets of non-parametric structural equations may yield identical distributions of $X = (X_1, ..., X_n)$ and $\{X_{V=v}\}_{V \subseteq X, v \in supp(V)}$ (Pearl, 2000). In the context of characterizations of causal directed acyclic graphs that make reference to counterfactuals but not to non-parametric structural equations (Robins, 2003), a positive monotonic effect could instead be defined to be present if for all $\widetilde{pa}_Y$ and $a_1 \geq a_0$, $P(Y_{a_1, \widetilde{pa}_Y} \geq Y_{a_0, \widetilde{pa}_Y}) = 1$. If this latter condition holds with respect to one set of non-parametric structural equations it will hold for any set of non-parametric structural equations which yields the same distribution for X and $\{X_{V=v}\}_{V \subseteq X, v \in supp(V)}$. We note that if for $a_1 \geq a_0$ the set $\{\omega : Y_{a_1, \widetilde{pa}_Y}(\omega) < Y_{a_0, \widetilde{pa}_Y}(\omega)\}$ is of measure zero then $Y_{a_1, \widetilde{pa}_Y}$ and $Y_{a_0, \widetilde{pa}_Y}$ could be re-defined on this set so that $Y_{a_1, \widetilde{pa}_Y}(\omega) \geq Y_{a_0, \widetilde{pa}_Y}(\omega)$ for all ω and so that the distributions of X and $\{X_{V=v}\}_{V \subseteq X, v \in supp(V)}$ remain unchanged.

Because for any value ω we observe the outcome only under one particular value of the intervention variable, the presence of a monotonic effect is not identifiable. The results presented in this paper are in fact true under slightly weaker conditions which are identifiable when data on all of the directed acyclic graph's variables are observed. We thus introduce the concept of a weak monotonic effect which is a special case of Wellman's positive qualitative influence (Wellman, 1990). The definition of a weak monotonic effect does not make reference to counterfactuals and thus can be used in characterizations of causal directed acyclic graphs that do not employ the concept of counterfactuals (Spirtes et al., 2000; Dawid, 2002). The stronger notion of a monotonic effect given above is useful in the context of testing for synergistic relationships (VanderWeele and Robins, 2008).

Definition 2. *Suppose that variable A is a parent of some variable Y and let $\widetilde{pa}_Y$ denote the parents of Y other than A. We say that A has a weak positive monotonic effect on Y if the survivor function $S(y|a, \widetilde{pa}_Y) = P(Y > y|A = a, \widetilde{pa}_Y)$ is such that whenever $a_1 \geq a_0$ we have $S(y|a_1, \widetilde{pa}_Y) \geq S(y|a_0, \widetilde{pa}_Y)$ for all y and all $\widetilde{pa}_Y$; the variable A is said to have a weak negative monotonic effect on Y if whenever $a_1 \geq a_0$ we have $S(y|a_1, \widetilde{pa}_Y) \leq S(y|a_0, \widetilde{pa}_Y)$ for all y and all $\widetilde{pa}_Y$.*

Proposition 2. *If A has a positive monotonic effect on Y then A has a weak positive monotonic effect on Y.*

We note that for parent A and child Y, the definition of a weak monotonic effect coincides with Wellman's (1990) definition of positive qualitative influence when the "context" for qualitative influence is chosen to be the parents of Y other than A.

A monotonic effect is a relation between two nodes on a directed acyclic graph and as such it is associated with an edge. The definition of the sign of an edge can be given either in terms of monotonic effects or weak monotonic effects. We can define the sign of an edge as the sign of the monotonic effect or weak monotonic effect to which the edge corresponds; this in turn gives rise to a natural definition for the sign of a path.

Definition 3. *An edge on a causal directed acyclic graph from X to Y is said to be of positive sign if X has a positive monotonic effect on Y. An edge from X to Y is said to be of negative sign if X has a negative monotonic effect on Y. If X has neither a positive monotonic effect nor a negative monotonic effect on Y, then the edge from X to Y is said to be without a sign.*

Definition 4. *The sign of a path on a causal directed acyclic graph is the product of the signs of the edges that constitute that path. If one of the edges on a path is without a sign then the sign of the path is said to be undefined.*

We will call a causal directed acyclic graph with signs on those edges which allow them a signed causal directed acyclic graph. The theorems in this paper are given in terms of signed paths so as to be applicable to both monotonic effects and weak monotonic effects. One further definition will be useful in the development of the theory below.

Definition 5. *Two variables X and Y are said to be positively monotonically associated if all directed paths from X to Y or from Y to X are of positive sign and all common causes C_i of X and Y are such that all directed paths from C_i to X are of the same sign as all directed paths from C_i to Y; the variables X and Y are said to be negatively monotonically associated if all directed paths between X and Y are of negative sign and all common causes C_i of X and Y are such that all directed paths from C_i to X are of the opposite sign as all directed paths from C_i to Y.*

It has been shown elsewhere (VanderWeele and Robins, 2009) that if X and Y are positively monotonically associated then $Cov(X,Y) \geq 0$ and if X and Y are negatively monotonically associated then $Cov(X,Y) \leq 0$. We now develop several results concerning the monotonicity in the conditioning argument of certain conditional expectations.

5. Monotonic Effects and Conditional Expectations

Lemma 1 below can be proved by integration by parts and will be used in the proofs of the subsequent propositions. We will assume throughout the remainder of this paper that the random variables under consideration satisfy regularity conditions that allow for the integration by parts required in the proof of Lemma 1. If conditional cumulative distribution functions are continuously differentiable then the regularity conditions will be satisfied; the regularity conditions will also be satisfied if all variables are discrete. Härdle et al. (1998, p72) also gives relatively weak conditions under which such integration by parts is possible. Alternatively, the existence of the Lebesgue-Stieltjes integrals found in the proof of Lemma 1 suffices to allow integration by parts. Note that Lemma 1 will always be applied either to the function $h(y,a,r) = y$ or to conditional survivor functions which will satisfy the relevant regularity conditions; thus the conditions which are required for integration by parts are only regularity conditions on the distribution of the random variables.

Lemma 1. *If $h(y,a,r)$ is non-decreasing in y and in a and $S(y|a,r) = P(Y > y|A = a, R = r)$ is non-decreasing in a for all y then $E[h(Y,A,R)|A = a, R = r]$ is non-decreasing in a.*

Proposition 3 immediately follows from Lemma 1.

Proposition 3. *Suppose that the A–Y edge, if it exists, is positive. Let X denote some set of non-descendents of Y that includes $\widetilde{pa}_Y$, the parents of Y other than A, then $E[Y|X = x, A = a]$ is non-decreasing in a for all values of x.*

Proposition 4 gives the basic result for the monotonicity of conditional expectations. For the conditional expectation of some variable Y to be monotonic in a conditioning argument A, it requires that the conditioning set includes variables that block all backdoor paths from A to Y. In order to prove Proposition 4 we will make use of the following two lemmas.

Lemma 2. *Suppose that A is a non-descendent of Y and let Q denote the set of ancestors of A or Y which are not descendents of A. Let $R = (R_1, ..., R_m)$ denote an ordered list of some set of nodes on directed paths from A to Y such that for each i the backdoor paths from R_i to Y are blocked by $R_1, ..., R_{i-1}, A$, and Q. Let $V_0 = A$ and $V_n = Y$ and let $V_1, ..., V_{n-1}$ be an ordered list of all the nodes which are not in R but which are on directed paths from A to Y such that at least one of the directed paths from each node to Y is not blocked by R. Let $\overline{V}_k = \{V_1, ..., V_k\}$ then $S(v_k|a, \overline{v}_{k-1}, q, r) = S(v_k|pa_{v_k})$.*

Lemma 3. *If under the conditions of Lemma 2 all directed paths from A to Y are positive except possibly through R then $S(y|a,q,r)$ is non-decreasing in a.*

These two lemmas allow us to prove Proposition 4 given below.

Proposition 4. *Suppose that A is a non-descendent of Y and let X denote some set of non-descendents of A that blocks all backdoor paths from A to Y. Let $R = (R_1, ..., R_m)$ denote an ordered list of some set of nodes on directed paths from A to Y such that for each i the backdoor paths from R_i to Y are blocked by $R_1, ..., R_{i-1}, A$ and X. If all directed paths from A to Y are positive except possibly through R then $S(y|a,x,r)$ and $E[y|a,x,r]$ are non-decreasing in a.*

If $R = \varnothing$ the statement of Proposition 4 is considerably simplified and is stated in the following corollary.

Corollary. *Let X denote some set of non-descendents of A that blocks all backdoor paths from A to Y. If all directed paths between A and Y are positive then $S(y|a,x)$ and $E[y|a,x]$ are non-decreasing in a.*

Lemma 3 and Proposition 4 are generalizations of results given by Wellman (1990) and Druzdzel and Henrion (1993). In particular, in Lemma 3 if $R = \varnothing$, then the result follows immediately from repeated application of Theorems 4.2 and 4.3 in Wellman (1990) or more directly from the work of Druzdzel and Henrion (1993, Theorem 4). Lemma 3 generalizes the results of Wellman (1990) and Druzdzel and Henrion (1993) by allowing for conditioning on nodes $R = (R_1, ..., R_m)$ which are on directed paths from A to Y. Proposition 4 further generalizes Lemma 3 by replacing the set Q in

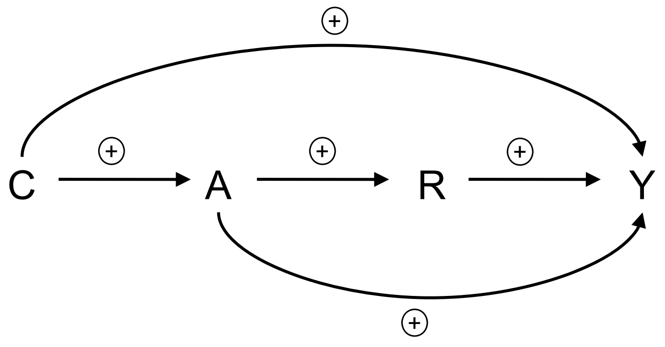

$$\begin{tikzcd} C \arrow[r, "\oplus"] \arrow[rrr, bend left, "\oplus"] & A \arrow[r, "\oplus"] \arrow[rr, bend right, "\oplus"'] & R \arrow[r, "\oplus"] & Y \end{tikzcd}$$

Figure 2: Example illustrating Propositions 4-6.

Lemma 3 which consists of the set of ancestors of A or Y which are not descendents of A with some other set X which consists of some set of non-descendents of A that blocks all backdoor paths from A to Y.

Propositions 5-8 relax the condition that the conditioning set includes variables that block all backdoor paths A to Y and impose certain other conditions; the proofs of each of these propositions make use of Proposition 4.

Proposition 5. *Suppose that A is not a descendent of Y, that A is binary, and that A and Y are positively monotonically associated then $E[Y|A]$ is non-decreasing in A.*

Proposition 6. *Suppose that A is not a descendent of Y, that Y is binary, and that A and Y are positively monotonically associated then $E[A|Y]$ is non-decreasing in Y.*

Propositions 5 and 6 require that the conditioning variable be binary. Counterexamples can be constructed to show that if the conditioning variable is not binary then the conditional expectation may not be non-decreasing in the conditioning argument even if A and Y are positively monotonically associated (see Appendix B, counterexamples 1 and 2).

Propositions 5 and 6 can be combined to give the following corollary which makes no reference to the ordering of A and Y.

Corollary. *Suppose that A is binary and that A and Y are positively monotonically associated then $E[Y|A]$ is non-decreasing in A.*

Example 2. Consider the signed directed acyclic graph given in Figure 2. By Proposition 4, we have that $E[Y|A=a,C=c,R=r]$ and $E[Y|A=a,C=c]$ are non-decreasing in a. If A is binary then by Proposition 5, it is also the case that $E[Y|A=a]$ is non-decreasing in a. If Y is binary, then by Proposition 6, $E[A|Y=y]$ is non-decreasing in y. The monotonicity of $E[Y|A=a,C=c,R=r]$ and $E[Y|A=a,C=c]$ also follow directly from the results of Wellman (1990) and Druzdzel and Henrion (1993); the monotonicity of $E[Y|A=a]$ and $E[A|Y=y]$ do not.

Propositions 7 and 8 consider the monotonicity of conditional expectations while conditioning on variables other than the variable in which monotonicity holds but not conditioning on variables that are sufficient to block all backdoor paths between A and Y. Propositions 7 and 8 generalize

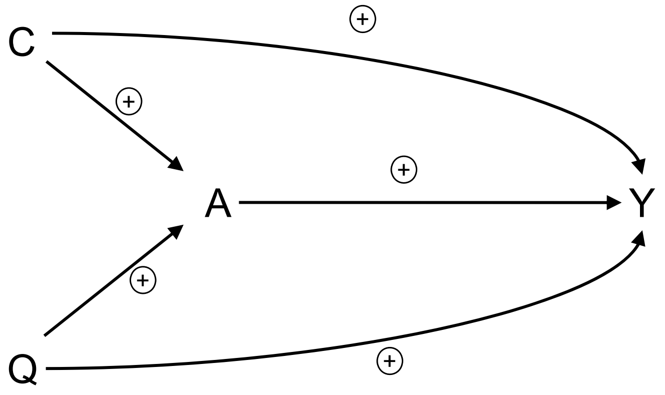

$$\begin{tikzcd}
C \arrow[dr, "\oplus"] \arrow[drr, bend left, "\oplus"] & & \\
& A \arrow[r, "\oplus"] & Y \\
Q \arrow[ur, "\oplus"] \arrow[urr, bend right, "\oplus"'] & &
\end{tikzcd}$$

Figure 3: Example illustrating Propositions 7 and 8.

Propositions 5 and 6 respectively.

Proposition 7. *Suppose that A is not a descendent of Y and that A is binary. Let Q be some set of variables that are not descendents of Y nor of A and let C be the common causes of A and Y not in Q. If all directed paths from A to Y are of positive sign and all directed paths from C to A not through Q are of the same sign as all directed paths from C to Y not through $\{Q,A\}$ then $E[Y|A,Q]$ is non-decreasing in A.*

Proposition 8 is similar to Proposition 7 but the conditional expectation $E[A|Y,Q]$ is considered rather than $E[Y|A,Q]$ and Y rather than A is assumed to be binary. The form of the proof differs.

Proposition 8. *Suppose that A is not a descendent of Y and that Y is binary. Let Q be some set of variables that are not descendents of Y nor of A and let C be the common causes of A and Y not in Q. If all directed paths from A to Y are of positive sign and all directed paths from C to A not through Q are of the same sign as all directed paths from C to Y not through $\{Q,A\}$ then $E[A|Y,Q]$ is non-decreasing in Y.*

Propositions 7 and 8 can be combined to give the following corollary which makes no reference to the ordering of A and Y.

Corollary. *Suppose that A is binary. Let Q be some set of variables that are not descendents of Y nor of A and let C be the common causes of A and Y not in Q. If all directed paths from A to Y (or from A to Y) are of positive sign and all directed paths from C to A not through $\{Q,Y\}$ are of the same sign as all directed paths from C to Y not through $\{Q,A\}$ then $E[Y|A,Q]$ is non-decreasing in Y.*

Example 3. Consider the signed directed acyclic graph given in Figure 3. If A is binary, then by Proposition 7, $E[Y|A=a,C=c,Q=q]$, $E[Y|A=a,Q=q]$, $E[Y|A=a,C=c]$ and $E[Y|A=a]$ are all non-decreasing in a. If Y is binary then by Proposition 8, $E[A|Y=y,C=c,Q=q]$, $E[A|Y=y,Q=q]$, $E[A|Y=y,C=c]$ and $E[A|Y=y]$ are all non-decreasing in y. The monotonicity of $E[Y|A=a,C=c,Q=q]$ follows directly from the results of Wellman (1990) and Druzdzel and Henrion (1993); the monotonicity of the other conditional expectations do not.

We now return to Example 1 concerning potential unmeasured confounding in the estimation of controlled direct effects.

Example 1 (Revisited). Consider once again the causal directed acyclic graph given in Figure 1. Suppose that we may assume that A has a weak monotonic effect on Y. Under the null hypothesis that U is not a cause of R (i.e., does not confound the relationship between R and Y) we could conclude by Proposition 4 that $E[Y|A=a,R=r,Q=q]$ is non-decreasing in a for all r and q. Under the alternative hypothesis that U is a cause of R, we could not apply Proposition 4 because of the unblocked backdoor path $R-U-Y$ between R and Y. The monotonicity relationship would thus not necessarily hold. Consequently, if $E[Y|A=a,R=r,Q=q]$ were found not to be monotonic in a then we could reject the null hypothesis that U is not a cause of R. Note that the monotonicity of $E[Y|A=a,R=r,Q=q]$ in a also follows from the results of Wellman (1990) and Druzdzel and Henrion (1993). If, however, there were an edge from U to Q for example, or in more complicated scenarios, the results of Wellman (1990) and Druzdzel and Henrion (1993) would no longer suffice to conclude the monotonicity of $E[Y|A=a,R=r,Q=q]$ in a; one would need to employ Proposition 4.

We now construct a simple statistical test in the case that A,R and Y are all binary (cf. Robins and Greenland, 1992) of the null hypothesis that U is absent from Figure 1. Let n_{ijq} denote the number of individuals in stratum $Q=q$ with $A=i$ and $R=j$ and let let d_{ijq} denote the number of individuals in stratum $Q=q$ with $A=i$ and $R=j$ and $Y=1$. Let p_{ijq} denote the true probability $P(Y=1|A=i,R=j,Q=q)$. From the null hypothesis that U is absent from Figure 1, it follows by Proposition 4 that $p_{1jq}-p_{0jq}\leq 0$ for all j and q. Thus we have $d_{ijq}\sim Bin(n_{ijq},p_{ijq})$ with $\mathbb{E}[\frac{d_{ijq}}{n_{ijq}}]=p_{ijq}$ and $Var(\frac{d_{ijq}}{n_{ijq}})=\frac{p_{ijq}(1-p_{ijq})}{n_{ijq}}$. By the central limit central limit theorem $\frac{(\frac{d_{1jq}}{n_{1jq}}-\frac{d_{0jq}}{n_{0jq}})-(p_{1jq}-p_{0jq})}{\sqrt{\frac{p_{1jq}(1-p_{1jq})}{n_{1jq}}+\frac{p_{0jq}(1-p_{0jq})}{n_{0jq}}}}\dot{\sim}N(0,1)$ and by Slutsky's theorem we have $\frac{(\frac{d_{1jq}}{n_{1jq}}-\frac{d_{0jq}}{n_{0jq}})-(p_{1jq}-p_{0jq})}{\sqrt{\frac{d_{1jq}(n_{1jq}-d_{1jq})}{n_{1jq}^3}+\frac{d_{0jq}(n_{0jq}-d_{0jq})}{n_{0jq}^3}}}\dot{\sim}N(0,1)$. To test the null hypothesis that the edge from U to R is absent from Figure 1 one may thus use the test statistic $\frac{(\frac{d_{1jq}}{n_{1jq}}-\frac{d_{0jq}}{n_{0jq}})}{\sqrt{\frac{d_{1jq}(n_{1jq}-d_{1jq})}{n_{1jq}^3}+\frac{d_{0jq}(n_{0jq}-d_{0jq})}{n_{0jq}^3}}}$ with critical regions of the form: $\{\frac{(\frac{d_{1jq}}{n_{1jq}}-\frac{d_{0jq}}{n_{0jq}})}{\sqrt{\frac{d_{1jq}(n_{1jq}-d_{1jq})}{n_{1jq}^3}+\frac{d_{0jq}(n_{0jq}-d_{0jq})}{n_{0jq}^3}}}>Z_{1-\alpha}\}$ to carry out a one-sided (upper tail) test. The derivation of the power of such a test would require providing explicit structural equations for each of the variables in the model. Similar tests could be constructed for other scenarios. We note that if the test fails to reject the null, one cannot conclude that the arrow from U to R is absent; if the inequality $E[Y|A=a_1,R=r,Q=q]\leq E[Y|A=a_2,R=r,Q=q]$ holds for all $a_1\leq a_2$ this is potentially consistent with both the presence and the absence of an edge from U to R. If, however, the test rejects the null then one can conclude that an edge from U to R must be present, provided the other model assumptions hold. With observational data, the assumption that no unmeasured confounding variable is present can be falsified but it cannot be verified regardless of the approach one takes. It is nevertheless worthwhile testing any empirical implications of the no unmeasured confounding variables assumptions which can be derived, such as those following from Proposition 4.

Tian and Pearl (2002) and Kang and Tian (2007) derived various equality constraints that arise from causal directed acyclic graphs with hidden variables; Kang and Tian (2006) derived various inequality constraints that arise from causal directed acyclic graphs with hidden variables. We note that the inequality constraint $E[Y|A=a_1,R=r,Q=q] \leq E[Y|A=a_2,R=r,Q=q]$ for $a_1 \leq a_2$ does not follow from the results of Tian and Pearl (2002) or Kang and Tian (2006, 2007). The equality and inequality constraints which follow from their work will apply to all causal models consistent with the directed acyclic graph in Figure 1 (without the sign); the inequality constraint $E[Y|A=a_1,R=r,Q=q] \leq E[Y|A=a_2,R=r,Q=q]$ follows only if it can be assumed in Figure 1 that A has a weak positive monotonic effect on Y. More generally, the results in this paper provide a supplementary set of constraints to those of Tian and Pearl (2002) and Kang and Tian (2006, 2007).

6. Effect Modification and Monotonic Effects

If when conditioning on a particular variable, the sign of the effect of another variable on the outcome varies between strata of the conditioning variable, then the conditioning variable is said to be a qualitative effect modifier. The following definition gives the condition for qualitative effect modification more formally.

Definition 6. *A variable Q is said to be an effect modifier for the causal effect of A on Y if Q is not a descendent of A and if there exist two levels of A, a_0 and a_1 say, such that $E[Y_{A=a_1}|Q=q]-E[Y_{A=a_0}|Q=q]$ is not constant in q. Furthermore Q is said to be a qualitative effect modifier if there exist two levels of A, a_0 and a_1, and two levels of Q, q_0 and q_1, such that $sign(E[Y_{A=a_1}|Q=q_1]-E[Y_{A=a_0}|Q=q_1]) \neq sign(E[Y_{A=a_1}|Q=q_0]-E[Y_{A=a_0}|Q=q_0])$.*

Monotonic effects and weak monotonic effects are closely related to the concept of qualitative effect modification. Essentially, the presence of a monotonic effect precludes the possibility of qualitative effect modification. This is stated precisely in Propositions 9 and 10.

Proposition 9. *Suppose that some parent A_1 of Y is such that the A_1-Y edge is of positive sign then there can be no other parent, A_2, of Y which is a qualitative effect modifier for causal effect of A_1 on Y, either unconditionally or within some stratum $C=c$ of the parents of Y other than A_1 and A_2.*

A similar result clearly holds if the A_1-Y edge is of negative sign. We give the contrapositive of Proposition 9 as a corollary.

Corollary. *Suppose that some parent of Y, A_2, is a qualitative effect modifier for causal effect of another parent of Y, A_1, either unconditionally or within some stratum $C=c$ of the parents of Y other than A_1 and A_2 then A_1 can have neither a weak positive monotonic effect nor a weak negative monotonic effect on Y.*

If there are intermediate variables between A and Y then Proposition 9 can be generalized to give Proposition 10.

Proposition 10. *Suppose that all directed paths from A to Y are of positive sign (or are all of negative sign) then there exists no qualitative effect modifier Q on the directed acyclic graph for the causal effect of A on Y.*

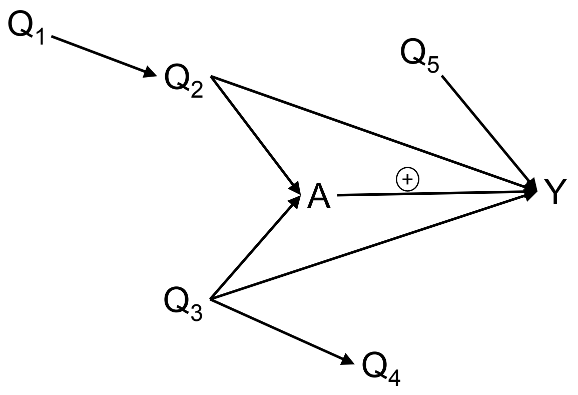


Figure 4: Example illustrating the use of Propositions 9 and 10.

Example 4. Consider the signed directed acyclic graph given in Figure 4 in which the $A-Y$ edge is of positive sign. It can be shown that any of Q_1, Q_2, Q_3, Q_4 or Q_5 can serve as effect modifiers for the causal effect of A on Y (VanderWeele and Robins, 2007). However, by Proposition 9 or 10, since A has a (weak) monotonic effect on Y, none of Q_1, Q_2, Q_3, Q_4 or Q_5 can serve as *qualitative* effect modifiers for the causal effect of A on Y. Conversely, if it is found that one of Q_1, Q_2, Q_3, Q_4 or Q_5 *is* a qualitative effect modifier for the causal effect of A on Y then the $A-Y$ edge cannot be of positive (or negative) sign.

7. Concluding Remarks

In this paper we have related weak monotonic effects to the monotonicity of certain conditional expectations in the conditioning argument and to qualitative effect modification. When the variables on a causal directed acyclic graph exhibit weak monotonic effects the results can be used to construct tests for the presence of unmeasured confounding variables. Future work could examine whether it is possible to weaken the restrictions on R in Proposition 4; another area of future research would include developing an algorithm for what relationships need systematic evaluation in order to test for particular confounding patterns; further research could also be done on the derivation of statistical tests of the type considered at the end of Section 5 for cases in which A, R and Y are not binary and for dealing with issues related to multiple testing problems.

Acknowledgments

We would like to thank the editors and three anonymous referees for helpful comments on this paper.

Appendix A. Proofs

This appendix contains the proofs for all of the results in this paper.

A.1 Proof of Proposition 1

By the definition of a non-parametric structural equation, $Y_{a,\widetilde{pa}_Y}(\omega) = f(\widetilde{pa}_Y, a, \varepsilon_Y(\omega))$ and from this the result follows.

A.2 Proof of Proposition 2

Since A has a positive monotonic effect on Y, for any $a_1 \geq a_0$ we have that $S(y|a_1, \widetilde{pa}_Y) = P(Y > y|a_1, \widetilde{pa}_Y) = P\{f(\widetilde{pa}_Y, a_1, \varepsilon_Y) > y\} \geq P\{f(\widetilde{pa}_Y, a_0, \varepsilon_Y) > y\} = P(Y > y|a_0, \widetilde{pa}_Y) = S(y|a_1, \widetilde{pa}_Y)$.

A.3 Proof of Lemma 1

For $a \geq a'$ we have $E[h(Y,A,R)|A = a, R = r] - E[h(Y,A,R)|A = a', R = r]$ $= \int_{y=-\infty}^{y=\infty} h(y,a,r)dF(y|a,r) - \int_{y=-\infty}^{y=\infty} h(y,a',r)dF(y|a',r) = \int_{y=-\infty}^{y=\infty} h(y,a,r)d\{F(y|a,r) - F(y|a',r)\} + \int_{y=-\infty}^{y=\infty} \{h(y,a,r) - h(y,a',r)\}dF(y|a',r) = [h(y,a,r)\{F(y|a,r) - F(y|a',r)\}]_{y=-\infty}^{y=\infty} - \int_{y=-\infty}^{y=\infty} \{F(y|a,r) - F(y|a',r)\}dh(y,a,r) + \int_{y=-\infty}^{y=\infty} \{h(y,a,r) - h(y,a',r)\}dF(y|a',r)$
$= \int_{y=-\infty}^{y=\infty} \{S(y|a,r) - S(y|a',r)\}dh(y,a,r) + \int_{y=-\infty}^{y=\infty} \{h(y,a,r) - h(y,a',r)\}dF(y|a',r)$. This final expression is non-negative since the integrands of both integrals are non-negative for $a \geq a'$.

A.4 Proof of Proposition 3

We have that $E[Y|X = x, A = a] = E[Y|\widetilde{pa}_Y, A = a]$ and since A has a (weak) positive monotonic effect on Y, we have that $S(y|a, \widetilde{pa}_Y)$ is non-decreasing in a and it follows from Lemma 1 that $E[Y|X = x, A = a] = E[Y|\widetilde{pa}_Y, A = a]$ is non-decreasing in a.

A.5 Proof of Lemma 2

We will say a path from A to B is a frontdoor path from A to B if the path begins with a directed edge with the arrowhead pointing out of A. Let Q^k and R^k be the subsets of Q and R respectively that are ancestors of $\overline{V}_k$. We will show that

$$\begin{aligned} S(v_k|a, v_1, ..., v_{k-1}, q, r) &= S(v_k|a, v_1, ..., v_{k-1}, q, r^k) \\ &= S(v_k|a, v_1, ..., v_{k-1}, q^k, r^k) = S(v_k|pa_{v_k}). \end{aligned}$$

If $R^k = R$, the first equality holds trivially. Suppose that $R^k \neq R$ so that R_m is not an ancestor of V_k. All frontdoor paths from R_m to V_k must include a collider since R_m is not an ancestor of V_k. This collider will not be in $A, V_1, ..., V_{k-1}, Q, R_1, ..., R_{m-1}$ since all these variables are non-descendents of R_m. Thus all frontdoor paths from R_m to V_k will be blocked given $A, V_1, ..., V_{k-1}, Q, R_1, ..., R_{m-1}$. All backdoor paths from R_m to V_k with an edge going into V_k will be blocked given $A, V_1, ..., V_{k-1}, Q, R_1, ..., R_{m-1}$ by pa_{V_k}; note by hypothesis it can be seen that pa_{V_k} will be contained by the variables $A, V_1, ..., V_{k-1}$, Q, R^k since there is a directed path from V_k to Y and Q includes all ancestors of Y not on directed paths from A to Y. All backdoor paths from R_m to V_k with an edge going out from V_k will be blocked given $A, Q, R_1, ..., R_{m-1}$ by hypothesis; otherwise there would be a backdoor path from R_m through V_k to Y not blocked by $A, Q, R_1, ..., R_{m-1}$. But all backdoor paths from R_m to V_k with an edge going out from V_k which are blocked by $A, Q, R_1, ..., R_{m-1}$ will also be blocked by

$A,V_1,...,V_{k-1},Q,R_1,...,R_{m-1}$. This is because such a path concluding with an edge going out from V_k which is blocked by $A,Q,R_1,...,R_{m-1}$ but not blocked by $A,V_1,...,V_{k-1},Q,R_1,...,R_{m-1}$ would require that one of $V_1,...,V_{k-1}$, say V_p, be a collider on the path or a descendent of a collider. If one of $V_1,...,V_{k-1}$ were a collider then the path would in fact be blocked by the parents of the collider since all the parents of $V_1,...,V_{k-1}$ are in $A,V_1,...,V_{k-1},Q,R_1,...,R_{m-1}$. If one of $V_1,...,V_{k-1}$, say V_p, were a descendent of the collider then none of the directed paths from the collider to V_p could contain nodes in $R_1,...,R_{m-1}$ for otherwise the path would not be blocked by $A,Q,R_1,...,R_{m-1}$; for the same reason the collider itself could not be in $R_1,...,R_{m-1}$. But it then follows that the collider must itself be one of $V_1,...,V_{p-1}$ since it is an ancestor of V_p with a directed path to V_p not blocked by R. However, if the collider is one of $V_1,...,V_{p-1}$ then the path would in fact be blocked by the parents of the collider since all the parents of $V_1,...,V_{k-1}$ are in $A,V_1,...,V_{k-1},Q,R_1,...,R_{m-1}$. From this it follows that all backdoor paths from R_m to V_k with an edge going out from V_k are blocked by $A,V_1,...,V_{k-1},Q,R_1,...,R_{m-1}$.

We have thus shown that V_k and R_m are d-separated given $A,V_1,...,V_{k-1},Q,R_1,...,R_{m-1}$ and so

$$S(v_k|a,v_1,...,v_{k-1},q,r) = S(v_k|a,v_1,...,v_{k-1},q,r_1,...,r_{m-1}).$$

Similarly, V_k and R_{m-1} are d-separated given $A,V_1,...,V_{k-1},Q,R_1,...,R_{m-2}$ and so

$$S(v_k|a,v_1,...,v_{k-1},q,r_1,...,r_{m-1}) = S(v_k|a,v_1,...,v_{k-1},q,r_1,...,r_{m-2}).$$

We may carry this argument forward to get

$$S(v_k|a,v_1,...,v_{k-1},q,r) = S(v_k|a,v_1,...,v_{k-1},q,r^k).$$

All backdoor paths from V_k to $Q\backslash Q^k$ will be blocked given $A,V_1,...,V_{k-1},Q^k,R^k$ by pa_{v_k}. Since V_k is not a descendent of $Q\backslash Q^k$ all frontdoor paths from V_k to $Q\backslash Q^k$ will involve at least one collider which is a descendent of V_k. This collider is not in the conditioning set $A,V_1,...,V_{k-1},Q^k,R^k$ since this entire set consists of non-descendents of V_k and so the collider will block the frontdoor path from V_k to $Q\backslash Q^k$.

Thus V_k and $Q\backslash Q^k$ are d-separated given $A,V_1,...,V_{k-1},Q^k,R^k$ and so

$$S(v_k|a,v_1,...,v_{k-1},q,r^k) = S(v_k|a,v_1,...,v_{k-1},q^k,r^k).$$

Furthermore, $A,V_1,...,V_{k-1},Q^k,R^k$ are non-descendents of V_k and include all of the parents of V_k and so

$$S(v_k|a,v_1,...,v_{k-1},q^k,r^k) = S(v_k|pa_{v_k}).$$

We have thus shown as desired that

$$\begin{aligned} S(v_k|a,v_1,...,v_{k-1},q,r) &= S(v_k|a,v_1,...,v_{k-1},q,r^k) \\ &= S(v_k|a,v_1,...,v_{k-1},q^k,r^k) = S(v_k|pa_{v_k}). \end{aligned}$$

A.6 Proof of Lemma 3

Let $V_0 = A$ and $V_n = Y$ and let $V_1,...,V_{n-1}$ be an ordered list of all the nodes which are not in R but which are on directed paths from A to Y such that at least one of the directed paths from each node to Y is not blocked by R. Let $\overline{V}_k = \{V_1,...,V_k\}$. It can be shown by induction that by starting with

$n = k$ and for each k iteratively replacing by their negations the parents of V_k with negative edges into V_k suffices to obtain a graph such that all edges on all directed paths from A to Y not blocked by R have positive sign.

We can express $E[1(V_n > v)|A,Q,R]$ as

$$E[E[...E[E[1(V_n > v)|A,\overline{V}_{n-1},Q,R]|A,\overline{V}_{n-2},Q,R]|...|A,V_1,Q,R]|A,Q,R].$$

Now conditional on $A,\overline{V}_{n-1}\backslash V_i,Q,R$ we have that

$$E[1(V_n > v)|,A,\overline{V}_{n-1},Q,R]$$

is non-decreasing in v_i for $i = 1,...,n-1$ since V_i has either a weak positive monotonic effect or no effect on V_n. Thus conditional on $A,\overline{V}_{n-1}\backslash\{V_i,V_{n-1}\},Q,R$ we have that

$$E[1(V_n > v)|A,\overline{V}_{n-1},Q,R]$$

is a non-decreasing function of v_i and v_{n-1}. Furthermore, by Lemma 2 we have that $S(v_{n-1}|a,v_1,...,v_{n-2},q,r) = S(v_{n-1}|pa_{v_{n-1}})$ and so $S(v_{n-1}|a,v_1,...,v_{n-2},q,r) = S(v_{n-1}|pa_{v_{n-1}})$ is a non-decreasing in v_i for all $a,v_1,...,v_{i-1},v_{i+1},...,v_{n-2},q,r$ since V_i has either a weak positive monotonic effect or no effect on V_{n-1}. Thus by Lemma 1 we have that conditional on $A,\overline{V}_{n-2}\backslash V_i,Q,R$,

$$E[E[1(V_n > v)|A,\overline{V}_{n-1},Q,R]|A,\overline{V}_{n-2},Q,R]$$

is non-decreasing in v_i for $i = 1,...,n-2$. Carrying the argument forward, conditional on A,Q,R, we will have that

$$E[...E[E[1(V_n > v)|A,\overline{V}_{n-1},Q,R]|A,\overline{V}_{n-2},Q,R]|...|A,V_1,Q,R]$$

is a non-decreasing function of v_1 and $v_0 = a$ and since A has either a weak positive monotonic effect or no effect on V_1, $S(v_1|a,q,r) = S(v_1|pa_{v_1})$ will be non-decreasing in a and thus by Lemma 1,

$$\begin{aligned} S(y|a,q,r) &= E[1(V_n > y)|A,Q,R] \\ &= E[E[...E[E[1(V_n > y)|A,\overline{V}_{n-1},Q,R]|A,\overline{V}_{n-2},Q,R]|...|A,V_1,Q,R]|A,Q,R] \end{aligned}$$

will be non-decreasing in a.

A.7 Proof of Proposition 4

Let Q denote the set of ancestors of A or Y which are not descendents of A. Note that if for each i the backdoor paths from R_i to Y are blocked by $R_1,...,R_{i-1},A$ and X then these backdoor paths will also be blocked by $R_1,...,R_{i-1},A$ and Q since for each backdoor path from R_i to X there must be some member of $\{A\}\bigcup Q$ through which the path passes. We may thus apply Lemma 3 to conclude that $E[1(Y > y)|a,Q,r]$. Since Q blocks all backdoor paths from A to Y we have

$$\begin{aligned} S(y|a,x,r) &= E[E[1(Y > y)|a,Q,x,r]|a,x,r] \\ &= E[E[1(Y > y)|a,Q,r]|a,x,r] = E[E[1(Y > y)|a,W,r]|a,x,r] \end{aligned}$$

where W is the subset of Q which are either parents of Y or parents of a node on a directed path from A to Y. Let W' denote the subset of W for which there is a path to Y not blocked by A,X,R

then $E[E[1(Y>y)|a,W,r]|a,x,r] = E[E[1(Y>y)|a,W',r]|a,x,r]$. All backdoor paths from A to W' are blocked given R and X by X since X blocks all backdoor paths from A to Y. Any frontdoor path from A to W' will include a collider since the nodes in W' are not descendents of A. The collider cannot be in X because X includes only non-descendents of A. Suppose the collider were some node R_i; by hypothesis all backdoor paths from R_i to Y are blocked by $R_1,...,R_{i-1},A$ and X; thus the frontdoor path from A to W' would have to be blocked by $A,R_1,...,R_{i-1}$ and X for otherwise there would be a backdoor path from R_i through W' to Y not blocked by $A,R_1,...,R_{i-1}$ and X. From this it follows that every frontdoor path from A to W' must be blocked given R and X either by a collider or by a node in R or X. We have thus shown that all paths from A to W' are blocked given R and X and so W' is conditionally independent of A given R and X and so we have

$$\begin{aligned} E[E[1(Y>y)|a,W',r]|a,x,r] &= E[E[1(Y>y)|a,W',r]|x,r] \\ &= E[E[1(Y>y)|a,Q,r]|x,r]. \end{aligned}$$

We have thus shown that $S(y|a,x,r) = E[E[1(Y>y)|a,Q,r]|x,r]$. Since $E[1(Y>y)|a,Q,r]$ is non-decreasing in a for all q we also have that

$$S(y|a,x,r) = E[E[1(Y>y)|a,Q,r]|x,r]$$

is non-decreasing in a. Finally, since $S(y|a,x,r)$ is non-decreasing in a, it follows from Lemma 1 that $E[y|a,x,r]$ is also non-decreasing in a.

A.8 Proof of Proposition 5

Proposition 5 is in fact a special case of Proposition 7 with $R = \varnothing$ and $Q = \varnothing$. The proof of Proposition 7 is given below.

A.9 Proof of Proposition 6

Proposition 6 is in fact a special case of Proposition 8 with $R = \varnothing$ and $Q = \varnothing$. The proof of Proposition 8 is given below.

A.10 Proof of Proposition 7

By the law of iterated expectations,

$$\begin{aligned} &E[Y|A=a,Q=q] \\ &\quad = \sum_c E[Y|A=a,C=c,Q=q]P(C=c|A=a,Q=q) \end{aligned}$$

We have by Proposition 4 that $E[Y|A,Q,C]$ is non-decreasing in A. Let $(C_1,...,C_n)$ denote an ordered list of the variables in C. Let Q^c be variables in Q which are common causes of C and let $Q^n = Q\backslash Q^c$. Let Q_i^d be the variables in Q^c that are descendents of C_i. Let C_i^d denote the variables in C that are descendents of C_i and let $C_i^n = C\backslash\{C_i,C_i^d\}$. By Proposition 4 we have that $E[Y|A,Q,C]$ is non-decreasing in each component C_i of C by choosing for each i, A in Proposition 4 to be C_i, X in Proposition 4 to be the set $\{Q^n,Q^c\backslash Q_i^d,C_i^n\}$ and R in Proposition 4 to be the set $\{Q_i^d,C_i^d,A\}$. Furthermore,

$$P(C=c|A=a,Q=q) = \frac{P(A=a|C=c,Q=q)P(C=c|Q=q)}{P(A=a|Q=q)}$$

and so

$$P(C=c|A=1,Q=q)=\nu_q(c)P(C=c|A=0,Q=q)$$

where

$$\nu_q(c)=\frac{P(A=0|Q=q)P(A=1|C=c,Q=q)}{P(A=1|Q=q)P(A=0|C=c,Q=q)}$$

which is non-decreasing in each dimension of c since the numerator is non-decreasing in each dimension of c and the denominator is non-increasing in each dimension of c by Proposition 4 by choosing for each i, A in Proposition 4 to be C_i, X in Proposition 4 to be the set $\{Q^n, Q^c \backslash Q_i^c, C_i^n\}$ and R in Proposition 4 to be the set $\{Q_i^c, C_i^d\}$. Thus

$$\begin{aligned}
&E[Y|A=1,Q=q]\\
&=\sum_c E[Y|A=1,C=c,Q=q]P(C=c|A=1,Q=q)\\
&\geq \sum_c E[Y|A=0,C=c,Q=q]P(C=c|A=1,Q=q)\\
&=\sum_c E[Y|A=0,C=c,Q=q]\nu_q(c)P(C=c|A=0,Q=q)\\
&\geq \sum_c E[Y|A=0,C=c,Q=q]P(C=c|A=0,Q=q)\\
&=E[Y|A=0,Q=q].
\end{aligned}$$

The second inequality holds because by an argument similar to that above $E[Y|A=0,Q=q,C=c]$ is non-decreasing in each dimension of c and $P(C=c|A=1,Q=q)=\nu_q(c)P(C=c|A=0,Q=q)$ weights more heavily higher values of each dimension of c than does $P(C=c|A=0,Q=q)$ since $\nu_q(c)$ is non-decreasing in each dimension of c. Thus $E[Y|A=a,Q=q]$ is non-decreasing in a.

A.11 Proof of Proposition 8

By the law of iterated expectations we have that

$$\begin{aligned}
E[A|Y=y,Q=q]&=\sum_c E[A|Y=y,C=c,Q=q]P(C=c|Y=y,Q=q)\\
&=\sum_{c,a} aP(A=a|Y=y,C=c,Q=q)P(C=c|Y=y,Q=q)\\
&=\sum_{c,a} a\frac{P(Y=y,A=a,C=c|Q=q)}{P(Y=y,C=c|Q=q)}P(C=c|Y=y,Q=q)\\
&=\sum_{c,a} a\frac{P(Y=y|A=a,C=c,Q=q)}{P(Y=y|Q=q)}P(A=a,C=c|Q=q)\\
&=E_{C,A}[A\frac{P(Y=y|A,C,Q=q)}{P(Y=y|Q=q)}|Q=q].
\end{aligned}$$

As in the proof of Proposition 7, we have by Proposition 4 we have that conditional on and $Q=q$, $\frac{P(Y=1|A,C,Q=q)}{P(Y=1|Q=q)}$ is a non-decreasing function of A and of each dimension of C. Similarly, $\frac{P(Y=0|A,C,Q=q)}{P(Y=0|Q=q)}$ is a non-increasing function of A and each dimension of C. Over c and a, conditional on and $Q=q$, $\frac{P(Y=y|A=a,C=c,Q=q)}{P(Y=y|Q=q)}$ is a weight function that sums to 1, that is, $E_{C,A}[\frac{P(Y=y|A=a,C=c,Q=q)}{P(Y=y|Q=q)}]=$

$\frac{P(Y=y|Q=q)}{P(Y=y|Q=q)} = 1$. Furthermore, by Proposition 4, $S(a|c,q)$ is non-decreasing in c and we thus have that

$$\begin{aligned} E[A|Y=1,Q=q] &= E_{C,A}[A\frac{P(Y=1|A,C,Q=q)}{P(Y=1|Q=q)}|Q=q] \\ &\geq E_{C,A}[A\frac{P(Y=0|A,C,Q=q)}{P(Y=0|Q=q)}|Q=q] \\ &= E[A|Y=0,Q=q] \end{aligned}$$

and so $E[A|Y,Q]$ is non-decreasing in Y.

A.12 Proof of Proposition 9

Note that by Proposition 3 above if A_1 has a weak positive monotonic effect on Y then $E[Y|A_1 = a_1, A_2 = a_2, C = c]$ must be non-decreasing in a_1 and if A_1 has a weak negative monotonic effect on Y then $E[Y|A_1 = a_1, A_2 = a_2, C = c]$ must be non-increasing in a_1. Since $(Y \coprod A_1|\{A_2,C\})_{G_{\underline{E_1}}}$ where $G_{\underline{E_1}}$ is the original directed acyclic graph G with all edges emanating from A_1 removed, we have $Y_{A_1=a} \coprod A_1|\{A_2,C\}$ (Pearl, 1995). Thus $E[Y_{A_1=a_1}|A_2 = a_2, C = c] = E[Y|A_1 = a_1, A_2 = a_2, C = c]$ and so if A_2 is a qualitative effect modifier for the causal effect of A_1 on Y for stratum $C = c$ then we must two values of A_1, a_1^* and a_1^{**}, and two levels of A_2, a_2' and a_2'', such that $E[Y|A_1 = a_1^{**}, A_2 = a_2'', C = c] - E[Y|A_1 = a_1^*, A_2 = a_2'', C = c] < 0$ and $E[Y|A_1 = a_1^{**}, A_2 = a_2', C = c] - E[Y|A_1 = a_1^*, A_2 = a_2', C = c] > 0$. Either $a_1^{**} > a_1^*$ or $a_1^{**} < a_1^*$. Consider the first case (the second is analogous) then since $E[Y|A_1 = a_1^{**}, A_2 = a_2'', C = c] - E[Y|A_1 = a_1^*, A_2 = a_2'', C = c] < 0$, A_1 does not have a weak positive monotonic effect on Y and since $E[Y|A_1 = a_1^{**}, A_2 = a_2', C = c] - E[Y|A_1 = a_1^*, A_2 = a_2', C = c] > 0$, A_1 does not have a weak negative monotonic effect on Y. Now if A_2 is a qualitative effect modifier for the causal effect of A_1 unconditionally then we must have two values of A_1, a_1^* and a_1^{**}, and two levels of A_2, a_2' and a_2'', such that $E[Y_{A_1=a_1^{**}}|A_2 = a_2''] - E[Y_{A_1=a_1^*}|A_2 = a_2''] < 0$ and $E[Y_{A_1=a_1^{**}}|A_2 = a_2'] - E[Y_{A_1=a_1^*}|A_2 = a_2'] > 0$. Once again either $a_1^{**} > a_1^*$ or $a_1^{**} < a_1^*$. We will consider the first case (the second is analogous). We thus have that $\sum_c E[Y|A_1 = a_1^{**}, A_2 = a_2'', C = c]P(C = c|A_2 = a_2'') = \sum_c E[Y_{A_1=a_1^{**}}|A_2 = a_2'', C = c]P(C = c|A_2 = a_2'') = E[Y_{A_1=a_1^{**}}|A_2 = a_2''] < E[Y_{A_1=a_1^*}|A_2 = a_2''] = \sum_c E[Y_{A_1=a_1^*}|A_2 = a_2'', C = c]P(C = c|A_2 = a_2'') = \sum_c E[Y|A_1 = a_1^*, A_2 = a_2'', C = c]P(C = c|A_2 = a_2'')$ and so A_1 cannot have a weak positive monotonic effect on Y and similarly, $\sum_c E[Y|A_1 = a_1^{**}, A_2 = a_2', C = c]P(C = c|A_2 = a_2') = \sum_c E[Y_{A_1=a_1^{**}}|A_2 = a_2', C = c]P(C = c|A_2 = a_2') = E[Y_{A_1=a_1^{**}}|A_2 = a_2'] > E[Y_{A_1=a_1^*}|A_2 = a_2'] = \sum_c E[Y_{A_1=a_1^*}|A_2 = a_2', C = c]P(C = c|A_2 = a_2') = \sum_c E[Y|A_1 = a_1^*, A_2 = a_2', C = c]P(C = c|A_2 = a_2')$ and so A_1 cannot have a weak negative monotonic effect on Y.

A.13 Proof of Proposition 10

We prove the Theorem for weak positive monotonic effects. The proof for weak negative monotonic effects is similar. Let C denote all non-descendents of A which are either parents of Y or parents of a node on a directed path between A and Y. By the law of iterated expectations we have $E[Y_{A=a_1}|Q = q] - E[Y_{A=a_0}|Q = q] = \sum_c E[Y_{A=a_1}|C = c, Q = q]P(C = c|Q = q) - \sum_c E[Y_{A=a_0}|C = c, Q = q]P(C = c|Q = q)$. We will show that this latter expression is equal to $\sum_c E[Y_{A=a_1}|C = c]P(C = c|Q = q) - \sum_c E[Y_{A=a_0}|C = c]P(C = c|Q = q)$. By Theorem 3 of Pearl (1995) it suffices

$$\begin{tikzcd} C \arrow[r, "\oplus"] \arrow[rr, bend right, "\oplus"'] & A \arrow[r, "\oplus"] & Y \end{tikzcd}$$

Figure 5: Directed acyclic graph illustrating counterexamples to Propositions 5 and 6 when A is not binary.

to show that $(Y\coprod Q|C,A)_{G_{\overline{A}}}$ where $G_{\overline{A}}$ denotes the graph obtained by deleting from the original directed acyclic graph all arrows pointing into A. Any front door path from Y to Q in $G_{\overline{A}}$ will be blocked by a collider. Any backdoor path from Y to Q in $G_{\overline{A}}$ will be blocked by C. We thus have that $E[Y_{A=a_1}|Q=q]-E[Y_{A=a_0}|Q=q]=\sum_c E[Y_{A=a_1}|C=c]P(C=c|Q=q)-\sum_c E[Y_{A=a_0}|C=c]P(C=c|Q=q)$. Since C will block all backdoor paths from A to Y we have by the backdoor path adjustment theorem $\sum_c E[Y|C=c,A=a_1]P(C=c|Q=q)-\sum_c E[Y|C=c,A=a_0]P(C=c|Q=q)=\sum_c\{E[Y|C=c,A=a_1]-E[Y|C=c,A=a_0]\}P(C=c|Q=q)$. If there were a qualitative effect modifier Q for the causal effect of A on Y then there would exist a value q_0 such that $E[Y_{A=a_1}|Q=q_0]-E[Y_{A=a_0}|Q=q_0]<0$. But since all paths between A and Y are of positive sign and since C blocks all backdoor paths from A to Y we have by Proposition 4 that $E[Y|C=c,A=a]$ is non-decreasing in a and so $E[Y_{A=a_1}|Q=q_0]-E[Y_{A=a_0}|Q=q_0]=\sum_c\{E[Y|C=c,A=a_1]-E[Y|C=c,A=a_0]\}P(C=c|Q=q_0)\geq 0$.

Appendix B. Counterexamples

This appendix contains the details of the counterexamples mentioned in the body of the paper.

B.1 Counterexample 1

Consider the directed acyclic graph given in Figure 5. In this example C and Y are binary and A is ternary. Suppose that $C\sim Ber(0.5)$, $\varepsilon_A\sim Ber(0.5)$ and that $P(A=0|\varepsilon_A=0)=1$ and $P(A=C+1|\varepsilon_A=1)=1$. Suppose also that $P(Y=1|A=2)=1$ and that if $P(Y=C|A=0)=1$ and $P(Y=C|A=1)=1$. Clearly then C has a positive monotonic effect on A and on Y and A has a positive monotonic effect on Y and so A and Y are positively monotonically associated. However, we have that $E[Y|A=1]=E[C|A=1]=0*P(C=1|A=1)=0$ but $E[Y|A=0]=E[C|A=0]=1*P(C=1|A=0)+0*P(C=0|A=0)=1/2$.

B.2 Counterexample 2

Consider again the directed acyclic graph given in Figure 5. In this example we will assume that C and A are binary and that Y is ternary. Suppose that $C\sim Ber(0.5)$ and that ε_A takes on the values 0, 1 and 2, each with probability $1/3$. Suppose also that $P(A=0|\varepsilon_A=0)=1$, $P(A=C|\varepsilon_A=1)=1$ and

$P(A=1|\varepsilon_A=2)=1$. Suppose further that $P(Y=0|C=0)=1$ and if $P(Y=A+1|C=1)$. Clearly then C has a positive monotonic effect on A and on Y and A has a positive monotonic effect on Y and so A and Y are positively monotonically associated. However, we have that $E[A|Y=1]=0$ but $E[A|Y=0]=E[A|C=0]=1/3$.

References

Alexander Philip Dawid. Influence diagrams for causal modelling and inference. *Int. Statist. Rev.*, 70:161–189, 2002.

Marek J. Druzdzel and Max Henrion. Efficient reasoning in qualitative probabilistic networks. In *Proceedings of the National Conference on Artificial Intelligence*, pages 548–553, Washington D. C., 1993.

Dan Geiger, Thomas S. Verma, and Judea Pearl. Identifying independence in bayesian networks. *Networks*, 20:507–534, 1990.

Trygve Haavelmo. The statistical implications of a system of simultaneous equations. *Econometrica*, 11:1–12, 1943.

Changsung Kang and Jin Tian. Inequality constraints in causal models with hidden variables. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, pages 233–240, Cambridge, 2006. AUAI Press.

Changsung Kang and Jin Tian. Polynomial constraints in causal bayesian networks. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, pages 200–208, Cambridge, 2007. AUAI Press.

Sefan L. Lauritzen, Alexander Philip Dawid, B. N. Larsen, and H. G. Leimer. Independence properties of directed markov fields. *Networks*, 20:491–505, 1990.

Judea Pearl. Causal diagrams for empirical research. *Biometrika*, 82:669–688, 1995.

Judea Pearl. *Causality: Models, Reasoning, and Inference*. Cambridge University Press, Cambridge, 2000.

Judea Pearl. Direct and indirect effects. In *Proceedings of the Conference on Uncertainty and Artificial Intelligence*, pages 411–420, San Francisco, 2001. Morgan Kaufmann.

James M. Robins. Semantics of causal dag models and the identification of direct and indirect effects. In P. Green, N.L. Hjort, and S. Richardson, editors, *Highly Structured Stochastic Systems*, pages 70–81. Oxford University Press, New York, 2003.

James M. Robins and Sander Greenland. Identifiability and exchangeability for direct and indirect effects. *Epidemiol.*, 3:143–155, 1992.

Peter Spirtes, Clark Glymour, and Richard Scheines. *Causation, Prediction and Search*. Springer-Verlag, New York, 2000.

Jin Tian and Judea Pearl. On the testable implications of causal models with hidden variables. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, pages 519–527, San Francisco, 2002. Morgan Kaufmann.

Linda C. van der Gaag, Hans L. Bodlaender, and Ad Feelders. Monotonicity in bayesian networks. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, pages 519–527, Banff Canada, 2004. AUAI Press.

Tyler J. VanderWeele and James M. Robins. Four types of effect modification, a classification based on directed acyclic graphs. *Epidemiol.*, 18:561–568, 2007.

Tyler J. VanderWeele and James M. Robins. Empirical and counterfactual conditions for sufficient cause interactions. *Biometrika*, 95:49–61, 2008.

Tyler J. VanderWeele and James M. Robins. Signed directed acyclic graphs for causal inference. *Journal of the Royal Statistical Society Series B*, in press, 2009.

Thomas S. Verma and Judea Pearl. Causal networks: Semantics and expressiveness. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, volume 4, pages 352–359, 1988.

Michael P. Wellman. Fundamental concepts of qualitative probabilistic networks. *Artificial Intelligence*, 44:257–303, 1990.

Sewall Wright. Correlation and causation. *J. Agric. Res.*, 20:557–585, 1921.

Journal of Machine Learning Research 10 (2009) 719-742 Submitted 9/07; Revised 9/08; Published 3/09

On Efficient Large Margin Semisupervised Learning: Method and Theory

Junhui Wang JWANG@STAT.COLUMBIA.EDU
Department of Statistics
Columbia University
New York, NY 10027, USA

Xiaotong Shen XSHEN@STAT.UMN.EDU
School of Statistics
University of Minnesota
Minneapolis, MN 55455, USA

Wei Pan WEIP@BIOSTAT.UMN.EDU
Division of Biostatistics
University of Minnesota
Minneapolis, MN 55455, USA

Editor: John Shawe-Taylor

Abstract

In classification, semisupervised learning usually involves a large amount of unlabeled data with only a small number of labeled data. This imposes a great challenge in that it is difficult to achieve good classification performance through labeled data alone. To leverage unlabeled data for enhancing classification, this article introduces a large margin semisupervised learning method within the framework of regularization, based on an efficient margin loss for unlabeled data, which seeks efficient extraction of the information from unlabeled data for estimating the Bayes decision boundary for classification. For implementation, an iterative scheme is derived through conditional expectations. Finally, theoretical and numerical analyses are conducted, in addition to an application to gene function prediction. They suggest that the proposed method enables to recover the performance of its supervised counterpart based on complete data in rates of convergence, when possible.

Keywords: difference convex programming, classification, nonconvex minimization, regularization, support vectors

1. Introduction

Semisupervised learning occurs in classification, where only a small number of labeled data is available with a large amount of unlabeled data, because of the difficulty of labeling. In artificial intelligence, one central issue is how to integrate human's intelligence with machine's processing capacity. This occurs, for instance, in webpage classification and spam email detection, where webpages and emails are automatically collected, yet require labeling manually or classification by experts. The reader may refer to Blum and Mitchell (1998), Amini and Gallinari (2003), and Balcan et al. (2005) for more details. In genomics applications, functions of many genes in sequenced genomes remain unknown, and are predicted using available biological information, see Xiao and Pan (2005). In situations as such, the primary goal is to leverage unlabeled data to enhance predictive performance of classification (Zhu, 2005).

In semisupervised learning, labeled data $\{(x_i, y_i)_{i=1}^{n_l}\}$ are sampled from an unknown distribution $P(x, y)$, together with an independent unlabeled sample $\{x_j\}_{j=n_l+1}^{n}$ from its marginal distribution $q(x)$. Here label $y_i \in \{-1, 1\}$, $x_i = (x_{i1}, \cdots, x_{id})$ is an d-dimensional input, $n_l \ll n_u$ and $n = n_l + n_u$ is the combined size of labeled and unlabeled samples.

Two types of approaches—distributional and margin-based, have been proposed in the literature. The distributional approach includes, among others, co-training (Blum and Mitchell, 1998), the EM method (Nigam et al., 1998), the bootstrap method (Collins and Singer, 1999), Gaussian random fields (Zhu, Ghahramani and Lafferty, 2003), and structure learning models (Ando and Zhang, 2005). The distributional approach relies on an assumption relating the class probability given input $p(x) = P(Y = 1|X = x)$ to $q(x)$ for an improvement to occur. However, the assumption of this sort is often not verifiable or met in practice.

A margin approach uses the concept of regularized separation. It includes Transductive SVM (TSVM; Vapnik, 1998; Chapelle and Zien, 2005; Wang, Shen and Pan, 2007), and a large margin method of Wang and Shen (2007). These methods use the notation of separation to borrow information from unlabeled data to enhance classification, which relies on the clustering assumption (Chapelle and Zien, 2005) that the clustering boundary can precisely approximate the Bayes decision boundary which is the focus of classification.

This article develops a large margin semisupervised learning method, which aims to extract the information from unlabeled data for estimating the Bayes decision boundary. This is achieved by constructing an efficient loss for unlabeled data with regard to reconstruction of the Bayes decision boundary and by incorporating some knowledge from an estimate of p. This permits efficient use of unlabeled data for accurate estimation of the Bayes decision boundary thus enhancing the classification performance based on labeled data alone. The proposed method, using both the grouping (clustering) structure of unlabeled data and the smoothness structure of p, is designed to recover the classification performance based on complete data without missing labels, when possible.

The proposed method has been implemented through an iterative scheme, which can be thought of as an analogy of Fisher's efficient scoring method (Fisher, 1946). That is, given a consistent initial classifier, an iterative improvement can be obtained through the constructed loss function. Numerical analysis indicates that the proposed method performs well against several state-of-the-art semisupervised methods, including TSVM and Wang and Shen (2007), where Wang and Shen (2007) compares favorably against several smooth and clustering based semisupervised methods.

A novel statistical learning theory for ψ-loss is developed to provide an insight into the proposed method. The theory reveals that the ψ-learning classifier's generalization performance based on complete data can be recovered by its semisupervised counterpart based on incomplete data in rates of convergence, when some regularity assumptions are satisfied. The theory also says that the least favorable situation for a semisupervised problem occurs at points near $p(x) = 0$ or 1 because little information can be provided by these points for reconstructing the classification boundary as discussed in Section 2.3. This is in contrast to the fact that the least favorable situation for a supervised problem occurs near $p(x) = 0.5$. In conclusion, this semisupervised method achieves the desired objective of delivering higher generalization performance.

This article also examines one novel application in gene function prediction, which has been a primary focus of biomedical research. In gene function prediction, microarray gene expression profiles can be used to predict gene functions, because genes sharing the same function tend to co-express, see Zhou, Kao and Wong (2002). Unfortunately, biological functions of many discovered genes remain unknown at present. For example, about 1/3 to 1/2 of the genes in the genome of bacterium *E. coli* have unknown functions. Therefore, gene function prediction is an ideal application for semisupervised methods and also employed in this article as a real numerical example.

This article is organized in six sections. Section 2 introduces the proposed method. Section 3 develops an iterative algorithm for implementation. Section 4 presents some numerical examples, together with an application to gene function prediction. Section 5 develops a learning theory. Section 6 contains a discussion, and the appendix is devoted to technical proofs.

2. Methodology

In this section, we present our proposed efficient large margin semisupervised learning method as well its connection to other existing popular methodologies.

2.1 Large Margin Classification

Consider large margin classification with labeled data $(x_i, y_i)_{i=1}^{n_l}$. In linear classification, given a class of candidate decision functions $\mathcal{F}$, a cost function

$$C\sum_{i=1}^{n_l} L(y_i f(x_i)) + J(f) \tag{1}$$

is minimized over $f \in \mathcal{F} = \{f(x) = \tilde{w}_f^T x + w_{f,0} \equiv (1, x^T)w_f\}$ to yield the minimizer $\hat{f}$ leading to classifier sign$(\hat{f})$. Here $J(f)$ is the reciprocal of the geometric margin of various form with the usual L_2 margin $J(f) = \|\tilde{w}_f\|^2/2$ to be discussed in further detail, and $L(\cdot)$ is a margin loss defined by functional margin $z = yf(x)$, and $C > 0$ is a regularization parameter. In nonlinear classification, a kernel $K(\cdot,\cdot)$ is introduced for flexible representations: $f(x) = \sum_{i=1}^{n_l} \alpha_i K(x, x_i) + b$. For this reason, it is referred to as kernel-based learning, where the reproducing kernel Hilbert spaces (RKHS) are useful, see Gu (2000) and Wahba (1990).

Different margin losses correspond to different learning methodologies. Margin losses include, among others, the hinge loss $L(z) = (1-z)_+$ for SVM with its variant $L(z) = (1-z)_+^q$ for $q > 1$; see Lin (2002); the ψ-losses $L(z) = \psi(z)$, with $\psi(z) = 1 - \text{sign}(z)$ if $z \geq 1$ or $z < 0$, and $2(1-z)$ otherwise, see Shen et al. (2003), the logistic loss $V(z) = \log(1 + e^{-z})$, see Zhu and Hastie (2005); the η-hinge loss $L(z) = (\eta - z)_+$ for nu-SVM (Schölkopf et al., 2000) with $\eta > 0$ being optimized; the sigmoid loss $L(z) = 1 - \tanh(cz)$; see Mason, Baxter, Bartlett and Frean (2000). A margin loss $L(z)$ is said to be large margin if $L(z)$ is non-increasing in z, penalizing small margin values. In this article, we fix $L(z) = \psi(z)$.

2.2 Loss Construction for Unlabeled Data

In classification, the optimal Bayes rule is defined by $\bar{f}_{.5} = \text{sign}(f_{.5})$ with $f_{.5}(x) = P(Y = 1|X = x) - 0.5$ being a global minimizer of the generalization error $GE(f) = EI(Y \neq \text{sign}(f(X)))$, which is usually estimated by labeled data through $L(\cdot)$ in (1). In absence of sufficient labeled data, the focus is on how to improve (1) by using additional unlabeled data. For this, we construct a margin loss U to measure the performance of estimating $\bar{f}_{.5}$ for classification through unlabeled data. Specifically, we seek the best loss U from a class of candidate losses of form $T(f)$, which minimizes the L_2-

distance between the target classification loss $L(yf)$ and $T(f)$. The expression of this loss U is given in Lemma 1.

Lemma 1 *(Optimal loss) For any margin loss $L(z)$,*

$$\operatorname*{argmin}_T E(L(Yf(X)) - T(f(X)))^2 = E(L(Yf(X))|X = x) = U(f(x)),$$

where $U(f(x)) = p(x)L(f(x)) + (1 - p(x))L(-f(x))$ and $p(x) = P(Y = 1|X = x)$. Moreover, $\operatorname{argmin}_{f\in\mathcal{F}} EU(f(X)) = \operatorname{argmin}_{f\in\mathcal{F}} EL(Yf(X))$.

Based on Lemma 1, we define $\hat{U}(f)$ to be $\hat{p}(x)L(f(x)) + (1 - \hat{p}(x))L(-f(x))$ by replacing p in $U(f)$ by $\hat{p}$. Clearly, $\hat{U}(f)$ approximates the ideal loss $U(f)$ for reconstructing the Bayes decision function $f_{.5}$ when $\hat{p}$ is a good estimate of p, as suggested by Corollary 5. This is analogous to construction of the efficient scores for Fisher's scoring method: an optimal estimate can be obtained iteratively through an efficient score function, provided that a consistent initial estimate is supplied, see McCullagh and Nelder (1983) for more details. Through (approximately) optimal loss $\hat{U}(f)$, an iterative improvement of estimation accuracy is achieved by starting with a consistent estimate $\hat{p}$ of p, which, for instance, can be obtained through SVM or TSVM. For $\hat{U}(f)$, its optimality is established through its closeness to $U(f)$ in Corollary 5, where our iterative method based on $\hat{U}$ is shown to yield an iterative improvement in terms of the classification accuracy, recovering the generalization error rate of its supervised counterpart based on complete data ultimately.

As a technical remark, we note that the explicit relationship between p and f is usually unavailable in practice. As a result, several large margin classifiers such as SVM and ψ-learning do not directly yield an estimate of p given $\hat{f}$. Therefore p needs to be either assumed or estimated. For instance, the methods of Wahba (1999) and Platt (1999) assume a parametric form of p so that an estimated $\hat{f}$ yields an estimated p, whereas Wang, Shen and Liu (2008) estimates p given $\hat{f}$ nonparametrically.

The preceding discussion leads to our proposed cost function:

$$s(f) = C\left(n_l^{-1}\sum_{i=1}^{n_l} L(y_i f(x_i)) + n_u^{-1}\sum_{j=n_l+1}^{n} \hat{U}(f(x_j))\right) + J(f). \tag{2}$$

Minimization of (2) with respect to $f \in \mathcal{F}$ gives our estimated decision function $\hat{f}$ for classification.

2.3 Connection with Clustering Assumption

We now intuitively explain advantages of $\hat{U}(f)$ over a popular large margin loss $L(|f|) = (1 - |f(x)|)_+$ (Vapnik, 1998; Wang and Shen, 2007), and its connection with the clustering assumption (Chapelle and Zien, 2005) that assumes closeness between the classification and grouping (clustering) boundaries.

First, $\hat{U}(f)$ has an optimality property, as discussed in Section 2.2, which leads to better performance as suggested by Theorem 3. Second, it has a higher discriminative power over its counterpart $L(|f|)$. To see this aspect, note that $L(|f|) = \inf_p U(f)$ by Lemma 1 of Wang and Shen (2007). This says that $L(|f|)$ is a version of $\hat{U}(f)$ in the least favorable situation where unknown p is estimated by $\operatorname{sign}(f)$, completely ignoring the magnitude of p. As displayed in Figure 1, $\hat{U}(f)$ corresponds to an "asymmetric" hat function or the solid line, whereas $L(|f|)$ corresponds to a "symmetric" one or the dashed line. By comparison, $\hat{U}(f)$ enables not only to identify the clustering boundary through the hat function as $L(|f|)$ does but also to discriminate $f(x)$ from $-f(x)$ through an estimated $\hat{p}(x)$. That is, $\hat{U}(f)$ has a smaller value for $f(x) > 0$ than for $-f(x) < 0$ when $\hat{p} > 0.5$, and vice versa, whereas $L(|f|)$ is in-discriminative with regard to the sign of $f(x)$.

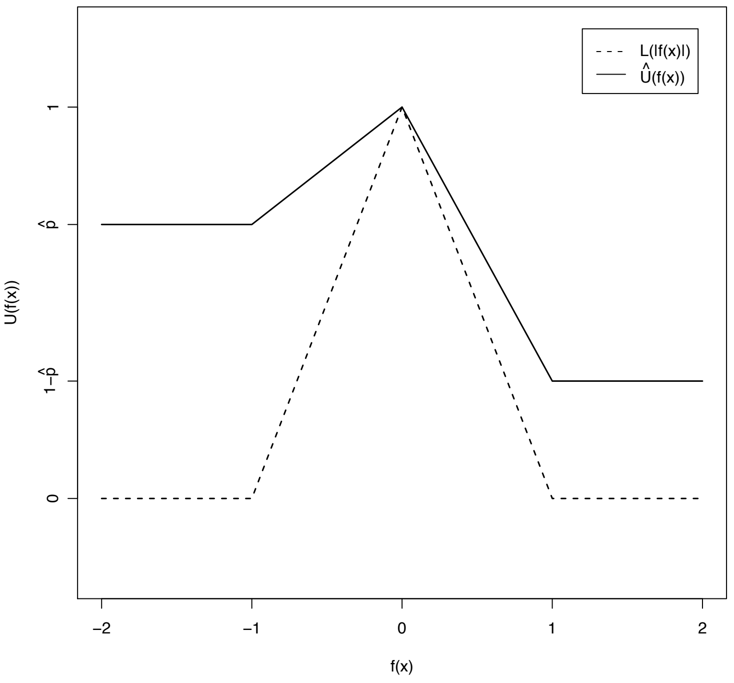


Figure 1: Plots of $L(|f(x)|)$ and $\hat{U}(f(x))$.

To reinforce the second point in the foregoing discussion, we examine one specific example with two possible clustering boundaries as described in Figure 3 of Zhu (2005). There $\hat{U}(f)$ favors one clustering boundary for classification if a consistent $\hat{p}$ is provided, whereas $L(|f|)$ fails to discriminate these two. More details are deferred to Section 4.1, where the simulated example 2 of this nature is studied.

In conclusion, $\hat{U}(f)$ yields a more efficient loss for a semisupervised problem as it uses the clustering information from the unlabeled data as $L(|f|)$ does, in addition to guidance about labeling through $\hat{p}$ to gain a higher discriminative power.

3. Computation

In this section, we implement the proposed semisupervised method through an iterative scheme as well as a nonconvex optimization technique.

3.1 Iterative Scheme

Effectiveness of $\hat{U}$ depends largely on the accuracy of $\hat{p}$ in estimating p. Given an estimate $\hat{p}^{(0)}$, (2) yields an estimate $\hat{f}^{(1)}$, which leads to a new estimate $\hat{p}^{(1)}$ through **Algorithm 0** below. The $\hat{p}^{(1)}$ is expected to be more accurate than $\hat{p}^{(0)}$ for p because additional information from unlabeled data has been used in estimation of $\hat{f}^{(1)}$ through $\hat{p}^{(0)}$ and additional smoothness structure has been used in **Algorithm 0** in estimation of $\hat{p}^{(1)}$ given $\hat{f}^{(1)}$. Specifically, an improvement in the process from $\hat{p}^{(0)}$ to $\hat{f}^{(1)}$ and that from $\hat{f}^{(1)}$ to $\hat{p}^{(1)}$ are assured by Assumptions B and D in Section 5.1, respectively, which are a more general version of the clustering assumption and a smoothness assumption of p. In other words, the marginal information from unlabeled data has been effectively incorporated in each iteration of **Algorithm 1** for improving estimation accuracy of $\hat{f}$ and $\hat{p}$.

Detailed implementation of the preceding scheme as well as the conditional probability estimation are summarized as follows.

Algorithm 0: (Conditional probability estimation; Wang, Shen and Liu, 2008)
Step 1. Specify m and initialize $\pi_t = (t-1)/m$, for $t = 1, \ldots, m+1$.
Step 2. Train weighted margin classifiers $\hat{f}_{\pi_t}$ by solving

$$\min_{f \in \mathcal{F}} Cn^{-1}\left((1-\pi_t)\sum_{y_i=1} L(y_i f(x_i)) + \pi_t \sum_{y_i=-1} L(y_i f(x_i))\right) + J(f),$$

with $1-\pi_t$ associated with positive instances and π_t associated with negative instances.
Step 3. Estimate labels of x by $\text{sign}(\hat{f}_{\pi_t}(x))$.
Step 4. Sort $\text{sign}\{\hat{f}_{\pi_t}(x)\}$, $t = 1, \ldots, m+1$, to compute $\pi^* = \max\{\pi_t : \text{sign}(\hat{f}_{\pi_t}(x)) = 1\}$, $\pi_* = \min\{\pi_t : \text{sign}(\hat{f}_{\pi_t}(x)) = -1\}$. The estimated class probability is $\hat{p}(x) = \frac{1}{2}(\pi^* + \pi_*)$.

Algorithm 1: (Efficient semisupervised learning)
Step 1. (Initialization) Given any initial classifier $\text{sign}(\hat{f}^{(0)})$, compute $\hat{p}^{(0)}$ through **Algorithm 0**. Specify precision tolerance level ε.
Step 2. (Iteration) At iteration $k+1$; $k = 0, 1, \cdots$, minimize $s(f)$ in (2) for $\hat{f}^{(k+1)}$ with $\hat{U} = \hat{U}^{(k)}$ defined by $\hat{p} = \hat{p}^{(k)}$ there. This is achieved through sequential QP for the ψ-loss. Details for sequential QP are deferred to Section 3.2. Compute $\hat{\hat{p}}^{(k+1)}$ through **Algorithm 0**, based on complete data with unknown labels imputed by $\text{sign}(\hat{f}^{(k+1)})$. Define $\hat{p}^{(k+1)} = \max(\hat{p}^{(k)}, \hat{\hat{p}}^{(k+1)})$ when $\hat{f}^{(k+1)} \geq 0$ and $\min(\hat{p}^{(k)}, \hat{\hat{p}}^{(k+1)})$ otherwise.
Step 3. (Stopping rule) Terminate when $|s(\hat{f}^{(k+1)}) - s(\hat{f}^{(k)})| \leq \varepsilon |s(\hat{f}^{(k)})|$. The final solution $\hat{f}_C$ is $\hat{f}^{(K)}$, with K the number of iterations to termination in **Algorithm 1**.

Theorem 2 *(Monotonicity) $s(\hat{f}^{(k)})$ is non-increasing in k. As a consequence,* **Algorithm 1** *converges to a stationary point $s(\hat{f}^{(\infty)})$ in that $s(\hat{f}^{(k)}) \geq s(\hat{f}^{(\infty)})$. Moreover,* **Algorithm 1** *terminates finitely.*

Algorithm 1 differs from the EM algorithm and its variant MM algorithm (Hunter and Lange, 2000) in that little marginal information has been used in these algorithms as argued in Zhang and Oles (2000). **Algorithm 1** also differs from Yarowsky's algorithm (Yarowsky, 1995; Abney, 2004) in that Yarowsky's algorithm solely relies on the strength of the estimated $\hat{p}$, ignoring the potential information from the clustering assumption.

There are several important aspects of **Algorithm 1**. First, loss $L(\cdot)$ in (2) may not be a likelihood regardless of if labeling missingness occurs at random. Secondly, the monotonicity property, as established in Theorem 2, is assured by constructing $\hat{p}^{(k+1)}$ to satisfy $(\hat{p}^{(k+1)} - \hat{p}^{(k)})\hat{f}^{(k+1)} \geq 0$, as opposed to the property of likelihood in the EM algorithm. Most importantly, the smoothness and clustering assumptions have been used in estimating p, and thus semisupervised learning. This is in contrast to the EM, where only likelihood is used in estimating p in a supervised manner.

Finally, we note that in **Step 2** of **Algorithm 1**, given $\hat{p}^{(k)}$, minimization in (2) involves nonconvex minimization when $L(\cdot)$ is ψ-loss. Next we shall discuss how to solve (2) for $\hat{f}^{(k+1)}$ through difference convex (DC) programming for nonconvex minimization.

3.2 Nonconvex Minimization

This section develops a nonconvex minimization method based on DC programming (An and Tao, 1997) for (2) with the ψ-loss, which was previously employed in Liu, Shen and Wong (2005) for supervised ψ-learning. As a technical remark, we note that DC programming has a high chance to locate an ε-global minimizer (An and Tao, 1997), although it can not guarantee globality. In fact, when combined with the method of branch-and-bound, it yields a global minimizer, see Liu et al. (2005). For a computational consideration, we shall use the DC programming algorithm without seeking an exact global minimizer.

Key to DC programming is decomposing the cost function $s(f)$ in (2) with $L(z) = \psi(z)$ into a difference of two convex functions as follows:

$$\begin{aligned} s(f) &= s_1(f) - s_2(f); \qquad (3) \\ s_1(f) &= C\big(n_l^{-1}\sum_{i=1}^{n_l}\psi_1(y_i f(x_i)) + n_u^{-1}\sum_{j=n_l+1}^{n}\hat{U}_{\psi_1}^{(k)}(f(x_j))\big) + J(f); \\ s_2(f) &= C\big(n_l^{-1}\sum_{i=1}^{n_l}\psi_2(y_i f(x_i)) + n_u^{-1}\sum_{j=n_l+1}^{n}\hat{U}_{\psi_2}^{(k)}(f(x_j))\big), \end{aligned}$$

where $\hat{U}_{\psi_t}^{(k)}(f(x_j)) = \hat{p}^{(k)}(x_j)\psi_t(f(x_j)) + (1-\hat{p}^{(k)}(x_j))\psi_t(-f(x_j));\ t = 1,2,\ \psi_1 = 2(1-z)_+$ and $\psi_2 = 2(-z)_+$. Here ψ_1 and ψ_2 are obtained through a convex decomposition of $\psi = \psi_1 - \psi_2$ as displayed in Figure 2.

With these decompositions, we treat (2) with the ψ-loss and $\hat{p} = \hat{p}^{(k)}$ by solving a sequence of quadratic problems described in **Algorithm 2**.

Algorithm 2: (Sequential QP)

Step 1. (Initialization) Set initial $\hat{f}^{(k+1,0)}$ to be the solution of $\min_f s_1(f)$. Specify precision tolerance level ε as in **Algorithm 1**.

Step 2. (Iteration) At iteration $l+1$, compute $\hat{f}^{(k+1,l+1)}$ by solving

$$\min_f \big(s_1(f) - \langle w_f, \nabla s_2(\hat{f}^{(k+1,l)})\rangle\big), \qquad (4)$$

where $\nabla s_2(f^{(k+1,l)})$ is a gradient vector of $s_2(f)$ at $w_{\hat{f}^{(k+1,l)}}$.

Step 3. (Stopping rule) Terminate when $|s(\hat{f}^{(k+1,l+1)}) - s(\hat{f}^{(k+1,l)})| \le \varepsilon |s(\hat{f}^{(k+1,l)})|$.
Then the estimate $\hat{f}^{(k+1)}$ is the best solution among $\hat{f}^{(k+1,l)};\ l = 0,1,\cdots$.

In (4), gradient $\nabla s_2(f^{(k+1,l)})$ is defined as the sum of partial derivatives of s_2 over each observation, with $\nabla\psi_2(z) = 0$ if $z > 0$ and $\nabla\psi_2(z) = -2$ otherwise. By the definition of $\nabla s_2(f^{(k+1,l)})$ and convexity of $s_2(f^{(k+1,l)})$, (4) gives a sequence of non-increasing upper envelops of (3), which can be solved via their dual forms.

The speed of convergence of **Algorithm 2** is super-linear, following the proof of Theorem 3 in Liu et al. (2005). This means that the number of iterations required for **Algorithm 2** to achieve the precision ε is $o(\log(1/\varepsilon))$.

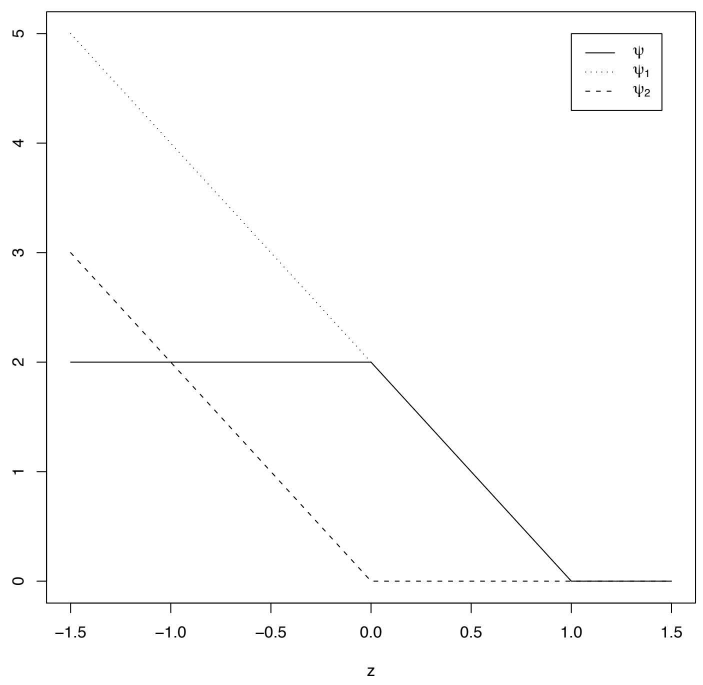


Figure 2: Plot of ψ, ψ_1 and ψ_2 for the DC decomposition of $\psi = \psi_1 - \psi_2$. Solid, dotted and dashed lines represent ψ, ψ_1 and ψ_2, respectively.

4. Numerical Comparison

This section examines effectiveness of the proposed method through numerical examples. A test error, averaged over 100 independent simulation replications, is used to measure a classifier's generalization performance. For simulation comparison, the amount of improvement of our method over $\text{sign}(\hat{f}^{(0)})$ is defined as the percent of improvement in terms of the Bayesian regret

$$\frac{(T(\text{Before}) - \mathit{Bayes}) - (T(\text{After}) - \mathit{Bayes})}{T(\text{Before}) - \mathit{Bayes}}, \tag{5}$$

where $T(\text{Before})$, $T(\text{After})$, and *Bayes* denote the test errors of $\text{sign}(\hat{f}^{(0)})$, the proposed method based on initial classifier $\text{sign}(\hat{f}^{(0)})$, and the Bayes error. The Bayes error is the ideal performance and serves as a benchmark for comparison, which can be computed when the distribution is known. For benchmark examples, the amount of improvement over $\text{sign}(\hat{f}^{(0)})$ is defined as

$$\frac{T(\text{Before}) - T(\text{After})}{T(\text{Before})}, \tag{6}$$

which actually underestimates the amount of improvement in absence of knowledge of the Bayes error.

Numerical analyses are conducted in R2.1.1. In linear learning, $K(x,y) = \langle x,y \rangle$; in Gaussian kernel learning, $K(x,y) = \exp(-\frac{\|x-y\|^2}{\sigma^2})$, where σ is set to be the median distance between positive and negative classes to reduce computational cost for tuning σ^2, see Jaakkola, Diekhans and Haussler (1999).

4.1 Simulations and Benchmarks

Two simulated and five benchmark data sets are examined, based on four state-of-the-art classifiers $\text{sign}(\hat{f}^{(0)})$'s. They are SVM (with labeled data alone), TSVM (TSVM^{DCA}; Wang, Shen and Pan, 2007) , and the methods of Wang and Shen (2007) with the hinge loss (SSVM) and with the ψ-loss (SPSI), where SSVM and SPSI compare favorably against their competitors. Corresponding to these methods, our method, with $m = n^{1/2}$ and $\varepsilon = 10^{-3}$, yields four semisupervised classifiers denoted as ESVM, ETSVM, ESSVM and ESPSI.

4.1.1 SIMULATED EXAMPLES

Examples 1 and 2 are taken from Wang and Shen (2007), where 200 and 800 labeled instances are randomly selected for training and testing. For training, 190 out 200 instances are randomly chosen for removing their labels. Here the Bayes errors are 0.162 and 0.089, respectively.

4.1.2 BENCHMARKS

Six benchmark examples include Wisconsin breast cancer (WBC), Pima Indians diabetes (PIMA), HEART, MUSHROOM, Spam email (SPAM) and Brain computer interface (BCI). The first five datasets are available in the UCI Machine Learning Repository (Blake and Merz, 1998) and the last one can be found in Chapelle et al. (2006). WBC discriminates a benign breast tissue from a malignant one through 9 diagnostic characteristics; PIMA differentiates between positive and negative cases for female diabetic patients of Pima Indian heritage based on 8 biological or diagnostic attributes; HEART concerns diagnosis status of the heart disease based on 13 clinic attributes; MUSHROOM separates an edible mushroom from a poisonous one through 22 biological records; SPAM identifies spam emails using 57 frequency attributes of a text, such as frequencies of particular words and characters; BCI concerns the difference of brain images when imagining left-hand and right-hand movements, based on 117 autoregressive model parameters fitted over human's electroencephalography.

Instances in WBC, PIMA, HEART, and MUSHROOM are randomly divided into two halves with 10 labeled and 190 unlabeled instances for training, and the remaining 400 for testing. Instances in SPAM are randomly divided into halves with 20 labeled and 380 unlabeled instances for training, and the remaining instances for testing. Twelve splits for BCI have already given at http://www.kyb.tuebingen.mpg.de/ssl-book/benchmarks.html, with 10 labeled and 390 unlabeled instances while no instance for testing. An averaged error rate over the unlabeled set is used in BCI example to approximate the test error.

In each example, the smallest test errors of all methods in comparison are computed over 61 grid points $\{10^{-3+k/10};\ k = 0, \cdots, 60\}$ for tuning C in (2) through a grid search. The results are summarized in Tables 1-2.

As suggested in Tables 1-2, ESVM, ETSVM, ESSVM and ESPSI perform no worse than their counterparts in almost all examples, except ESVM in SPAM where the performance is slightly worse but indistinguishable from its counterpart. The amount of improvement, however, varies over examples and different types of classifiers. In linear learning, the improvements of the proposed method are from 1.0% to 51.7% over its counterparts, except in SPAM where ESVM performs slightly worse than SVM; in kernel learning, the improvements range from 0.0% to 23.2% over its counterparts. Overall, large improvement occurs for less accurate initial classifiers when they are sufficiently accurate. However, if the initial classifier is too accurate, the potential for an improve-

Data	Example 1	Example 2	WBC	PIMA	HEART	MUSHROOM	SPAM	BCI[1]
Size	1000 × 2	1000 × 2	682 × 9	768 × 8	303 × 13	8124 × 22	4601 × 57	400 × 117
SVM	.345(.0081)	.333(.0129)	.053(.0071)	.328(.0092)	.284(.0085)	.232(.0135)	.216(.0097)	.479(.0059)
ESVM	.281(.0143)	.297(.0177)	.031(.0007)	.320(.0059)	.214(.0066)	.172(.0084)	.217(.0178)	.474(.0052)
Improv.	35.0%	14.8%	41.5%	2.4%	24.6%	25.9%	-0.5%	1.0%
TSVM	.220(.0103)	.203(.0088)	.037(.0024)	.314(.0086)	.270(.0082)	.206(.113)	.196(.0132)	.479(.0054)
ETSVM	.190(.0074)	.147(.0131)	.029(.0009)	.309(.0063)	.211(.0062)	.153(.0054)	.179(.0101)	.474(.0076)
Improv.	51.7%	49.1%	21.6%	1.6%	21.9%	25.7%	8.7%	1.0%
SSVM	.188(.0084)	.129(.0031)	.032(.0025)	.307(.0054)	.240(.0074)	.186(.0095)	.191(.0114)	.479(.0071)
ESSVM	.182(.0065)	.124(.0034)	.028(.0006)	.293(.0029)	.205(.0059)	.162(.0054)	.169(.0107)	.474(.0041)
Improv.	23.1%	12.5%	12.5%	4.6%	14.6%	11.8%	11.5%	1.0%
SPSI	.184(.0084)	.128(.0084)	.029(.0022)	.291(.0032)	.232(.0067)	.184(.0095)	.189(.0107)	.476(.0068)
ESPSI	.182(.0065)	.123(.0029)	.027(.0006)	.284(.0026)	.181(.0052)	.137(.0067)	.167(.0107)	.471(.0046)
Improv.	9.1%	12.8%	6.9%	2.4%	22.0%	25.5%	11.6%	1.1%
SVM$_c$	.164(.0084)	.115(.0032)	.027(.0020)	.238(.0011)	.176(.0031)	.041(.0018)	.095(.0022)	.173(.0012)[2]

Table 1: Linear learning. Averaged test errors as well as estimated standard errors (in parenthesis) of ESVM, ETSVM, ESSVM, ESPSI, and their initial counterpartsand testing samples, in the simulated and benchmark examples. SVM$_c$ denotes the performance of SVM with complete labeled data. Here the amount of improvement is defined in (5) or (6).

Data	Example 1	Example 2	WBC	PIMA	HEART	MUSHROOM	SPAM	BCI [1]
Size	1000 × 2	1000 × 2	682 × 9	768 × 8	303 × 13	8124 × 22	4601 × 57	400 × 117
SVM	.385(.0099)	.347(.0119)	.047(.0038)	.353(.0089)	.331(.0094)	.217(.0135)	.226(.0108)	.488(.0073)
ESVM	.368(.0077)	.322(.0109)	.039(.0067)	.335(.0035)	.308(.0107)	.187(.0118)	.212(.0104)	.482(.0076)
Improv.	7.6%	9.7%	17.0%	5.1%	6.9%	13.8%	6.2%	1.2%
TSVM	.232(.0122)	.205(.0091)	.037(.0015)	.330(.0107)	.281(.0113)	.185(.0080)	.192(.0110)	.484(.0087)
ETSVM	.216(.0090)	.187(.0084)	.030(.0005)	.304(.0028)	.263(.0094)	.171(.0093)	.181(.0106)	.484(.0086)
Improv.	22.9%	15.5%	18.9%	7.9%	6.4%	7.6%	5.7%	0.0%
SSVM	.201(.0072)	.175(.0092)	.030(.0005)	.304(.0044)	.226(.0063)	.173(.0126)	.189(.0120)	.479(.0080)
ESSVM	.201(.0072)	.170(.0083)	.030(.0005)	.304(.0042)	.223(.0054)	.147(.0105)	.170(.0103)	.476(.0085)
Improv.	0.0%	5.8%	0.0%	0.0%	1.3%	15.0%	10.1%	0.6%
SPSI	.200(.0069)	.175(.0092)	.030(.0005)	.295(.0037)	.215(.0057)	.164(.0123)	.189(.0112)	.475(.0072)
ESPSI	.198(.0072)	.169(.0082)	.030(.0005)	.294(.0033)	.215(.0054)	.126(.0083)	.169(.0091)	.475(.0081)
Improv.	1.0%	7.0%	0.0%	0.3%	0.0%	23.2%	10.6%	0.0%
SVM$_c$	.196(.0015)	.151(.0021)	.030(.0004)	.254(.0013)	.196(.0031)	.021(.0014)	.099(.0018)	.280(.0015)[2]

Table 2: Gaussian kernel learning. Averaged test errors as well as estimated standard errors (in parenthesis) of ESVM, ETSVM, ESSVM, ESPSI, and their initial counterpartsin the simulated and benchmark examples. Here the amount of improvement is defined in (5) or (6).

ment becomes small or null, such as the cases of SSVM and SPSI with Gaussian kernel in PIMA. If the initial classifier is too poor, then no improvement may occur. This is the case for ESVM with linear kernel in SPAM, where ESVM performs worse than SVM with $n_l = 10$ labeled data alone. This suggests that a better initial estimate should be used together with unlabeled data.

In summary, we recommend SPSI to be an initial classifier for $\hat{f}^{(0)}$ based on its overall performance across all the examples. Moreover, ESPSI nearly recovers the classification performance of its counterpart SVM with complete labeled data in the two simulated examples, WBC and HEART.

4.2 Gene Function Prediction Through Expression Profiles

This section applies the proposed method to predict gene functions through gene data in Hughes et al. (2000), consisting of expression profiles of a total of 6316 genes for yeast S. *cerevisiae* from

1. The error rate is computed on the unlabeled data and averaged over twelve splits.
2. This error rate is approximated by the 10-fold cross validation.

300 microarray experiments. In this case almost half of the genes have unknown functions although gene expression profiles are available for almost the entire yeast genome.

Our specific focus is predicting functional categories defined by the MIPS, a multifunctional classification scheme (Mewes et al., 2002). For simplicity, we examine two functional categories, namely "transcriptional control" and "mitochondrion", with 334 and 346 annotated genes, respectively. The goal is to predict gene functional categories for genes annotated within these two categories by training our semisupervised classifier on expression profiles of genes, where some genes are treated as if their functions are unknown to mimic the semisupervised scenario in complete dataset. At present, detection of novel class is not permitted in our formulation, which remains to be an open research question.

For the purpose of evaluation, we divide the entire dataset into two sets of training and testing. The training set involves a random sample of $n_l = 20$ labeled and $n_u = 380$ unlabeled gene profiles, while the testing set contains 280 remaining profiles.

		SVM	TSVM	SSVM	SPSI
	I	.298(.0066)	.303(.0087)	.270(.0075)	.272(.0063)
Linear	*E*	.278(.0069)	.272(.0080)	.261(.0052)	.252(.0112)
	Improv.	6.7%	14.0%	3.3 %	7.4%
	I	.290(.0081)	.287(.0027)	.284(.0111)	.283(.0063)
Gaussian	*E*	.279(.0085)	.279(.0076)	.275(.0086)	.256(.0082)
	Improv.	3.8%	2.8%	3.2%	9.5%

Table 3: Averaged test errors as well as estimated standard errors (in parenthesis) of ESVM, ETSVM, ESSVM, ESPSI, and their initial counterparts, over 100 pairs of training and testing samples, in gene function prediction. Here I stands for an initial classifier, E stands for our proposed method with the initial method, and the amount of improvement is defined in (6).

As indicated in Table 3, ESVM, ETSVM, ESSVM and ESPSI all improve predictive accuracy of their initial counterparts in linear learning and Gaussian kernel learning. It appears that ESPSI performs best. Most importantly, it demonstrates predictive power of the proposed method for predicting which of the two categories a gene belongs to.

5. Statistical Learning Theory

In the literature, several theories have been developed to understand the problem of semisupervised learning, including Rigollet (2007) and Singh, Nowak and Zhu (2008). Both the theories rely on a different clustering assumption that homogeneous labels are assumed over local clusters. Based on the original clustering assumption, as well as a smoothness assumption on the conditional probability $p(x)$, this section develops a novel statistical learning theory. Specifically, finite-sample and asymptotical upper bounds of the generalization error are derived for ESPSI $\hat{f}_C$ defined by the ψ-loss in **Algorithm 1**. The generalization accuracy is measured by the Bayesian regret $e(\hat{f}_C, \bar{f}_{.5}) = GE(\hat{f}_C) - GE(\bar{f}_{.5}) \geq 0$ with $GE(f)$ defined in Section 2.2.

5.1 Statistical Learning Theory

The error bounds of $e(\hat{f}_C, \bar{f}_{.5})$ are expressed in terms of complexity of the candidate class $\mathcal{F}$, the sample size n, tuning parameter $\lambda = (nC)^{-1}$, the error rate of the initial classifier $\delta_n^{(0)}$, and the maximum number of iteration K in **Algorithm 1**. The results imply that ESPSI, without knowing labels of the unlabeled data, enables to recover the classification accuracy of ψ-learning based on complete data under regularity conditions.

We first introduce some notations. Let $L(z) = \psi(z)$ be the ψ-loss. Define the margin loss $V_\pi(f, Z)$ for unequal cost classification to be $S_\pi(y)L(yf(x))$, with cost $0 < \pi < 1$ for the positive class and $S_\pi(y) = 1 - \pi$ if $y = 1$, and π otherwise. Let $e_{V_\pi}(f, \bar{f}_\pi)E(V_\pi(f,Z) - V_\pi(\bar{f}_\pi, Z)) \geq 0$ for $f \in \mathcal{F}$ with respect to unequal cost π, where $\bar{f}_\pi(x) = \text{sign}(f_\pi(x)) = \arg\min_f EV_\pi(f, Z)$ is the Bayes rule, with $f_\pi(x) = p(x) - \pi$.

Assumption A: (Approximation) For any $\pi \in (0,1)$, there exist some positive sequence $s_n \to 0$ as $n \to \infty$ and $f_\pi^* \in \mathcal{F}$ such that $e_{V_\pi}(f_\pi^*, \bar{f}_\pi) \leq s_n$.

Assumption A is an analog of that of Shen et al. (2003), which ensures that the Bayes rule $\bar{f}_\pi$ can be well approximated by elements in $\mathcal{F}$.

Assumption B. (Conversion) For any $\pi \in (0,1)$, there exist constants $0 < \alpha,\ \beta_\pi < \infty$, $0 \leq \zeta < \infty$, $a_i > 0$; $i = 0,1,2$, such that for any sufficiently small $\delta > 0$,

$$\sup_{\{f \in \mathcal{F}: e_{V_{.5}}(f, \bar{f}_{.5}) \leq \delta\}} e(f, \bar{f}_{.5}) \leq a_0 \delta^\alpha, \tag{7}$$

$$\sup_{\{f \in \mathcal{F}: e_{V_\pi}(f, \bar{f}_\pi) \leq \delta\}} \|\text{sign}(f) - \text{sign}(\bar{f}_\pi)\|_1 \leq a_1 \delta^{\beta_\pi}, \tag{8}$$

$$\sup_{\{f \in \mathcal{F}: e_{V_\pi}(f, \bar{f}_\pi) \leq \delta\}} \text{Var}(V_\pi(f,Z) - V_\pi(\bar{f}_\pi, Z)) \leq a_2 \delta^\zeta. \tag{9}$$

Assumption B describes local smoothness of the Bayesian regret $e(f, \bar{f}_{.5})$ in terms of a first-moment function $\|\text{sign}(f) - \text{sign}(\bar{f}_\pi)\|_1$ and a second-moment function $\text{Var}(V_\pi(f,Z) - V_\pi(\bar{f}_\pi, Z))$ relative to $e_{V_\pi}(f, \bar{f}_\pi)$ with respect to unequal cost π. Here the degrees of smoothness are defined by exponents α, β_π and ζ. Note that (7) and (9) are related to the "no noise assumption" of Tsybakov (2004); and (8) has been used in Wang et al. (2008) for quantifying the error rate of probability estimation, which plays a key role in controlling the error rate of ESPSI. For simplicity, denote $\beta_{.5}$ and $\inf_{\pi \neq 0.5}\{\beta_\pi\}$ as β and γ respectively, where β quantifies the clustering assumption through the degree to which the positive and negative clusters are distinguishable, and γ measures the conversion rate between the classification and probability estimation accuracies.

For Assumption C, we define a complexity measure—the L_2-metric entropy with bracketing, describing the cardinality of $\mathcal{F}$. Given any $\varepsilon > 0$, denote $\{(f_r^l, f_r^u)\}_{r=1}^R$ as an ε-bracketing function set of $\mathcal{F}$ if for any $f \in \mathcal{F}$, there exists an r such that $f_r^l \leq f \leq f_r^u$ and $\|f_r^l - f_r^u\|_2 \leq \varepsilon; r = 1, \cdots, R$. Then the L_2-metric entropy with bracketing $H_B(\varepsilon, \mathcal{F})$ is defined as the logarithm of the cardinality of the smallest ε-bracketing function set of $\mathcal{F}$. See Kolmogorov and Tihomirov (1959) for more details.

Define $\mathcal{F}(k) = \{L(f,z) - L(f_\pi^*, z) : f \in \mathcal{F}, J(f) \leq k\}$ to be a space defined by candidate decision functions, with $J(f) = \frac{1}{2}\|f\|_K^2$. Let $J_\pi^* = \max(J(f_\pi^*), 1)$. In (11), we specify an entropy integral to establish a relationship between the complexity of $\mathcal{F}(k)$ and convergence speed ε_n for the Bayesian regret.

Assumption C. (Complexity) For some constants $a_i > 0; i = 3, \cdots, 5$ and $\varepsilon_n > 0$,

$$\sup_{k \geq 2} \phi(\varepsilon_n, k) \leq a_5 n^{1/2}, \tag{10}$$

where $\phi(\varepsilon, k) = \int_{a_4 M}^{a_3^{1/2} M^{\min(1,\zeta)/2}} H_B^{1/2}(w, \mathcal{F}(k)) dw / M$, and $M = M(\varepsilon, \lambda, k) = \min(\varepsilon^2 + \lambda(k/2 - 1) J_\pi^*, 1)$.
Assumption D. (Smoothness of $p(x)$) There exist some constants $0 \leq \eta \leq 1$, $d \geq 0$ and $a_6 > 0$ such that $\|\Delta^j(p)\|_\infty \leq a_6$ for $j = 0, 1, \cdots, d$, and $|\Delta^d(p(x_1)) - \Delta^d(p(x_2))| \leq a_6 \|x_1 - x_2\|_1^\eta + d/m$ for any $\|x_1 - x_2\|_1 \leq \delta$ with some sufficiently small $\delta > 0$, where Δ^j is the j-th order difference operator and m is defined as in **Algorithm 0**.

Assumption D specifies the degree of smoothness of the conditional density $p(x)$.
Assumption E. (Degree of least favorable situation) There exist some constants $0 \leq \theta \leq \infty$ and $a_7 > 0$ such that $P\big(X : \min(p(X), 1 - p(X)) \leq \delta\big) \leq a_7 \delta^\theta$ for any sufficiently small $\delta > 0$.

Assumption E describes the behavior of $p(x)$ near 0 and 1, corresponding to the least favorable situation, as described in Section 2.3.

Theorem 3 *In addition to Assumptions A-E, let the precision parameter m be* $[\delta_n^{-\beta\gamma}]$ *and* $\delta_n^2 = \min(\max(\varepsilon_n^2, 16 s_n), 1)$. *Then for ESPSI* $\hat{f}_C$*, there exist some positive constants* a_8*-*a_{10} *such that*

$$\begin{aligned} P\Big(e(\hat{f}_C, \bar{f}_{.5}) \geq a_{10} \max(\delta_n^{2\alpha}, (a_{11} \rho_n \delta_n^{(0)})^{2\alpha \max(1, B^K)})\Big) \leq \\ P\Big(e_L(\hat{f}_{.5}^{(0)}, \bar{f}_{.5}) \geq 2 a_{11} \rho_n (\delta_n^{(0)})^2\Big) + 3.5 K \exp(-a_8 n_l (\lambda J_{.5}^*)^{\max(1, 2-\zeta)}) + \\ 3.5 K \exp(-a_9 n (\lambda J_\pi^*)^{\max(1, 2-\zeta)}) + 2 K \rho_n^{-\min(1, \beta)}. \end{aligned}$$

Here $B = \frac{(\theta+1)(d+\eta)\beta\gamma}{2(1+\max(0, 1-\beta)\theta)(d+\eta+1)}$, $a_{11} = \max\left(1, 2^{\frac{3\gamma(d+\eta)}{d+\eta+1}+2} a_1^{\frac{(2\gamma+1)(d+\eta)}{d+\eta+1}}\right)$ *and* $\rho_n > 0$ *is any real number satisfying* $a_{11} \rho_n \delta_n^2 \leq 4 \lambda J_\pi^*$.

Theorem 3 provides a finite-sample probability bound for the Bayesian regret $e(\hat{f}_C, \bar{f}_{.5})$, where the parameter B measures the level of difficulty of a semisupervised problem, with small value of B indicating more difficulty. Note that the value of B is proportional to those of α, β, γ, d, η and θ, as defined in Assumptions A-E. In fact, α, β and γ quantify the local smoothness of the Bayesian regret $e(f, \bar{f}_{.5})$, and d, η and θ describe the smoothness of $p(x)$ as well as its behavior near 0 and 1.

Next, by letting n_l, n_u tending infinity, we obtain the rates of convergence of ESPSI in terms of the error rate $\delta_n^{2\alpha}$ of its supervised counterpart ψ-learning based on complete data, and the initial error rate $\delta_n^{(0)}$, B, and the maximum number K of iteration.

Corollary 4 *Under the assumptions of Theorem 3, as* $n_u, n_l \to \infty$,

$$\begin{aligned} |e(\hat{f}_C, \bar{f}_{.5})| &= O_p\Big(\max(\delta_n^{2\alpha}, (\rho_n \delta_n^{(0)})^{2\alpha \max(1, B^K)})\Big), \\ E|e(\hat{f}_C, \bar{f}_{.5})| &= O\Big(\max(\delta_n^{2\alpha}, (\rho_n \delta_n^{(0)})^{2\alpha \max(1, B^K)})\Big), \end{aligned}$$

provided that the initial classifier converges in that $P\Big(e_L(\hat{f}_{.5}^{(0)}, \bar{f}_{.5}) \geq 2 a_{11} \rho_n (\delta_n^{(0)})^2\Big) \to 0$, *with any slow varying sequence* $\rho_n \to \infty$ *and* $\rho_n \delta_n^{(0)} \to 0$, *and the tuning parameter* λ *is chosen such that* $n(\lambda J_\pi^*)^{\max(1, 2-\zeta)}$ *and* $n_l(\lambda J_{.5}^*)^{\max(1, 2-\zeta)}$ *are bounded away from 0.*

Note that there are two important cases defined by the value of B. When $B > 1$, ESPSI achieves the convergence rate $\delta_n^{2\alpha}$ of its supervised counterpart ψ-learning based on complete data, c.f., Theorem 1 of (Shen and Wang, 2007). When $B \leq 1$, ESPSI performs no worse than its initial classifier because $(\delta_n^{(0)})^{2\alpha\max(1,B^K)}(\delta_n^{(0)})^{2\alpha}$. Therefore, it is critical to compute the value of B. For instance, if the two classes are perfectly separated and located very densely within respective regions, then $B = \infty$ and our method recovers the rate $\delta_n^{2\alpha}$; if the two classes are completely indistinguishable, then $B = 0$ and our method yields the rate $(\delta_n^{(0)})^2$.

For the optimality claimed in Section 2.2, we show that $\hat{U}(f)$ is sufficiently close to $U(f)$ so that optimality of $U(f)$ can be translated into $\hat{U}(f)$. As a result, minimization of $\hat{U}(f)$ over f mimics that of $U(f)$.

Corollary 5 *(Optimality) Under the assumptions of Corollary 4, as $n_u, n_l \to \infty$,*

$$\sup_{f\in\mathcal{F}} \|\hat{U}(f) - U(f)\|_1 \quad = \quad O_p\Big(\max(\delta_n^{\beta\gamma}, (\rho_n\delta_n^{(0)})^{\beta\gamma\max(1,B^K)})\Big),$$

where $\hat{U}(f)$ is estimated $U(f)$ loss with p estimated based on $\hat{f}_C$.

To argue that the approximation error rate of $\hat{U}(f)$ to $U(f)$ is sufficiently small, note that $\hat{f}_C$ obtained from minimizing (2) recovers the classification error rate of its supervised counterpart based on complete data, as suggested by Corollary 4. Otherwise, a poor approximation precision could impede the error rate of ESPSI.

In conclusion, ESPSI, without knowing label values of unlabeled instances, enables to reconstruct the classification and estimation performance of ψ-learning based on complete data in rates of convergence, when possible.

5.2 Theoretical Example

We now apply Corollary 4 to linear and kernel learning examples to derive generalization errors rates for ESPSI in terms of the Bayesian regret. In all cases, ESPSI (nearly) achieves the generalization error rates of ψ-learning for complete data when unlabeled data provides useful information, and yields no worse performance then its initial classifier otherwise.

Consider a learning example in which $X = (X_{\cdot 1}, X_{\cdot 2})$ are independent, following marginal distribution $q(x) = \frac{1}{2}(\kappa_1 + 1)|x|^{\kappa_1}$ for $x \in [-1, 1]$ for $\kappa_1 > 0$. Given $X = 1$, $P(Y = 1|X = x) = p(x) = \frac{2}{5}\,\mathrm{sign}(x_{\cdot 1})|x_{\cdot 1}|^{\kappa_2} + \frac{1}{2}$ with $\kappa_2 > 0$. Note that $f_\pi(x)$ is $x_{\cdot 1} - \mathrm{sign}(\pi - \frac{1}{2})(\frac{5}{4}|2\pi - 1|)^{\frac{1}{\kappa_2}}$, which in turn yields the vertical line as the decision boundary for classification with unequal cost π. The value of κ_i; $i = 1, 2$ describe the behavior of the marginal distribution around the origin, and that of the conditional distribution $p(x)$ in the neighborhood of $1/2$, respectively.

For illustration, Figure 3 displays the marginal and conditional densities from the data distribution with $\kappa_1 = 2$ and $\kappa_2 = 1$. It is evident that the clustering assumption (**Assumption B**) is met since the neighborhood of $f_{.5}(x)$ has low density as showed in the left panel of Figure 3, and the smoothness assumption (**Assumption D**) and the boundedness assumption of $p(x)$ (**Assumption E**) are met as well since $p(x)$ is a hyperplane bounded by $(0.1, 0.9)$ as showed in the right panel of Figure 3. Technical details of verifying assumptions are deferred to Appendix B.

5.2.1 LINEAR LEARNING

Here it is natural to consider linear learning in which candidate decision functions are linear in $\mathcal{F} = \{f(x) = (1, x^T)w : w \in \mathcal{R}^3, x = (x_{\cdot 1}, x_{\cdot 2}) \in \mathcal{R}^2\}$.

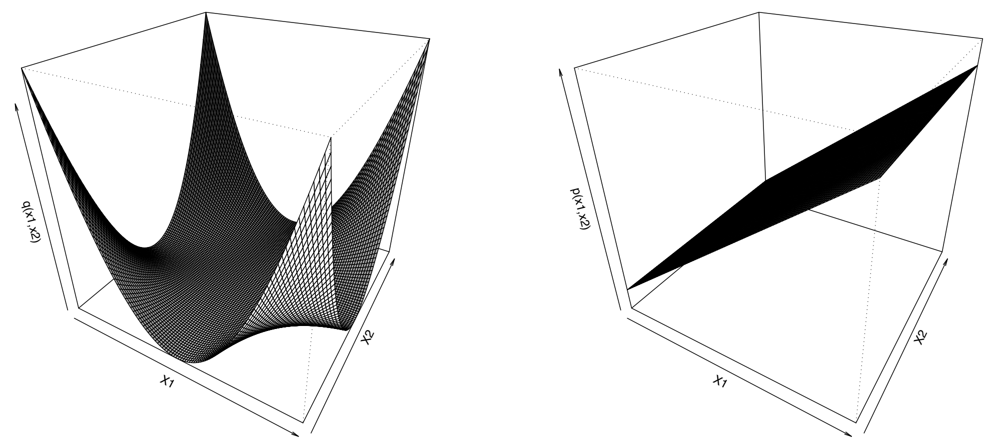


Figure 3: Plots of the marginal and conditional densities from the data distribution with $\kappa_1 = 2$ and $\kappa_2 = 1$.

For ESPSI $\hat{f}_C$, we choose $\delta_n^{(0)} = n_l^{-1}\log n_l$, the convergence rate of supervised linear ψ-learning, $\rho_n \to \infty$ to be an arbitrarily slow sequence and $C = O((\log n)^{-1})$. An application of Corollary 4 yields that $E|e(\hat{f}_C, \bar{f}_{.5})| = O(\max(n^{-1}\log n,\, (n_l^{-1}(\log n_l)^2)^{\max(1,2B^K)}))$, with $B = \frac{(1+\kappa_1)^2}{2\kappa_2(1+\kappa_1+\kappa_2)}$. When $B > 1$, equivalently, $\kappa_1 + 1 > (1+\sqrt{3})\kappa_2$, this rate reduces to $O(n^{-1}\log n)$ when K is sufficiently large. Otherwise, the rate is $O(n_l^{-1}\log n_l)$.

The fast rate $n^{-1}\log n$ is achieved when κ_1 is large but κ_2 is relatively small. Interestingly, large κ_1 value implies that $q(x)$ has a low density around $x = 0$, corresponding to the low density separation assumption in Chapelle and Zien (2005) for a semisupervised problem, whereas large κ_1 value and small κ_2 value indicate that $p(x)$ has a small probability to be close to the decision boundary $p(x) = 1/2$ for a supervised problem.

5.2.2 KERNEL LEARNING

Consider a flexible representation defined by a Gaussian kernel, where $\mathcal{F} = \{x \in \mathcal{R}^2 : f(x)w_{f,0} + \sum_{k=1}^{n} w_{f,k}K(x,x_k) : w_f = (w_{f,1},\cdots,w_{f,n})^T \in \mathcal{R}^n\}$ by the representation theorem of RKHS, see Wahba (1990). Here $K(x,z) = \exp(-\frac{\|x-z\|^2}{2\sigma^2})$ is the Gaussian kernel.

Similarly, we choose $\delta_n^{(0)} = n_l^{-1}(\log n_l)^3$ to be the convergence rate of supervised ψ-learning with Gaussian kernel, $\rho_n \to \infty$ to be an arbitrarily slow sequence and $C = O((\log n)^{-3})$. By Corollary 4, $E|e(\hat{f}_C, \bar{f}_{.5})| = O(\max((n_l^{-1}(\log n_l)^3)^{\max(1,2B^K)},\, n^{-1}(\log n)^3)) = O(n^{-1}(\log n)^3)$ when $\kappa_1 + 1 > 2\kappa_2(1+\kappa_2)$ and K is sufficiently large, and $O(n_l^{-1}(\log n_l)^3)$ otherwise. Again, large κ_1 and small κ_2 lead to the fast rate.

6. Summary

This article introduces a large margin semisupervised learning method through an iterative scheme based on an efficient loss for unlabeled data. In contrast to most methods assuming a relationship between the conditional and the marginal distributions, the proposed method integrates labeled and unlabeled data through using the clustering structure of unlabeled data as well as the smoothness structure of the estimated p. The theoretical and numerical results suggest that the method compares favorably against top competitors, and achieves the desired goal of reconstructing the classification performance of its supervised counterpart on complete labeled data.

With regard to tuning parameter C, further investigation is necessary. One critical issue is how to use unlabeled data to enhance the accuracy of estimating the generalization error so that adaptive tuning is possible.

Acknowledgments

This research is supported in part by NSF grants IIS-0328802 and DMS-0604394, by NIH grants HL65462 and 1R01GM081535-01, and a UM AHC FRD grant.

Appendix A. Technical Proofs

Proof of Lemma 1: Let $U(f(x)) = E(L(Yf(X))|X = x)$. By orthogonality, $E(L(Yf(X)) - T(f(X)))^2 = E(L(Yf(X)) - U(f(X)))^2 + E(U(f(X)) - T(f(X)))^2$, implying that $U(f(x))$ minimizes $E(L(Yf(X)) - T(f(X)))^2$ over any T. Then the proof follows from the fact that $EL(Yf(X)) = E(E(L(Yf(X))|X))$.

Proof of Theorem 2: For clarity, we write $s(\hat{f})$ as $s(\hat{f}, \hat{p})$ in this proof. Then it suffices to show that $s(\hat{f}^{(k)}, \hat{p}^{(k)}) \geq s(\hat{f}^{(k+1)}, \hat{p}^{(k+1)})$. First, $s(\hat{f}^{(k)}, \hat{p}^{(k)}) \geq s(\hat{f}^{(k+1)}, \hat{p}^{(k)})$ since $\hat{f}^{(k+1)}$ minimizes $s(f, \hat{p}^{(k)})$. Then $s(\hat{f}^{(k+1)}, \hat{p}^{(k)}) - s(\hat{f}^{(k+1)}, \hat{p}^{(k+1)}) = \sum_{j=n_l+1}^{n} (\hat{p}^{(k)} - \hat{p}^{(k+1)})(L(\hat{f}^{(k+1)}(x_j)) - L(-\hat{f}^{(k+1)}(x_j)))$, which is nonnegative by the definition of $\hat{p}^{(k+1)}$.

Proof of Theorem 3: The proof involves two steps. In **Step 1**, given $f^{(k)}$, we derive a probability upper bound for $\|\hat{p}^{(k)} - p\|_1$, where $\hat{p}^{(k)}$ is obtained from **Algorithm 0**. In **Step 2**, based on the result of **Step 1**, the difference between the tail probability of $e_\pi(\hat{f}_\pi^{(k+1)}, \bar{f}_\pi)$ and that of $e_\pi(\hat{f}_\pi^{(k)}, \bar{f}_\pi)$ is bounded through a large deviation inequality of Wang and Shen (2007); $k = 0, 1, \cdots$. This in turn results in a faster rate for $e(\hat{f}_{.5}^{(k+1)}, \bar{f}_{.5})$, thus $e(\hat{f}_C, \bar{f}_{.5})$. In this proof, we denote labeled and unlabeled samples by $\{(X_i, Y_i)\}_{i=1}^{n_l}$ and $\{X_j\}_{j=n_l+1}^{n}$ to indicate that they are all random variables.

Step 1: First we bound the probability of the percentage of wrongly labeled unlabeled instances by $\text{sign}(\hat{f}^{(k)})$ by the tail probability of $e_{V.5}(\hat{f}^{(k)}, \bar{f}_{.5})$. For this purpose, define $D_f = \{\text{sign}(\hat{f}^{(k)}(X_j)) \neq \text{sign}(\bar{f}_{.5}(X_j)); n_l + 1 \leq j \leq n\}$ to be the set of unlabeled data that are wrongly labeled by $\text{sign}(\hat{f}^{(k)})$, with $n_f = \#\{D_f\}$ being its cardinality. According to Markov's inequality, the fact that $E(\frac{n_f}{n}) = \frac{n_u}{n} E\|\text{sign}(\hat{f}^{(k+1)}) - \text{sign}(\bar{f}_{.5})\|_1$, and (8), we have

$$\begin{aligned} P\Big(\frac{n_f}{n} \geq a_1(a_{11}\rho_n^2(\delta_n^{(k)})^2)^\beta\Big) &\leq P\Big(\|\text{sign}(\hat{f}^{(k)}) - \text{sign}(\bar{f}_{.5})\|_1 \geq a_1(a_{11}\rho_n(\delta_n^{(k)})^2)^\beta\Big) \\ &\quad + P\Big(\frac{n_f}{n} \geq \rho_n^\beta \|\text{sign}(\hat{f}^{(k+1)}) - \text{sign}(\bar{f}_{.5})\|_1\Big) \\ &\leq P\Big(e_{V.5}(\hat{f}^{(k)}, \bar{f}_{.5}) \geq a_{11}\rho_n(\delta_n^{(k)})^2\Big) + \rho_n^{-\beta}. \end{aligned} \tag{11}$$

Next we bound the tail probability of $\|\hat{p}^{(k)} - p\|_1$ based on "complete" data consisting of unlabeled data assigned by $\mathrm{sign}(\hat{f}^{(k)})$. An application of similar treatment to that in the proof of Theorem 3 of Wang et al. (2008) leads to

$$\begin{aligned} P\Big(\|\hat{p}^{(k)} - p\|_1 \geq & 8^{\gamma} a_1^{2\gamma+1}(a_{11}\rho_n\delta_n^{(k)})^{\beta\gamma}\Big) \leq \\ & P(\exists j : \|\,\mathrm{sign}(\hat{f}^{(k)}_{\pi_j}) - \mathrm{sign}(\bar{f}_{\pi_j})\|_1 \geq 8^{\gamma} a_1^{2\gamma+1}(a_{11}\rho_n\delta_n^{(k)})^{2\beta\gamma}), \end{aligned} \tag{12}$$

with $\pi_j = j/\lceil (a_{11}\rho_n\delta_n^{(k)})^{-\beta\gamma}\rceil$. By (8), it suffices to bound $P(e_{V_\pi}(\hat{f}^{(k)}_{\pi_j}, \bar{f}_{\pi_j}) \geq 8a_1^2(a_{11}\rho_n\delta_n^{(k)})^{2\beta})$ for all π_j in what follows.

We introduce some notations to be used. Let $\tilde{V}_\pi(f,Z) = V_\pi(f,Z) + \lambda J(f)$, and $Z_j = (X_j, Y_j)$ with $Y_j = \mathrm{sign}(\hat{f}^{(k)}(X_j))$; $n_l + 1 \leq j \leq n$. Define a scaled empirical process $E_n(\tilde{V}_\pi(f^*_\pi, Z) - \tilde{V}_\pi(f,Z)) = n^{-1}\Big(\sum_{i\in D_f} + \sum_{i\notin D_f}\Big)\Big(\tilde{V}_\pi(f^*_\pi, Z_i) - \tilde{V}_\pi(f,Z_i) - E(\tilde{V}_\pi(f^*_\pi, Z_i) - \tilde{V}_\pi(f,Z_i))\Big) \equiv E_n(V_\pi(f^*_\pi,Z) - V_\pi(f,Z))$.

By the definition of $\hat{f}^{(k)}_\pi$ and (11),

$$\begin{aligned} P\Big(e_{V_\pi}(\hat{f}^{(k)}_\pi, \bar{f}_\pi) \geq \delta_k^2\Big) \quad &\leq \quad P\Big(\frac{n_f}{n} \geq a_1(a_{11}\rho_n^2(\delta_n^{(k)})^2)^{\beta}\Big) + \\ & \qquad P^*\Big(\sup_{N_k} \frac{1}{n}\sum_{i=1}^{n}(\tilde{V}_\pi(f^*_\pi, Z_i) - \tilde{V}_\pi(f,Z_i)) \geq 0, \frac{n_f}{n} \leq a_1(a_{11}\rho_n^2(\delta_n^{(k)})^2)^{\beta}\Big) \\ &\leq \quad P\Big(e_{V_{.5}}(\hat{f}^{(k)}, \bar{f}_{.5}) \geq a_{11}\rho_n(\delta_n^{(k)})^2\Big) + \rho_n^{-\beta} + I_1, \end{aligned} \tag{13}$$

where $N_k = \{f \in \mathcal{F} : e_{V_\pi}(f, \bar{f}_\pi) \geq \delta_k^2\}$, $\delta_k^2 = 8a_1^2(a_{11}\rho_n\delta_n^{(k)})^{2\beta}$, $I_1 = P^*\Big(\sup_{N_k} E_n(V_\pi(f^*_\pi, Z) - V_\pi(f,Z)) \geq \inf_{N_k}\nabla(f, f^*_\pi), \frac{n_f}{n} \leq a_1(a_{11}\rho_n^2(\delta_n^{(k)})^2)^{\beta}\Big)$, and $\nabla(f, f^*_\pi) = \frac{n_f}{n}E_{i\in D_f}(\tilde{V}_\pi(f,Z_i) - \tilde{V}_\pi(f^*_\pi, Z_i)) + \frac{n-n_f}{n}E_{i\notin D_f}(\tilde{V}_\pi(f,Z_i) - \tilde{V}_\pi(f^*_\pi, Z_i))$.

To bound I_1, we partition N_k into a union of $A_{s,t}$ with

$$\begin{aligned} A_{s,t} &= \{f \in \mathcal{F} : 2^{s-1}\delta_k^2 \leq e_{V_\pi}(f, \bar{f}_\pi) < 2^s\delta_k^2, 2^{t-1}J^*_\pi \leq J(f) < 2^t J^*_\pi\}; \\ A_{s,0} &= \{f \in \mathcal{F} : 2^{s-1}\delta_k^2 \leq e_{V_\pi}(f, \bar{f}_\pi) < 2^s\delta_k^2, J(f) < J^*_\pi\}, \end{aligned}$$

for $s, t = 1, 2, \cdots$. Then it suffices to bound the corresponding probability over each $A_{s,t}$. Toward this end, we need to bound the first and second moments of $\tilde{V}_\pi(f,Z) - \tilde{V}_\pi(f^*_\pi, Z)$ over $f \in A_{s,t}$. Without loss of generality, assume that $4s_n < \delta_k^2 < 1$, $J(f^*_\pi) \geq 1$, and thus $J^*_\pi = \max(J(f^*_\pi), 1) = J(f^*_\pi)$.

For the first moment, note that $\nabla(f, f^*_\pi) \geq e_{V_\pi}(f, f^*_\pi) - \frac{n_f}{n}E|V_\pi(f,Z) - V_\pi(f^*_\pi, Z) + \bar{V}_\pi(f,Z) - \bar{V}_\pi(f^*_\pi, Z)| \geq e_{V_\pi}(f, f^*_\pi) - 4\frac{n_f}{n}$ with $\bar{V}_\pi(f,z) = S_\pi(-y)L(-yf(x))$. Using the assumption that $4\lambda J(f^*_\pi) \leq \delta_k^2$, and Assumptions A and B, we obtain

$$\begin{aligned} \inf_{A_{s,t}} \nabla(f, f^*_\pi) \quad &\geq \quad M(s,t) = (2^{s-1} - 1/2)\delta_k^2 + \lambda(2^{t-1} - 1)J(f^*_\pi), \\ \inf_{A_{s,0}} \nabla(f, f^*_\pi) \quad &\geq \quad (2^{s-1} - 3/4)\delta_k^2 \geq M(s,0) = 2^{s-3}\delta_k^2. \end{aligned}$$

For the second moment, by Assumptions A and B and $|\bar{V}_\pi(f,Z)-\bar{V}_\pi(\bar{f}_\pi,Z)|\le 2$ for any $0<\pi<1$, we have, for any $s,t=1,2,\cdots$ and some constant $a_3>0$,

$$\begin{aligned}
&\sup_{A_{s,t}} \mathrm{Var}(V_\pi(f,Z)-V_\pi(f^*_\pi,Z))\\
\le\ & \sup_{A_{s,t}} \frac{2(n-n_f)}{n}\Big(\mathrm{Var}(V_\pi(f,Z)-V_\pi(\bar{f}_\pi,Z))+\mathrm{Var}(V_\pi(f^*_\pi,Z)-V_\pi(\bar{f}_\pi,Z))\Big)+\\
& \frac{2n_f}{n}\Big(\mathrm{Var}(\bar{V}_\pi(f,Z)-\bar{V}_\pi(\bar{f}_\pi,Z))+\mathrm{Var}(\bar{V}_\pi(f^*_\pi,Z)-\bar{V}_\pi(\bar{f}_\pi,Z))\Big)\\
\le\ & 2a_2M(s,t)^\zeta+8a_1(a_{11}\rho_n^2(\delta_n^{(k)})^2)^\beta+4s_n\le a_3M(s,t)^{\min(1,\zeta)}=v^2(s,t).
\end{aligned}$$

Note that $I_1\le I_2+I_3$ with

$$\begin{aligned}
I_2 &= \sum_{s,t=1}^{\infty} P^*(\sup_{A_{s,t}} E_n(V_\pi(f^*_\pi,Z)-V_\pi(f,Z))\ge M(s,t), n_f/n\le a_1(a_{11}\rho_n^2(\delta_n^{(k)})^2)^\beta);\\
I_3 &= \sum_{s=1}^{\infty} P^*(\sup_{A_{s,0}} E_n(V_\pi(f^*_\pi,Z)-V_\pi(f,Z))\ge M(s,0), n_f/n\le a_1(a_{11}\rho_n^2(\delta_n^{(k)})^2)^\beta).
\end{aligned}$$

Then we bound I_2 and I_3 separately using Lemma 1 of Wang et al. (2007). For I_2, we verify conditions (8)-(10) there. Note that $\int_{aM(s,t)}^{v(s,t)} H_B^{1/2}(w,\mathcal{F}(2^w))dw/M(s,t)$ is non-increasing in s and $M(s,t)$, we have

$$\int_{aM(s,t)}^{v(s,t)} H_B^{1/2}(w,\mathcal{F}(2^t))dw/M(s,t)\le\int_{a_3M(1,t)}^{aM(1,t)^{\min(1,\zeta)/2}} H_B^{1/2}(w,\mathcal{F}(2^t))dw/M(1,t),$$

which is bounded by $\phi(\varepsilon_n^2,2^t)$ with $a=2a_4\varepsilon$ and $\varepsilon_n^2\le\delta_k^2$. Then Assumption C implies (8)-(10) there with $\varepsilon=1/2$, the choices of $M(s,t)$ and $v(s,t)$ and some constants $a_i>0; i=3,4$. It then follows that for some constant $0<\xi<1$,

$$\begin{aligned}
I_2 &\le \sum_{s,t=1}^{\infty} 3\exp\left(-\frac{(1-\xi)n(M(s,t))^2}{2(4(v(s,t))^2+2M(s,t)/3)}\right)\\
&\le \sum_{s,t=1}^{\infty} 3\exp(-a_8n(M(s,t))^{\max(1,2-\zeta)})\\
&\le \sum_{s,t=1}^{\infty} 3\exp(-a_8n(2^{s-1}\delta_k^2+\lambda(2^{t-1}-1)J^*_\pi)^{\max(1,2-\zeta)})\\
&\le 3\exp(-a_8n(\lambda J^*_\pi)^{\max(1,2-\zeta)})/(1-\exp(-a_8n(\lambda J^*_\pi)^{\max(1,2-\zeta)}))^2.
\end{aligned}$$

Similarly $I_3\le 3\exp(-a_8n(\lambda J^*_\pi)^{\max(1,2-\zeta)})/(1-\exp(-a_8n(\lambda J^*_\pi)^{\max(1,2-\zeta)}))^2$. Combining the bounds for I_i; $i=2,3$, we have $I_1\le 3.5\exp(-a_8n(\lambda J^*_\pi)^{\max(1,2-\zeta)})$. Consequently, by (8), (12) and (13)

$$\begin{aligned}
&P\Big(\|\hat{p}^{(k)}-p\|_1\ge 8^\gamma a_1^{2\gamma+1}(a_{11}\rho_n\delta_n^{(k)})^{\beta\gamma}\Big)\le\\
&\quad P\Big(e_{V.5}(\hat{f}^{(k)}_{.5},\bar{f}_{.5})\ge a_{11}\rho_n(\delta_n^{(k)})^2\Big)+\rho_n^{-\beta}+3.5\exp(-a_8n(\lambda J^*_\pi)^{\max(1,2-\zeta)}).
\end{aligned}\tag{14}$$

Step 2: To begin, note that $P\Big(e_{V_{.5}}(\hat{f}^{(k+1)}_{.5},\bar{f}_{.5}) \geq a_{11}\rho_n(\delta_n^{(k+1)})^2\Big) \leq I_4+I_5$ with

$$\begin{aligned} I_4 &= P\Big(e_{V_{.5}}(\hat{f}^{(k+1)}_{.5},\bar{f}_{.5}) \geq a_{11}\rho_n(\delta_n^{(k+1)})^2 \|\hat{p}^{(k)}-p\|_1 < 8^\gamma a_1^{2\gamma+1}(a_{11}\rho_n\delta_n^{(k)})^{\beta\gamma}\Big), \\ I_5 &= P\Big(\|\hat{p}^{(k)}-p\|_1 \geq 8^\gamma a_1^{2\gamma+1}(a_{11}\rho_n\delta_n^{(k)})^{\beta\gamma}\Big), \end{aligned}$$

where $a_{11}\rho_n(\delta_n^{(k+1)})^2 = (a_{12}(a_{11}\rho_n\delta_n^{(k)})^{\frac{(d+\eta)\beta\gamma}{d+\eta+1}})^{\frac{\theta+1}{1+\max(0,1-\beta)\theta}}$ and $a_{12} = 2a_6^{1/(d+\eta+1)}(4a_7)^{-\frac{1}{\theta}}$. By (14), it suffices to bound I_4.

For I_4, we need some notations. Let the ideal cost function be $V_{.5}(f,z)+U_{.5}(f(x))$, the ideal version of (2), where $V_{.5}(f,z)=\frac{1}{2}L(yf(x))$, and $U_{.5}(f(x))=\frac{1}{2}(p(x)L(f(x))+(1-p(x))L(-f(x)))$ is the ideal loss for unlabeled data. Denote by $\hat{U}^{(k)}_{.5}(f(x))=\frac{1}{2}(\hat{p}^{(k)}(x)L(f(x))+(1-\hat{p}^{(k)}(x))L(-f(x)))$ an estimate of $U_{.5}(f(x))$ at **Step** k. So the cost function in (2) can be written as $\tilde{W}(f,z)=W(f,z)+\lambda J(f)$ with $W(f,z)=V_{.5}(f,z)+U_{.5}(f(x))$. For simplicity, we denote a weighted empirical process by $E_n(W(f^*_{.5},z)-W(f,z)) = n_l^{-1}\sum_{i=1}^{n_l}\big(V_{.5}(f^*_{.5},Z_i)-V_{.5}(f,Z_i)\ -E(V_{.5}(f^*_{.5},Z)-V_{.5}(f,Z))\big) + n_u^{-1}\sum_{j=n_l+1}^{n}\big(U_{.5}(f^*_{.5}(X_j))-U_{.5}(f(X_j))-E(U_{.5}(f^*_{.5}(X))-U_{.5}(f(X)))\big)$.

By the definition of $\hat{f}^{(k+1)}_{.5}$, we have

$$\begin{aligned} I_4 \leq P\Big(&\sup_{N'_k} n_l^{-1}\sum_{i=1}^{n_l}(V_{.5}(f^*_{.5},Z_i)-V_{.5}(f,Z_i)) + n_u^{-1}\sum_{j=n_l+1}^{n}(\hat{U}^{(k)}_{.5}(f^*_{.5}(X_j)) - \\ &\hat{U}^{(k)}_{.5}(f(X_j))) + \lambda(J(f^*_{.5})-J(f)) \geq 0,\ \|\hat{p}^{(k)}-p\|_1 < 8^\gamma a_1^{2\gamma+1}(a_{11}\rho_n\delta_n^{(k)})^{\beta\gamma}\Big), \end{aligned}$$

where $N'_k=\{f\in\mathcal{F}: e_{V_{.5}}(f,\bar{f}_{.5}) \geq a_{11}\rho_n(\delta_n^{(k+1)})^2\}$. Then $I_4 \leq I_6+I_7$ with

$$\begin{aligned} I_6 &= P\Big(\sup_{N'_k} n_u^{-1}\sum_{j=n_l+1}^{n} D(f,X_j) \geq \\ &\qquad 8^{\frac{\gamma(d+\eta)}{d+\eta+1}} a_1^{\frac{(2\gamma+1)(d+\eta)}{d+\eta+1}} \rho_n(e_{V_{.5}}(f,f^*_{.5}))^{\frac{\min(1,\beta)\theta}{\theta+1}}(a_{11}\rho_n(\delta_n^{(k+1)})^2)^{\frac{1+\max(0,1-\beta)\theta}{\theta+1}}\Big), \\ I_7 &= P\Big(\sup_{N'_k} E_n(W(f^*_{.5},Z)-W(f,Z)) \geq \inf_{N'_k} E(\tilde{W}(f,Z)-\tilde{W}(f^*_{.5},Z)) \\ &\qquad - 8^{\frac{\gamma(d+\eta)}{d+\eta+1}} a_1^{\frac{(2\gamma+1)(d+\eta)}{d+\eta+1}} \rho_n(e_{V_{.5}}(f,f^*_{.5}))^{\frac{\min(1,\beta)\theta}{\theta+1}}(a_{11}\rho_n(\delta_n^{(k+1)})^2)^{\frac{1+\max(0,1-\beta)\theta}{\theta+1}}\Big), \end{aligned}$$

where $D(f,X_j)=\hat{U}^{(k)}_{.5}(f^*_{.5}(X_j))-\hat{U}^{(k)}_{.5}(f(X_j))-U_{.5}(f^*_{.5}(X_j))+U_{.5}(f(X_j))$.

For I_6, we note that

$$\begin{aligned} E|D(f,X)| &= \frac{1}{2}E|\hat{p}^{(k)}(X)-p(X)|\big|L(f^*_{.5}(X))-L(f(X))-L(-f^*_{.5}(X))+L(-f(X))\big| \\ &\leq \frac{1}{2}\|\hat{p}^{(k)}-p\|_\infty E\big(|L(f^*_{.5}(X))-L(f(X))|+|L(-f^*_{.5}(X))-L(-f(X))|\big). \end{aligned}$$

It thus suffices to bound $\|\hat{p}^{(k)}-p\|_\infty$ and $E\big(|L(f^*_{.5}(X))-L(f(X))|+|L(-f^*_{.5}(X))-L(-f(X))|\big)$ separately.

To bound $\|\hat{p}^{(k)} - p\|_\infty$, note that $EJ(\hat{f}^{(k)}_{\pi_j})$ is bounded for all π_j following the same argument as in Lemma 5 of Wang and Shen (2007). By Sobolev's interpolation theorem (Adams, 1975), $\|\hat{p}^{(k)}\|_\infty \leq 1$ and the fact that $|\hat{p}^{(k,d)}(x_1) - \hat{p}^{(k,d)}(x_2)| \leq \sup_j |\hat{f}^{(k,d)}_{\pi_j}(x_1) - \hat{f}^{(k,d)}_{\pi_j}(x_2)| + d(m^{(k)})^{-1}$ for any x_1 and x_2 based on the construction of $\hat{p}^{(k)}$ with $\hat{f}^{(k,d)}_{\pi_j} = \Delta^d(\hat{f}^{(k)}_{\pi_j})$, there exists a constant a_{13} such that $\|\hat{p}^{(k,d)}\|_\infty \leq a_{13}$ and $|\hat{p}^{(k,d)}(x_1) - \hat{p}^{(k,d)}(x_2)| \leq a_{13}|x_1 - x_2|^\eta + d(m^{(k)})^{-1}$ when $|x_1 - x_2|$ is sufficiently small. Without loss of generality, we assume $a_{13} \leq a_6$. By Assumption D and $m^{(k)} = \lceil (a_{11}\rho_n\delta_n^{(k)})^{-\beta\gamma} \rceil$, we have

$$\begin{aligned}
&\big| |\hat{p}^{(k,d)}(x_1) - p^{(d)}(x_1)| - |\hat{p}^{(k,d)}(x_2) - p^{(d)}(x_2)| \big| \\
\leq\ & |\hat{p}^{(k,d)}(x_1) - \hat{p}^{(k,d)}(x_2)| + |p^{(d)}(x_1) - p^{(d)}(x_2)| \leq 2a_6|x_1 - x_2|^\eta + 2d(a_{11}\rho_n\delta_n^{(k)})^{\frac{\beta\gamma(d+\eta)}{d+\eta+1}}.
\end{aligned}$$

It then follows from Proposition 6 of Shen (1997) that $\|\hat{p}^{(k)} - p\|_\infty \leq 2a_6^{\frac{1}{d+\eta+1}} \|\hat{p}^{(k)} - p\|_1^{\frac{d+\eta}{d+\eta+1}} + 2d(a_{11}\rho_n\delta_n^{(k)})^{\frac{\beta\gamma(d+\eta)}{d+\eta+1}} \leq 2a_6^{\frac{1}{d+\eta+1}} (8^\gamma a_1^{2\gamma+1})^{\frac{d+\eta}{d+\eta+1}} (a_{11}\rho_n\delta_n^{(k)})^{\frac{\beta\gamma(d+\eta)}{d+\eta+1}}$.

To bound $E\big(|L(f^*_{.5}(X)) - L(f(X))| + |L(-f^*_{.5}(X)) - L(-f(X))|\big)$, we note that

$$\begin{aligned}
& E|V_{.5}(f,Z) - V_{.5}(f^*_{.5},Z)| \\
=\ & \frac{1}{2}E\big(p(X)|L(f^*_{.5}) - L(f)| + (1-p(X))|L(-f) - L(-f^*_{.5})|\big) \\
\geq\ & E\frac{\delta}{2}\big(|L(f^*_{.5}) - L(f)| + |L(-f) - L(-f^*_{.5})|\big) I(\min(p(X), 1-p(X)) \geq \delta) \\
\geq\ & \delta\big(E\frac{1}{2}\big(|L(f^*_{.5}) - L(f)| + |L(-f) - L(-f^*_{.5})|\big) - 2a_7\delta^\theta\big)
\end{aligned}$$

by Assumption E. With $\delta = \big(E(|L(f^*_{.5}) - L(f)| + |L(-f) - L(-f^*_{.5})|)/8a_7\big)^{1/\theta}$, it yields that $E\big(|L(f^*_{.5}) - L(f)| + |L(-f) - L(-f^*_{.5})|\big) \leq (4a_7)^{-1/\theta}\big(E|V_{.5}(f,Z) - V_{.5}(f^*_{.5},Z)|\big)^{\theta/\theta+1}$, where

$$\begin{aligned}
& E|V_{.5}(f,Z) - V_{.5}(f^*_{.5},Z)| \leq E|V_{.5}(f,Z) - V_{.5}(\bar{f}_{.5},Z)| + E|V_{.5}(f^*_{.5},Z) - V_{.5}(\bar{f}_{.5},Z)| \\
\leq\ & P(\text{sign}(f) \neq \text{sign}(\bar{f}_{.5})) + E(V_{.5}(f,Z) - V_{.5}(\bar{f}_{.5},Z)) + \\
& \qquad P(\text{sign}(f^*_{.5}) \neq \text{sign}(\bar{f}_{.5})) + E(V_{.5}(f^*_{.5},Z) - V_{.5}(\bar{f}_{.5},Z)) \\
\leq\ & (e_{V_{.5}}(f,f^*_{.5}))^\beta + e_{V_{.5}}(f,f^*_{.5}) + s_n^\beta + s_n \leq 4(e_{V_{.5}}(f,f^*_{.5}))^{\min(1,\beta)}
\end{aligned}$$

by Assumptions A and B. Therefore,

$$E|D(f,X)| \leq 8^{\frac{\beta(d+\eta)}{d+\eta+1}} a_1^{\frac{(2\beta+1)(d+\eta)}{d+\eta+1}} (e_{V_{.5}}(f,f^*_{.5}))^{\frac{\min(1,\beta)\theta}{\theta+1}} (a_{11}\rho_n(\delta_n^{(k+1)})^2)^{\frac{1+\max(0,1-\beta)\theta}{\theta+1}}$$

and $I_6 \leq \rho_n^{-1}$ by Markov's inequality.

To bound I_7, we apply a similar treatment as in bounding I_1 in **Step 1** to yield that $I_7 \leq 3.5\exp(-a_8 n_l(\lambda J^*_{.5})^{\max(1,2-\zeta)})$. Combining the upper bounds of I_6 and I_7, $P(e_{V_{.5}}(\hat{f}^{(k+1)}_{.5}, \bar{f}_{.5}) \geq a_{11}\rho_n(\delta_n^{(k+1)})^2) \leq P(e_{V_{.5}}(\hat{f}^{(k)}_{.5}, \bar{f}_{.5}) \geq a_{11}\rho_n(\delta_n^{(k)})^2) + 3.5\exp(-a_9 n(\lambda J^*_{.5})^{\max(1,2-\zeta)}) + 3.5\exp(-a_8 n_l(\lambda J^*_{.5})^{\max(1,2-\zeta)}) + \rho_n^{-\beta} + \rho_n^{-1}$. Iterating this inequality yields that

$$\begin{aligned}
& P\Big(e_{V_{.5}}(\hat{f}^{(K)}_{.5}, \bar{f}_{.5}) \geq (a_{12}^{\frac{2B(d+\eta+1)}{\beta\gamma(d+\eta)}} (a_{11}\rho_n)^{2B-1})^{\frac{B^{K+1}-1}{B-1}} (\delta_n^{(0)})^{2B^K}\Big) \\
& \leq P\Big(e_{V_{.5}}(\hat{f}^{(0)}_{.5}, \bar{f}_{.5}) \geq a_{11}\rho_n(\delta_n^{(0)})^2\Big) + 3.5K\exp(-a_8 n_l(\lambda J^*_{.5})^{\max(1,2-\zeta)}) + \qquad (15) \\
& \qquad 3.5K\exp(-a_9 n(\lambda J^*_\pi)^{\max(1,2-\zeta)}) + K\rho_n^{-\beta} + K\rho_n^{-1},
\end{aligned}$$

where $B = \frac{(\theta+1)(d+\eta)\beta\gamma}{2(1+\max(0,1-\beta)\theta)(d+\eta+1)}$. Then the desired result follows from Assumption B and the fact that $\delta_k^2 = 8a_1^2(a_{11}\rho_n\delta_n^{(k)})^{2\beta} \geq \max(\varepsilon_n^2, 16s_n) = \delta_n^2$ for any k.

Proof of Corollary 4: It follows from Theorem 3 immediately and the proof is omitted.

Proof of Corollary 5: It follows from (14) and Corollary 4 that $\|\hat{p}_C - p\|_1 = O_p\Big(\max(\delta_n^{\beta\gamma}, (\rho_n\delta_n^{(0)})^{\max(1,1\beta\gamma B^K)})\Big)$, where $\hat{p}_C$ is the estimated probability through $\hat{f}_C$. The desired results follows from the fact that $\|\hat{U}_C(f) - U(f)\|_1 \leq 4\|\hat{p}_C - p\|_1$.

Appendix B. Verification of Assumptions in Section 5.2

We now verify Assumptions A-E for the theoretical examples in Section 5.2.

B.1 Linear Learning

First, note that $(X_{\cdot 1}, Y)$ is independent of $X_{\cdot 2}$, which implies that $ES(f;C) = E(E(S(f;C)|X_{\cdot 2})) \geq ES(\tilde{f}_C^*;C)$ for any $f \in \mathcal{F}$, where $\tilde{f}_C^* = \arg\min_{\tilde{f}\in\mathcal{F}_1} ES(\tilde{f};C)$ with $\mathcal{F}_1 = \{x_{\cdot 1} \in \mathcal{R} : \tilde{f}(x) = (1, x_{\cdot 1})^T w : w \in \mathcal{R}^2\} \subset \mathcal{F}$ and $S(f;C) = C(L(Yf(X)) + U(f(X))) + J(f)$.

Assumption A follows from $e_{V_\pi}(f_\pi^*, \bar{f}_\pi) \leq 2P(|nf_\pi(X)| \leq 1) \leq (\kappa_1 + 1)n^{-1} = s_n$ with $f_\pi^* = nf_\pi$. Easily, (7) in Assumption B holds for $\alpha = 1$. To verify (8), direct calculation yields that there exist some constants $b_1 > 0$ and $b_2 > 0$ such that for any $f \in \mathcal{F}_1$, we have $e_{V_\pi}(f, \bar{f}_\pi) \geq e_\pi(f, \bar{f}_\pi) = b_1\big((\frac{5}{4}(2\pi - 1) + e)^{\kappa_1+\kappa_2+1} - (\frac{5}{4}(2\pi - 1))^{\kappa_1+\kappa_2+1}\big)$ and $E|\text{sign}(f_\pi) - \text{sign}(f)| = b_2\big((\frac{5}{4}(2\pi - 1) + e)^{\kappa_1+1} - (\frac{5}{4}(2\pi - 1))^{\kappa_1+1}\big)$ with $w_f = w_{f_\pi} + (e_0, e_1)^T$ and $e = -\frac{50e_1(\frac{5}{4}(2\pi-1))+10e_0}{4+10e_1} > 0$. This implies (8) with $\beta = \gamma = \frac{1+\kappa_1}{1+\kappa_1+\kappa_2}$. For (9) in Assumption B, by the triangle inequality, $\text{Var}(V_\pi(f,Z) - V_\pi(f_\pi^*,Z)) \leq 2E|V_\pi(f,Z) - V_\pi(\bar{f}_\pi,Z)| \leq 2(\Lambda_1 + \Lambda_2)$, where $\Lambda_1 = E|l_\pi(f,Z) - V_\pi(\bar{f}_\pi,Z)|E|S_\pi(Y)|$ $|\text{sign}(f) - \text{sign}(\bar{f}_\pi)| \leq \big(2^{1+\kappa_2}(\kappa_1+1)^{\kappa_2}\big)^{\frac{1+\kappa_1}{1+\kappa_1+\kappa_2}} e_{V_\pi}(f, \bar{f}_\pi)^{\frac{1+\kappa_1}{1+\kappa_1+\kappa_2}}$, and $\Lambda_2 = E(V_\pi(f,Z) - l_\pi(f,Z)) = E(V_\pi(f,Z) - V_\pi(\bar{f}_\pi,Z)) + E(l_\pi(\bar{f}_\pi, Z) - l_\pi(f,Z)) \leq 2e_{V_\pi}(f, \bar{f}_\pi)$. Therefore (9) is met with $\zeta = \frac{1+\kappa_1}{1+\kappa_1+\kappa_2}$. For Assumption C, we define $\phi_1(\varepsilon, k) = a_3(\log(1/M^{1/2}))^{1/2}/M^{1/2}$ with $M = M(\varepsilon, \lambda, k)$. By Lemma 6 of Wang and Shen (2007), solving (10) yields $\varepsilon_n = (\frac{\log n}{n})^{1/2}$ when $C/J_\pi^* \sim \delta_n^{-2}n^{-1} \sim (\log n)^{-1}$. Assumption D is satisfied with $d = \infty$ and $\eta = 0$, and Assumption E is met with $\theta = \infty$ by noting that $\min(p(x), 1 - p(x)) \geq 1/10$. In this case, $B = \frac{(1+\kappa_1)^2}{2\kappa_2(1+\kappa_1+\kappa_2)}$, and the desired result follows from Corollary 4.

B.2 Kernel Learning

Similar to the linear case, we restrict our attention to $\mathcal{F}_1 = \{x_{\cdot 1} \in \mathcal{R} : f(x_{\cdot 1}) = w_{f,0} + \sum_{k=1}^n w_{f,k}K(x_{\cdot 1}, x_{k1}) : w_f = (w_{f,1}, \cdots, w_{f,n})^T \in \mathcal{R}^n\}$.

Note that $\mathcal{F}_1$ is rich for sufficiently large n in that for function f_π^* as defined in the linear example, there exists a $\tilde{f}_\pi^* \in \mathcal{F}_1$ such that $\|\tilde{f}_\pi^* - f_\pi^*\|_\infty \leq s_n$, and hence $e_{V_\pi}(\tilde{f}_\pi^*, \bar{f}_\pi) \leq 2s_n$. Assumption A is then met. Easily, (7) is satisfied for $\alpha = 1$. To verify (8), note that there exists constant $b_3 > 0$ such that for small $\delta > 0$, $P(|p(x) - 1/2| \geq \delta) = 2P(0 \leq p(x) - 1/2 \leq \delta) = 2P(0 \leq x_{(1)} \leq \frac{5}{2}\delta^{\frac{1}{\kappa_2}}) \leq b_3\delta^{\frac{1+\kappa_1}{\kappa_2}}$. Therefore, $e_{V_{.5}}(f, \bar{f}_{.5}) \geq e_{.5}(f, \bar{f}_{.5}) \geq \delta E|\text{sign}(f) - \text{sign}(\bar{f}_{.5})|I(|p(x)| \geq \delta) \geq 2^{-1}(4b_3)^{-\frac{\kappa_2}{1+\kappa_1}}\|\text{sign}(f) - \text{sign}(\bar{f}_{.5})\|_1^{\frac{1+\kappa_1+\kappa_2}{1+\kappa_1}}$ with $\delta = (\|\text{sign}(f) - \text{sign}(\bar{f}_{.5})\|_1/4b_3)^{\frac{\kappa_2}{1+\kappa_1}}$. This implies $\beta = \frac{1+\kappa_1}{1+\kappa_1+\kappa_2}$ in (8). Similarly, we can verify that there exists a constant $b_4 > 0$ such that

$P(|p(x)-\pi| \geq \delta) = 2P\left(\frac{5}{2}(\pi - \frac{1}{2}) \leq x_{(1)} \leq \frac{5}{2}(\pi - \frac{1}{2} + \delta^{\frac{1}{\kappa_2}})\right) \leq b_4 \delta^{\frac{1}{\kappa_2}}$ when $\pi > \frac{1}{2}$, which implies (8) with $\gamma = \frac{1}{1+\kappa_2}$. For (9), an application of the similar argument leads to $\zeta = \frac{1}{1+\kappa_2}$. For Assumption C, we define $\phi_1(\varepsilon, k) = a_3(\log(1/M^{1/2}))^{3/2}/M^{1/2}$ with $M = M(\varepsilon, \lambda, k)$. By Lemma 7 of Wang and Shen (2007), solving (10) yields $\varepsilon_n = ((\log n)^3 n^{-1})^{1/2}$ when $C/J^*_\pi \sim \delta_n^{-2} n^{-1} \sim (\log n)^{-3}$. Assumption D is satisfied with $d = \infty$ and $\eta = 0$, and Assumption E is met with $\theta = \infty$. Finally, $B = \frac{(1+\kappa_1)}{2\kappa_2(1+\kappa_2)}$, and the desired result follows from Corollary 4.

References

S. Abney. Understanding the Yarowsky algorithm. *Computat. Linguistics*, 30: 365-395, 2004.

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. *J. Mach. Learn. Res.*, 6:1817–1853, 2005.

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.

C. L. Blake and C. J. Merz. UCI repository of machine learning databases. 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 *Proc. 11th Ann. Conf. on Computat. Learn. Theory*, 1998.

O. Chapelle, B. Schölkopf, and A. Zien. *Semi-supervised Learning*. MIT press, Cambridge, 2006.

O. Chapelle and A. Zien. Semi-supervised classification by low density separation. In *Proc. Int. Workshop on Artif. Intell. and Statist.*, pages 57-64, 2005.

M. Collins and Y. Singer. Unsupervised models for named entity classification. In *Proc. Joint SIGDAT Conf. on Empirical Methods in Natural Language Processing and Very Large Corpora*, pages 100-110, 1999.

R. A. Fisher. A system of scoring linkage data, with special reference to the pied factors in mice. *Amer. Nat.*, 80:568-578, 1946.

C. Gu. Multidimension smoothing with splines. In, M. G. Shimek, (ed.), *Smoothing and Regression: Approaches, Computation and Application*, 2000.

T. Hughes, M. Marton, A. Jones, C. Roberts, R. Stoughton, C. Armour, H. Bennett, E. Coffey, H. Dai, Y. He, M. Kidd, A. King, M. Meyer, D. Slade, P. Lum, S. Stepaniants, D. Shoemaker, D. Gachotte, K. Chakraburtty, J. Simon, M. Bard and S. Friend. Functional discovery via a compendium of expression profiles. *Cell*, 102:109-126, 2000.

D. Hunter and K. Lange. Quantile regression via an MM algorithm. *J. Computat. & Graph. Statist.*, 9:60-77, 2000.

T. Jaakkola, M. Diekhans and D. Haussler. Using the Fisher kernel method to detect remote protein homologies. In *Proc. Int. Conf. on Intelligent Systems for Molecular Biology*, pages 149-158, 1999.

A. N. Kolmogorov and V. M. Tihomirov. ε-entropy and ε-capacity of sets in function spaces. *Uspekhi Mat. Nauk.*, 14:3-86, 1959. [In Russian. English translation, *Ameri. Math. Soc. Transl.*, 14:277-364, 1961.

Y. Lin. Support vector machines and the Bayes rule in classification. *Data Mining and Knowledge Discovery*, 6:259-275, 2002.

S. Liu, X. Shen and W. Wong. Computational development of ψ-learning. In *Proc. SIAM 2005 Int. Data Mining Conf.*, 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*, 12:512-518. MIT Press, Cambridge, 2000.

P. McCullagh and J. Nelder. *Generalized Linear Models*, 2nd edition. Chapman and Hall/CRC, 1983.

H. Mewes, K. Albermann, K. Heumann, S. Liebl and F. Pfeiffer. MIPS: a database for protein sequences, homology data and yeast genome information. *Nucleic Acids Res.*, 25:28-30, 2002.

K. Nigam, A. McCallum, S. Thrun and T. Mitchell . Text classification from labeled and unlabeled documents using EM. *Mach. Learn.*, 39:103–134, 1998.

J. Platt. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In *Advances in Large Margin Classifiers*, pages 61-74, MIT press, Cambridge, 1999.

P. Rigollet. Generalization Error Bounds in Semi-supervised Classification Under the Cluster Assumption. *J. Mach. Learn. Res.*, 8:1369-1392, 2007.

B. Schölkopf, A. Smola, R. Williamson and P. Bartlett. New support vector algorithms. *Neural Computation*, 12:1207-1245, 2000.

X. Shen. On method of sieves and penalization. *Ann. Statist.*, 25:2555-2591, 1997.

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. Generalization error for multi-class margin classification. *Electronic J. of Statist.*, 1:307-330, 2007.

X. Shen and W. Wong. Convergence rate of sieve estimates. *Ann. Statist.*, 22:580-615, 1994.

A. Singh, R. Nowak and X. Zhu. Unlabeled data: Now it helps, now it doesn't. In *NIPS*, 2008.

A. Tsybakov. Optimal aggregation of classifiers in statistical learning. *Ann. Statist.*, 32:135-166, 2004.

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.

G. Wahba. Support vector machines, reproducing kernel Hilbert spaces and the randomized GACV. In *Advances in Kernel Methods: Support Vector Learning*, edited by B. Schoelkopf, C. Burges and A. Smola, MIT Press, Cambridge, 1998.

J. Wang and X. Shen. Large margin semi-supervised learning. *J. Mach. Learn. Res.*, 8:1867-1891, 2007.

J. Wang, X. Shen and Y. Liu. Probability estimation for large margin classifiers. *Biometrika*, 95:149-167, 2008.

J. Wang, X. Shen and W. Pan. On transductive support vector machine. *Contemp. Math.*, 43:7-19, 2007.

G. Xiao and W. Pan. Gene function prediction by a combined analysis of gene expression data and protein-protein interaction data. *J. Bioinformatics and Computat. Biol.*, 3:1371-1390, 2005.

D. Yarowsky. Unsupervised word sense disambiguation rivaling supervised methods. In *Proc. of the 33rd Annual Meeting of the Association for Computational Linguistics*, pages 189-196, 1995.

T. Zhang and F. Oles. A probability analysis on the value of unlabeled data for classification problems. In *ICML*, 2000.

X. Zhou, M. Kao and W. Wong. Transitive functional annotation by shortest-path analysis of gene expression data. *Proc. Nat. Acad. Sci.*, 99:12783-12788, 2000.

J. Zhu and T. Hastie. Kernel logistic regression and the import vector machine. *J. Computat. 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. Semi-supervised learning literature survey. Technical Report 1530, University of Wisconsin, Madison, 2005.

Journal of Machine Learning Research 10 (2009) 743-746 Submitted 11/07; Revised 7/08; Published 3/09

Nieme: Large-Scale Energy-Based Models

Francis Maes FRANCIS.MAES@LIP6.FR
Universite Pierre et Marie Curie
Laboratoire d'Informatique de Paris 6, UMR CNRS 7606
104, Avenue du President Kennedy
Paris, France

Editor: Cheng Soon Ong

Abstract

In this paper we introduce NIEME,[1] a machine learning library for large-scale classification, regression and ranking. NIEME relies on the framework of *energy-based models* (LeCun et al., 2006) which unifies several learning algorithms ranging from simple perceptrons to recent models such as the pegasos support vector machine or l1-regularized maximum entropy models. This framework also unifies batch and stochastic learning which are both seen as energy minimization problems. NIEME can hence be used in a wide range of situations, but is particularly interesting for large-scale learning tasks where both the examples and the features are processed incrementally. Being able to deal with new incoming features at any time within the learning process is another original feature of the NIEME toolbox. NIEME is released under the GPL license. It is efficiently implemented in C++, it works on Linux, Mac OS X and Windows and provides interfaces for C++, Java and Python.

Keywords: large-scale machine learning, classification, ranking, regression, energy-based models, machine learning software

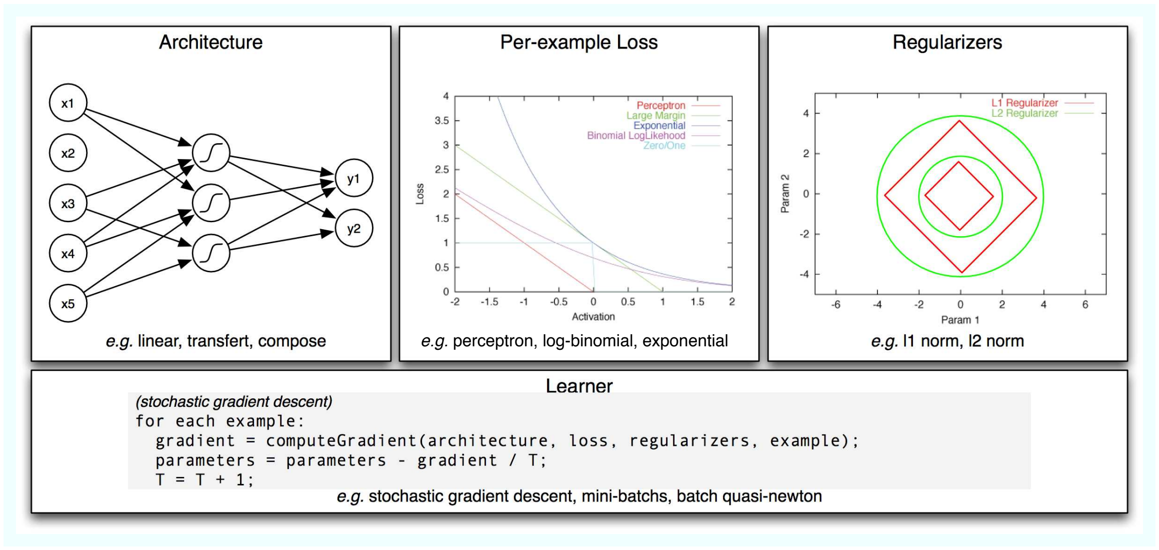
1. Introduction

Although many machine learning toolkits have been developed in the past years, it is often the case that they do not scale well to real-world applications. Developing learning algorithms for large-scale applications is a challenging design and implementation problem. In this paper, we introduce NIEME, a C++ library that provides generic tools for large-scale learning. NIEME covers two domains: supervised learning and learning in decision processes. In this paper, we focus on the supervised learning part, made of classification, regression and ranking algorithms. The support of decision processes is still in active development and will be more detailed in further versions of the library.

2. Framework: Energy-based Models

Most learning machines in NIEME rely on the unified framework of energy-based models introduced by LeCun et al. (2006). In this framework, illustrated in Figure 1, a learning machine can be interpreted as a combination of an architecture $\mathcal{A}$ with parameters θ, a per-example loss L , a set of regularizers $\{\Omega_1, \ldots, \Omega_R\}$ and a learner $\mathcal{L}$. Given the architecture, the per-example loss and the

1. Code for NIEME can be found at `http://nieme.lip6.fr`.



```
(stochastic gradient descent)
for each example:
  gradient = computeGradient(architecture, loss, regularizers, example);
  parameters = parameters - gradient / T;
  T = T + 1;
```

Figure 1: This figure illustrates the framework of energy-based models. Top: the three components that define the learning energy. Bottom: the *learner* component that performs energy minimization.

Model	Architecture	Loss	Regularizers	Learner
Perceptron	linear	perceptron	none	stochastic descent
Logistic regression	linear	log-binomial	none	batch quasi-newton
Pegasos linear SVM	linear	hinge loss	l2	pegasos learner
Multilayer perceptron	linear ∘ transfer ∘ linear	perceptron	none	stochastic descent
L1-maxent classifier	multi-class linear	log-binomial	l1	batch quasi-newton
Pegasos multi-class SVM	multi-class linear	hinge loss	l2	pegasos learner
Least-square regression	linear	squared loss	none	batch quasi-newton
Custom	linear ∘ transfer	absolute loss	l1 + l2	batch rprop
Many others	...	...	...	...

Table 1: This table gives some examples of energy-based models. Each model is defined by its architecture, per-example loss, regularizers and learner. The ∘ symbols denotes architecture composition.

regularizers, we can define a learning loss for a set of examples $S = \{e_1, \dots e_N\}$:

$$\mathcal{L}_S(\theta) = \frac{1}{N}\sum_{i=1}^{N} L(e_i, \mathcal{A}_\theta) + \sum_{i=1}^{R} \Omega_i(\theta).$$

The aim of the learner is then to find parameters θ that minimize the learning loss given the training set S. As illustrated in Table 1, many combinations are possible in the energy-based framework. Some correspond to well-known learning machines, others to more original approaches. We describe the three components that define the learning loss $\mathcal{L}_S(\theta)$ in Section 2.1, describe learners in Section 2.2 and discuss some aspects of features in NIEME in Section 2.3.

2.1 Learning Loss components

The architecture is a parameterized function which computes predictions given input vectors. A simple example is the linear architecture, which computes a single output as a scalar product between the input vector and the parameter vector. NIEME supports elementary architectures (linear, multi-class linear, neural network transfer function) as well as a *composition* operation which allows users to create a new architecture by chaining existing ones. The per-example loss quantifies *how bad* an architecture and its parameters perform on a given learning example. Learning aims at finding parameters which lead to low expected per-example loss. Currently NIEME implements four different loss functions for discriminant learning (perceptron loss, hinge loss, log-binomial loss and exponential loss) and two loss functions for regression (squared and absolute loss). Regularizers are functions which measure the *complexity* of an architecture and its parameters. It has been shown (Vapnik, 1999) that penalizing complex models often leads to better generalization performance. Up to now, NIEME includes the two most commonly used regularizers: the l1-norm and l2-norm of the parameters.

2.2 Learners

The learner is the algorithmic component that performs learning loss minimization. NIEME implements three batch learners: the large-scale limited-memory quasi-Newton method of Lio and Nocedal (1989), the recently proposed method of Andrew and Gao (2007) for minimizing large-scale l1-regularized models and the rprop method of Riedmiller and Braun (1993). If batch learning is not possible for a given problem, NIEME proposes classical online methods such as stochastic gradient descent. Finally, NIEME also offers mini-batch methods including the recent SVM solver of Shalev-Shwartz et al. (2007).

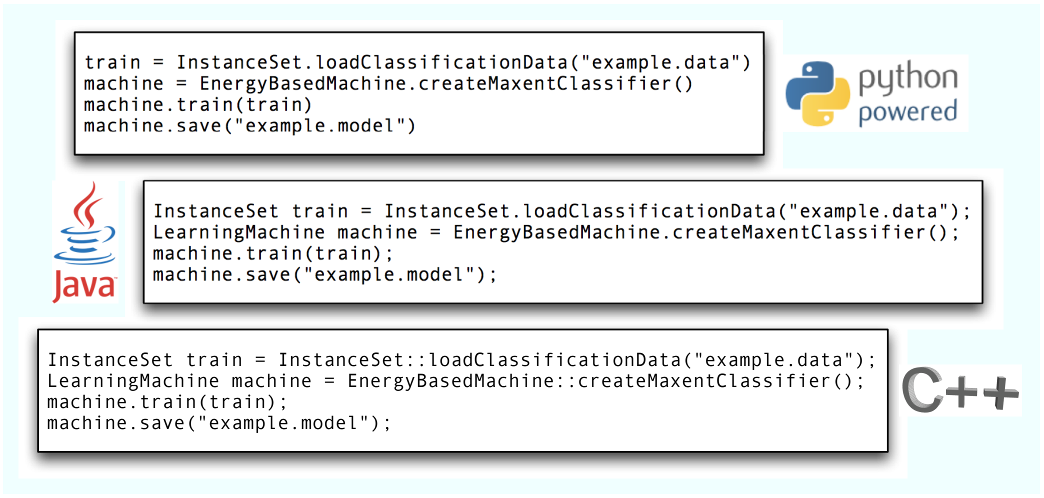
2.3 Feature Space

In large-scale applications, such as text processing, natural language processing, or bioinformatics, examples are often described with sparse feature representations. NIEME has thus been designed for efficient processing of such vectors. Moreover, in online settings, NIEME has the key ability to handle new incoming features at any time within the learning process. Everytime a new feature appears, the parameters of the learning machines are automatically extended to include this new feature. This is particularly interesting when dealing with large data streams for which the feature set cannot be known entirely before learning. Conceptually, all those features exist from the beginning of the learning process although most of them have zero values. Practically, a feature does not affect the learning parameters until it has been seen once.

3. Implementation

The core of NIEME is implemented in portable C++ and compiles currently under Windows (with Visual C++), Mac OS X (with Xcode or Makefiles) and Linux (with Makefiles or KDevelop). The implementation of about 14,000 lines of code is fully object-oriented and makes use of several design-patterns. The code is clear and easy to extend. However, it is not necessary to have a detailed understanding of the implementation in order to use NIEME. Indeed, NIEME includes an easy-to-use interface that can be used from C++, Java or Python, as illustrated in Figure 2. Wrappers for Java and Python are automatically generated thanks to the SWIG tool[2] that makes the glue code. NIEME could even be extended to support languages such as C# or OCaml.

2. SWIG can be found at `http://www.swig.org`.



```
train = InstanceSet.loadClassificationData("example.data")
machine = EnergyBasedMachine.createMaxentClassifier()
machine.train(train)
machine.save("example.model")
```

```
InstanceSet train = InstanceSet.loadClassificationData("example.data");
LearningMachine machine = EnergyBasedMachine.createMaxentClassifier();
machine.train(train);
machine.save("example.model");
```

```
InstanceSet train = InstanceSet::loadClassificationData("example.data");
LearningMachine machine = EnergyBasedMachine::createMaxentClassifier();
machine.train(train);
machine.save("example.model");
```

Figure 2: This figure shows the same program in three languages (Python, Java and C++). The program loads a classification data set, trains a maximum entropy classifier and saves the resulting model in a file called "example.model".

3.1 Tutorials, Documentation, License, Unit Tests

The NIEME website includes a quick-start guide with compilation instructions and tutorials to get started with NIEME in C++, Python or Java. Furthermore, it includes a complete reference documentation of the interface. NIEME is released under the GPL license. All functions of NIEME's interface are unit-tested within the *python unittest* framework.

References

G. Andrew and J. Gao. Scalable training of L1-regularized log-linear models. In Zoubin Ghahramani, editor, *ICML 2007*, pages 33–40. Omnipress, 2007.

Yann LeCun, Sumit Chopra, Raia Hadsell, Ranzato Marc'Aurelio, and Fu-Jie Huang. A tutorial on energy-based learning. In *Predicting Structured Data*. MIT Press, 2006.

D. C. Lio and J. Nocedal. On the limited memory BFGS method for large scale optimization. *Math. Programming*, 45(3):503–528, 1989.

Martin Riedmiller and Heinrich Braun. A direct adaptive method for faster backpropagation learning: The RPROP algorithm. In *ICNN*, pages 586–591, San Francisco, CA, 1993.

S. Shalev-Shwartz, Y. Singer, and N. Srebro. Pegasos: primal estimated sub-gradient solver for SVM. In Zoubin Ghahramani, editor, *ICML 2007*, pages 807–814. Omnipress, 2007.

V. N. Vapnik. An overview of statistical learning theory. *Neural Networks, IEEE Transactions on*, 10(5):988–999, 1999.

Journal of Machine Learning Research 10 (2009) 747-776 Submitted 9/08; Revised 2/09; Published 3/09

Similarity-based Classification: Concepts and Algorithms

Yihua Chen YHCHEN@U.WASHINGTON.EDU
Eric K. Garcia GARCIAER@U.WASHINGTON.EDU
Maya R. Gupta GUPTA@EE.WASHINGTON.EDU
Department of Electrical Engineering
University of Washington
Seattle, WA 98195, USA

Ali Rahimi ALI.RAHIMI@INTEL.COM
1100 NE 45th Street
Intel Research
Seattle, WA 98105, USA

Luca Cazzanti LUCA@APL.WASHINGTON.EDU
Applied Physics Lab
University of Washington
Seattle, WA 98105, USA

Editor: Alexander J. Smola

Abstract

This paper reviews and extends the field of similarity-based classification, presenting new analyses, algorithms, data sets, and a comprehensive set of experimental results for a rich collection of classification problems. Specifically, the generalizability of using similarities as features is analyzed, design goals and methods for weighting nearest-neighbors for similarity-based learning are proposed, and different methods for consistently converting similarities into kernels are compared. Experiments on eight real data sets compare eight approaches and their variants to similarity-based learning.

Keywords: similarity, dissimilarity, similarity-based learning, indefinite kernels

1. Introduction

Similarity-based classifiers estimate the class label of a test sample based on the similarities between the test sample and a set of labeled training samples, and the pairwise similarities between the training samples. Like others, we use the term *similarity-based classification* whether the pairwise relationship is a similarity or dissimilarity. Similarity-based classification does not require direct access to the features of the samples, and thus the sample space can be any set, not necessarily a Euclidean space, as long as the similarity function is well defined for any pair of samples. Let Ω be the sample space and $\mathcal{G}$ be the finite set of class labels. Let $\psi : \Omega \times \Omega \to \mathbb{R}$ be the similarity function. We assume that the pairwise similarities between n training samples are given as an $n \times n$ similarity matrix S whose (i, j)-entry is $\psi(x_i, x_j)$, where $x_i \in \Omega$, $i = 1, \dots, n$, denotes the ith training sample, and $y_i \in \mathcal{G}$, $i = 1, \dots, n$ the corresponding ith class label. The problem is to estimate the class label $\hat{y}$ for a test sample x based on its similarities to the training samples $\psi(x, x_i)$, $i = 1, \dots, n$ and its self-similarity $\psi(x, x)$.

Similarity-based classification is useful for problems in computer vision, bioinformatics, information retrieval, natural language processing, and a broad range of other fields. Similarity functions may be asymmetric and fail to satisfy the other mathematical properties required for metrics or inner products (Santini and Jain, 1999). Some simple example similarity functions are: travel time from one place to another, compressibility of one random process given a code built for another, and the minimum number of steps to convert one sequence into another (edit distance). Computer vision researchers use many similarities, such as the tangent distance (Duda et al., 2001), earth mover's distance (EMD) (Rubner et al., 2000), shape matching distance (Belongie et al., 2002), and pyramid match kernel (Grauman and Darrell, 2007) to measure the similarity or dissimilarity between images in order to do image retrieval and object recognition. In bioinformatics, the Smith-Waterman algorithm (Smith and Waterman, 1981), the FASTA algorithm (Lipman and Pearson, 1985) and the BLAST algorithm (Altschul et al., 1990) are popular methods to compute the similarity between different amino acid sequences for protein classification. The cosine similarity between term frequency-inverse document frequency (tf-idf) vectors is widely used in information retrieval and text mining for document classification.

Notions of similarity appear to play a fundamental role in human learning, and thus psychologists have done extensive research to model human similarity judgement. Tversky's *contrast model* and *ratio model* (Tversky, 1977) represent an important class of similarity functions. In these two models, each sample is represented by a set of features, and the similarity function is an increasing function of set overlap but a decreasing function of set differences. Tversky's set-theoretic similarity models have been successful in explaining human judgement in various similarity assessment tasks, and are consistent with the observations made by psychologists that metrics do not account for cognitive judgement of similarity in complex situations (Tversky, 1977; Tversky and Gati, 1982; Gati and Tversky, 1984). Therefore, similarity-based classification may be useful for imitating or understanding how humans categorize.

The main contributions of this paper are: (1) we distill and analyze concepts and issues specific to similarity-based learning, including the generalizability of using similarities as features, (2) we propose similarity-based nearest-neighbor design goals and methods, and (3) we present a comprehensive set of experimental results for eight similarity-based learning problems and eight different similarity-based classification approaches and their variants. First, we discuss the idea of similarities as inner products in Section 2, then the concept of treating similarities as features in Section 3. The generalizability of using similarities as features and that of using similarities as kernels are compared in Section 4. In Section 5, we propose design goals and solutions for similarity-based weighted nearest-neighbor learning. Generative similarity-based classifiers are discussed in Section 6. Then in Section 7 we describe eight similarity-based classification problems, detail our experimental setup, and discuss the results. The paper concludes with some open questions in Section 8. For the reader's reference, key notation is summarized in Table 1.

2. Similarities as Inner Products

A popular approach to similarity-based classification is to treat the given similarities as inner products in some Hilbert space or to treat dissimilarities as distances in some Euclidean space. This approach can be roughly divided into two categories: one is to explicitly embed the samples in a Euclidean space according to the given (dis)similarities using multidimensional scaling (see Borg and Groenen, 2005, for further reading); the other is to modify the similarities to be kernels and

Ω	sample space	S	$n \times n$ matrix with (i,j)-entry $\psi(x_i,x_j)$
$\mathcal{G}$	set of class labels	s_i	$n \times 1$ vector with jth element $\psi(x_i,x_j)$
n	number of training samples	s	$n \times 1$ vector with jth element $\psi(x,x_j)$
$x_i \in \Omega$	ith training sample	$\mathbf{1}$	column vector of 1's
$x \in \Omega$	test sample	I	identity matrix
$y_i \in \mathcal{G}$	class label of ith training sample	$I_{\{\cdot\}}$	indicator function
$y \in \mathcal{G}^n$	$n \times 1$ vector with ith element y_i	K	kernel matrix or kernel function
$\hat{y} \in \mathcal{G}$	estimated class label for x	k	neighborhood size
$\mathcal{D}$	n training sample pairs $\{(x_i,y_i)\}_{i=1}^n$	L	hinge loss function
$\psi : \Omega \times \Omega \to \mathbb{R}$	similarity function	$\operatorname{diag}(a)$	diagonal matrix with a as the diagonal

Table 1: Key Notation

apply kernel classifiers. We discuss different methods for modifying similarities into kernels in Section 2.1. An important technicality is how to handle test samples, which is addressed in Section 2.2.

2.1 Modify Similarities into Kernels

The power of kernel methods lies in the implicit use of a reproducing kernel Hilbert space (RKHS) induced by a positive semidefinite (PSD) kernel (Schölkopf and Smola, 2002). Although the mathematical meaning of a kernel is the inner product in some Hilbert space, a standard interpretation of a kernel is the pairwise similarity between different samples. Conversely, many researchers have suggested treating similarities as kernels, and applying any classification algorithm that only depends on inner products. Using similarities as kernels eliminates the need to explicitly embed the samples in a Euclidean space.

Here we focus on the support vector machine (SVM), which is a well-known representative of kernel methods, and thus appears to be a natural approach to similarity-based learning. All the SVM algorithms that we discuss in this paper are for binary classification[1] such that $y_i \in \{\pm 1\}$. Let y be the $n \times 1$ vector whose ith element is y_i. The SVM dual problem can be written as

$$\begin{aligned} \underset{\alpha}{\text{maximize}} \quad & \mathbf{1}^T\alpha - \frac{1}{2}\alpha^T \operatorname{diag}(y) K \operatorname{diag}(y)\alpha \\ \text{subject to} \quad & 0 \preceq \alpha \preceq C\mathbf{1}, \quad y^T\alpha = 0, \end{aligned} \tag{1}$$

with variable $\alpha \in \mathbb{R}^n$, where $C > 0$ is the hyperparameter, K is a PSD kernel matrix whose (i,j)-entry is $K(x_i,x_j)$, $\mathbf{1}$ is the column vector with all entries one, and $\preceq$ denotes component-wise inequality for vectors. The corresponding decision function is Schölkopf and Smola (2002)

$$\hat{y} = \operatorname{sgn}\left(\sum_{i=1}^{n} \alpha_i y_i K(x,x_i) + b\right),$$

where

$$b = y_i - \sum_{j=1}^{n} \alpha_j y_j K(x_i,x_j)$$

for any i that satisfies $0 < \alpha_i < C$. The theory of RKHS requires the kernel to satisfy Mercer's condition, and thus the corresponding kernel matrix K must be PSD. However, many similarity

1. We refer the reader to Hsu and Lin (2002) for multiclass SVM.

functions do not satisfy the properties of an inner product, and thus the similarity matrix S can be indefinite. In the following subsections we discuss several methods to modify similarities into kernels; a previous review can be found in Wu et al. (2005). Unless mentioned otherwise, in the following subsections we assume that S is symmetric. If not, we use its symmetric part $\frac{1}{2}\left(S+S^T\right)$ instead. Notice that the symmetrization does not affect the SVM objective function in (1) since $\alpha^T\frac{1}{2}\left(S+S^T\right)\alpha = \frac{1}{2}\alpha^T S\alpha + \frac{1}{2}\alpha^T S^T\alpha = \alpha^T S\alpha$.

2.1.1 INDEFINITE KERNELS

One approach is to simply replace K with S, and ignore the fact that S is indefinite. For example, although the SVM problem given by (1) is no longer convex when S is indefinite, Lin and Lin (2003) show that the sequential minimal optimization (SMO) (Platt, 1998) algorithm will still converge with a simple modification to the original algorithm, but the solution is a stationary point instead of a global minimum. Ong et al. (2004) interpret this as finding the stationary point in a reproducing kernel Kreĭn space (RKKS), while Haasdonk (2005) shows that this is equivalent to minimizing the distance between reduced convex hulls in a pseudo-Euclidean space. A Kreĭn space, denoted by $\mathcal{K}$, is defined to be the direct sum of two disjoint Hilbert spaces, denoted by $\mathcal{H}_+$ and $\mathcal{H}_-$, respectively. So for any $a, b \in \mathcal{K} = \mathcal{H}_+ \oplus \mathcal{H}_-$, there are unique $a_+, b_+ \in \mathcal{H}_+$ and unique $a_-, b_- \in \mathcal{H}_-$ such that $a = a_+ + a_-$ and $b = b_+ + b_-$. The "inner product" on $\mathcal{K}$ is defined as

$$\langle a, b\rangle_{\mathcal{K}} = \langle a_+, b_+\rangle_{\mathcal{H}_+} - \langle a_-, b_-\rangle_{\mathcal{H}_-},$$

which no longer has the property of positive definiteness. Pseudo-Euclidean space is a special case of Kreĭn space where $\mathcal{H}_+$ and $\mathcal{H}_-$ are two Euclidean spaces. Ong et al. (2004) provide a representer theorem for RKKS that poses learning in RKKS as a problem of finding a stationary point of the risk functional, in contrast to minimizing a risk functional in RKHS. Using indefinite kernels in empirical risk minimization (ERM) methods such as SVM can lead to a saddle point solution and thus does not ensure minimizing the risk functional, so this approach does not guarantee learning in the sense of a good function approximation. Also, the nonconvexity of the problem may require intensive computation.

2.1.2 SPECTRUM CLIP

Since S is assumed to be symmetric, it has an eigenvalue decomposition $S = U^T\Lambda U$, where U is an orthogonal matrix and Λ is a diagonal matrix of real eigenvalues, that is, $\Lambda = \operatorname{diag}(\lambda_1, \ldots, \lambda_n)$. Spectrum clip makes S PSD by clipping all the negative eigenvalues to zero. Some researchers assume that the negative eigenvalues of the similarity matrix are caused by noise and view spectrum clip as a denoising step (Wu et al., 2005). Let

$$\Lambda_{\text{clip}} = \operatorname{diag}\left(\max(\lambda_1, 0), \ldots, \max(\lambda_n, 0)\right),$$

and the modified PSD similarity matrix be $S_{\text{clip}} = U^T\Lambda_{\text{clip}}U$. Let u_i denote the ith column vector of U. Using S_{clip} as a kernel matrix for training the SVM is equivalent to implicitly using $x_i = \Lambda_{\text{clip}}^{1/2}u_i$ as the representation of the ith training sample since $\langle x_i, x_j\rangle$ is equal to the (i,j)-entry of S_{clip}. A mathematical justification for spectrum clip is that S_{clip} is the nearest PSD matrix to S in terms of the Frobenius norm (Higham, 1988), that is,

$$S_{\text{clip}} = \arg\min_{K\succeq 0} \|K - S\|_F,$$

where $\succeq$ denotes the generalized inequality with respect to the PSD cone.

Recently, Luss and d'Aspremont (2007) have proposed a robust extension of SVM for indefinite kernels. Instead of only considering the nearest PSD matrix S_{clip}, they consider all the PSD matrices within distance β to S, that is, $\{K \succeq 0 \mid \|K-S\|_F \leq \beta\}$, where $\beta > \min_{K\succeq 0} \|K-S\|_F$, and propose to maximize the worst case of the SVM dual objective among these matrices:

$$\begin{array}{ll} \underset{\alpha}{\text{maximize}} & \underset{K\succeq 0, \|K-S\|_F\leq\beta}{\min} \left(\mathbf{1}^T\alpha - \frac{1}{2}\alpha^T \operatorname{diag}(y) K \operatorname{diag}(y)\alpha\right) \\ \text{subject to} & 0 \preceq \alpha \preceq C\mathbf{1}, \quad y^T\alpha = 0. \end{array}$$

This model is more flexible in the sense that the set of possible K in the hypothesis space lies in a ball with radius β centered around S. Different draws of training samples will change the candidate set of K and could cause overfitting. In practice, they replace the hard constraint $\|K-S\|_F \leq \beta$ with a penalty term and propose the following problem as the robust SVM for S:

$$\begin{array}{ll} \underset{\alpha}{\text{maximize}} & \underset{K\succeq 0}{\min} \left(\mathbf{1}^T\alpha - \frac{1}{2}\alpha^T \operatorname{diag}(y) K \operatorname{diag}(y)\alpha + \rho\|K-S\|_F^2\right) \\ \text{subject to} & 0 \preceq \alpha \preceq C\mathbf{1}, \quad y^T\alpha = 0, \end{array} \tag{2}$$

where $\rho > 0$ is the parameter to control the trade-off. They point out that the inner problem of (2) has a closed-form solution, and the outer problem is convex since its objective is a pointwise minimum of a set of concave quadratic functions of α and thus concave. A fast algorithm to solve (2) is given by Chen and Ye (2008).

2.1.3 SPECTRUM FLIP

In contrast to the interpretation that negative eigenvalues are caused by noise, Laub and Müller (2004) and Laub et al. (2006) show that the negative eigenvalues of some similarity data can code useful information about object features or categories, which agrees with some fundamental psychological studies (Tversky and Gati, 1982; Gati and Tversky, 1982). In order to use the negative eigenvalues, Graepel et al. (1998) propose an SVM in pseudo-Euclidean space, and Pekalska et al. (2001) also consider a generalized nearest mean classifier and Fisher linear discriminant classifier in the same space. Following the notation in Section 2.1.1, they assume that the samples lie in a Kreĭn space $\mathcal{K} = \mathcal{H}_+ \oplus \mathcal{H}_-$ with similarities given by $\psi(a,b) = \langle a_+, b_+\rangle_{\mathcal{H}_+} - \langle a_-, b_-\rangle_{\mathcal{H}_-}$. These proposed classifiers are their standard versions in the Hilbert space $\mathcal{H} = \mathcal{H}_+ \oplus \mathcal{H}_-$ with associated inner product $\langle a,b\rangle_{\mathcal{H}} = \langle a_+, b_+\rangle_{\mathcal{H}_+} + \langle a_-, b_-\rangle_{\mathcal{H}_-}$. This is equivalent to flipping the sign of the negative eigenvalues of the similarity matrix S: let $\Lambda_{\text{flip}} = \operatorname{diag}(|\lambda_1|, \ldots, |\lambda_n|)$, and then the similarity matrix after spectrum flip is $S_{\text{flip}} = U^T\Lambda_{\text{flip}}U$. Wu et al. (2005) note that this is the same as replacing the original eigenvalues of S with its singular values.

2.1.4 SPECTRUM SHIFT

Spectrum shift is another popular approach to modifying a similarity matrix into a kernel matrix: since $S + \lambda I = U^T(\Lambda + \lambda I)U$, any indefinite similarity matrix can be made PSD by shifting its spectrum by the absolute value of its minimum eigenvalue $|\lambda_{\min}(S)|$. Let $\Lambda_{\text{shift}} = \Lambda + |\min(\lambda_{\min}(S), 0)|I$, which is used to form the modified similarity matrix $S_{\text{shift}} = U^T\Lambda_{\text{shift}}U$. Compared with spectrum clip and flip, spectrum shift only enhances all the self-similarities by the amount of $|\lambda_{\min}(S)|$ and does not change the similarity between any two different samples. Roth et al. (2003) propose spectrum shift for clustering nonmetric proximity data and show that S_{shift} preserves the group structure of the original data represented by S. Let $\mathcal{X}$ be the set of samples to cluster, and $\{\mathcal{X}_\ell\}_{\ell=1}^N$ be a partition of $\mathcal{X}$ into N sets. Specifically, they consider minimizing the clustering cost function[2]

$$f\left(\{\mathcal{X}_\ell\}_{\ell=1}^N\right) = -\sum_{\ell=1}^{N} \sum_{\substack{i,j\in\mathcal{X}_\ell \\ i\neq j}} \frac{\psi(x_i,x_j)}{|\mathcal{X}_\ell|}, \tag{3}$$

where $|\mathcal{X}_\ell|$ denotes the cardinality of set $\mathcal{X}_\ell$. It is easy to see that (3) is invariant under spectrum shift.

Recently, Zhang et al. (2006) proposed training an SVM only on the k-nearest neighbors of each test sample, called SVM-KNN. They used spectrum shift to produce a kernel from the similarity data. Their experimental results on image classification demonstrated that SVM-KNN performs comparably to a standard SVM classifier, but with significant reduction in training time.

2.1.5 SPECTRUM SQUARE

The fact that $SS^T \succeq 0$ for any $S \in \mathbb{R}^{n\times n}$ led us to consider using SS^T as a kernel, which is valid even when S is not symmetric. For symmetric S, this is equivalent to squaring its spectrum since $SS^T = U^T\Lambda^2 U$. It is also true that using SS^T is the same as defining a new similarity function $\tilde{\psi}$ for any $a,b \in \Omega$ as

$$\tilde{\psi}(a,b) = \sum_{i=1}^{n} \psi(a,x_i)\psi(x_i,b).$$

We note that for symmetric S, treating SS^T as a kernel matrix K is equivalent to representing each x_i by its similarity feature vector $s_i = \begin{bmatrix}\psi(x_i,x_1) & \dots & \psi(x_i,x_n)\end{bmatrix}^T$ since $K_{ij} = \langle s_i, s_j\rangle$. The concept of treating similarities as features is discussed in more detail in Section 3.

2.2 Consistent Treatment of Training and Test Samples

Consider a test sample x that is the same as a training sample x_i. Then if one uses an ERM classifier trained with modified similarities $\tilde{S}$, but uses the unmodified test similarities, represented by vector $s = \begin{bmatrix}\psi(x,x_1) & \dots & \psi(x,x_n)\end{bmatrix}^T$, the same sample will be treated inconsistently. In general, one would like to modify the training and test similarities in a consistent fashion, that is, to modify the underlying similarity function rather than only modifying the S. In this context, given S and $\tilde{S}$, we term a transformation T on test samples *consistent* if $T(s_i)$ is equal to the ith row of $\tilde{S}$ for $i = 1,\dots,n$.

One solution is to modify the training and test samples all at once. However, when test samples are not known beforehand, this may not be possible. For such cases, Wu et al. (2005) proposed to first modify S and train the classifier using the modified $n\times n$ similarity matrix $\tilde{S}$, and then for each test sample modify its s in an effort to be consistent with the modified similarities used to train the model. Their approach is to re-compute the same modification on the augmented $(n+1)\times(n+1)$ similarity matrix

$$S' = \begin{bmatrix} S & s \\ s^T & \psi(x,x) \end{bmatrix}$$

2. They originally use dissimilarities in their cost function, and we reformulate it into similarities with the assumption that the relationship between dissimilarities and similarities is affine.

to form $\tilde{S}'$, and then let the modified test similarities $\tilde{s}$ be the first n elements of the last column of $\tilde{S}'$. The classifier that was trained on $\tilde{S}$ is then applied on $\tilde{s}$. To implement this approach, Wu et al. (2005) propose a fast algorithm to perform eigenvalue decomposition of S' by using the results of the eigenvalue decomposition of S. However, this approach does not guarantee consistency.

To attain consistency, we note that both the spectrum clip and flip modifications can be represented by linear transformations, that is, $\tilde{S} = PS$, where P is the corresponding transformation matrix, and we propose to apply the same linear transformation P on s such that $\tilde{s} = Ps$. For spectrum flip, the linear transformation is $P_{\text{flip}} = U^T M_{\text{flip}} U$, where

$$M_{\text{flip}} = \operatorname{diag}\left(\operatorname{sgn}(\lambda_1), \ldots, \operatorname{sgn}(\lambda_n)\right).$$

For spectrum clip, the linear transformation is $P_{\text{clip}} = U^T M_{\text{clip}} U$, where

$$M_{\text{clip}} = \operatorname{diag}\left(I_{\{\lambda_1 \geq 0\}}, \ldots, I_{\{\lambda_n \geq 0\}}\right),$$

and $I_{\{\cdot\}}$ is the indicator function. Recall that using $\tilde{S}$ implies embedding the training samples in a Euclidean space. For spectrum clip, this linear transformation is equivalent to embedding the test sample as a feature vector into the same Euclidean space of the embedded training samples:

Proposition 1 *Let S_{clip} be the Gram matrix of the column vectors of $X \in \mathbb{R}^{m \times n}$, where $\operatorname{rank}(X) = m$. For a given s, let $x = \arg\min_{z \in \mathbb{R}^m} \|X^T z - s\|_2$, then $X^T x = P_{\text{clip}} s$.*

The proof is in the appendix.

Proposition 1 states that if the n training samples are embedded in $\mathbb{R}^m$ with S_{clip} as the Gram matrix, and we embed the test sample in $\mathbb{R}^m$ by finding the feature vector whose inner products with the embedded training samples are closest to the given s, then the inner products between the embedded test sample and the embedded training samples are indeed $\tilde{s} = P_{\text{clip}} s$.

On the other hand, there is no linear transformation to ensure consistency for spectrum shift. For our experiments using spectrum shift, we adopt the approach of Wu et al. (2005), which for this case is to let $\tilde{s} = s$, because spectrum shift only affects self-similarities.

3. Similarities as Features

Similarity-based classification problems can be formulated into standard learning problems in Euclidean space by treating the similarities between a sample x and the n training samples as features (Graepel et al., 1998, 1999; Pekalska et al., 2001; Pekalska and Duin, 2002; Liao and Noble, 2003). That is, represent sample x by the similarity feature vector s. As detailed in Section 4, the generalizability analysis yields different results for using similarities as features and using similarities as inner products.

Graepel et al. (1998) consider applying a linear SVM on similarity feature vectors by solving the following problem:

$$\underset{w,b}{\text{minimize}} \quad \frac{1}{2}\|w\|_2^2 + C\sum_{i=1}^{n} L(w^T s_i + b, y_i) \tag{4}$$

with variables $w \in \mathbb{R}^n$, $b \in \mathbb{R}$ and hyperparameter $C > 0$, where $L(\alpha, \beta) \triangleq \max(1 - \alpha\beta, 0)$ is the hinge loss function. Liao and Noble (2003) also propose to apply an SVM on similarity feature vectors; they use a Gaussian radial basis function (RBF) kernel.

In order to make the solution w sparser, which helps ease the computation of the discriminant function $f(s) = w^T s + b$, Graepel et al. (1999) substitute the ℓ_1-norm regularization for the squared ℓ_2-norm regularization in (4), and propose a linear programming (LP) machine:

$$\underset{w,b}{\text{minimize}} \quad \|w\|_1 + C \sum_{i=1}^{n} L(w^T s_i + b, y_i). \tag{5}$$

Balcan et al. (2008a) provide a theoretical analysis for using similarities as features, and show that if a similarity is good in the sense that the expected intraclass similarity is sufficiently large compared to the expected interclass similarity, then given n training samples, there exists a linear separator on the similarities as features that has a specifiable maximum error at a margin that depends on n. Specifically, Theorem 4 in Balcan et al. (2008a) gives a sufficient condition on the similarity function ψ for (4) to achieve good generalization. Their latest results for ℓ_1-margin (inversely proportional to $\|w\|_1$) provide similar theoretical guarantees for (5) (Balcan et al., 2008b, Theorem 11). Wang et al. (2007) show that under slightly less restrictive assumptions on the similarity function there exists with high probability a convex combination of simple classifiers on the similarities as features which has a maximum specifiable error.

Another approach is the potential support vector machine (P-SVM) (Hochreiter and Obermayer, 2006; Knebel et al., 2008), which solves

$$\begin{aligned} \underset{\alpha}{\text{minimize}} \quad & \frac{1}{2}\|y - S\alpha\|_2^2 + \varepsilon\|\alpha\|_1 \\ \text{subject to} \quad & \|\alpha\|_\infty \le C, \end{aligned} \tag{6}$$

where $C > 0$ and $\varepsilon > 0$ are two hyperparameters. We note that by strong duality (6) is equivalent to

$$\underset{\alpha}{\text{minimize}} \quad \frac{1}{2}\|y - S\alpha\|_2^2 + \varepsilon\|\alpha\|_1 + \gamma\|\alpha\|_\infty \tag{7}$$

for some $\gamma > 0$. One can see from (7) that P-SVM is equivalent to the lasso regression (Tibshirani, 1996) with an extra ℓ_∞-norm regularization term. The use of multiple regularization terms in P-SVM is similar to the elastic net (Zou and Hastie, 2005), which uses ℓ_1 and squared ℓ_2 regularization together.

The algorithms above minimize the empirical risk with regularization. In addition, Pekalska et al. consider generative classifiers for similarity feature vectors; they propose a regularized Fisher linear discriminant classifier (Pekalska et al., 2001) and a regularized quadratic discriminant classifier (Pekalska and Duin, 2002).

We note that treating similarities as features may not capture discriminative information if there is a large intraclass variance compared to the interclass variance, even if the classes are well-separated. A simple example is if the two classes are generated by Gaussian distributions with highly-ellipsoidal covariances, and the similarity function is taken to be a negative linear function of the distance.

4. Generalization Bounds of Similarity SVM Classifiers

To investigate the generalizability of SVM classifiers using similarities, we analyze two forms of SVMs: using similarity as a kernel as discussed in Section 2, and a linear SVM using the similarities as features as given by (4). When similarities are used as features, we show that good generalization performance can be achieved by training the SVM on a small subset of m ($< n$) randomly selected training examples, and we compare this to the established analysis for the kernelized SVM.

The SVM learns a discriminant function $f(s) = w^T s + b$ by minimizing the empirical risk

$$\hat{R}_{\mathcal{D}}(f, L) = \frac{1}{n} \sum_{i=1}^{n} L(f(s_i), y_i),$$

where $\mathcal{D}$ denotes the training set, subject to some smoothness constraint $\mathcal{N}$, that is,

$$\underset{f}{\text{minimize}} \quad \hat{R}_{\mathcal{D}}(f, L) + \lambda_n \mathcal{N}(f),$$

where $\lambda_n = \frac{1}{2nC}$. We note that using (arbitrary) similarities as features corresponds to setting $\mathcal{N}(f) = w^T w$, while using (PSD) similarities as a kernel changes the smoothness constraint to $\mathcal{N}(f) = w^T S w$ (Rifkin, 2002, Appendix B), and in fact, this change of regularizer is the only difference between these two similarity-based SVM approaches. In this section, we examine how this small change in regularization affects the generalization ability of the classifiers.

To simplify the following analysis, we do not directly investigate the SVM classifier as presented; instead, as is standard in SVM learning theory, we investigate the following constrained version of the problem:

$$\begin{aligned} &\underset{f}{\text{minimize}} && \hat{R}_{\mathcal{D}}(f, L_t) \\ &\text{subject to} && \mathcal{N}(f) \leq \beta^2, \end{aligned} \tag{8}$$

with truncated hinge loss $L_t \triangleq \min(L, 1) \in [0, 1]$ and $f(s) = w^T s$ stripped of the intercept b.

The following generalization bound for the SVM using (PSD) similarities[3] as a kernel follows directly from the results in Bartlett and Mendelson (2002).

Theorem 1 (Generalization Bound of Similarities as Kernel) *Suppose* (x, y) *and* $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^n$ *are drawn i.i.d. from a distribution on* $\Omega \times \{\pm 1\}$. *Let* ψ *be a positive definite similarity such that* $\psi(a, a) \leq \kappa^2$ *for some* $\kappa > 0$ *and all* $a \in \Omega$. *Let* S *be the* $n \times n$ *matrix with* (i, j)*-entry* $\psi(x_i, x_j)$ *and* s *be the* $n \times 1$ *vector with ith element* $\psi(x, x_i)$. *Define* F_S *to be the set of real-valued functions* $\{f(s) = w^T s \,|\, w^T S w \leq \beta^2\}$ *for a finite* β. *Then with probability at least* $1 - \delta$ *with respect to* $\mathcal{D}$, *every function* f *in* F_S *satisfies*

$$P(yf(s) \leq 0) \leq \hat{R}_{\mathcal{D}}(f, L_t) + 4\beta\kappa\sqrt{\frac{1}{n}} + \sqrt{\frac{\ln(2/\delta)}{2n}}.$$

The proof is in the appendix.

Theorem 1 says that with high probability, as $n \to \infty$, the misclassification rate is tightly bounded by the empirical risk $\hat{R}_{\mathcal{D}}(f, L_t)$, implying that a discriminant function trained by (8) with $\mathcal{N}(f) = w^T S w$ generalizes well to unseen data.

Next, we state a weaker result in Theorem 2 for the SVM using (arbitrary) similarities as features. Let the features be the similarities to m ($< n$) prototypes $\{(\tilde{x}_1, \tilde{y}_1), \ldots, (\tilde{x}_m, \tilde{y}_m)\} \subseteq \mathcal{D}$ randomly chosen from the training set so that $\tilde{s}$ is the $m \times 1$ vector with ith element $\psi(x, \tilde{x}_i)$. Results will be obtained on the remaining $n - m$ training data $\widetilde{\mathcal{D}} = \mathcal{D} \backslash \{(\tilde{x}_1, \tilde{y}_1), \ldots, (\tilde{x}_m, \tilde{y}_m)\}$.

3. If the original similarities are not PSD, then they must be modified to be PSD before this result applies; see Section 2 for a discussion of common PSD modifications.

Theorem 2 (Generalization Bound of Similarities as Features) *Suppose (x,y) and $\mathcal{D} = \{(x_i,y_i)\}_{i=1}^n$ are drawn i.i.d. from a distribution on $\Omega \times \{\pm 1\}$. Let ψ be a similarity such that $\psi(a,b) \leq \kappa^2$ for some $\kappa > 0$ and all $a,b \in \Omega$. Let $\{(\tilde{x}_1,\tilde{y}_1),\ldots,(\tilde{x}_m,\tilde{y}_m)\} \subseteq \mathcal{D}$ be a set of randomly chosen prototypes, and denote $\widetilde{\mathcal{D}} = \mathcal{D}\backslash\{(\tilde{x}_1,\tilde{y}_1),\ldots,(\tilde{x}_m,\tilde{y}_m)\}$. Let $\tilde{s}$ be the $m \times 1$ vector with ith element $\psi(x,\tilde{x}_i)$. Define F_l to be the set of real-valued functions $\left\{f(\tilde{s}) = w^T\tilde{s} \,\middle|\, w^Tw \leq \beta^2\right\}$ for a finite β. Then with probability at least $1-\delta$ with respect to $\widetilde{\mathcal{D}}$, every function f in F_l satisfies*

$$P\left(yf(s) \leq 0\right) \leq \hat{R}_{\widetilde{\mathcal{D}}}(f,L_t) + 4\beta\kappa^2\sqrt{\frac{m}{n}} + \sqrt{\frac{\ln(2/\delta)}{2n}}.$$

The proof is in the appendix.

Theorem 2 only differs significantly from Theorem 1 in the term $\sqrt{m/n}$, which means that if m, the number of prototypes used, grows no faster than $o(n)$, then with high probability, as $n \to \infty$, the misclassification rate is tightly bounded by the empirical risk on the remaining training set $\hat{R}_{\widetilde{\mathcal{D}}}(f,L_t)$. Note that Theorem 2 is unable to claim anything about the generalization when $m = n$, that is, the entire training set is chosen as prototypes. For a further discussion, see the appendix.

5. Similarity-based Weighted Nearest-Neighbors

In this section, we consider design goals and propose solutions for weighted nearest-neighbors for similarity-based classification. Nearest-neighbor learning is the algorithmic parallel of the *exemplar* model of human learning (Goldstone and Kersten, 2003). Weighted nearest-neighbor algorithms are task-flexible because the weights on the neighbors can be used as probabilities as long as they are non-negative and sum to one. For classification, such weights can be summed for each class to form posteriors, which is helpful for use with asymmetric misclassification costs and when the similarity-based classifier is a component of a larger decision-making system. As a lazy learning method, weighted nearest-neighbor classifiers do not require training before the arrival of test samples. This can be advantageous to certain applications where the amount of training data is huge, or there are a large number of classes, or the training data is constantly evolving.

5.1 Design Goals for Similarity-based Weighted k-NN

In this section, for a test sample x, we use x_i to denote its ith nearest neighbor from the training set as defined by the similarity function ψ for $i = 1,\ldots,k$, and y_i to denote the label of x_i. Also, we redefine S as the $k \times k$ matrix of the similarities between the k-nearest neighbors and s the $k \times 1$ vector of the similarities between the test sample x and its k-nearest neighbors. For each test sample, weighted k-NN assigns weight w_i to the ith nearest neighbor for $i = 1,\ldots,k$. Weighted k-NN classifies the test sample x as the class $\hat{y}$ that is assigned the most weight,

$$\hat{y} = \arg\max_{g\in\mathcal{G}} \sum_{i=1}^{k} w_i I_{\{y_i=g\}}. \qquad (9)$$

It is common to additionally require that the weights be nonnegative and normalized such that the weights form a posterior distribution over the set of classes $\mathcal{G}$. Then the estimated probability for class g is $\sum_{i=1}^k w_i I_{\{y_i=g\}}$, which can be used with asymmetric misclassification costs.

An intuitive and standard approach to weighting nearest neighbors is to give larger weight to neighbors that are more similar to the test sample. Formally, we state:

Design Goal 1 (Affinity): w_i should be an increasing function of $\psi(x, x_i)$.

In addition, we propose a second design goal. In practice, some samples in the training set are often very similar, for example, a random sampling of the emails by one person may include many emails from the same thread that contain repeated text due to replies and forwarding. Such similar training samples provide highly-correlated information to the classifier. In fact, many of the nearest neighbors may provide very similar information which can bias the classifier. Moreover, we consider those training samples that are similar to many other training samples less valuable based on the same motivation for tf-idf. To address this problem, one can choose weights to down-weight highly similar samples and ensure that a diverse set of the neighbors has a voice in the classification decision. We formalize this goal as:

Design Goal 2 (Diversity): w_i should be a decreasing function of $\psi(x_i, x_j)$.

Next we propose two approaches to weighting neighbors for similarity-based classification that aim to satisfy these goals.

5.2 Kernel Ridge Interpolation Weights

First, we describe kernel regularized linear interpolation, and we show that it leads to weights that satisfy the design goals. Gupta et al. (2006) proposed weights for k-NN in Euclidean space that satisfy a linear interpolation with maximum entropy (LIME) objective:

$$\begin{aligned} \underset{w}{\text{minimize}} \quad & \left\| \sum_{i=1}^{k} w_i x_i - x \right\|_2^2 - \lambda H(w) \\ \text{subject to} \quad & \sum_{i=1}^{k} w_i = 1,\ w_i \geq 0,\ i = 1, \ldots, k, \end{aligned} \tag{10}$$

with variable $w \in \mathbb{R}^k$, where $H(w) = -\sum_{i=1}^{k} w_i \log w_i$ is the entropy of the weights and $\lambda > 0$ is a regularization parameter. The first term of the convex objective in (10) tries to solve the linear interpolation equations, which balances the weights so that the test point is best approximated by a convex combination of the training samples. Additionally, the entropy maximization pushes the LIME weights toward the uniform weights.

We simplify (10) to a quadratic programming (QP) problem by replacing the negative entropy regularization with a ridge regularizer[4] $w^T w$, and we rewrite (10) in matrix form:

$$\begin{aligned} \underset{w}{\text{minimize}} \quad & \frac{1}{2} w^T X^T X w - x^T X w + \frac{\lambda}{2} w^T w \\ \text{subject to} \quad & w \succeq 0, \quad \mathbf{1}^T w = 1, \end{aligned} \tag{11}$$

where $X = \begin{bmatrix} x_1 & x_2 & \cdots & x_k \end{bmatrix}$. Note that (11) is completely specified in terms of the inner products of the feature vectors: $\langle x_i, x_j \rangle$ and $\langle x, x_i \rangle$, and thus we term the solution to (11) as *kernel ridge*

4. Due to the constraint $\mathbf{1}^T w = 1$, the ridge regularizer actually regularizes the variance of the weights and thus has similar effect as the negative entropy regularizer.

interpolation (KRI) weights. Generalizing from inner products to similarities, we form the KRI similarity-based weights:

$$\begin{aligned} \underset{w}{\text{minimize}} \quad & \frac{1}{2}w^T S w - s^T w + \frac{\lambda}{2} w^T w \\ \text{subject to} \quad & w \succeq 0, \quad \mathbf{1}^T w = 1. \end{aligned} \tag{12}$$

There are three terms in the objective function of (12). Acting alone, the linear term $-s^T w$ would give all the weight to the 1-nearest neighbor. This is prevented by the ridge regularization term $\frac{1}{2}\lambda w^T w$, which regularizes the variance of w and hence pushes the weights toward the uniform weights. These two terms work together to give more weight to the training samples that are more similar to the test sample, and thus help the resulting weights satisfy the first design goal of rewarding neighbors with high affinity to the test sample. The quadratic term in (12) can be expanded as follows,

$$\frac{1}{2}w^T S w = \frac{1}{2}\sum_{i,j} \psi(x_i, x_j) w_i w_j.$$

From the above expansion, one sees that holding all else constant in (12), the bigger $\psi(x_i, x_j)$ and $\psi(x_j, x_i)$ are, the smaller the chosen w_i and w_j will be. Thus the quadratic term tends to down-weight the neighbors that are similar to each other and acts to achieve the second design goal of spreading the weight among a diverse set of neighbors.

A sensitivity analysis further verifies the above observations. Let $g(w; S, s)$ be the objective function of (12), and $w^\star$ denote the optimal solution. To simplify the analysis, we only consider $w^\star$ in the interior of the probability simplex and thus $\nabla g(w^\star; S, s) = 0$. We first perturb s by adding $\delta > 0$ to its ith element, that is, $\tilde{s} = s + \delta e_i$, where e_i denotes the standard basis vector whose ith element is 1 and 0 elsewhere. Then

$$\nabla g(w^\star; S, \tilde{s}) = (S + \lambda I) w^\star - \tilde{s} = -\delta e_i,$$

whose projection on the probability simplex is

$$\nabla g(w^\star; S, \tilde{s}) - \frac{1}{k}\left(\mathbf{1}^T \nabla g(w^\star; S, \tilde{s})\right)\mathbf{1} = \delta\left(\frac{1}{k}\mathbf{1} - e_i\right). \tag{13}$$

The direction of the steepest descent given by the negative of the projected gradient in (13) indicates that the new optimal solution will have an increased w_i, which satisfies the first design goal.

Similarly, if we instead perturb S by adding $\delta > 0$ to its (i, j)-entry ($i \neq j$), that is, $\tilde{S} = S + \delta E_{ij}$, where E_{ij} denotes the matrix with (i, j)-entry 1 and 0 elsewhere, then

$$\nabla g(w^\star; \tilde{S}, s) = (\tilde{S} + \lambda I) w^\star - s = \delta w_j^\star e_i,$$

whose projection on the probability simplex is

$$\nabla g(w^\star; \tilde{S}, s) - \frac{1}{k}\left(\mathbf{1}^T \nabla g(w^\star; \tilde{S}, s)\right)\mathbf{1} = \delta w_j^\star\left(e_i - \frac{1}{k}\mathbf{1}\right). \tag{14}$$

The direction of the steepest descent given by the negative of the projected gradient in (14) indicates that the optimal solution will have a decreased w_i, which satisfies the second design goal.

Experimentally, we found little statistically significant difference between using negative entropy or ridge regularization for the KRI weights. Analytically, entropy regularization leads to an exponential form for the weights that can be used to prove consistency (Friedlander and Gupta, 2006). Computationally, the ridge regularizer is more practical because it results in a QP with box constraints and an equality constraint if the S matrix is PSD or approximated by a PSD matrix, and can thus be solved by the SMO algorithm (Platt, 1998).

5.2.1 KERNEL RIDGE REGRESSION WEIGHTS

A closed-form solution to (12) is possible if one relaxes the problem by removing the constraints $w_i \in [0,1]$ and $\sum_i w_i = 1$ that ensure the weights form a probability mass function. Then for PSD S, the objective $\frac{1}{2}w^T Sw - s^T w + \frac{1}{2}\lambda w^T w$ is solved by

$$w = (S + \lambda I)^{-1}s. \tag{15}$$

The k-NN decision rule (9) using these weights is equivalent to classifying by maximizing the discriminant of a local kernel ridge regression. For each class $g \in \mathcal{G}$, local kernel ridge regression (without intercept) solves

$$\underset{\beta_g}{\text{minimize}} \quad \sum_{i=1}^{k} \left(I_{\{y_i=g\}} - \langle \beta_g, \phi(x_i) \rangle \right)^2 + \lambda \langle \beta_g, \beta_g \rangle, \tag{16}$$

where ϕ denotes the mapping from the sample space Ω to a Hilbert space with inner product $\langle \phi(x_i), \phi(x_j) \rangle = \psi(x_i, x_j)$. Each solution to (16) yields the discriminant $f_g(x) = \langle \beta_g, \phi(x) \rangle = \nu_g^T(S + \lambda I)^{-1}s$ for class g (Cristianini and Shawe-Taylor, 2000), where $\nu_g = \begin{bmatrix} I_{\{y_1=g\}} & \dots & I_{\{y_k=g\}} \end{bmatrix}^T$. Maximizing $f_g(x)$ over $g \in \mathcal{G}$ produces the same estimated class label as (9) using the weights given in (15), thus we refer to these weights as *kernel ridge regression* (KRR) weights.

For a non-PSD S, it is possible that $S + \lambda I$ is singular. In the experiments, we compare handling non-PSD S by clip, flip, shift, or taking the pseudo-inverse (pinv) $(S + \lambda I)^{\dagger}$.

5.2.2 ILLUSTRATIVE EXAMPLES

We illustrate the KRI and KRR[5] weights with three toy examples shown on the next page. For each example, there are $k = 4$ nearest-neighbors, and the KRI and KRR weights are shown for a range of regularization parameter λ.

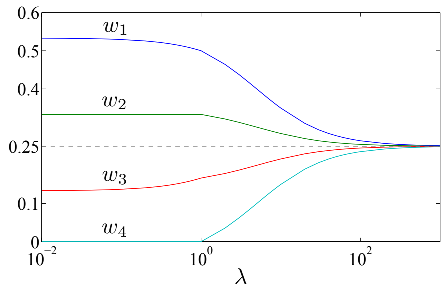
In Example 1, affinity is illustrated. One sees from s that the four distinct training samples are not equally similar to the test sample, and from S that the training samples have zero similarity to each other. Both KRI and KRR give more weight to training samples that are more similar to the test sample, illustrating that the weighting methods achieve the design goal of affinity.

In Example 2, diversity is illustrated. The four training samples all have similarity 3 to the test sample, but x_2 and x_3 are very similar to each other, and are thus weighted down as prescribed the by the design goal of diversity. Because of the symmetry of the similarities, the weights for x_2 and x_3 are exactly the same for both KRI and KRR.

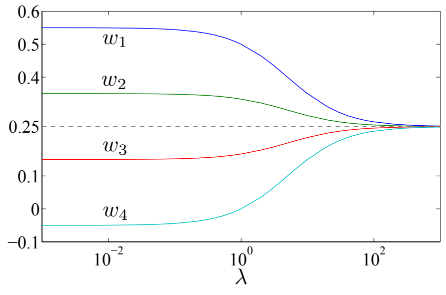
In Example 3, the interaction between the two design goals is illustrated. The S matrix is the same as in Example 2, but s is no longer uniform. In fact, although x_1 is less similar to the test sample than x_3, x_1 receives more weight because it is less similar to other training samples. The

5. For the purpose of comparison, we show the normalized KRR weights $\tilde{w}$ where $\tilde{w} = \left(I - \frac{1}{k}\mathbf{1}\mathbf{1}^T\right)w + \frac{1}{k}\mathbf{1}$. This does not affect the result of classification since each weight is shifted by the same constant.

Example 1: $s^T = \begin{bmatrix} 4 & 3 & 2 & 1 \end{bmatrix}, \quad S = \begin{bmatrix} 5 & 0 & 0 & 0 \\ 0 & 5 & 0 & 0 \\ 0 & 0 & 5 & 0 \\ 0 & 0 & 0 & 5 \end{bmatrix}$

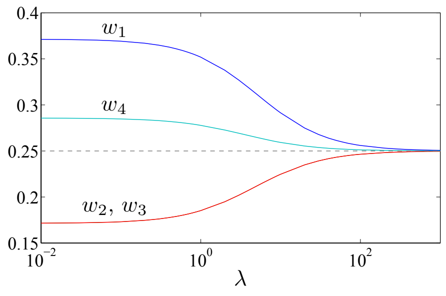


KRI weights

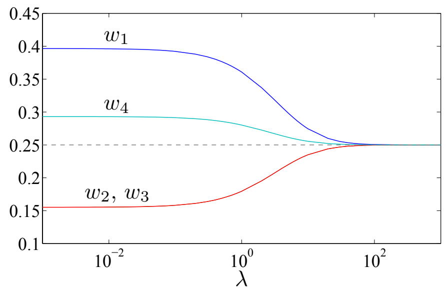


KRR weights

Example 2: $s^T = \begin{bmatrix} 3 & 3 & 3 & 3 \end{bmatrix}, \quad S = \begin{bmatrix} 5 & 1 & 1 & 1 \\ 1 & 5 & 4 & 2 \\ 1 & 4 & 5 & 2 \\ 1 & 2 & 2 & 5 \end{bmatrix}$

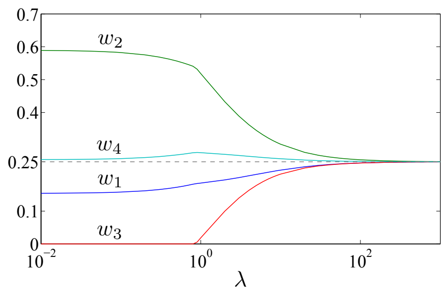


KRI weights

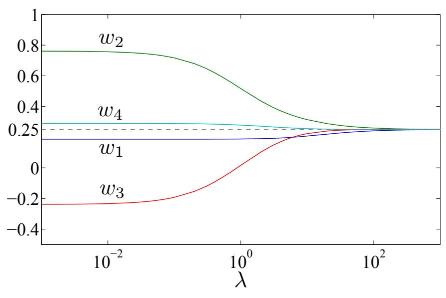


KRR weights

Example 3: $s^T = \begin{bmatrix} 2 & 4 & 3 & 3 \end{bmatrix}, \quad S = \begin{bmatrix} 5 & 1 & 1 & 1 \\ 1 & 5 & 4 & 2 \\ 1 & 4 & 5 & 2 \\ 1 & 2 & 2 & 5 \end{bmatrix}$



KRI weights



KRR weights

affinity goal allots the largest weight to the most similar neighbor x_2, but because x_2 and x_3 are highly similar, the diversity goal forces them to share weight, resulting in x_3 receiving little weight.

One observes from these examples that the KRR weights tend to be smoother than the KRI weights, because the KRI weights are constrained to a probability simplex, while the KRR weights are unconstrained.

6. Generative Classifiers

Generative classifiers provide probabilistic outputs that can be easily fused with other probabilistic information or used to minimize expected misclassification costs. One approach to generative classification given similarities is using the similarities as features to define an n-dimensional feature space, and then fitting standard generative models in that space, as discussed in Section 3. However, this requires fitting generative models with n data points (or less for class-conditional models) in an n-dimensional space. Another generative framework for similarity-based classification termed *similarity discriminant analysis* (SDA) has been proposed that models the class-conditional distributions of similarity statistics (Cazzanti et al., 2008). First we review the basic SDA model, then consider a local variant (Cazzanti and Gupta, 2007) and a mixture model variant that were both designed to reduce bias.

Let X denote a random test sample and x denote a realization of X. Assume that the relevant information about the class label of X is captured by a finite set $\mathcal{T}(X)$ of M descriptive statistics, where the mth descriptive statistic is denoted $\mathcal{T}_m(X)$. In this paper, we take the set of descriptive statistics to be the similarities to class centroids:

$$\mathcal{T}(x) = \{\psi(x,\mu_1), \psi(x,\mu_2), \ldots, \psi(x,\mu_G)\}, \tag{17}$$

where $\mu_g \in \Omega$ is a centroid for the gth class, and G is the number of classes. Although there are many possible definitions of a centroid given similarities, in this paper a class centroid is defined to be the training sample that has maximum sum-similarity to the other training samples of its class. The centroid-based descriptive statistics given by (17) were shown to be overall more effective than other considered descriptive statistics on simulations and a small set of real-data experiments (Cazzanti, 2007).

The classification rule for SDA to minimize the expected misclassification cost is: classify x as the class

$$\hat{y} = \arg\min_{g\in\mathcal{G}} \sum_{h\in\mathcal{G}} C(g,h) P\left(\mathcal{T}(x)\,|\,Y=h\right) P(Y=h), \tag{18}$$

where $C(g,h)$ is the cost of classifying a sample as class g if the truth is class h.

To estimate the class-conditional distributions $\{P(\mathcal{T}(x)\,|\,Y=g)\}_{g=1}^{G}$ the SDA model estimates the expected value of the mth descriptive statistic $\mathcal{T}_m(X)$ with respect to the class conditional distribution $P(\mathcal{T}(x)\,|\,Y=g)$ to be the average of the training sample data for each class g:

$$E_{P(\mathcal{T}(x)\,|\,g)}\left(\mathcal{T}_m(X)\right) = \frac{1}{|\mathcal{X}_g|} \sum_{z\in\mathcal{X}_g} \mathcal{T}_m(z), \tag{19}$$

where $\mathcal{X}_g$ is the set of training samples from class g. Given the $G\times G$ constraints specified by (19), the SDA model estimates each class-conditional distribution as the solution to (19) with maximum entropy, which is the exponential (Jaynes, 1982):

$$\hat{P}(\mathcal{T}(x)\,|\,g) = \prod_{m=1}^{G} \gamma_{gm} e^{\lambda_{gm}\mathcal{T}_m(x)}. \tag{20}$$

Substituting the maximum entropy solution (20) into (18) yields the SDA classification rule: classify x as the class

$$\hat{y} = \arg\min_{g\in\mathcal{G}} \sum_{h\in\mathcal{G}} C(g,h)P(Y=h) \prod_{m=1}^{G} \gamma_{hm} e^{\lambda_{hm}\mathcal{T}_m(x)}.$$

Each pair of parameters $(\lambda_{gm}, \gamma_{gm})$ can be separately calculated from the constraints given in (19) by one-dimensional optimization and normalization.

6.1 Reducing SDA Model Bias

The SDA model may not be flexible enough to capture a complicated decision boundary. To address this model bias issue, one could apply SDA locally to a neighborhood of k nearest neighbors for each test point, or learn a mixture SDA model.

In this paper we experimentally compare to *local SDA* (Cazzanti and Gupta, 2007) with local centroid-similarity descriptive statistics given by (17), in which SDA is applied to the k nearest neighbors of a test point, where the parameter k is trained by cross-validation. If any class in the neighborhood has fewer than three samples, there are not enough data samples to fit distributions of similarities, so every λ is assumed to be zero, and the local SDA model is reduced to a simple local centroid classifier. Given a discrete set of possible similarities, local SDA has been shown to be a consistent classifier in the sense that its error converges to the Bayes error under the usual asymptotic assumption that when the number of training samples $n \to \infty$, the neighborhood size $k \to \infty$ but k grows relatively slowly such that $k/n \to 0$ (Cazzanti and Gupta, 2007).

Cazzanti (2007) explored mixture SDA models analogous to Gaussian mixture models (GMM). He fits SDA mixture models, producing the following decision rule:

$$\hat{y} = \arg\min_{g\in\mathcal{G}} \sum_{h\in\mathcal{G}} C(g,h)P(Y=h) \left(\prod_{m=1}^{G} \sum_{l=1}^{c_m} w_{gml}\gamma_{gml} e^{\lambda_{gml}\psi(x,\mu_{ml})} \right),$$

where $\sum_{l=1}^{c_m} w_{ml} = 1$, and $w_{ml} > 0$. The number of components c_m are determined by cross-validation. The component weights $\{w_{gml}\}$ and the component SDA parameters $\{(\lambda_{gml}, \gamma_{gml})\}$ are estimated by an expectation-maximization (EM) algorithm, analogous to the EM-fitting of a GMM, except that the centroids are calculated only once (rather than iteratively) at the beginning using a k-medoids algorithm (Hastie et al., 2001). Simulations and a small set of experiments showed that this mixture SDA performed similarly to local SDA, but the model training for mixture SDA was much more computationally intensive.

7. Experiments

We compare eight similarity-based classification approaches: a linear and an RBF SVM using similarities as features, the P-SVM (Hochreiter and Obermayer, 2006), a local SVM (SVM-KNN) (Zhang et al., 2006) and a global SVM using the given similarities as a kernel, local SDA, k-NN, and the three weighted k-NN methods discussed in Section 5: the proposed KRR and KRI weights, and affinity weights as a control, defined by $w_i = a\psi(x, x_i)$, $i = 1, \ldots, k$, where a is a normalization constant. Results are shown in Table 3 and 4.

For algorithms that require a PSD S, we make S PSD by clip, flip or shift as discussed in Section 2, and pinv for KRR weights. The results in Table 3 are for clip; the experimental differences between clip, flip and shift, and pinv are shown in Table 4 and discussed in Section 7.4.

7.1 Data Sets

We tested the proposed classifiers on eight real data sets[6] representing a diverse set of similarities ranging from the human judgement of audio signals to sequence alignment of proteins.

The *Amazon-47* data set, created for this paper, consists of 204 books written by 47 authors. Each book listed on `amazon.com` links to the top four books that customers bought after viewing it, along with the percentage of customers who did so. We take the similarity of book A to book B to be the percentage of customers who bought B after viewing A, and the classification problem is to determine the author of the book.

The *Aural Sonar* data set is from a recent paper which investigated the human ability to distinguish different types of sonar signals by ear (Philips et al., 2006). The signals were returns from a broadband active sonar system, with 50 target-of-interest signals and 50 clutter signals. Every pair of signals was assigned a similarity score from 1 to 5 by two randomly chosen human subjects unaware of the true labels, and these scores were added to produce a 100×100 similarity matrix with integer values from 2 to 10.

The *Caltech-101* data set (Fei-Fei et al., 2004) is an object recognition benchmark data set consisting of 8677 images from 101 object categories. Similarities between images were computed using the pyramid match kernel (Grauman and Darrell, 2007) on SIFT features (Lowe, 2004). Here, the similarity is PSD.

The *Face Rec* data set consists of 945 sample faces of 139 people from the NIST Face Recognition Grand Challenge data set.[7] There are 139 classes, one for each person. Similarities for pairs of the original three-dimensional face data were computed as the cosine similarity between integral invariant signatures based on surface curves of the face (Feng et al., 2007). The original paper demonstrated comparable results to the state-of-the-art using these similarities with a 1-NN classifier.

The *Mirex07* data set was obtained from the human-rated, fine-scale audio similarity data used in the MIREX 2007 Audio Music Similarity and Retrieval[8] task. Mirex07 consists of 3090 samples, divided roughly evenly among 10 classes that correspond to different music genres. Humans judged how similar two songs are on a 0–10 scale with 0.1 increments. Each song pair was evaluated by three people, and the three similarity values were averaged. Self-similarity was assumed to be 10, the maximum similarity. The classification task is to correctly label each song with its genre.

The *Patrol* data set was collected by Driskell and McDonald (2008). Members of seven patrol units were asked to name five members of their unit; in some cases the respondents inaccurately named people who were not in their unit, including people who did not belong to any unit. Of the original 385 respondents and named people, only the ones that were named at least once were

6. The data sets along with the randomized partitions are available at `http://idl.ee.washington.edu/SimilarityLearning/`.
7. For more information, see `http://face.nist.gov/frgc/`.
8. For more information, see `http://www.music-ir.org/mirex/2007/index.php/Audio_Music_Similarity_and_Retrieval`.

kept, reducing the data set to 241 samples. The similarity between any two people a and b is $(N(a,b)+N(b,a))/2$, where $N(a,b)$ is the number of times person a names person b. Thus, this similarity ϕ has a range $\{0,0.5,1\}$. The classification problem is to estimate to which of the seven patrol units a person belongs, or to correctly place them in an eighth class that corresponds to "not in any of the units."

The *Protein* data set has sequence-alignment similarities for 213 proteins from 4 classes,[9] where class one through four contains 72, 72, 39, and 30 samples, respectively (Hofmann and Buhmann, 1997). As further discussed in the results, we define an additional similarity termed *RBF-sim* for the Protein data set: $\psi_{\text{RBF}}(x_i,x_j) = e^{-\|s(x_i)-s(x_j)\|_2}$, where $s(x)$ is the 213×1 vector of similarities with mth component $\psi(x,x_m)$.

The *Voting* data set comes from the UCI Repository (Asuncion and Newman, 2007). It is a two-class classification problem with 435 samples, where each sample is a categorical feature vector with 16 components and three possibilities for each component. We compute the value difference metric (Stanfill and Waltz, 1986) from the categorical data, which is a dissimilarity that uses the training class labels to weight different components differently so as to achieve maximum probability of class separation.

Shown in Figure 1 are the similarity matrices of all the data sets. The rows and columns are ordered by class label; on many of the data sets, particularly those with a fewer number of classes, a block-diagonal structure is visible along the class boundaries, indicated by tick marks. Note that a purely block-diagonal similarity matrix would indicate a particularly easy classification problem, as objects have nonzero similarity only to objects of the same class.

7.2 Other Experimental Details

For each data set, we randomly selected 20% of the data for testing and used the remaining 80% for training. We chose the classifier parameters such as C for the SVM, λ for the KRI and KRR weights, and k for local classifiers by 10-fold cross-validation on the training set, and then used them to classify the held out test data. This process was repeated for 20 random partitions of test and training data, and the statistical significance of the classification error was computed by a one-sided Wilcoxon signed-rank test. Multiclass implementations of the SVM classifiers used the "one-vs-one" scheme (Hsu and Lin, 2002).

Nearest neighbors for local methods were determined using symmetrized similarities[10] $\frac{1}{2}(\psi(x,x_i)+\psi(x_i,x))$. Cross-validation choices are listed in Table 2. These choices were based on recommendations and usage in previous literature, and on preliminary experiments we conducted with a larger range of cross-validation parameters on the Voting and Protein data sets.

7.3 Results

The mean and standard deviation (in parentheses) of the error across the 20 randomized test/training partitions are shown in Table 3. The bold results in each column indicate the classifier with lowest average error; also bolded are any classifiers that were not statistically significantly worse than the classifier with lowest average error.

9. The original data set has 226 samples with 9 classes. As is standard practice with this data set, we removed those classes which contain less than 7 samples.
10. Only Amazon-47 and Patrol are natively asymmetric.

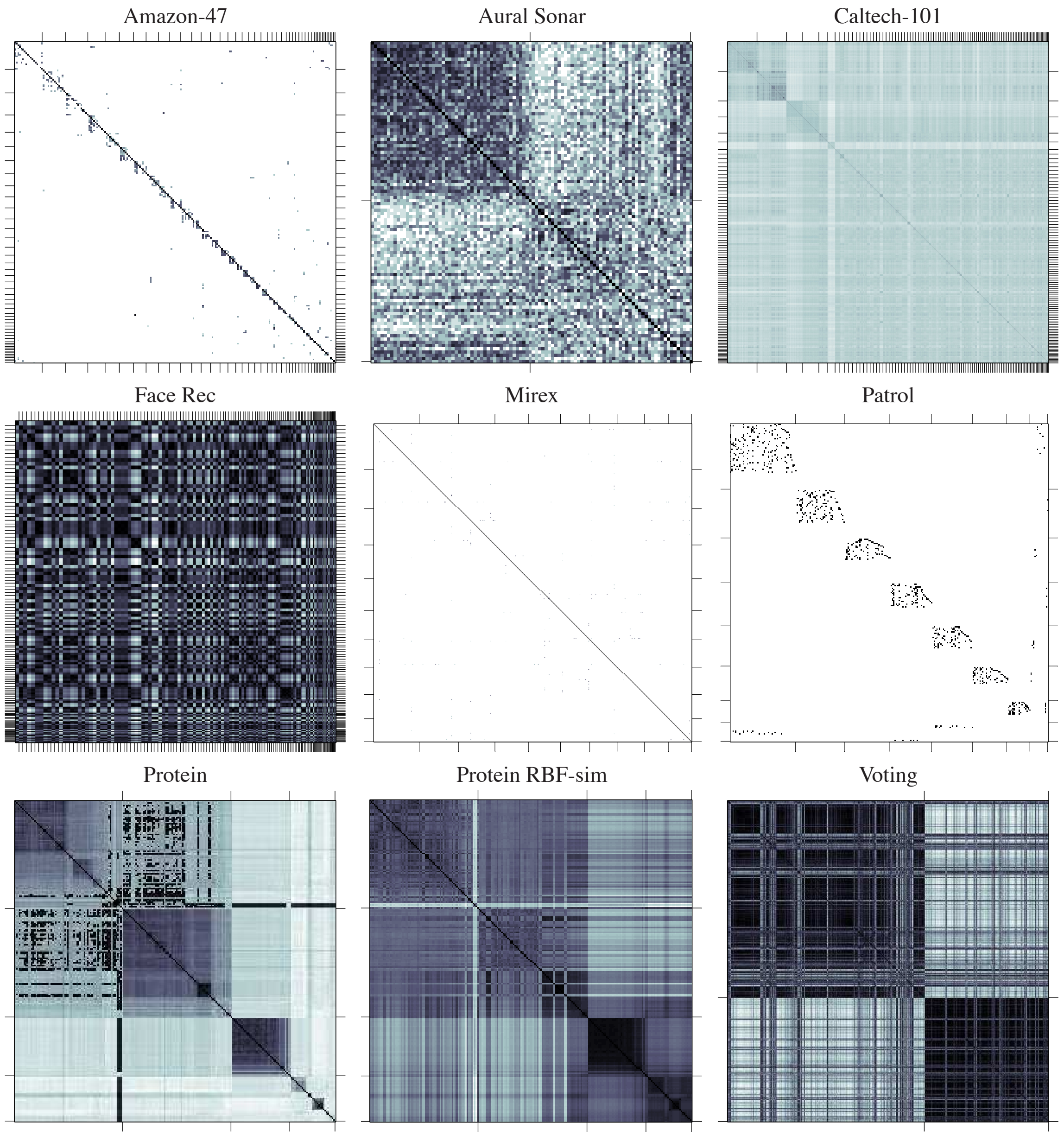


Figure 1: Similarity matrices of each data set with class divisions indicated by tick marks; black corresponds to maximum similarity and white to zero.

The similarity matrices in Figure 1 show that Aural Sonar and Voting exhibit fairly nice block-diagonal structures, indicating that these are somewhat easy classification problems. This is reflected in the relatively low errors across the board and few statistically significant differences in performance. More interesting results can be observed on the more difficult classification problems posed by the other data sets.

All local methods	k:	$1, 2, 3, \ldots, 16, 32, 64, 128$
KRR	λ:	$10^{-3}, 10^{-2}, \ldots, 10$
KRI	λ:	$10^{-6}, 10^{-5}, \ldots, 10$, and 10^{6}
PSVM	ε:	$10^{-4}, 10^{-3}, \ldots, 10$
PSVM	C:	$10^{0}, 10^{1}, \ldots, 10^{4}$
SVM-KNN	C:	$10^{-3}, 10^{-2}, \ldots, 10^{5}$
SVM (linear, clip, flip, shift)	C:	$10^{-3}, 10^{-2}, \ldots, 10^{5}$
SVM (RBF)	C:	$10^{-3}, \ldots, 10$
SVM (RBF)	γ:	$10^{-5}, 10^{-4}, \ldots, 10$

Table 2: Cross-validation Parameter Choices

The Amazon-47 data set is very sparse with at most four non-zero similarities per row. With such sparse data, one might expect a 1-NN classifier to perform well; indeed for the uniform k-NN classifier, the cross-validation chose $k = 1$ on all 20 of the randomized partitions. For all of the local classifiers, the k chosen by cross-validation on this data set was never larger than $k = 3$, and out of the 20 randomized partitions, $k = 1$ was chosen the majority of the time for all the local classifiers. In contrast, the global SVM classifiers perform poorly on this sparse data set. The Patrol data set is the next sparsest data set, and the results show a similar pattern. However, the Mirex data set is also relatively sparse, and yet the global classifiers do well, in particular the SVMs that use similarities as features. The Amazon-47 and Patrol data sets do differ from the Mirex data set in that the self-similarities are not always maximal. Whether (and how) this difference causes or correlates the relative differences in performance is an open question.

The Protein data set exhibits large differences in performance, with the statistically significantly best performances achieved by two of the three SVMs that use similarity as features. The reason that similarities on features performs so well while others do so poorly can be seen in the Protein similarity matrix in Figure 1. The first and second classes (roughly the first and second thirds of the matrix) exhibit a strong interclass similarity, and rows belonging to the same class exhibit very similar patterns, thus treating the rows of the similarity matrix as feature vectors provides good discrimination of classes. To investigate this effect, we transformed the entire data set using a radial basis function (RBF) to create a similarity based on the Euclidean distance between rows of the original similarity matrix, yielding the 213×213 Protein RBF similarity matrix. One sees from Table 3 that this transformation increases the performance of classifiers across the board, indicating that this is indeed a better measure of similarity for this data set. Furthermore, after this transformation we see a complete turnaround in performance: for Protein RBF, the SVMs that use similarities as features are among the worst performers (with P-SVM still performing decently), and the basic k-NN does better than the best classifier given the original Protein similarities.

The Caltech-101 data set is the largest data set, and with 8677 samples in 101 classes, an analysis of the structure of this similarity matrix is difficult. Here we see the most dramatic enhancement in performance (25% lower error) by using the KRR and KRI weights rather than k-NN or the affinity

	Amazon-47		Aural Sonar		Caltech-101	
k-NN	16.95	(4.85)	17.00	(7.65)	41.55	(0.95)
affinity *k*-NN	**15.00**	(4.77)	**15.00**	(6.12)	39.20	(0.86)
KRI *k*-NN (clip)	17.68	(4.75)	**14.00**	(6.82)	30.13	(0.42)
KRR *k*-NN (pinv)	**16.10**	(4.90)	15.25	(6.22)	**29.90**	(0.44)
Local SDA	16.83	(5.11)	17.75	(7.66)	41.99	(0.52)
SVM-KNN (clip)	17.56	(4.60)	**13.75**	(7.40)	36.82	(0.60)
SVM-similarities as kernel (clip)	81.34	(4.77)	**13.00**	(5.34)	33.49	(0.78)
SVM-similarities as features (linear)	76.10	(6.92)	**14.25**	(6.94)	38.18	(0.78)
SVM-similarities as features (RBF)	75.98	(7.33)	**14.25**	(7.46)	38.16	(0.75)
P-SVM	70.12	(8.82)	**14.25**	(5.97)	34.23	(0.95)
	Face Rec		Mirex		Patrol	
k-NN	**4.23**	(1.43)	61.21	(1.97)	**11.88**	(4.42)
affinity *k*-NN	**4.23**	(1.48)	61.15	(1.90)	**11.67**	(4.08)
KRI *k*-NN (clip)	**4.15**	(1.32)	61.20	(2.03)	**11.56**	(4.54)
KRR *k*-NN (pinv)	4.31	(1.86)	61.18	(1.96)	12.81	(4.62)
Local SDA	4.55	(1.67)	60.94	(1.94)	**11.77**	(4.62)
SVM-KNN (clip)	4.23	(1.25)	61.25	(1.95)	**11.98**	(4.36)
SVM-similarities as kernel (clip)	4.18	(1.25)	57.83	(2.05)	38.75	(4.81)
SVM-similarities as features (linear)	4.29	(1.36)	**55.54**	(2.52)	42.19	(5.85)
SVM-similarities as features (RBF)	**3.92**	(1.29)	**55.72**	(2.06)	40.73	(5.95)
P-SVM	**4.05**	(1.44)	63.81	(2.70)	40.42	(5.94)
	Protein		Protein RBF		Voting	
k-NN	29.88	(9.96)	**0.93**	(1.71)	5.80	(1.83)
affinity *k*-NN	30.81	(6.61)	**0.93**	(1.71)	5.86	(1.78)
KRI *k*-NN (clip)	30.35	(9.71)	**1.05**	(1.72)	**5.29**	(1.80)
KRR *k*-NN (pinv)	9.53	(5.04)	**1.05**	(1.72)	**5.52**	(1.69)
Local SDA	17.44	(6.52)	**0.93**	(1.71)	6.38	(2.07)
SVM-KNN (clip)	11.86	(5.50)	**1.16**	(1.72)	**5.23**	(2.25)
SVM-similarities as kernel (clip)	5.35	(4.60)	**1.16**	(1.72)	**4.89**	(2.05)
SVM-similarities as features (linear)	3.02	(2.76)	2.67	(2.12)	5.40	(2.03)
SVM-similarities as features (RBF)	**2.67**	(2.97)	2.44	(2.60)	5.52	(1.77)
P-SVM	**1.86**	(1.89)	**1.05**	(1.56)	**5.34**	(1.72)

Table 3: % Test misclassified averaged over 20 randomized test/training partitions.

k-NN, suggesting that there are highly correlated samples that bias the classification. In contrast, for the Amazon-47, Aural Sonar, Face Rec, Mirex, and Patrol data sets there is only a small win by using the KRI or KRR weights, or a statistically insignificant small decline in performance (we hypothesize this occurs because of overfitting due to the additional parameter λ). On Protein the KRR error is 1/3 the error of the other weighted methods; this is a consequence of using the pinv rather than other types of spectrum modification, as can be seen from Table 4. The other significant difference between the weighting methods is a roughly 10% improvement in average error on Voting by using the KRR or KRI weights. In conclusion, the use of diverse weights may not matter on some data sets, but can be very helpful on certain data sets.

SVM-KNN was proposed by Zhang et al. (2006) in part as a way to reduce the computations required to train a global SVM using similarities as a kernel, and their results showed that it performed similarly to a global SVM. That is somewhat true here, but some differences emerge. For the Amazon-47 and Patrol data sets the local methods all do well including SVM-KNN, but the global methods do poorly, including the global SVM using similarities as a kernel. On the other hand, the global SVM using similarities as a kernel is statistically significantly better than SVM-KNN on Caltech-101, even though the best performance on that data set is achieved by a local method (KRR). From this sampling of data sets, we conclude that applying the SVM locally or globally can in fact make a difference, but whether it is a positive or negative difference depends on the application.

Among the four global SVMs (including P-SVM), it is hard to draw conclusions about the performance of the one that uses similarities as a kernel versus the three that use similarities as features. In terms of statistical significance, the SVM using similarities as a kernel outperforms the others on Patrol and Caltech-101 whereas it is the worst on Amazon-47 and Protein, and there is no clear division on the remaining data sets. Lastly, the results do not show statistically significant differences between using the linear or RBF version of the SVM with similarities as features except for small differences on Face Rec and Patrol.

7.4 Clip, Flip, or Shift?

Different approaches to modifying similarities to form a kernel were discussed in Section 2.1. We experimentally compared clip, flip, and shift for the KRI weights, SVM-KNN, and SVM, and flip, clip, shift and pinv for the KRR weights on the nine data sets. Table 4 shows the five data sets for which at least one method showed statistically significantly different results depending on the choice of spectrum modification.

For KRR weights, one sees that the pinv solution is never statistically significantly worse than clip, flip, or shift, which are worse than pinv at least once. For KRI weights, the differences are negligible, but based on average error we recommend clip.

Flip takes the absolute value of the eigenvalues, which is similar to the effect of using SS^T (as discussed in Section 2.1.5), which for an SVM is equivalent to using the SVM on similarities-as-features. Thus it is not surprising that for the Protein data set, which we have seen in Table 3 works best with similarities as features, flip makes a large positive difference for SVM-KNN and SVM. One sees different effects of the spectrum modification on the local methods versus the global SVMs because for the local methods the modification is only done locally.

	KRI					KRR				
	Amazon	Mirex	Patrol	Protein	Voting	Amazon	Mirex	Patrol	Protein	Voting
clip	17.68	61.20	11.56	30.35	5.34	16.22	*61.22*	11.67	*30.35*	5.34
flip	17.56	61.17	11.67	31.28	5.34	16.22	61.12	*12.08*	*30.47*	5.29
shift	17.68	61.25	*13.23*	30.35	5.29	16.34	*61.25*	11.88	*30.35*	5.52
pinv	-	-	-	-	-	16.10	*61.18*	*12.81*	9.53	5.52

	SVM-KNN					SVM				
	Amazon	Mirex	Patrol	Protein	Voting	Amazon	Mirex	Patrol	Protein	Voting
clip	17.56	61.25	11.98	11.86	5.23	81.34	57.83	38.75	5.35	4.89
flip	17.56	61.25	11.88	1.74	5.23	*84.27*	56.34	*47.29*	1.51	4.94
shift	17.56	61.25	11.88	*30.23*	5.34	77.68	*85.29*	40.83	*23.49*	5.17

Table 4: Clip, flip, shift, and pinv comparison. Table shows % test misclassified averaged over 20 randomized test/training partitions for the five data sets that exhibit statistically significant differences between these spectrum modifications. If there are statistically significant differences for a given algorithm and a given data set, then the worst score, and scores not statistically better, are shown in italics.

8. Conclusions and Some Open Questions

Similarity-based learning is a practical learning framework for many problems in bioinformatics, computer vision, and those regarding human similarity judgement. Kernel methods can be applied in this framework, but similarity-based learning creates a richer set of challenges because the data may not be natively PSD.

In this paper we explored four different approximations of similarities: clipping, flipping, and shifting the spectrum, and in some cases a pseudoinverse solution. Experimental results show small but sometimes statistically significant differences. Based on the theoretical justification and results, we suggest practitioners clip. Flipping the spectrum does create significantly better performance for the original Protein problem because, as we noted earlier, flipping the spectrum has a similar effect to using the similarities as features, which works favorably for the original Protein problem. However, it should be easy to recognize when flip will be advantageous, modify the similarity as we did for the Protein RBF problem, and possibly achieve even better results. Concerning approximating similarities, we addressed the issue of consistent treatment of training and test samples when approximating the similarities to be PSD. Although our linear solution is consistent, we do not argue it is optimal, and consider this issue still open.

A fundamental question is whether it is more advisable to use the similarities as kernels or features. We showed that the difference for SVMs is in the regularization, and that for similarities-as-kernels generalization bounds can be proven using standard learning theory machinery. However, it is not straightforward to apply standard learning theory machinery to similarities-as-features because the normal Rademacher complexity bounds do not hold with the resulting adaptive non-independent functions. To address this, we considered splitting the training set into prototypes for similarities-as-features and a separate set to evaluate the empirical risk. Even then, we were only able to show generalization bounds if the number of prototypes grows slowly. Complementary results have been shown for similarities-as-features by Balcan et al. (2008a) and Wang et al. (2007), but further analysis would be valuable. Experimentally, it would be interesting to investigate methods using prototypes and performance as the number of training samples increases, ideally with larger data sets than those used in our experiments.

We proposed design goals of affinity and diversity for weighting nearest neighbors, and suggested two methods for constructing weights that satisfy these design goals. Experimental results on eight diverse data sets demonstrated that the proposed weighted k-NN methods can significantly reduce error compared to standard k-NN. In particular, on the largest data set Caltech-101, the proposed KRI and KRR weights provided a roughly 25% improvement over k-NN and affinity-weighted k-NN. The Caltech-101 similarities are PSD, and it may be that the KRI and KRR methods are sensitive to approximations of the matrix S. Preliminary experiments using the unmodified local S and solving KRI or KRR objective functions using a global optimizer showed an increase in performance but at the price of greatly increased computational cost. Compared to the local SVM (SVM-KNN), the proposed KRR weights were statistically significantly worse on the two-class data sets Aural Sonar and Patrol, but statistically significantly better on both of the highly multi-class data sets, Amazon-47 and Caltech-101.

Overall, the results show that local methods are effective for similarity-based learning. It is tempting from the obvious discrepancy between the performance of local and global methods on Amazon and Patrol to argue that local methods will work best for sparse similarities. However, Mirex is also relatively sparse, and the best methods are global. The Amazon and Patrol data sets are different from the other data sets in that they are the only two data sets with non-maximal self-similarities, and this issue may be the root of the discrepancy. For local methods an open question not addressed here is how to efficiently find nearest-neighbors given only similarities. Some research has been done in this area, for example Goyal et al. (2008) developed methods for fast search with logarithmic query times that depend on how "disordered" the similarity is, where they measure a disorder constant D of a set of samples as the D that ensures that if x_i is the kth most similar sample to x, and x_j is the qth most similar sample to x, then x is among the $D(q+k)$ most similar samples to x_j.

Lastly, we note that similarity-based learning can be considered a special case of graph-based learning (see, for example, Zhu, 2005), where the graph is fully-connected. However, most graph-based learning literature addresses the problem of semi-supervised learning, while the similarity-based learning algorithms discussed in this paper are mainly for supervised learning. We have seen no cross-referential literature between these two fields, and it remains an open question which techniques developed for one problem will be useful for the other problem and vice versa.

Acknowledgments

This work was funded by the United States Office of Naval Research and Intel.

Appendix A.

Proof of Proposition 1: Recall the eigenvalue decomposition $S_{\text{clip}} = U^T \Lambda_{\text{clip}} U$. After removing the zero eigenvalues in Λ_{clip} and their corresponding eigenvectors in U and U^T, one can express $S_{\text{clip}} = \check{U}^T \check{\Lambda}_{\text{clip}} \check{U}$, where $\check{\Lambda}_{\text{clip}}$ is an $m \times m$ diagonal matrix with m the number of nonzero eigenvalues and $\check{U}$ an $m \times n$ matrix satisfying $\check{U}\check{U}^T = I$. The vector representation of the training samples implicitly used via S_{clip} is $X = \check{\Lambda}_{\text{clip}}^{1/2} \check{U}$. Given test similarity vector s, the least-squares solution to the equation $X^T x = s$ is $x = \left(XX^T\right)^{-1} Xs$. Let $\tilde{s}$ be the vector of the inner products between the embedded test sample x and the embedded training samples X, then

$$\tilde{s} = X^T x = X^T \left(XX^T\right)^{-1} Xs = \check{U}^T \check{U} s = U^T M_{\text{clip}} U s = P_{\text{clip}} s. \qquad \blacksquare$$

The proofs of the generalization bounds of Theorem 1 and Theorem 2 rely on bounding the Rademacher complexity of the function classes F_S and F_I, respectively. We provide the definition of Rademacher complexity here for convenience.

Definition 1 (Rademacher Complexity) *Suppose $\mathcal{X} = \{X_1, X_2, \ldots, X_n\}$ are samples drawn independently from a distribution on Ω, and let F be a class of functions mapping from Ω to $\mathbb{R}$. Then, the Rademacher complexity of F is*

$$\mathcal{R}_{\mathcal{X}}(F) = E_{\sigma,\mathcal{X}} \left(\sup_{f \in F} \left| \frac{2}{n} \sum_{i=1}^{n} \sigma_i f(X_i) \right| \right),$$

where $\sigma = \{\sigma_1, \sigma_2, \ldots, \sigma_n\}$ is a set of independent uniform $\{\pm 1\}$-valued random variables[11] *such that $P(\sigma_i = 1) = P(\sigma_i = -1) = 1/2$ for all i.*

The following lemma establishes that for a class of bounded functions F, the generalization error for any $f \in F$ is bounded above by a function of $\hat{R}_{\mathcal{D}}(f, L)$ and $\mathcal{R}_{\mathcal{D}}(F)$.

Lemma 1 (Bartlett and Mendelson, 2002, Theorem 7) *Suppose (X,Y) and the elements of $\mathcal{D}$ are drawn i.i.d. from a distribution on $\Omega \times \{\pm 1\}$. Let F be a class of bounded real-valued functions defined on Ω such that $\sup_{f \in F} \sup_{x \in \Omega} |f(x)| < \infty$. Suppose $L : \mathbb{R} \to [0,1]$ is Lipschitz with constant C and satisfies $L(a) \geq I_{\{a \leq 0\}}$. Then with probability at least $1 - \delta$ with respect to $\mathcal{D}$, every function in F satisfies*

$$P(Yf(X) \leq 0) \leq \hat{R}_{\mathcal{D}}(f, L) + 2C\mathcal{R}_{\mathcal{D}}(F) + \sqrt{\frac{\ln(2/\delta)}{2n}}.$$

For the proof of Theorem 1, we also require the following bound on the Rademacher complexity of kernel methods.

Lemma 2 (Bartlett and Mendelson, 2002, Lemma 22) *Suppose the elements of $\mathcal{D}$ are drawn i.i.d. from a distribution on $\Omega \times \{\pm 1\}$. Let F_K denote the set of functions $\left\{ f(x) = \sum_i \alpha_i K(x, X_i) \,\middle|\, \sum_{i,j} \alpha_i \alpha_j K(X_i, X_j) \leq \beta^2 \right\}$, then by Jensen's inequality,*

$$\mathcal{R}_{\mathcal{D}}(F_K) \leq 2\beta \sqrt{\frac{E\left(K(X,X)\right)}{n}}.$$

11. Such random variables are called *Rademacher random variables* and their distribution is called the *Rademacher distribution*.

Proof of Theorem 1: Theorem 1 is an application of Lemma 1 and 2 for the function class $F_S = \{f(s) = w^T s \,|\, w^T S w \leq \beta^2\}$. By replacing the kernel function K in Lemma 2 with ψ, and noticing $E(\psi(X,X)) \leq \kappa^2$ since $\psi(a,a) \leq \kappa^2$ for all $a \in \Omega$, we have $\mathcal{R}_{\mathcal{D}}(F_S) \leq 2\beta\kappa\sqrt{1/n}$. It can be verified that the function class F_S is bounded as $|f(s)| \leq \beta\kappa$ for all $f \in F_S$, and thus we can apply Lemma 1. Noting that L_t is Lipschitz with $C = 1$ completes the proof. ■

Proof of Theorem 2: Recall that $\tilde{s} = \begin{bmatrix}\psi(x,\tilde{x}_1) & \dots & \psi(x,\tilde{x}_m)\end{bmatrix}^T$ where $\{(\tilde{x}_1,\tilde{y}_1),\dots,(\tilde{x}_m,\tilde{y}_m)\} \subseteq \mathcal{D}$ is a subset of the training data and $\widetilde{\mathcal{D}} = \mathcal{D}\backslash\{(\tilde{x}_1,\tilde{y}_1),\dots,(\tilde{x}_m,\tilde{y}_m)\}$ is the remaining training set. It is tempting to apply Lemma 2 with the linear kernel, but in this case, this does not satisfy the definition of the function class $F_I = \{f(\tilde{s}) = w^T\tilde{s} \,|\, w^T w \leq \beta^2\}$ defined on these prototypes. The following bound on the Rademacher complexity mirrors that of Bartlett and Mendelson (2002, Lemma 22), but requires an important modification noted below:

$$\begin{aligned}
\mathcal{R}_{\widetilde{\mathcal{D}}}(F_I) &= E_{\widetilde{\mathcal{D}},\sigma}\left(\sup_{f\in F_I}\left|\frac{2}{n}\sum_{i=1}^{n-m}\sigma_i f(\tilde{s}_i)\right|\right) \\
&\leq \frac{2}{n}E_{\widetilde{\mathcal{D}},\sigma}\left(\sup_{\|w\|_2\leq\beta}\left|w^T\left(\sum_{i=1}^{n-m}\sigma_i\tilde{s}_i\right)\right|\right) \\
&\overset{(a)}{\leq} \frac{2}{n}E_{\widetilde{\mathcal{D}},\sigma}\left(\beta\left\|\sum_{i=1}^{n-m}\sigma_i\tilde{s}_i\right\|_2\right) \\
&= \frac{2\beta}{n}E_{\widetilde{\mathcal{D}},\sigma}\left(\sqrt{\sum_{i,j}\sigma_i\sigma_j\tilde{s}_i^T\tilde{s}_j}\right) \\
&\overset{(b)}{\leq} \frac{2\beta}{n}\sqrt{\sum_{i,j}E_{\widetilde{\mathcal{D}},\sigma}\left(\sigma_i\sigma_j\tilde{s}_i^T\tilde{s}_j\right)} \\
&= \frac{2\beta}{n}\sqrt{\sum_{i=1}^{n-m}E_{\widetilde{\mathcal{D}}}\left(\tilde{s}_i^T\tilde{s}_i\right)} \\
&= \frac{2\beta}{n}\sqrt{\sum_{i=1}^{n-m}\sum_{j=1}^{m}E_{\widetilde{\mathcal{D}}}\psi^2(x_i,\tilde{x}_j)} \\
&\leq 2\beta\kappa^2\sqrt{\frac{m}{n}-\frac{m^2}{n^2}} \\
&\leq 2\beta\kappa^2\sqrt{\frac{m}{n}}
\end{aligned}$$

where (a) follows from the Cauchy-Schwarz inequality[12] and (b) follows from Jensen's inequality.

It can be verified that the function class is bounded as $|f(\tilde{s})| \leq \beta\kappa^2\sqrt{m}$ for all $f \in F_I$, and thus we can apply Lemma 1. As before, noting that L_t is Lipschitz with $C = 1$ completes the proof. ■

The proof of Theorem 2 illustrates why using similarities as features has a poorer guarantee on the generalization than using similarities as a kernel. Specifically, the function class corresponding

12. Note that in the proof of Theorem 1 and in Bartlett and Mendelson (2002, Lemma 22) the Cauchy-Schwartz inequality is applied in the RKHS space whereas here it is applied in $\mathbb{R}^m$.

to regularization on $w^T w$ is too large. Of course, this flexibility can be mitigated by using only a set of m prototypes whose size grows as $o(n)$, which can be seen as an additional form of capacity control.

References

S. F. Altschul, W. Gish, W. Miller, E. W. Myers, and D. J. Lipman. Basic local alignment search tool. *J. Molecular Biology*, 215(3):403–410, Oct. 1990.

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

M.-F. Balcan, A. Blum, and N. Srebro. A theory of learning with similarity functions. *Machine Learning*, 72(1–2):89–112, Aug. 2008a.

M.-F. Balcan, A. Blum, and N. Srebro. Improved guarantees for learning via similarity functions. In *Proc. Ann. Conf. Learning Theory*, 2008b.

P. L. Bartlett and S. Mendelson. Rademacher and Gaussian complexities: Risk bounds and structural results. *J. Machine Learning Research*, 3:463–482, Nov. 2002.

S. Belongie, J. Malik, and J. Puzicha. Shape matching and object recognition using shape contexts. *IEEE Trans. Pattern Anal. and Machine Intel.*, 24(4):509–522, April 2002.

I. Borg and P. J. F. Groenen. *Modern Multidimensional Scaling: Theory and Applications*. Springer, New York, 2nd edition, 2005.

L. Cazzanti. *Generative Models for Similarity-based Classification*. PhD thesis, Dept. of Electrical Engineerng, University of Washington, 2007.

L. Cazzanti and M. R. Gupta. Local similarity discriminant analysis. In *Proc. Intl. Conf. Machine Learning*, 2007.

L. Cazzanti, M. R. Gupta, and A. J. Koppal. Generative models for similarity-based classification. *Pattern Recognition*, 41(7):2289–2297, July 2008.

J. Chen and J. Ye. Training SVM with indefinite kernels. In *Proc. Intl. Conf. Machine Learning*, 2008.

N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines*. Cambridge University Press, Cambridge, UK, 2000.

J. E. Driskell and T. McDonald. Identification of incomplete networks. Technical report, Florida Maxima Corporation, 2008.

R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. John Wiley & Sons, New York, 2nd edition, 2001.

L. Fei-Fei, R. Fergus, and P. Perona. Learning generative visual models from few training examples: An incremental Bayesian approach tested on 101 object categories. In *Proc. IEEE Computer Soc. Conf. Computer Vision and Pattern Recognition*, 2004.

S. Feng, H. Krim, and I. A. Kogan. 3D face recognition using Euclidean integral invariants signature. In *Proc. IEEE Workshop Statistical Signal Processing*, 2007.

M. P. Friedlander and M. R. Gupta. On minimizing distortion and relative entropy. *IEEE Trans. Information Theory*, 52(1):238–245, Jan. 2006.

I. Gati and A. Tversky. Representations of qualitative and quantitative dimensions. *J. Experimental Psychology: Human Perception & Performance*, 8(2):325–340, April 1982.

I. Gati and A. Tversky. Weighting common and distinctive features in perceptual and conceptual judgments. *Cognitive Psychology*, 16(3):341–370, July 1984.

R. L. Goldstone and A. Kersten. *Comprehensive Handbook of Psychology*, volume 4, chapter 22: Concepts and Categorization, pages 599–621. Wiley, New Jersey, 2003.

N. Goyal, Y. Lifshits, and H. Schütze. Disorder inequality: A combinatorial approach to nearest neighbor search. In *Proc. ACM Symposium Web Search and Data Mining*, 2008.

T. Graepel, R. Herbrich, P. Bollmann-Sdorra, and K. Obermayer. Classification on pairwise proximity data. In *Advances in Neural Information Processing Systems*, 1998.

T. Graepel, R. Herbrich, B. Schölkopf, A. Smola, P. Bartlett, K.-R. Müller, K. Obermayer, and R. Williamson. Classification on proximity data with LP–machines. In *Proc. Intl. Conf. Artificial Neural Networks*, 1999.

K. Grauman and T. Darrell. The pyramid match kernel: Efficient learning with sets of features. *J. Machine Learning Research*, 8:725–760, April 2007.

M. R. Gupta, R. M. Gray, and R. A. Olshen. Nonparametric supervised learning by linear interpolation with maximum entropy. *IEEE Trans. Pattern Anal. and Machine Intel.*, 28(5):766–781, May 2006.

B. Haasdonk. Feature space interpretation of SVMs with indefinite kernels. *IEEE Trans. Pattern Anal. and Machine Intel.*, 27(4):482–492, April 2005.

T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction.* Springer, New York, 2001.

N. J. Higham. Computing a nearest symmetric positive semidefinite matrix. *Linear Algebra and its Applications*, 103:103–118, May 1988.

S. Hochreiter and K. Obermayer. Support vector machines for dyadic data. *Neural Computation*, 18(6):1472–1510, June 2006.

T. Hofmann and J. M. Buhmann. Pairwise data clustering by deterministic annealing. *IEEE Trans. Pattern Anal. and Machine Intel.*, 19(1):1–14, Jan. 1997.

C.-W. Hsu and C.-J. Lin. A comparison of methods for multiclass support vector machines. *IEEE Trans. Neural Networks*, 13(2):415–425, March 2002.

E. T. Jaynes. On the rationale for maximum entropy methods. *Proc. IEEE*, 70(9):939–952, Sept. 1982.

T. Knebel, S. Hochreiter, and K. Obermayer. An SMO algorithm for the potential support vector-machine. *Neural Computation*, 20(1):271–287, Jan. 2008.

J. Laub and K.-R. Müller. Feature discovery in non-metric pairwise data. *J. Machine Learning Research*, 5:801–808, July 2004.

J. Laub, V. Roth, J. M. Buhmann, and K.-R. Müller. On the information and representation of non-Euclidean pairwise data. *Pattern Recognition*, 39(10):1815–1826, Oct. 2006.

L. Liao and W. S. Noble. Combining pairwise sequence similarity and support vector machines for detecting remote protein evolutionary and structural relationships. *J. Computational Biology*, 10 (6):857–868, 2003.

H.-T. Lin and C.-J. Lin. A study on sigmoid kernels for SVM and the training of non-PSD kernels by SMO-type methods. Technical report, National Taiwan University, March 2003.

D. J. Lipman and W. R. Pearson. Rapid and sensitive protein similarity searches. *Science*, 227 (4693):1435–1441, March 1985.

D. G. Lowe. Distinctive image features from scale-invariant keypoints. *Intl. J. Computer Vision*, 60 (2):91–110, 2004.

R. Luss and A. d'Aspremont. Support vector machine classification with indefinite kernels. In *Advances in Neural Information Processing Systems*, 2007.

C. S. Ong, X. Mary, S. Canu, and A. J. Smola. Learning with non-positive kernels. In *Proc. Intl. Conf. Machine Learning*, 2004.

E. Pekalska and R. P. W. Duin. Dissimilarity representations allow for building good classifiers. *Pattern Recognition Letters*, 23(8):943–956, June 2002.

E. Pekalska, P. Paclík, and R. P. W. Duin. A generalized kernel approach to dissimilarity-based classification. *J. Machine Learning Research*, 2:175–211, Dec. 2001.

S. Philips, J. Pitton, and L. Atlas. Perceptual feature identification for active sonar echoes. In *Proc. IEEE OCEANS Conf.*, 2006.

J. C. Platt. Using analytic QP and sparseness to speed training of support vector machines. In *Advances in Neural Information Processing Systems*, 1998.

R. M. Rifkin. *Everything Old is New Again: A Fresh Look at Historical Approaches in Machine Learning*. PhD thesis, MIT, 2002.

V. Roth, J. Laub, M. Kawanabe, and J. M. Buhmann. Optimal cluster preserving embedding of nonmetric proximity data. *IEEE Trans. Pattern Anal. and Machine Intel.*, 25(12):1540–1551, Dec. 2003.

Y. Rubner, C. Tomasi, and L. J. Guibas. The earth mover's distance as a metric for image retrieval. *Intl. J. Computer Vision*, 40(2):99–121, Nov. 2000.

S. Santini and R. Jain. Similarity measures. *IEEE Trans. Pattern Anal. and Machine Intel.*, 21(9): 871–883, Sept. 1999.

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

T. F. Smith and M. S. Waterman. Identification of common molecular subsequences. *J. Molecular Biology*, 147(1):195–197, March 1981.

C. Stanfill and D. Waltz. Toward memory-based reasoning. *Comm. ACM*, 29(12):1213–1228, Dec. 1986.

R. Tibshirani. Regression shrinkage and selection via the lasso. *J. Royal Statistical Society, Series B (Statistical Methodology)*, 58(1):267–288, 1996.

A. Tversky. Features of similarity. *Psychological Review*, 84(2):327–352, July 1977.

A. Tversky and I. Gati. Similarity, separability, and the triangle inequality. *Psychological Review*, 89(2):123–154, March 1982.

L. Wang, C. Yang, and J. Feng. On learning with dissimilarity functions. In *Proc. Intl. Conf. Machine Learning*, 2007.

G. Wu, E. Y. Chang, and Z. Zhang. An analysis of transformation on non-positive semidefinite similarity matrix for kernel machines. Technical report, University of California, Santa Barbara, March 2005.

H. Zhang, A. C. Berg, M. Maire, and J. Malik. SVM-KNN: Discriminative nearest neighbor classification for visual category recognition. In *Proc. IEEE Computer Soc. Conf. Computer Vision and Pattern Recognition*, 2006.

X. Zhu. *Semi-supervised Learning with Graphs*. PhD thesis, School of Computer Science, Carnegie Mellon University, 2005.

H. Zou and T. Hastie. Regularization and variable selection via the elastic net. *J. Royal Statistical Society: Series B (Statistical Methodology)*, 67(2):301–320, April 2005.

Journal of Machine Learning Research 10 (2009) 777-801 Submitted 6/08; Revised 11/08; Published 3/09

Sparse Online Learning via Truncated Gradient

John Langford JL@YAHOO-INC.COM
Yahoo! Research
New York, NY, USA

Lihong Li LIHONG@CS.RUTGERS.EDU
Department of Computer Science
Rutgers University
Piscataway, NJ, USA

Tong Zhang[*] TONGZ@RCI.RUTGERS.EDU
Department of Statistics
Rutgers University
Piscataway, NJ, USA

Editor: Manfred Warmuth

Abstract

We propose a general method called *truncated gradient* to induce sparsity in the weights of online-learning algorithms with convex loss functions. This method has several essential properties:

1. The degree of sparsity is continuous—a parameter controls the rate of sparsification from no sparsification to total sparsification.
2. The approach is theoretically motivated, and an instance of it can be regarded as an online counterpart of the popular L_1-regularization method in the batch setting. We prove that small rates of sparsification result in only small additional regret with respect to typical online-learning guarantees.
3. The approach works well empirically.

We apply the approach to several data sets and find for data sets with large numbers of features, substantial sparsity is discoverable.

Keywords: truncated gradient, stochastic gradient descent, online learning, sparsity, regularization, Lasso

1. Introduction

We are concerned with machine learning over large data sets. As an example, the largest data set we use here has over 10^7 sparse examples and 10^9 features using about 10^{11} bytes. In this setting, many common approaches fail, simply because they cannot load the data set into memory or they are not sufficiently efficient. There are roughly two classes of approaches which can work:

1. Parallelize a batch-learning algorithm over many machines (e.g., Chu et al., 2008).

2. Stream the examples to an online-learning algorithm (e.g., Littlestone, 1988; Littlestone et al., 1995; Cesa-Bianchi et al., 1996; Kivinen and Warmuth, 1997).

*. Partially supported by NSF grant DMS-0706805.

This paper focuses on the second approach.

Typical online-learning algorithms have at least one weight for every feature, which is too much in some applications for a couple reasons:

1. Space constraints. If the state of the online-learning algorithm overflows RAM it can not efficiently run. A similar problem occurs if the state overflows the L2 cache.

2. Test-time constraints on computation. Substantially reducing the number of features can yield substantial improvements in the computational time required to evaluate a new sample.

This paper addresses the problem of inducing sparsity in learned weights while using an online-learning algorithm. There are several ways to do this wrong for our problem. For example:

1. Simply adding L_1-regularization to the gradient of an online weight update doesn't work because gradients don't induce sparsity. The essential difficulty is that a gradient update has the form $a+b$ where a and b are two floats. Very few float pairs add to 0 (or any other default value) so there is little reason to expect a gradient update to accidentally produce sparsity.

2. Simply rounding weights to 0 is problematic because a weight may be small due to being useless or small because it has been updated only once (either at the beginning of training or because the set of features appearing is also sparse). Rounding techniques can also play havoc with standard online-learning guarantees.

3. Black-box wrapper approaches which eliminate features and test the impact of the elimination are not efficient enough. These approaches typically run an algorithm many times which is particularly undesirable with large data sets.

1.1 What Others Do

In the literature, the Lasso algorithm (Tibshirani, 1996) is commonly used to achieve sparsity for linear regression using L_1-regularization. This algorithm does not work automatically in an online fashion. There are two formulations of L_1-regularization. Consider a loss function $L(w,z_i)$ which is convex in w, where $z_i=(x_i,y_i)$ is an input/output pair. One is the *convex constraint formulation*

$$\hat{w} = \arg\min_w \sum_{i=1}^n L(w,z_i) \quad \text{subject to } \|w\|_1 \le s, \tag{1}$$

where s is a tunable parameter. The other is the *soft regularization formulation*, where

$$\hat{w} = \arg\min_w \sum_{i=1}^n L(w,z_i) + g\|w\|_1. \tag{2}$$

With appropriately chosen g, the two formulations are equivalent. The convex constraint formulation has a simple online version using the projection idea of Zinkevich (2003), which requires the projection of weight w into an L_1-ball at every online step. This operation is difficult to implement efficiently for large-scale data with many features even if all examples have sparse features although recent progress was made (Duchi et al., 2008) to reduce the *amortized* time complexity to $O(k\log d)$, where k is the number of nonzero entries in x_i, and d is the total number of features (i.e., the dimension of x_i). In contrast, the soft-regularization method is efficient for a batch setting (Lee et al., 2007) so we pursue it here in an online setting where we develop an algorithm whose complexity is linear in k but independent of d; these algorithms are therefore more efficient in problems where d is prohibitively large.

More recently, Duchi and Singer (2008) propose a framework for empirical risk minimization with regularization called *Forward Looking Subgradients*, or FOLOS in short. The basic idea is to solve a regularized optimization problem after every gradient-descent step. This family of algorithms allow general convex regularization function, and reproduce a special case of the truncated gradient algorithm we will introduce in Section 3.3 (with θ set to ∞) when L_1-regularization is used.

The Forgetron algorithm (Dekel et al., 2006) is an online-learning algorithm that manages memory use. It operates by decaying the weights on previous examples and then rounding these weights to zero when they become small. The Forgetron is stated for kernelized online algorithms, while we are concerned with the simpler linear setting. When applied to a linear kernel, the Forgetron is not computationally or space competitive with approaches operating directly on feature weights.

A different, Bayesian approach to learning sparse linear classifiers is taken by Balakrishnan and Madigan (2008). Specifically, their algorithms approximate the posterior by a Gaussian distribution, and hence need to store second-order covariance statistics which require $O(d^2)$ space and time per online step. In contrast, our approach is much more efficient, requiring only $O(d)$ space and $O(k)$ time at every online step.

After completing the paper, we learned that Carpenter (2008) independently developed an algorithm similar to ours.

1.2 What We Do

We pursue an algorithmic strategy which can be understood as an online version of an efficient L_1 loss optimization approach (Lee et al., 2007). At a high level, our approach works with the soft-regularization formulation (2) and decays the weight to a default value after every online stochastic gradient step. This simple approach enjoys minimal time complexity (which is linear in k and independent of d) as well as strong performance guarantee, as discussed in Sections 3 and 5. For instance, the algorithm never performs much worse than a standard online-learning algorithm, and the additional loss due to sparsification is controlled continuously with a single real-valued parameter. The theory gives a family of algorithms with convex loss functions for inducing sparsity—one per online-learning algorithm. We instantiate this for square loss and show how an efficient implementation can take advantage of sparse examples in Section 4. In addition to the L_1-regularization formulation (2), the family of algorithms we consider also include some non-convex sparsification techniques.

As mentioned in the introduction, we are mainly interested in sparse online methods for large scale problems with sparse features. For such problems, our algorithm should satisfy the following requirements:

- The algorithm should be computationally efficient: the number of operations per online step should be linear in the number of nonzero features, and independent of the total number of features.

- The algorithm should be memory efficient: it needs to maintain a list of active features, and can insert (when the corresponding weight becomes nonzero) and delete (when the corresponding weight becomes zero) features dynamically.

Our solution, referred to as *truncated gradient*, is a simple modification of the standard stochastic gradient rule. It is defined in (6) as an improvement over simpler ideas such as rounding and sub-gradient method with L_1 -regularization. The implementation details, showing our methods satisfy the above requirements, are provided in Section 5.

Theoretical results stating how much sparsity is achieved using this method generally require additional assumptions which may or may not be met in practice. Consequently, we rely on experiments in Section 6 to show our method achieves good sparsity practice. We compare our approach to a few others, including L_1 -regularization on small data, as well as online rounding of coefficients to zero.

2. Online Learning with Stochastic Gradient Descent

In the setting of standard online learning, we are interested in sequential prediction problems where repeatedly from $i = 1, 2, \ldots$:

1. An unlabeled example x_i arrives.
2. We make a prediction based on existing weights $w_i \in R^d$.
3. We observe y_i, let $z_i = (x_i, y_i)$, and incur some known loss $L(w_i, z_i)$ that is convex in parameter w_i.
4. We update weights according to some rule: $w_{i+1} \leftarrow f(w_i)$.

We want to come up with an update rule f, which allows us to bound the sum of losses

$$\sum_{i=1}^{t} L(w_i, z_i)$$

as well as achieving sparsity. For this purpose, we start with the standard stochastic gradient descent (SGD) rule, which is of the form:

$$f(w_i) = w_i - \eta \nabla_1 L(w_i, z_i), \tag{3}$$

where $\nabla_1 L(a, b)$ is a sub-gradient of $L(a, b)$ with respect to the first variable a. The parameter $\eta > 0$ is often referred to as the learning rate. In our analysis, we only consider constant learning rate with fixed $\eta > 0$ for simplicity. In theory, it might be desirable to have a decaying learning rate η_i which becomes smaller when i increases to get the so called *no-regret bound* without knowing T in advance. However, if T is known in advance, one can select a constant η accordingly so the regret vanishes as $T \to \infty$. Since our focus is on sparsity, not how to adapt learning rate, for clarity, we use a constant learning rate in the analysis because it leads to simpler bounds.

The above method has been widely used in online learning (Littlestone et al., 1995; Cesa-Bianchi et al., 1996). Moreover, it is argued to be efficient even for solving batch problems where we repeatedly run the online algorithm over training data multiple times. For example, the idea has been successfully applied to solve large-scale standard SVM formulations (Shalev-Shwartz et al., 2007; Zhang, 2004). In the scenario outlined in the introduction, online-learning methods are more suitable than some traditional batch-learning methods.

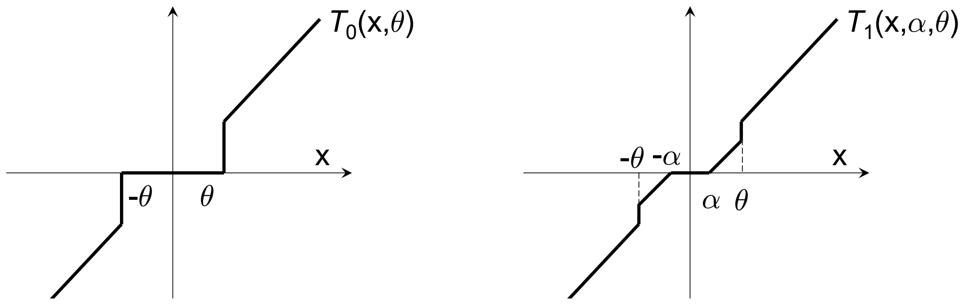


Figure 1: Plots for the truncation functions, T_0 and T_1, which are defined in the text.

However, a main drawback of (3) is that it does not achieve sparsity, which we address in this paper. In the literature, the stochastic-gradient descent rule is often referred to as gradient descent (GD). There are other variants, such as exponentiated gradient descent (EG). Since our focus in this paper is sparsity, not GD versus EG, we shall only consider modifications of (3) for simplicity.

3. Sparse Online Learning

In this section, we examine several methods for achieving sparsity in online learning. The first idea is simple coefficient rounding, which is the most natural method. We will then consider another method which is the online counterpart of L_1-regularization in batch learning. Finally, we combine such two ideas and introduce truncated gradient. As we shall see, all these ideas are closely related.

3.1 Simple Coefficient Rounding

In order to achieve sparsity, the most natural method is to round small coefficients (that are no larger than a threshold $\theta > 0$) to zero after every K online steps. That is, if i/K is not an integer, we use the standard GD rule in (3); if i/K is an integer, we modify the rule as:

$$f(w_i) = T_0(w_i - \eta \nabla_1 L(w_i, z_i), \theta), \tag{4}$$

where for a vector $v = [v_1, \ldots, v_d] \in R^d$, and a scalar $\theta \geq 0$, $T_0(v, \theta) = [T_0(v_1, \theta), \ldots, T_0(v_d, \theta)]$, with T_0 defined by (cf., Figure 1)

$$T_0(v_j, \theta) = \begin{cases} 0 & \text{if } |v_j| \leq \theta \\ v_j & \text{otherwise} \end{cases}.$$

That is, we first apply the standard stochastic gradient descent rule, and then round small coefficients to zero.

In general, we should not take $K = 1$, especially when η is small, since each step modifies w_i by only a small amount. If a coefficient is zero, it remains small after one online update, and the rounding operation pulls it back to zero. Consequently, rounding can be done only after every K steps (with a reasonably large K); in this case, nonzero coefficients have sufficient time to go above the threshold θ. However, if K is too large, then in the training stage, we will need to keep many more nonzero features in the intermediate steps before they are rounded to zero. In the extreme case, we may simply round the coefficients in the end, which does not solve the storage problem in the training phase. The sensitivity in choosing appropriate K is a main drawback of this method; another drawback is the lack of theoretical guarantee for its online performance.

3.2 A Sub-gradient Algorithm for L_1-Regularization

In our experiments, we combine rounding-in-the-end-of-training with a simple online sub-gradient method for L_1-regularization with a regularization parameter $g > 0$:

$$f(w_i) = w_i - \eta \nabla_1 L(w_i, z_i) - \eta g \operatorname{sgn}(w_i), \tag{5}$$

where for a vector $v = [v_1, \ldots, v_d]$, $\operatorname{sgn}(v) = [\operatorname{sgn}(v_1), \ldots, \operatorname{sgn}(v_d)]$, and $\operatorname{sgn}(v_j) = 1$ when $v_j > 0$, $\operatorname{sgn}(v_j) = -1$ when $v_j < 0$, and $\operatorname{sgn}(v_j) = 0$ when $v_j = 0$. In the experiments, the online method (5) plus rounding in the end is used as a simple baseline. This method does not produce sparse weights online. Therefore it does not handle large-scale problems for which we cannot keep all features in memory.

3.3 Truncated Gradient

In order to obtain an online version of the simple rounding rule in (4), we observe that the direct rounding to zero is too aggressive. A less aggressive version is to shrink the coefficient to zero by a smaller amount. We call this idea truncated gradient.

The amount of shrinkage is measured by a *gravity* parameter $g_i > 0$:

$$f(w_i) = T_1(w_i - \eta \nabla_1 L(w_i, z_i), \eta g_i, \theta), \tag{6}$$

where for a vector $v = [v_1, \ldots, v_d] \in R^d$, and a scalar $g \geq 0$, $T_1(v, \alpha, \theta) = [T_1(v_1, \alpha, \theta), \ldots, T_1(v_d, \alpha, \theta)]$, with T_1 defined by (cf., Figure 1)

$$T_1(v_j, \alpha, \theta) = \begin{cases} \max(0, v_j - \alpha) & \text{if } v_j \in [0, \theta] \\ \min(0, v_j + \alpha) & \text{if } v_j \in [-\theta, 0] \\ v_j & \text{otherwise} \end{cases}.$$

Again, the truncation can be performed every K online steps. That is, if i/K is not an integer, we let $g_i = 0$; if i/K is an integer, we let $g_i = Kg$ for a gravity parameter $g > 0$. This particular choice is equivalent to (4) when we set g such that $\eta K g \geq \theta$. This requires a large g when η is small. In practice, one should set a small, fixed g, as implied by our regret bound developed later.

In general, the larger the parameters g and θ are, the more sparsity is incurred. Due to the extra truncation T_1, this method can lead to sparse solutions, which is confirmed in our experiments described later. In those experiments, the degree of sparsity discovered varies with the problem.

A special case, which we will try in the experiment, is to let $g = \theta$ in (6). In this case, we can use only one parameter g to control sparsity. Since $\eta K g \ll \theta$ when ηK is small, the truncation operation is less aggressive than the rounding in (4). At first sight, the procedure appears to be an ad-hoc way to fix (4). However, we can establish a regret bound for this method, showing it is theoretically sound.

Setting $\theta = \infty$ yields another important special case of (6), which becomes

$$f(w_i) = T(w_i - \eta \nabla_1 L(w_i, z_i), g_i \eta), \tag{7}$$

where for a vector $v = [v_1, \dots, v_d] \in R^d$, and a scalar $g \geq 0$, $T(v, \alpha) = [T(v_1, \alpha), \dots, T(v_d, \alpha)]$, with

$$T(v_j, \alpha) = \begin{cases} \max(0, v_j - \alpha) & \text{if } v_j > 0 \\ \min(0, v_j + \alpha) & \text{otherwise} \end{cases}.$$

The method is a modification of the standard sub-gradient descent method with L_1-regularization given in (5). The parameter $g_i \geq 0$ controls the sparsity that can be achieved with the algorithm. Note when $g_i = 0$, the update rule is identical to the standard stochastic gradient descent rule. In general, we may perform a truncation every K steps. That is, if i/K is not an integer, we let $g_i = 0$; if i/K is an integer, we let $g_i = Kg$ for a gravity parameter $g > 0$. The reason for doing so (instead of a constant g) is that we can perform a more aggressive truncation with gravity parameter Kg after each K steps. This may lead to better sparsity. An alternative way to derive a procedure similar to (7) is through an application of convex hull projection idea of Zinkevich (2003) to the L_1-regularized loss, as in (5). However, instead of working with the original feature set, we need to consider a $2d$-dimensional duplicated feature set $[x_i, -x_i]$, with the non-negativity constraint $w^j \geq 0$ for each component of j (w will also have dimension $2d$ in this case). The resulting method is similar to ours, with a similar theoretical guarantee as in Theorem 3.1. The proof presented in this paper is more specialized to truncated gradient, and directly works with x_i instead of augmented data $[x_i, -x_i]$. Moreover, our analysis does not require the loss function to have bounded gradient, and thus can directly handle the least squares loss.

The procedure in (7) can be regarded as an online counterpart of L_1-regularization in the sense that it approximately solves an L_1-regularization problem in the limit of $\eta \to 0$. Truncated gradient for L_1-regularization is different from (5), which is a naïve application of stochastic gradient descent rule with an added L_1-regularization term. As pointed out in the introduction, the latter fails because it rarely leads to sparsity. Our theory shows even with sparsification, the prediction performance is still comparable to standard online-learning algorithms. In the following, we develop a general regret bound for this general method, which also shows how the regret may depend on the sparsification parameter g.

3.4 Regret Analysis

Throughout the paper, we use $\|\cdot\|_1$ for 1-norm, and $\|\cdot\|$ for 2-norm. For reference, we make the following assumption regarding the loss function:

Assumption 3.1 *We assume $L(w, z)$ is convex in w, and there exist non-negative constants A and B such that $\|\nabla_1 L(w, z)\|^2 \leq AL(w, z) + B$ for all $w \in R^d$ and $z \in R^{d+1}$.*

For linear prediction problems, we have a general loss function of the form $L(w, z) = \phi(w^T x, y)$. The following are some common loss functions $\phi(\cdot, \cdot)$ with corresponding choices of parameters A and B (which are not unique), under the assumption $\sup_x \|x\| \leq C$.

- Logistic: $\phi(p, y) = \ln(1 + \exp(-py))$; $A = 0$ and $B = C^2$. This loss is for binary classification problems with $y \in \{\pm 1\}$.
- SVM (hinge loss): $\phi(p, y) = \max(0, 1 - py)$; $A = 0$ and $B = C^2$. This loss is for binary classification problems with $y \in \{\pm 1\}$.
- Least squares (square loss): $\phi(p, y) = (p - y)^2$; $A = 4C^2$ and $B = 0$. This loss is for regression problems.

Our main result is Theorem 3.1 which is parameterized by A and B. The proof is left to the appendix. Specializing it to particular losses yields several corollaries. A corollary applicable to the least square loss is given later in Corollary 4.1.

Theorem 3.1 *(Sparse Online Regret) Consider sparse online update rule (6) with $w_1 = 0$ and $\eta > 0$. If Assumption 3.1 holds, then for all $\bar{w} \in R^d$ we have*

$$\frac{1-0.5A\eta}{T}\sum_{i=1}^{T}\left[L(w_i,z_i)+\frac{g_i}{1-0.5A\eta}\|w_{i+1}\cdot I(w_{i+1}\le\theta)\|_1\right]$$
$$\le\frac{\eta}{2}B+\frac{\|\bar{w}\|^2}{2\eta T}+\frac{1}{T}\sum_{i=1}^{T}[L(\bar{w},z_i)+g_i\|\bar{w}\cdot I(w_{i+1}\le\theta)\|_1],$$

where for vectors $v=[v_1,\ldots,v_d]$ and $v'=[v'_1,\ldots,v'_d]$, we let

$$\|v\cdot I(|v'|\le\theta)\|_1=\sum_{j=1}^{d}|v_j|I(|v'_j|\le\theta),$$

where $I(\cdot)$ is the set indicator function.

We state the theorem with a constant learning rate η. As mentioned earlier, it is possible to obtain a result with variable learning rate where $\eta = \eta_i$ decays as i increases. Although this may lead to a no-regret bound without knowing T in advance, it introduces extra complexity to the presentation of the main idea. Since our focus is on sparsity rather than adapting learning rate, we do not include such a result for clarity. If T is known in advance, then in the above bound, one can simply take $\eta = O(1/\sqrt{T})$ and the L_1-regularized regret is of order $O(1/\sqrt{T})$.

In the above theorem, the right-hand side involves a term $g_i\|\bar{w}\cdot I(w_{i+1}\le\theta)\|_1$ depending on w_{i+1} which is not easily estimated. To remove this dependency, a trivial upper bound of $\theta=\infty$ can be used, leading to L_1 penalty $g_i\|\bar{w}\|_1$. In the general case of $\theta<\infty$, we cannot replace w_{i+1} by $\bar{w}$ because the effective regularization condition (as shown on the left-hand side) is the non-convex penalty $g_i\|w\cdot I(|w|\le\theta)\|_1$. Solving such a non-convex formulation is hard both in the online and batch settings. In general, we only know how to efficiently discover a local minimum which is difficult to characterize. Without a good characterization of the local minimum, it is not possible for us to replace $g_i\|\bar{w}\cdot I(w_{i+1}\le\theta)\|_1$ on the right-hand side by $g_i\|\bar{w}\cdot I(\bar{w}\le\theta)\|_1$ because such a formulation implies we can efficiently solve a non-convex problem with a simple online update rule. Still, when $\theta<\infty$, one naturally expects the right-hand side penalty $g_i\|\bar{w}\cdot I(w_{i+1}\le\theta)\|_1$ is much smaller than the corresponding L_1 penalty $g_i\|\bar{w}\|_1$, especially when w_j has many components close to 0. Therefore the situation with $\theta<\infty$ can potentially yield better performance on some data. This is confirmed in our experiments.

Theorem 3.1 also implies a trade-off between sparsity and regret performance. We may simply consider the case where $g_i = g$ is a constant. When g is small, we have less sparsity but the regret term $g\|\bar{w}\cdot I(w_{i+1}\le\theta)\|_1\le g\|\bar{w}\|_1$ on the right-hand side is also small. When g is large, we are able to achieve more sparsity but the regret $g\|\bar{w}\cdot I(w_{i+1}\le\theta)\|_1$ on the right-hand side also becomes large. Such a trade-off (sparsity versus prediction accuracy) is empirically studied in Section 6. Our observation suggests we can gain significant sparsity with only a small decrease of accuracy (that is, using a small g).

Now consider the case $\theta = \infty$ and $g_i = g$. When $T \to \infty$, if we let $\eta \to 0$ and $\eta T \to \infty$, then Theorem 3.1 implies

$$\frac{1}{T}\sum_{i=1}^{T}[L(w_i, z_i) + g\|w_i\|_1] \leq \inf_{\bar{w}\in R^d}\left[\frac{1}{T}\sum_{i=1}^{T}L(\bar{w}, z_i) + g\|\bar{w}\|_1\right] + o(1).$$

In other words, if we let $L'(w,z) = L(w,z) + g\|w\|_1$ be the L_1-regularized loss, then the L_1-regularized regret is small when $\eta \to 0$ and $T \to \infty$. In particular, if we let $\eta = 1/\sqrt{T}$, then the theorem implies the L_1-regularized regret is

$$\begin{aligned}
&\sum_{i=1}^{T}(L(w_i, z_i) + g\|w_i\|_1) - \sum_{i=1}^{T}(L(\bar{w}, z_i) + g\|\bar{w}\|_1)\\
\leq\ &\frac{\sqrt{T}}{2}(B + \|\bar{w}\|^2)\left(1 + \frac{A}{2\sqrt{T}}\right) + \frac{A}{2\sqrt{T}}\left(\sum_{i=1}^{T}L(\bar{w}, z_i) + g\sum_{i=1}^{T}(\|\bar{w}\|_1 - \|w_{i+1}\|_1)\right) + o(\sqrt{T}),
\end{aligned}$$

which is $O(\sqrt{T})$ for bounded loss function L and weights w_i. These observations imply our procedure can be regarded as the online counterpart of L_1-regularization methods. In the stochastic setting where the examples are drawn iid from some underlying distribution, the sparse online gradient method proposed in this paper solves the L_1-regularization problem.

3.5 Stochastic Setting

SGD-based online-learning methods can be used to solve large-scale batch optimization problems, often quite successfully (Shalev-Shwartz et al., 2007; Zhang, 2004). In this setting, we can go through training examples one-by-one in an online fashion, and repeat multiple times over the training data. In this section, we analyze the performance of such a procedure using Theorem 3.1.

To simplify the analysis, instead of assuming we go through the data one by one, we assume each additional data point is drawn from the training data randomly with equal probability. This corresponds to the standard stochastic optimization setting, in which observed samples are iid from some underlying distributions. The following result is a simple consequence of Theorem 3.1. For simplicity, we only consider the case with $\theta = \infty$ and constant gravity $g_i = g$.

Theorem 3.2 *Consider a set of training data $z_i = (x_i, y_i)$ for $i = 1, \ldots, n$, and let*

$$R(w, g) = \frac{1}{n}\sum_{i=1}^{n}L(w, z_i) + g\|w\|_1$$

be the L_1-regularized loss over training data. Let $\hat{w}_1 = w_1 = 0$, and define recursively for $t = 1, 2, \ldots$

$$w_{t+1} = T(w_t - \eta\nabla_1(w_t, z_{i_t}), g\eta), \qquad \hat{w}_{t+1} = \hat{w}_t + \frac{w_{t+1} - \hat{w}_t}{t+1},$$

where each i_t is drawn from $\{1,\ldots,n\}$ uniformly at random. If Assumption 3.1 holds, then at any time T, the following inequalities are valid for all $\bar{w} \in R^d$:

$$\begin{aligned}
&\mathbf{E}_{i_1,\ldots,i_T}\left[(1-0.5A\eta)R\left(\hat{w}_T, \frac{g}{1-0.5A\eta}\right)\right] \\
\leq &\mathbf{E}_{i_1,\ldots,i_T}\left[\frac{1-0.5A\eta}{T}\sum_{i=1}^{T} R\left(w_i, \frac{g}{1-0.5A\eta}\right)\right] \\
\leq &\frac{\eta}{2}B + \frac{\|\bar{w}\|^2}{2\eta T} + R(\bar{w},g).
\end{aligned}$$

Proof Note the recursion of $\hat{w}_t$ implies

$$\hat{w}_T = \frac{1}{T}\sum_{t=1}^{T} w_t$$

from telescoping the update rule. Because $R(w,g)$ is convex in w, the first inequality follows directly from Jensen's inequality. It remains to prove the second inequality. Theorem 3.1 implies the following:

$$\frac{1-0.5A\eta}{T}\sum_{t=1}^{T}\left[L(w_t,z_{i_t}) + \frac{g}{1-0.5A\eta}\|w_t\|_1\right] \leq g\|\bar{w}\|_1 + \frac{\eta}{2}B + \frac{\|\bar{w}\|^2}{2\eta T} + \frac{1}{T}\sum_{t=1}^{T} L(\bar{w},z_{i_t}). \tag{8}$$

Observe that

$$\mathbf{E}_{i_t}\left[L(w_t,z_{i_t}) + \frac{g}{1-0.5A\eta}\|w_t\|_1\right] = R\left(w_t, \frac{g}{1-0.5A\eta}\right)$$

and

$$g\|\bar{w}\|_1 + \mathbf{E}_{i_1,\ldots,i_T}\left[\frac{1}{T}\sum_{t=1}^{T} L(\bar{w},z_{i_t})\right] = R(\bar{w},g).$$

The second inequality is obtained by taking the expectation with respect to $\mathbf{E}_{i_1,\ldots,i_T}$ in (8). ■

If we let $\eta \to 0$ and $\eta T \to \infty$, the bound in Theorem 3.2 becomes

$$\mathbf{E}\left[R(\hat{w}_T,g)\right] \leq \mathbf{E}\left[\frac{1}{T}\sum_{t=1}^{T} R(w_t,g)\right] \leq \inf_{\bar{w}} R(\bar{w},g) + o(1).$$

That is, on average, $\hat{w}_T$ approximately solves the L_1-regularization problem

$$\inf_{w}\left[\frac{1}{n}\sum_{i=1}^{n} L(w,z_i) + g\|w\|_1\right].$$

If we choose a random stopping time T, then the above inequalities says that on average w_T also solves this L_1-regularization problem approximately. Therefore in our experiment, we use the last solution w_T instead of the aggregated solution $\hat{w}_T$. For practice purposes, this is adequate even though we do not intentionally choose a random stopping time.

Since L_1-regularization is frequently used to achieve sparsity in the batch learning setting, the connection to L_1-regularization can be regarded as an alternative justification for the sparse-online algorithm developed in this paper.

Algorithm 1 Truncated Gradient for Least Squares

```
Inputs:
  • threshold $\theta \geq 0$
  • gravity sequence $g_i \geq 0$
  • learning rate $\eta \in (0,1)$
  • example oracle $O$

initialize weights $w^j \leftarrow 0$ ($j = 1,\ldots,d$)
for trial $i = 1,2,\ldots$
  1. Acquire an unlabeled example $x = [x^1,x^2,\ldots,x^d]$ from oracle $O$
  2. forall weights $w^j$ ($j = 1,\ldots,d$)
      (a) if $w^j > 0$ and $w^j \leq \theta$ then $w^j \leftarrow \max\{w^j - g_i\eta, 0\}$
      (b) elseif $w^j < 0$ and $w^j \geq -\theta$ then $w^j \leftarrow \min\{w^j + g_i\eta, 0\}$
  3. Compute prediction: $\hat{y} = \sum_j w^j x^j$
  4. Acquire the label $y$ from oracle $O$
  5. Update weights for all features $j$: $w^j \leftarrow w^j + 2\eta(y - \hat{y})x^j$
```

4. Truncated Gradient for Least Squares

The method in Section 3 can be directly applied to least squares regression. This leads to Algorithm 1 which implements sparsification for square loss according to Equation (6). In the description, we use superscripted symbol w^j to denote the j-th component of vector w (in order to differentiate from w_i, which we have used to denote the i-th weight vector). For clarity, we also drop the index i from w_i. Although we keep the choice of gravity parameters g_i open in the algorithm description, in practice, we only consider the following choice:

$$g_i = \begin{cases} Kg & \text{if } i/K \text{ is an integer} \\ 0 & \text{otherwise} \end{cases}.$$

This may give a more aggressive truncation (thus sparsity) after every K-th iteration. Since we do not have a theorem formalizing how much more sparsity one can gain from this idea, its effect will only be examined empirically in Section 6.

In many online-learning situations (such as web applications), only a small subset of the features have nonzero values for any example x. It is thus desirable to deal with sparsity only in this small subset rather than in all features, while simultaneously inducing sparsity on all feature weights. Moreover, it is important to store only features with non-zero coefficients (if the number of features is too large to be stored in memory, this approach allows us to use a hash table to track only the nonzero coefficients). We describe how this can be implemented efficiently in the next section.

For reference, we present a specialization of Theorem 3.1 in the following corollary which is directly applicable to Algorithm 1.

Corollary 4.1 *(Sparse Online Square Loss Regret) If there exists $C > 0$ such that for all x, $\|x\| \leq C$, then for all $\bar{w} \in R^d$, we have*

$$\frac{1-2C^2\eta}{T}\sum_{i=1}^{T}\left[(w_i^T x_i - y_i)^2 + \frac{g_i}{1-2C^2\eta}\|w_i \cdot I(|w_i| \leq \theta)\|_1\right]$$
$$\leq \frac{\|\bar{w}\|^2}{2\eta T} + \frac{1}{T}\sum_{i=1}^{T}\left[(\bar{w}^T x_i - y_i)^2 + g_{i+1}\|\bar{w} \cdot I(|w_{i+1}| \leq \theta)\|_1\right],$$

where $w_i = [w^1, \ldots, w^d] \in R^d$ is the weight vector used for prediction at the i-th step of Algorithm 1; (x_i, y_i) is the data point observed at the i-step.

This corollary explicitly states that average square loss incurred by the learner (the left-hand side) is bounded by the average square loss of the best weight vector $\bar{w}$, plus a term related to the size of $\bar{w}$ which decays as $1/T$ and an additive offset controlled by the sparsity threshold θ and the gravity parameter g_i.

5. Efficient Implementation

We altered a standard gradient-descent implementation, VOWPAL WABBIT(Langford et al., 2007), according to algorithm 1. VOWPAL WABBIT optimizes square loss on a linear representation $w^T x$ via gradient descent (3) with a couple caveats:

1. The prediction is normalized by the square root of the number of nonzero entries in a sparse vector, $w^T x / \sqrt{\|x\|_0}$. This alteration is just a constant rescaling on dense vectors which is effectively removable by an appropriate rescaling of the learning rate.

2. The prediction is clipped to the interval $[0,1]$, implying the loss function is not square loss for unclipped predictions outside of this dynamic range. Instead the update is a constant value, equivalent to the gradient of a linear loss function.

The learning rate in VOWPAL WABBIT is controllable, supporting $1/i$ decay as well as a constant learning rate (and rates in-between). The program operates in an entirely online fashion, so the memory footprint is essentially just the weight vector, even when the amount of data is very large.

As mentioned earlier, we would like the algorithm's computational complexity to depend linearly on the number of nonzero features of an example, rather than the total number of features. The approach we took was to store a time-stamp τ_j for each feature j. The time-stamp was initialized to the index of the example where feature j was nonzero for the first time. During online learning, we simply went through all nonzero features j of example i, and could "simulate" the shrinkage of w^j after τ_j in a batch mode. These weights are then updated, and their time stamps are set to i. This lazy-update idea of delaying the shrinkage calculation until needed is the key to efficient implementation of truncated gradient. Specifically, instead of using update rule (6) for weight w^j, we shrunk the weights of all nonzero feature j differently by the following:

$$f(w^j) = T_1\left(w^j + 2\eta(y - \hat{y})x^j, \left\lfloor \frac{i - \tau_j}{K} \right\rfloor K\eta g, \theta\right),$$

and τ_j is updated by

$$\tau_j \leftarrow \tau_j + \left\lfloor \frac{i - \tau_j}{K} \right\rfloor K.$$

This lazy-update trick can be applied to the other two algorithms given in Section 3. In the coefficient rounding algorithm (4), for instance, for each nonzero feature j of example i, we can first perform a regular gradient descent on the square loss, and then do the following: if $|w_j|$ is below the threshold θ and $i \geq \tau_j + K$, we round w_j to 0 and set τ_j to i.

This implementation shows the truncated gradient method satisfies the following requirements needed for solving large scale problems with sparse features.

- The algorithm is computationally efficient: the number of operations per online step is linear in the number of nonzero features, and independent of the total number of features.
- The algorithm is memory efficient: it maintains a list of active features, and a feature can be inserted when observed, and deleted when the corresponding weight becomes zero.

If we directly apply the online projection idea of Zinkevich (2003) to solve (1), then in the update rule (7), one has to pick the smallest $g_i \geq 0$ such that $\|w_{i+1}\|_1 \leq s$. We do not know an efficient method to find this specific g_i using operations independent of the total number of features. A standard implementation relies on sorting all weights, which requires $O(d \log d)$ operations, where d is the total number of (nonzero) features. This complexity is unacceptable for our purpose. However, in an important recent work, Duchi et al. (2008) proposed an efficient online ℓ_1-projection method. The idea is to use a balanced tree to keep track of weights, which allows efficient threshold finding and tree updates in $O(k \ln d)$ operations on average, where k denotes the number of nonzero coefficients in the current training example. Although the algorithm still has weak dependency on d, it is applicable to large-scale practical applications. The theoretical analysis presented in this paper shows we can obtain a meaningful regret bound by picking an arbitrary g_i. This is useful because the resulting method is much simpler to implement and is computationally more efficient per online step. Moreover, our method allows non-convex updates closely related to the simple coefficient rounding idea. Due to the complexity of implementing the balanced tree strategy in Duchi et al. (2008), we shall not compare to it in this paper and leave it as a future direction. However, we believe the sparsity achieved with their approach should be comparable to the sparsity achieved with our method.

6. Empirical Results

We applied VOWPAL WABBIT with the efficiently implemented sparsify option, as described in the previous section, to a selection of data sets, including eleven data sets from the UCI repository (Asuncion and Newman, 2007), the much larger data set rcv1 (Lewis et al., 2004), and a private large-scale data set Big_Ads related to ad interest prediction. While UCI data sets are useful for benchmark purposes, rcv1 and Big_Ads are more interesting since they embody real-world data sets with large numbers of features, many of which are less informative for making predictions than others. The data sets are summarized in Table 1.

The UCI data sets used do not have many features so we expect that a large fraction of these features are useful for making predictions. For comparison purposes as well as to better demonstrate the behavior of our algorithm, we also added 1000 random binary features to those data sets. Each feature has value 1 with probability 0.05 and 0 otherwise.

Data Set	#features	#train data	#test data	task
ad	1411	2455	824	classification
crx	47	526	164	classification
housing	14	381	125	regression
krvskp	74	2413	783	classification
magic04	11	14226	4794	classification
mushroom	117	6079	2045	classification
spambase	58	3445	1156	classification
wbc	10	520	179	classification
wdbc	31	421	148	classification
wpbc	33	153	45	classification
zoo	17	77	24	regression
rcv1	38853	781265	23149	classification
Big_Ads	3×10^9	26×10^6	2.7×10^6	classification

Table 1: Data Set Summary.

6.1 Feature Sparsification of Truncated Gradient

In the first set of experiments, we are interested in how much reduction in the number of features is possible without affecting learning performance significantly; specifically, we require the accuracy be reduced by no more than 1% for classification tasks, and the total square loss be increased by no more than 1% for regression tasks. As common practice, we allowed the algorithm to run on the training data set for multiple passes with decaying learning rate. For each data set, we performed 10-fold cross validation over the training set to identify the best set of parameters, including the learning rate η (ranging from 0.1 to 0.5), the sparsification rate g (ranging from 0 to 0.3), number of passes of the training set (ranging from 5 to 30), and the decay of learning rate across these passes (ranging from 0.5 to 0.9). The optimized parameters were used to train VOWPAL WABBIT on the whole training set. Finally, the learned classifier/regressor was evaluated on the test set. We fixed $K = 1$ and $\theta = \infty$, and will study the effects of K and θ in later subsections.

Figure 2 shows the fraction of reduced features after sparsification is applied to each data set. For UCI data sets, we also include experiments with 1000 random features added to the original feature set. We do not add random features to rcv1 and Big_Ads since the experiment is not as interesting.

For UCI data sets, with randomly added features, VOWPAL WABBIT is able to reduce the number of features by a fraction of more than 90%, except for the ad data set in which only 71% reduction is observed. This less satisfying result might be improved by a more extensive parameter search in cross validation. However, if we can tolerate 1.3% decrease in accuracy (instead of 1% as for other data sets) during cross validation, VOWPAL WABBIT is able to achieve 91.4% reduction, indicating that a large reduction is still possible at the tiny additional cost of 0.3% accuracy loss. With this slightly more aggressive sparsification, the test-set accuracy drops from 95.9% (when only 1% loss in accuracy is allowed in cross validation) to 95.4%, while the accuracy without sparsification is 96.5%.

Even for the original UCI data sets without artificially added features, VOWPAL WABBIT manages to filter out some of the less useful features while maintaining the same level of performance. For example, for the ad data set, a reduction of 83.4% is achieved. Compared to the results above, it seems the most effective feature reductions occur on data sets with a large number of less useful features, exactly where sparsification is needed.

For rcv1, more than 75% of features are removed after the sparsification process, indicating the effectiveness of our algorithm in real-life problems. We were not able to try many parameters in cross validation because of the size of rcv1. It is expected that more reduction is possible when a more thorough parameter search is performed.

The previous results do not exercise the full power of the approach presented here because the standard Lasso (Tibshirani, 1996) is or may be computationally viable in these data sets. We have also applied this approach to a large non-public data set Big_Ads where the goal is predicting which of two ads was clicked on given context information (the content of ads and query information). Here, accepting a 0.009 increase in classification error (from error rate 0.329 to error rate 0.338) allows us to reduce the number of features from about 3×10^9 to about 24×10^6, a factor of 125 decrease in the number of features.

For classification tasks, we also study how our sparsification solution affects AUC (Area Under the ROC Curve), which is a standard metric for classification.[1] Using the same sets of parameters from 10-fold cross validation described above, we find the criterion is not affected significantly by sparsification and in some cases, they are actually slightly improved. The reason may be that our sparsification method removed some of the features that could have confused VOWPAL WABBIT. The ratios of the AUC with and without sparsification for all classification tasks are plotted in Figures 3. Often these ratios are above 98%.

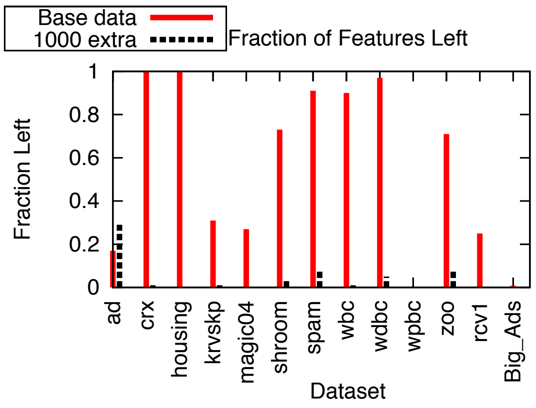
6.2 The Effects of K

As we argued before, using a K value larger than 1 may be desired in truncated gradient and the rounding algorithms. This advantage is empirically demonstrated here. In particular, we try $K = 1$, $K = 10$, and $K = 20$ in both algorithms. As before, cross validation is used to select parameters in the rounding algorithm, including learning rate η, number of passes of data during training, and learning rate decay over training passes.

Figures 4 and 5 give the AUC vs. number-of-feature plots, where each data point is generated by running respective algorithm using a different value of g (for truncated gradient) and θ (for the rounding algorithm). We used $\theta = \infty$ in truncated gradient.

The effect of K is large in the rounding algorithm. For instance, in the ad data set the algorithm using $K = 1$ achieves an AUC of 0.94 with 322 features, while 13 and 7 features are needed using $K = 10$ and $K = 20$, respectively. However, the same benefits of using a larger K is not observed in truncated gradient, although the performances with $K = 10$ or 20 are at least as good as those with $K = 1$ and for the spambase data set further feature reduction is achieved at the same level of performance, reducing the number of features from 76 (when $K = 1$) to 25 (when $K = 10$ or 20) with of an AUC of about 0.89.

1. We use AUC here and in later subsections because it is insensitive to threshold, which is unlike accuracy.



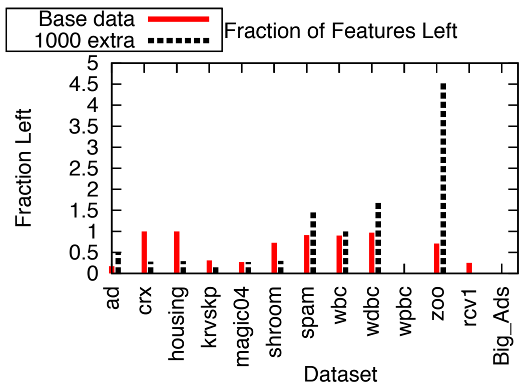


Figure 2: Plots showing the amount of features left after sparsification using truncated gradient for each data set, when the performance is changed by at most 1% due to sparsification. The solid bar: with the original feature set; the dashed bar: with 1000 random features added to each example. Plot on left: fraction left with respect to the total number of features (original with 1000 artificial features for the dashed bar). Plot on right: fraction left with respect to the original features (not counting the 1000 artificial features in the denominator for the dashed bar).

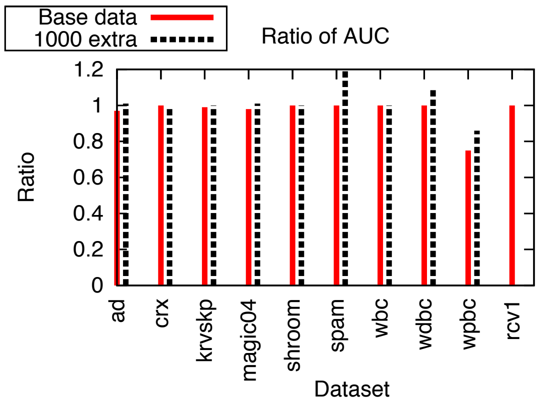


Figure 3: A plot showing the ratio of the AUC when sparsification is used over the AUC when no sparsification is used. The same process as in Figure 2 is used to determine empirically good parameters. The first result is for the original data set, while the second result is for the modified data set where 1000 random features are added to each example.

6.3 The Effects of θ in Truncated Gradient

In this subsection, we empirically study the effect of θ in truncated gradient. The rounding algorithm is also included for comparison due to its similarity to truncated gradient when $\theta = g$. Again, we used cross validation to choose parameters for each θ value tried, and focused on the AUC metric in the eight UCI classification tasks, except the degenerate one of wpbc. We fixed $K = 10$ in both algorithm.

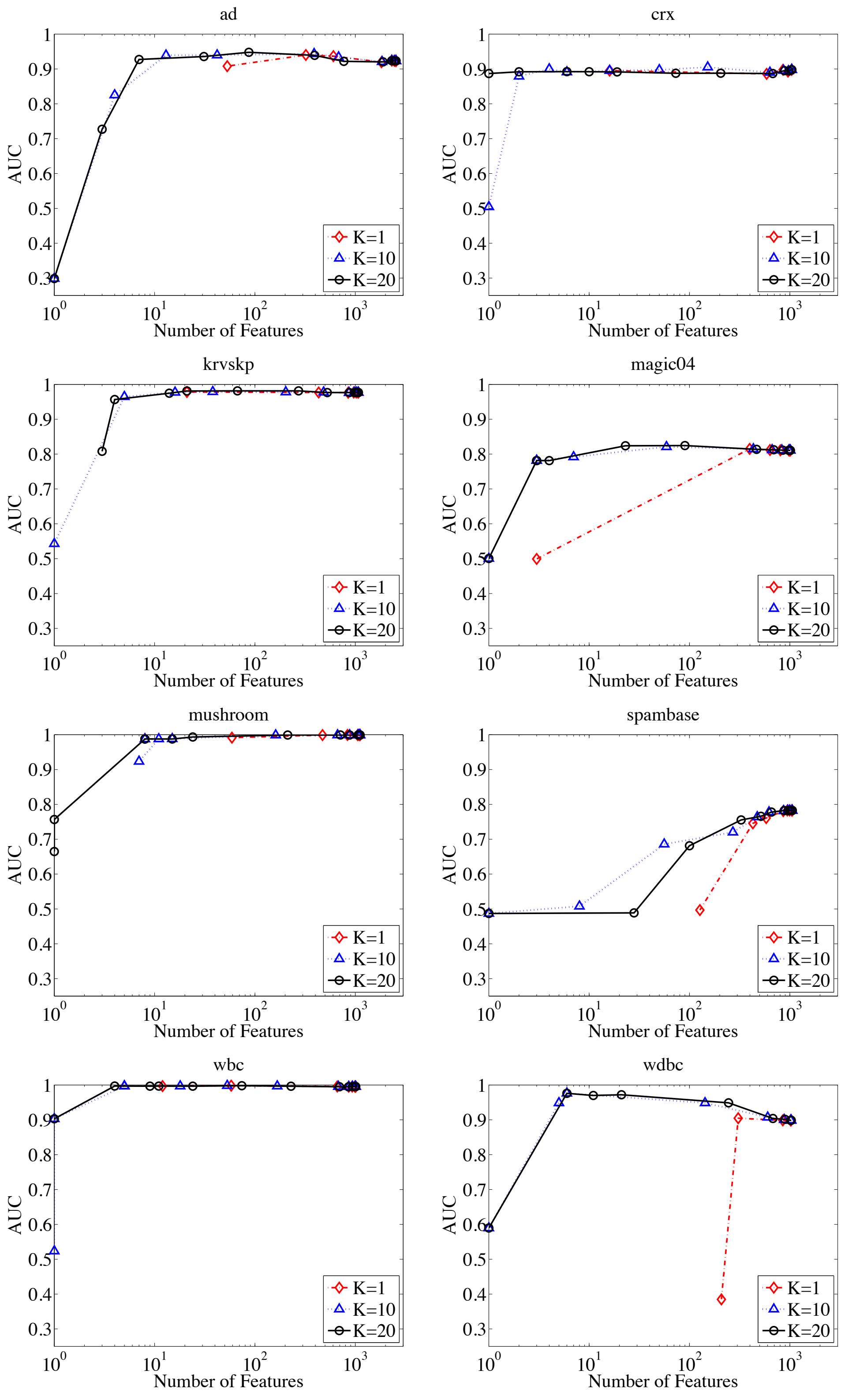


Figure 4: Effect of K on AUC in the rounding algorithm.

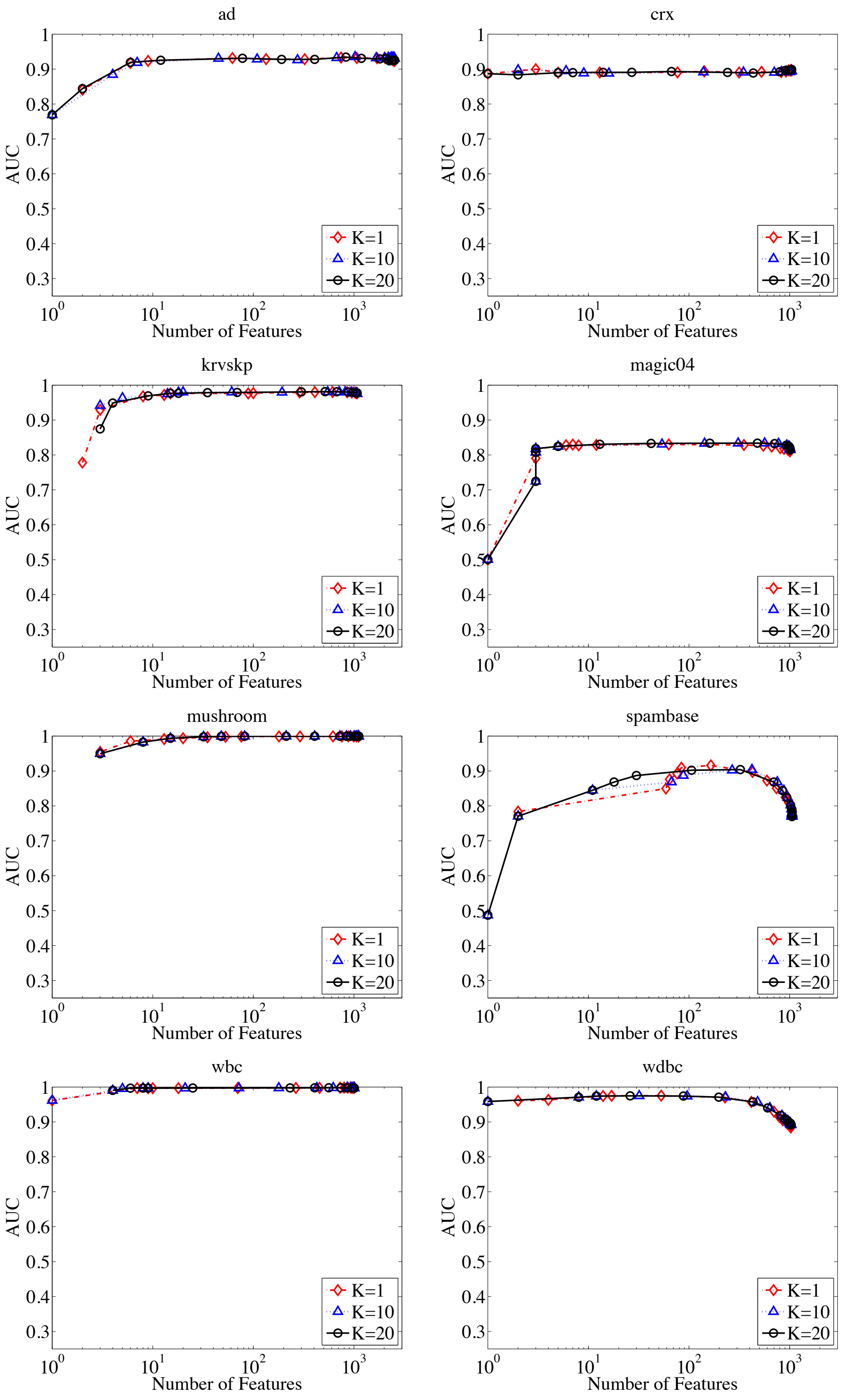


Figure 5: Effect of K on AUC in truncated gradient.

Figure 6 gives the AUC vs. number-of-feature plots, where each data point is generated by running respective algorithms using a different value of g (for truncated gradient) and θ (for the rounding algorithm). A few observations are in place. First, the results verify the observation that the behavior of truncated gradient with $\theta = g$ is similar to the rounding algorithm. Second, these results suggest that, in practice, it may be desirable to use $\theta = \infty$ in truncated gradient because it avoids the local-minimum problem.

6.4 Comparison to Other Algorithms

The next set of experiments compares truncated gradient to other algorithms regarding their abilities to balance feature sparsification and performance. Again, we focus on the AUC metric in UCI classification tasks except wpdc. The algorithms for comparison include:

- The truncated gradient algorithm: We fixed $K = 10$ and $\theta = \infty$, used crossed-validated parameters, and altered the gravity parameter g.
- The rounding algorithm described in Section 3.1: We fixed $K = 10$, used cross-validated parameters, and altered the rounding threshold θ.
- The subgradient algorithm described in Section 3.2: We fixed $K = 10$, used cross-validated parameters, and altered the regularization parameter g.
- The Lasso (Tibshirani, 1996) for batch L_1-regularization: We used a publicly available implementation (Sjöstrand, 2005).

Note that we do not attempt to compare these algorithms on rcv1 and Big_Ads simply because their sizes are too large for the Lasso.

Figure 7 gives the results. Truncated gradient is consistently competitive with the other two online algorithms and significantly outperformed them in some problems. This suggests the effectiveness of truncated gradient.

Second, it is interesting to observe that the qualitative behavior of truncated gradient is often similar to LASSO, especially when very sparse weight vectors are allowed (the left side in the graphs). This is consistent with theorem 3.2 showing the relation between them. However, LASSO usually has worse performance when the allowed number of nonzero weights is set too large (the right side of the graphs). In this case, LASSO seems to overfit, while truncated gradient is more robust to overfitting. The robustness of online learning is often attributed to early stopping, which has been extensively discussed in the literature (e.g., Zhang, 2004).

Finally, it is worth emphasizing that the experiments in this subsection try to shed some light on the relative strengths of these algorithms in terms of feature sparsification. For large data sets such as Big_Ads only truncated gradient, coefficient rounding, and the sub-gradient algorithms are applicable. As we have shown and argued, the rounding algorithm is quite ad hoc and may not work robustly in some problems, and the sub-gradient algorithm does not lead to sparsity in general during training.

7. Conclusion

This paper covers the first sparsification technique for large-scale online learning with strong theoretical guarantees. The algorithm, truncated gradient, is the natural extension of Lasso-style re-

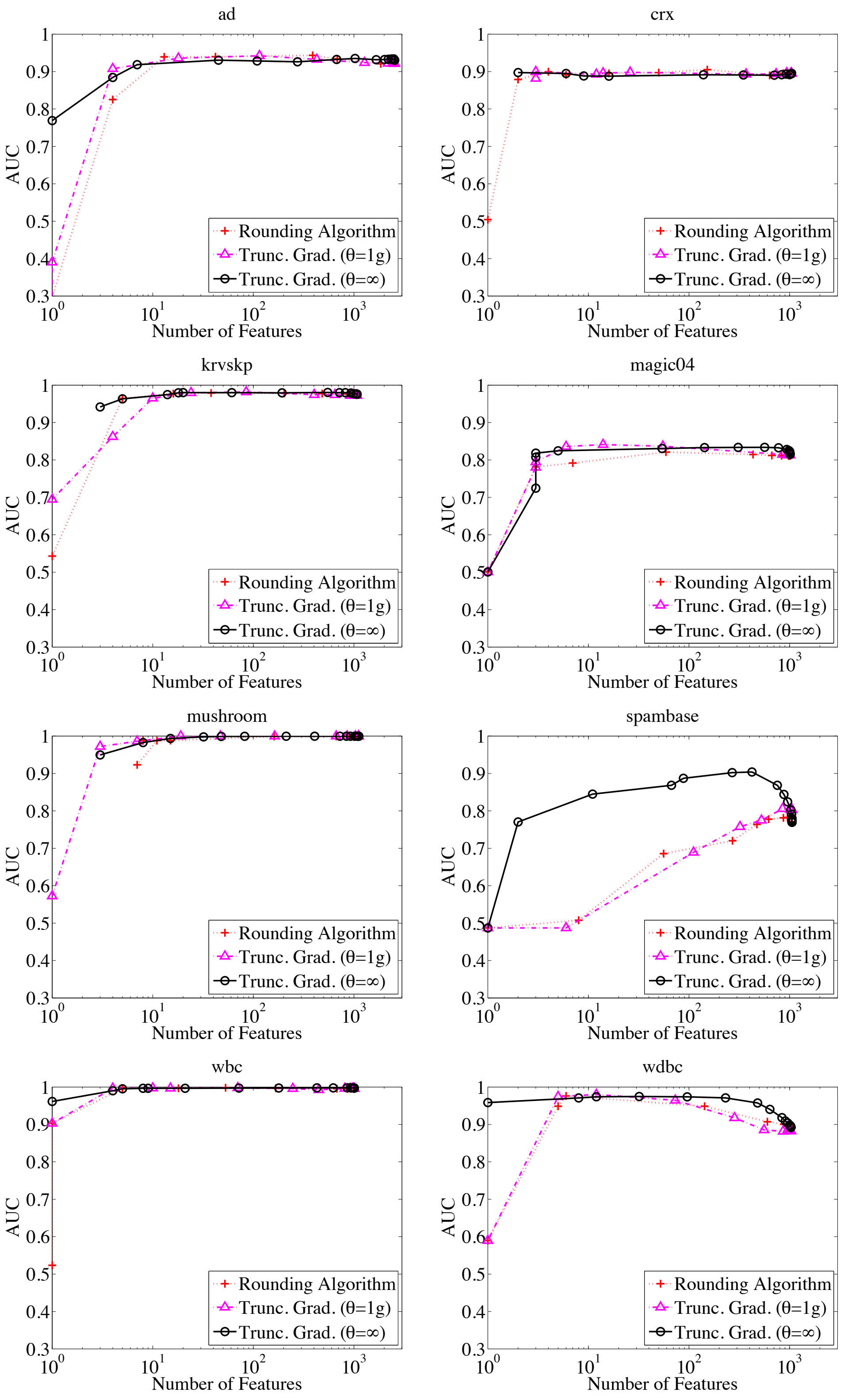


Figure 6: Effect of θ on AUC in truncated gradient.

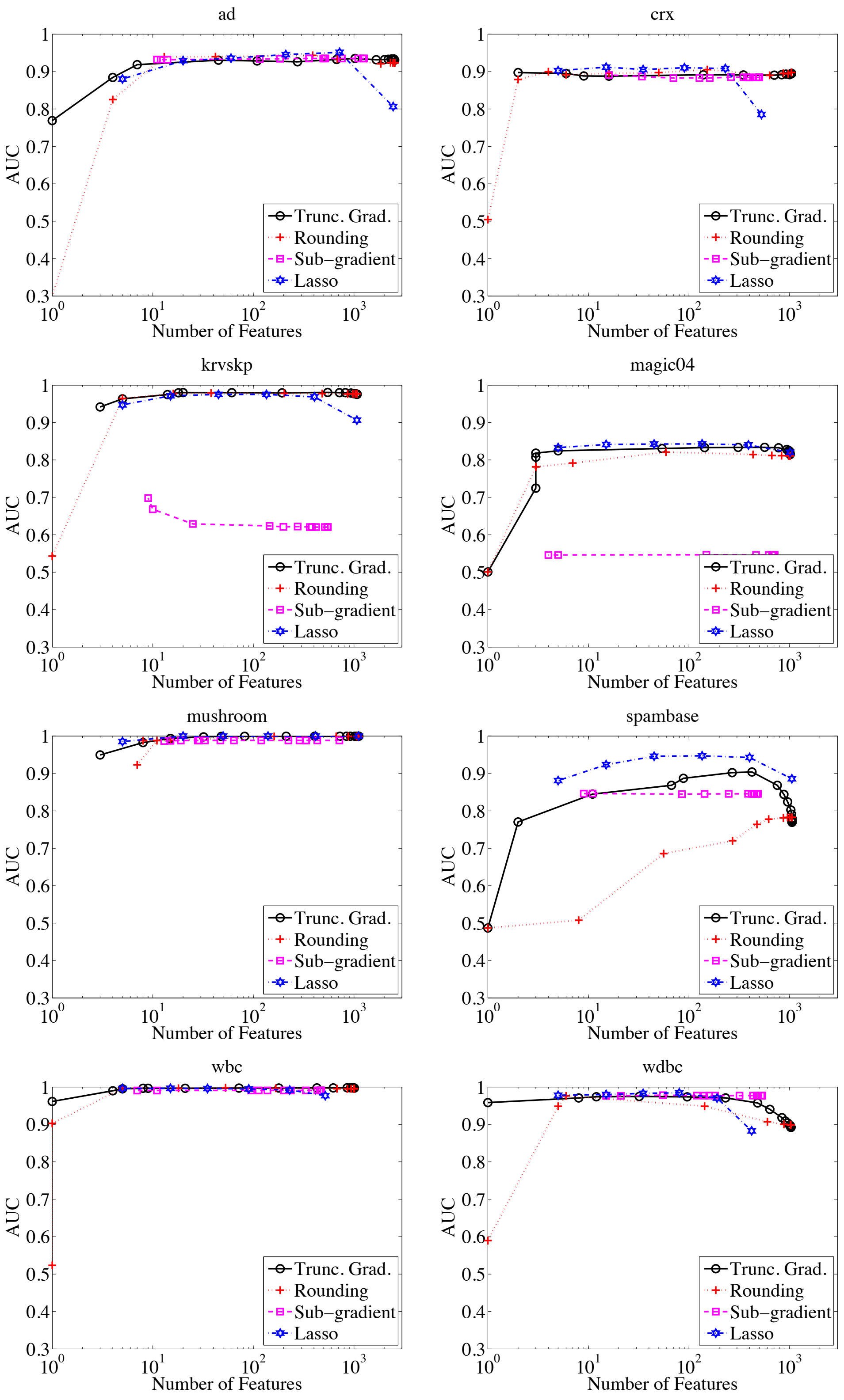


Figure 7: Comparison of four algorithms.

gression to the online-learning setting. Theorem 3.1 proves the technique is sound: it never harms performance much compared to standard stochastic gradient descent in adversarial situations. Furthermore, we show the asymptotic solution of one instance of the algorithm is essentially equivalent to the Lasso regression, thus justifying the algorithm's ability to produce sparse weight vectors when the number of features is intractably large.

The theorem is verified experimentally in a number of problems. In some cases, especially for problems with many irrelevant features, this approach achieves a one or two order of magnitude reduction in the number of features.

Acknowledgments

We thank Alex Strehl for discussions and help in developing VOWPAL WABBIT. Part of this work was done when Lihong Li and Tong Zhang were at Yahoo! Research in 2007.

Appendix A. Proof of Theorem 3.1

The following lemma is the essential step in our analysis.

Lemma 1 *Suppose update rule (6) is applied to weight vector w on example $z = (x, y)$ with gravity parameter $g_i = g$, and results in a weight vector w'. If Assumption 3.1 holds, then for all $\bar{w} \in R^d$, we have*

$$\begin{aligned}&(1 - 0.5A\eta)L(w,z) + g\|w' \cdot I(|w'| \le \theta)\|_1 \\ \le& L(\bar{w},z) + g\|\bar{w} \cdot I(|w'| \le \theta)\|_1 + \frac{\eta}{2}B + \frac{\|\bar{w} - w\|^2 - \|\bar{w} - w'\|^2}{2\eta}.\end{aligned}$$

Proof Consider any target vector $\bar{w} \in R^d$ and let $\tilde{w} = w - \eta\nabla_1 L(w,z)$. We have $w' = T_1(\tilde{w}, g\eta, \theta)$. Let

$$u(\bar{w}, w') = g\|\bar{w} \cdot I(|w'| \le \theta)\|_1 - g\|w' \cdot I(|w'| \le \theta)\|_1.$$

Then the update equation implies the following:

$$\begin{aligned}&\|\bar{w} - w'\|^2 \\ \le&\|\bar{w} - w'\|^2 + \|w' - \tilde{w}\|^2 \\ =&\|\bar{w} - \tilde{w}\|^2 - 2(\bar{w} - w')^T(w' - \tilde{w}) \\ \le&\|\bar{w} - \tilde{w}\|^2 + 2\eta u(\bar{w}, w') \\ =&\|\bar{w} - w\|^2 + \|w - \tilde{w}\|^2 + 2(\bar{w} - w)^T(w - \tilde{w}) + 2\eta u(\bar{w}, w') \\ =&\|\bar{w} - w\|^2 + \eta^2\|\nabla_1 L(w,z)\|^2 + 2\eta(\bar{w} - w)^T\nabla_1 L(w,z) + 2\eta u(\bar{w}, w') \\ \le&\|\bar{w} - w\|^2 + \eta^2\|\nabla_1 L(w,z)\|^2 + 2\eta(L(\bar{w},z) - L(w,z)) + 2\eta u(\bar{w}, w') \\ \le&\|\bar{w} - w\|^2 + \eta^2(AL(w,z) + B) + 2\eta(L(\bar{w},z) - L(w,z)) + 2\eta u(\bar{w}, w').\end{aligned}$$

Here, the first and second equalities follow from algebra, and the third from the definition of $\tilde{w}$. The first inequality follows because a square is always non-negative. The second inequality follows because $w' = T_1(\tilde{w}, g\eta, \theta)$, which implies $(w' - \tilde{w})^T w' = -g\eta \|w' \cdot I(|\tilde{w}| \leq \theta)\|_1 = -g\eta \|w' \cdot I(|w'| \leq \theta)\|_1$ and $|w'_j - \tilde{w}_j| \leq g\eta I(|w'_j| \leq \theta)$. Therefore,

$$\begin{aligned}
-(\bar{w} - w')^T (w' - \tilde{w}) &= -\bar{w}^T (w' - \tilde{w}) + w'^T (w' - \tilde{w}) \\
&\leq \sum_{j=1}^{d} |\bar{w}_j||w'_j - \tilde{w}_j| + (w' - \tilde{w})^T w' \\
&\leq g\eta \sum_{j=1}^{d} |\bar{w}_j| I(|w'_j| \leq \theta) + (w' - \tilde{w})^T w' = \eta u(\bar{w}, w'),
\end{aligned}$$

where the third inequality follows from the definition of sub-gradient of a convex function, implying

$$(\bar{w} - w)^T \nabla_1 L(w, z) \leq L(\bar{w}, z) - L(w, z)$$

for all w and $\bar{w}$; the fourth inequality follows from Assumption 3.1. Rearranging the above inequality leads to the desired bound. ■

Proof (of Theorem 3.1) Applying Lemma 1 to the update on trial i gives

$$\begin{aligned}
&(1 - 0.5A\eta) L(w_i, z_i) + g_i \|w_{i+1} \cdot I(|w_{i+1}| \leq \theta)\|_1 \\
\leq &L(\bar{w}, z_i) + \frac{\|\bar{w} - w_i\|^2 - \|\bar{w} - w_{i+1}\|^2}{2\eta} + g_i \|\bar{w} \cdot I(|w_{i+1}| \leq \theta)\|_1 + \frac{\eta}{2} B.
\end{aligned}$$

Now summing over $i = 1, 2, \ldots, T$, we obtain

$$\begin{aligned}
& \sum_{i=1}^{T} [(1 - 0.5A\eta) L(w_i, z_i) + g_i \|w_{i+1} \cdot I(|w_{i+1}| \leq \theta)\|_1] \\
\leq \quad & \sum_{i=1}^{T} \left[\frac{\|\bar{w} - w_i\|^2 - \|\bar{w} - w_{i+1}\|^2}{2\eta} + L(\bar{w}, z_i) + g_i \|\bar{w} \cdot I(|w_{i+1}| \leq \theta)\|_1 + \frac{\eta}{2} B \right] \\
= \quad & \frac{\|\bar{w} - w_1\|^2 - \|\bar{w} - w_T\|^2}{2\eta} + \frac{\eta}{2} TB + \sum_{i=1}^{T} [L(\bar{w}, z_i) + g_i \|\bar{w} \cdot I(|w_{i+1}| \leq \theta)\|_1] \\
\leq \quad & \frac{\|\bar{w}\|^2}{2\eta} + \frac{\eta}{2} TB + \sum_{i=1}^{T} [L(\bar{w}, z_i) + g_i \|\bar{w} \cdot I(|w_{i+1}| \leq \theta)\|_1].
\end{aligned}$$

The first equality follows from the telescoping sum and the second inequality follows from the initial condition (all weights are zero) and dropping negative quantities. The theorem follows by dividing with respect to T and rearranging terms. ■

References

Arthur Asuncion and David J. Newman. UCI machine learning repository, 2007. University of California, Irvine, School of Information and Computer Sciences, http://www.ics.uci.edu/∼mlearn/MLRepository.html.

Suhrid Balakrishnan and David Madigan. Algorithms for sparse linear classifiers in the massive data setting. *Journal of Machine Learning Research*, 9:313–337, 2008.

Bob Carpenter. Lazy sparse stochastic gradient descent for regularized multinomial logistic regression. Technical report, April 2008.

Nicolò Cesa-Bianchi, Philip M. Long, and Manfred Warmuth. Worst-case quadratic loss bounds for prediction using linear functions and gradient descent. *IEEE Transactions on Neural Networks*, 7(3):604–619, 1996.

Cheng-Tao Chu, Sang Kyun Kim, Yi-An Lin, YuanYuan Yu, Gary Bradski, Andrew Y. Ng, and Kunle Olukotun. Map-reduce for machine learning on multicore. In *Advances in Neural Information Processing Systems 20 (NIPS-07)*, 2008.

Ofer Dekel, Shai Shalev-Schwartz, and Yoram Singer. The Forgetron: A kernel-based perceptron on a fixed budget. In *Advances in Neural Information Processing Systems 18 (NIPS-05)*, pages 259–266, 2006.

John Duchi and Yoram Singer. Online and batch learning using forward looking subgradients. Unpublished manuscript, September 2008.

John Duchi, Shai Shalev-Shwartz, Yoram Singer, and Tushar Chandra. Efficient projections onto the ℓ_1-ball for learning in high dimensions. In *Proceedings of the Twenty-Fifth International Conference on Machine Learning (ICML-08)*, pages 272–279, 2008.

Jyrki Kivinen and Manfred K. Warmuth. Exponentiated gradient versus gradient descent for linear predictors. *Information and Computation*, 132(1):1–63, 1997.

John Langford, Lihong Li, and Alexander L. Strehl. Vowpal Wabbit (fast online learning), 2007. http://hunch.net/$\sim$vw/.

Honglak Lee, Alexis Battle, Rajat Raina, and Andrew Y. Ng. Efficient sparse coding algorithms. In *Advances in Neural Information Processing Systems 19 (NIPS-06)*, pages 801–808, 2007.

David D. Lewis, Yiming Yang, Tony G. Rose, and Fan Li. RCV1: A new benchmark collection for text categorization research. *Journal of Machine Learning Research*, 5:361–397, 2004.

Nick Littlestone. Learning quickly when irrelevant attributes abound: A new linear-threshold algorithms. *Machine Learning*, 2(4):285–318, 1988.

Nick Littlestone, Philip M. Long, and Manfred K. Warmuth. On-line learning of linear functions. *Computational Complexity*, 5(2):1–23, 1995.

Shai Shalev-Shwartz, Yoram Singer, and Nathan Srebro. Pegasos: Primal Estimated sub-GrAdient SOlver for SVM. In *Proceedings of the Twenty-Fourth International Conference on Machine Learning (ICML-07)*, 2007.

Karl Sjöstrand. Matlab implementation of LASSO, LARS, the elastic net and SPCA, June 2005. Version 2.0, http://www2.imm.dtu.dk/pubdb/p.php?3897.

Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society, B.*, 58(1):267–288, 1996.

Tong Zhang. Solving large scale linear prediction problems using stochastic gradient descent algorithms. In *Proceedings of the Twenty-First International Conference on Machine Learning (ICML-04)*, pages 919–926, 2004.

Martin Zinkevich. Online convex programming and generalized infinitesimal gradient ascent. In *Proceedings of the Twentieth International Conference on Machine Learning (ICML-03)*, pages 928–936, 2003.

Journal of Machine Learning Research 10 (2009) 803-826 Submitted 2/08; Revised 9/08; Published 3/09

A New Approach to Collaborative Filtering: Operator Estimation with Spectral Regularization

Jacob Abernethy JAKE@CS.BERKELEY.EDU
Division of Computer Science
University of California
387 Soda Hall, Berkeley, CA, USA

Francis Bach FRANCIS.BACH@MINES.ORG
INRIA - WILLOW Project-Team
Laboratoire d'Informatique de l'Ecole Normale Supérieure (CNRS/ENS/INRIA UMR 8548)
45, rue d'Ulm, 75230 Paris, France

Theodoros Evgeniou THEODOROS.EVGENIOU@INSEAD.EDU
Decision Sciences and Technology Management
INSEAD
Bd de Constance, 77300 Fontainebleau, France

Jean-Philippe Vert[*] JEAN-PHILIPPE.VERT@MINES-PARISTECH.FR
Mines ParisTech
Centre for Computational Biology
35 rue Saint-Honoré, 77300 Fontainebleau, France

Editor: Tommi Jaakkola

Abstract

We present a general approach for collaborative filtering (CF) using spectral regularization to learn linear operators mapping a set of "users" to a set of possibly desired "objects". In particular, several recent low-rank type matrix-completion methods for CF are shown to be special cases of our proposed framework. Unlike existing regularization-based CF, our approach can be used to incorporate additional information such as attributes of the users/objects—a feature currently lacking in existing regularization-based CF approaches—using popular and well-known *kernel methods*. We provide novel representer theorems that we use to develop new estimation methods. We then provide learning algorithms based on low-rank decompositions and test them on a standard CF data set. The experiments indicate the advantages of generalizing the existing regularization-based CF methods to incorporate related information about users and objects. Finally, we show that certain multi-task learning methods can be also seen as special cases of our proposed approach.

Keywords: collaborative filtering, matrix completion, kernel methods, spectral regularization

1. Introduction

Collaborative filtering (CF) refers to the task of predicting preferences of a given "user" for some "objects" (e.g., books, music, products, people, etc.) based on his/her previously revealed preferences—typically in the form of purchases or ratings—as well as the revealed preferences of other users. In a book recommender system, for example, one might like to suggest new books

∗. Also at Institut Curie, Inserm U900, 75248 Paris, France.

to a new user based on what she and other users have recently purchased or rated. The ultimate goal of CF is to infer the preferences of users in order to offer them new objects.

A number of CF methods have been developed in the past (Breese et al., 1998; Heckerman et al., 2000; Salakhutdinov et al., 2007). Recently there has been interest in CF using regularization-based methods (Srebro and Jaakkola, 2003). This work adds to that literature by developing a novel general approach to regularization-based CF methods.

Recent regularization-based CF methods assume that the only data available are the revealed preferences, where no other information such as background information on the objects or users is given. In this case, one may formulate the problem as that of inferring the contents of a partially observed *preference matrix*: each row represents a user, each column represents an object (e.g., books or movies), and entries in the matrix represent a given user's rating of a given object. When the only information available is a set of observed user/object ratings, the unknown entries in the matrix must be inferred from the known ones—of which there are typically very few relative to the size of the matrix.

To make useful predictions within this setting, regularization-based CF methods make certain assumptions about the *relatedness* of the objects and users. The most common assumption is that the preference function can be decomposed into a small number of factors, resulting in the search for a low-rank matrix which approximates the partially observed matrix of preferences (Srebro and Jaakkola, 2003). The rank constraint can be interpreted as a regularization on the hypothesis space. Since the rank constraint gives rise to a non-convex set of matrices, the associated optimization problem will be a difficult non-convex problem for which only heuristic algorithms exist (Srebro and Jaakkola, 2003). An alternative formulation, proposed by Srebro et al. (2005), suggests penalizing the predicted matrix by its *trace norm*, that is, the sum of its singular values. An added benefit of the trace norm regularization is that, with a sufficiently large regularization parameter, the final solution will be low-rank (Fazel et al., 2001; Bach, 2008).

However, a key limitation of current regularization-based CF methods is that they do not take advantage of additional information, such as known attributes of each user (e.g., gender, age) and object (e.g., book's author, genre), which is often available. Intuitively, such information might be useful to guide the inference of preferences, in particular for users and objects with very few known ratings. For example, at the extreme, users and objects with no prior ratings can not be considered in the standard CF formulation, while their attributes alone could provide some basic preference inference.

The main contribution of this paper is to develop a general framework for the CF setting when user/object attribute information is potentially available and, in particular, to provide several specific algorithms based on new "representer" theorems. More precisely we show that CF, while typically seen as a problem of matrix completion, can be thought of more generally as *linear operator estimation*, where the desired operator maps from a Hilbert space of users to a Hilbert space of objects. Equivalently, this can be viewed as learning a bilinear form between users and objects. We then develop *spectral regularization* based methods to learn such linear operators. When dealing with operators, rather than matrices, one may also work with infinite dimensions, allowing one to consider arbitrary feature space, possibly induced by some kernel function. Among key theoretical contributions of this paper are new representer theorems, allowing us to develop new general methods that learn finitely many parameters even when working in infinite dimensional user/object feature space. These representer theorems generalize the classical representer theorem for minimization of an empirical loss penalized by the norm in a Reproducing Kernel Hilbert Space (RKHS) to more general penalty functions and function classes.

We also show that, with the appropriate choice of kernels for both users and objects, we may consider a number of existing machine learning methods as special cases of our general framework. In particular, we show that several CF methods such as rank-constrained optimization, trace-norm regularization, and those based on Frobenius norm regularization, can all be cast as special cases of spectral regularization on operator spaces. Moreover, particular choices of kernels lead to specific sub-cases such as regular matrix completion and multi-task learning. In the specific application of collaborative filtering with the presence of attributes, we show that our generalization of these sub-cases leads to better predictive performance.

The outline of the paper is as follows. In Section 2, we review the notion of a compact operator on Hilbert Space, and we show how to cast the collaborative filtering problem within this framework. We then introduce spectral regularization and discuss how rank constraint, trace norm regularization, and Frobenius norm regularization are all special cases of spectral regularization. In Section 3, we show how our general framework encompasses many existing methods with proper choice of the loss function, the kernels, and the spectral regularizer. In Section 4, we provide three representer theorems for operator estimation with spectral regularization that allow for efficient learning algorithms. Finally in Section 5 we present a number of algorithms and describe several techniques to improve efficiency. We test these algorithms in Section 6 on synthetic examples and a widely used movie database.

2. Learning Compact Operators with Spectral Regularization

In this section we propose a mathematical formulation for a general CF problem with spectral regularization. We then show in Section 3 how several learning problems can be cast under this general framework.

2.1 A General CF Formulation

We consider a general CF problem in which our goal is to model the preference of a user described by $\mathbf{x}$ for an item described by $\mathbf{y}$. We denote by $\mathbf{x}$ and $\mathbf{y}$ the data objects containing all relevant or available information; this could, for example, include a unique identifier i for the i-th user or object. Of course, the users and objects may additionally be characterized by known attributes, in which case $\mathbf{x}$ or $\mathbf{y}$ might contain some representation of this extra information. Ultimately, we would like to consider such attribute information as encoded in some positive definite kernel between users, or equivalently between objects. This naturally leads us to model the users as elements in a Hilbert space $\mathcal{X}$, and the objects they rate as elements of another Hilbert space $\mathcal{Y}$.

We assume that our observation data is in the form of *ratings* from users to objects, a real-valued score representing the user's preference for the object. Alternatively, similar methods can be applied when the observations are binary, specifying for instance whether or not a user considered or selected an object.

Given a series of N observations $(\mathbf{x}_i, \mathbf{y}_i, t_i)_{i=1,\ldots,N}$ in $\mathcal{X} \times \mathcal{Y} \times \mathbb{R}$, where t_i represents the rating of user $\mathbf{x}_i$ for object $\mathbf{y}_i$, the generalized CF problem is then to infer a function $f : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$ that can then be used to predict the rating of any user $\mathbf{x} \in \mathcal{X}$ for any object $\mathbf{y} \in \mathcal{Y}$ by $f(\mathbf{x}, \mathbf{y})$. Note that in our notation $\mathbf{x}_i$ and $\mathbf{y}_i$ represent the user and object corresponding to the i-th rating available. If several ratings of a user for different objects are available, as is commonly the case, several $\mathbf{x}_i$'s will be identical in $\mathcal{X}$—a slight abuse of notation. We denote by $\mathcal{X}_N$ and $\mathcal{Y}_N$ the linear spans of $\{\mathbf{x}_i\,, i = 1,\ldots,N\}$ and $\{\mathbf{y}_i\,, i = 1,\ldots,N\}$ in $\mathcal{X}$ and $\mathcal{Y}$, with respective dimensions $m_{\mathcal{X}}$ and $m_{\mathcal{Y}}$.

In the present paper, we uniquely consider learning preference functions $f(\cdot,\cdot)$ that take the form of a linear operator from $\mathcal{X}$ to $\mathcal{Y}$; that is, bilinear forms on $\mathcal{X} \times \mathcal{Y}$,

$$f(\mathbf{x},\mathbf{y}) = \langle \mathbf{x}, F\mathbf{y} \rangle_{\mathcal{X}} \,, \tag{1}$$

for some compact operator F. We now denote by $\mathcal{B}_0(\mathcal{Y},\mathcal{X})$ the set of compact operators from $\mathcal{X}$ to $\mathcal{Y}$. For an introduction to relevant concepts in functional analysis, see Appendix A.

In the general case we consider below, if $\mathcal{X}$ and $\mathcal{Y}$ are not Hilbert spaces, one could also first map (implicitly) users $\mathbf{x}$ and objects $\mathbf{y}$ into possibly infinite dimensional Hilbert feature spaces $\Phi_{\mathcal{X}}(\mathbf{x})$ and $\Psi_{\mathcal{Y}}(\mathbf{y})$ and use kernels. We refer the reader to Appendix A for basic definitions and properties related to compact operators that are useful below. The inference problem can now be stated as follows:

Given a training set of ratings, how may we estimate a "good" compact operator F to predict future ratings?

We estimate the operator F in (1) from the training data using a standard regularization and statistical machine learning approach. In particular, we propose to define the operator as the solution of an optimization problem over $\mathcal{B}_0(\mathcal{Y},\mathcal{X})$ whose objective function balances a data fitting term $R_N(F)$, which is small for operators that can correctly explain the training data, with a regularization term $\Omega(F)$ that controls the complexity of the desired operator. We now describe these two terms in more details.

2.2 Data Fitting Term

Given a *loss function* $\ell(t',t)$ that quantifies how good a prediction $t' \in \mathbb{R}$ is if the true value is $t \in \mathbb{R}$, we consider a fitting term equal to the empirical risk, that is, the mean loss incurred on the training set:

$$R_N(F) = \frac{1}{N}\sum_{i=1}^{N} \ell\left(\langle \mathbf{x}_i, F\mathbf{y}_i \rangle_{\mathcal{X}}, t_i\right) . \tag{2}$$

The particular choice of the loss function should typically depend on the precise problem to be solved and on the nature of the variables t to be predicted. See more details in Section 3. In particular, while the representer theorems presented in Section 4 do not need any convexity with respect to this choice, the algorithms presented in Section 5 do.

2.3 Regularization Term

For the regularization term, we focus on a class of spectral functions defined as follows.

Definition 1 *A function* $\Omega : \mathcal{B}_0(\mathcal{Y},\mathcal{X}) \mapsto \mathbb{R} \cup \{+\infty\}$ *is called a* spectral penalty function *if it can be written as:*

$$\Omega(F) = \sum_{i=1}^{d} s_i\left(\sigma_i(F)\right) , \tag{3}$$

where for any $i \geq 1, s_i : \mathbb{R}^+ \mapsto \mathbb{R}^+ \cup \{+\infty\}$ *is a non-decreasing penalty function satisfying* $s(0) = 0$*, and* $(\sigma_i(F))_{i=1,\ldots,d}$ *are the d singular values of F in decreasing order—d possibly infinite.*

Note that, by the spectral theorem presented in Appendix A, any compact operator can be decomposed into singular vectors, with singular values being a sequence that tends to zero.

Spectral penalty functions include as special cases several functions often encountered in matrix completion problems:

- For a given integer r, consider taking $s_i = 0$ for $i = 1, \ldots, r$, $s_{r+1}(u) = 0$ for $u = 0$, and $s_{r+1}(u) = +\infty$ when $u > 0$. This leads to the function:

$$\Omega(F) = \begin{cases} 0 & \text{if } \operatorname{rank}(F) \leq r, \\ +\infty & \text{otherwise.} \end{cases} \tag{4}$$

  In other words, the set of operators F that satisfy $\Omega(F) < +\infty$ is the set of operators with rank smaller than r.

- Taking $s_i(u) = u$ for all i results in the trace norm penalty (see Appendix A):

$$\Omega(F) = \begin{cases} \|F\|_1 & \text{if } F \in \mathcal{B}_1(\mathcal{Y}, \mathcal{X}), \\ +\infty & \text{otherwise,} \end{cases} \tag{5}$$

  where we note with $\mathcal{B}_1(\mathcal{Y}, \mathcal{X})$ the set of operators with finite trace norm. Such operators are referred to as trace class operators.

- Taking $s_i(u) = u^2$ for all i results in the squared Hilbert-Schmidt norm penalty (also called squared Frobenius norm for matrices, see Appendix A):

$$\Omega(F) = \begin{cases} \|F\|_2^2 & \text{if } F \in \mathcal{B}_2(\mathcal{Y}, \mathcal{X}), \\ +\infty & \text{otherwise,} \end{cases} \tag{6}$$

  where we note with $\mathcal{B}_2(\mathcal{Y}, \mathcal{X})$ the set of operators with finite squared Hilbert-Schmidt norm. Such operators are referred to as Hilbert Schmidt operators.

These particular functions can be combined together in different ways. For example, we may constrain the rank to be smaller than r while penalizing the trace norm of the matrix, which can be obtained by setting $s_i(u) = u$ for $i = 1, \ldots, r$ and $s_{r+1}(u) = +\infty$ if $u > 0$. Alternatively, if we want to penalize the Frobenius norm while constraining the rank, we set $s_i(u) = u^2$ for $i = 1, \ldots, r$ and $s_{r+1}(u) = +\infty$ if $u > 0$. We state these two choices of Ω explicitly since we use these in the experiments (see Section 6) or to design efficient algorithms (see Section 5):

$$\text{Trace+Rank Penalty:} \qquad \Omega(F) = \begin{cases} \|F\|_1 & \text{if } \operatorname{rank}(F) \leq r, \\ +\infty & \text{otherwise.} \end{cases} \tag{7}$$

$$\text{Frobenius+Rank Penalty:} \qquad \Omega(F) = \begin{cases} \|F\|_2^2 & \text{if } \operatorname{rank}(F) \leq r, \\ +\infty & \text{otherwise.} \end{cases}$$

2.4 Operator Inference

With both a fitting term and a regularization term, we can now formally define our inference approach. It consists of finding an operator $\hat{F}$, if there exists one, that solves the following optimization problem:

$$\hat{F} \in \operatorname*{arg\,min}_{F \in \mathcal{B}_0(\mathcal{Y},\mathcal{X})} R_N(F) + \lambda\Omega(F) \,, \tag{8}$$

where $\lambda \in \mathbb{R}_+$ is a parameter that controls the trade-off between fitting and regularization, and where $R_N(F)$ and $\Omega(F)$ are respectively defined in (2) and (3). We note that if the set $\{F \in \mathcal{B}_0(\mathcal{Y},\mathcal{X}) \,, \Omega(F) < +\infty\}$ is not empty, then necessarily the solution $\hat{F}$ of this optimization problem must satisfy $\Omega(\hat{F}) < \infty$.

We show in Sections 4 and 5 how problem (8) can be solved in practice in particular for Hilbert spaces of infinite dimensions. Before exploring such implementation-related issues, in the following section we provide several examples of algorithms that can be derived as particular cases of (8) and highlight their relationships to existing methods.

3. Examples and Related Approaches

The general formulation (8) can result in a variety of practical algorithms potentially useful in different contexts. In particular, three elements can be tailored to one's particular needs: the loss function, the kernels (or equivalently the Hilbert spaces), and the spectral penalty term. We start this section by some generalities about the possible choices for these elements and their consequences, before highlighting some particular combinations of choices relevant for different applications.

1. **The loss function.** The choice of ℓ defines the empirical risk through (2). It is a classical component of many machine learning methods, and should typically depend on the type of data to be predicted (e.g., discrete or continuous) and of the final objective of the algorithm (e.g., classification, regression or ranking). The choice of ℓ also influences the algorithm, as discussed in Section 5. As a deeper discussion about the loss function is only tangential to the current work, we only consider the square loss here, knowing that other convex losses may be considered.

2. **The spectral penalty function.** The choice of $\Omega(F)$ defines the type of constraint we impose on the operator that we seek to learn. In Section 2.3, we gave several examples of such constraints including the rank constraint (4), the trace norm constraint (5), the Hilbert-Schmidt norm constraint (6), or the trace norm constraint over low-rank operators (7). The choice of a particular penalty might be guided by some considerations about the problem to be solved, for example, finding low-rank operators as a way to discover low-dimensional latent structures in the data. On the other hand, from an algorithmic perspective, the choice of the spectral penalty may affect the efficiency or feasibility of our learning algorithm. Certain penalty functions, such as the rank constraint for example, will lead to non-convex problems because the corresponding penalty function (4) is itself not convex. However, the same rank constraint can vastly reduce the number of parameters to be learned. These algorithmic considerations are discussed in more detail in Section 5.

3. **The kernels.** Our choice of kernels defines the inner products (i.e., embeddings) of the users and objects in their respective Hilbert spaces. We may use a variety of possible kernels

depending on the problem to be solved and on the available attributes. Interestingly, the choice of a particular kernel has no influence on the algorithm, as we show later (however, it can easily influence the running time of these algorithms). In the current work, we focus on two basic kernels (Dirac kernels and attribute kernels) and in Section 3.4 we discuss combining these.

- The first kernel we consider is the *Dirac* kernel. When two users (resp. two objects) are compared, the Dirac kernel returns 1 if they are the same user (resp. object), and 0 otherwise. In other words, the Dirac kernel amounts to representing the users (resp. the objects) by orthonormal vectors in $\mathcal{X}$ (resp. in $\mathcal{Y}$). This kernel can be used whether or not attributes are available for users and objects. We denote by $k_D^{\mathcal{X}}$ (resp. $k_D^{\mathcal{Y}}$) the Dirac kernel for the users (resp. objects).
- The second kernel we consider is a kernel between *attributes*, when attributes are available to describe the users and/or objects. We call this an "attribute kernel". This would typically be a kernel between vectors, such as the inner product or a Gaussian RBF kernel, when the descriptions of users and/or objects take the form of vectors of real-valued attributes, or any kernel on structured objects (Shawe-Taylor and Cristianini, 2004). We denote by $k_A^{\mathcal{X}}$ (resp. $k_A^{\mathcal{Y}}$) the attributes kernel for the users (resp. objects).

In the following section we illustrate how specific combinations of loss, spectral penalty and kernels can be relevant for various settings. In particular the choice of kernels leads to new methods for a range of different estimation problems; namely, matrix completion, multi-task learning, and pairwise learning. In Section 3.4 we consider a new representation that allows interpolation between these particular problem formulations.

3.1 Matrix Completion

When the Dirac kernel is used for both users and objects, then we can organize the data $\{\mathbf{x}_i, i = 1, \ldots, n\}$ into $n_{\mathcal{X}}$ groups of identical data points and similarly $\{\mathbf{y}_i, i = 1, \ldots, n\}$ into $n_{\mathcal{Y}}$ groups. Since we use the Dirac kernel, we can represent each of these groups by the elements of the canonical basis $(\mathbf{u}_1, \ldots, \mathbf{u}_{n_{\mathcal{X}}})$ and $(\mathbf{v}_1, \ldots, \mathbf{v}_{n_{\mathcal{Y}}})$ of $\mathbb{R}^{n_{\mathcal{X}}}$ and $\mathbb{R}^{n_{\mathcal{Y}}}$, respectively. A bilinear form using Dirac kernels only depends on the identities of the users and the objects, and we only predict the rating t_i based on the identities of the groups in both spaces. If we assume that each user/object pair is observed at most once, the data can be re-arranged into a $n_{\mathcal{X}} \times n_{\mathcal{Y}}$ incomplete matrix where the learning objective would be to "complete" the missing entries in this matrix (indeed, in this context, it is not possible to generalize to never seen points in $\mathcal{X}$ and $\mathcal{Y}$).

In this case, our bilinear form framework exactly corresponds to completing the matrix, since the bilinear function of $\mathbf{x}$ and $\mathbf{y}$ is exactly equal to $\mathbf{u}_i^\top M \mathbf{v}_j$ where $\mathbf{x} = \mathbf{u}_i$ (i.e., $\mathbf{x}$ is the i-th person) and $\mathbf{y} = \mathbf{v}_j$ (i.e., $\mathbf{y}$ is the j-th object). Thus, the (i, j)-th entry of the matrix M can be assimilated to the value of the bilinear form defined by the matrix M over the pair $(\mathbf{u}_i, \mathbf{v}_j)$. Moreover the spectral regularizer corresponds to the corresponding spectral function of the complete matrix $M \in \mathbb{R}^{n_{\mathcal{X}} \times n_{\mathcal{Y}}}$.

In this context, finding a low-rank approximation of the observed entries in a matrix is an appealing strategy, which corresponds to taking the rank penalty constraint (4) combined with, for example, the square loss error. This however leads to non-convex optimization problems with multiple local minima, for which only local search heuristics are known (Srebro and Jaakkola, 2003). To circumvent this issue, convex spectral penalty functions can be considered. Indeed, in the case of binary preferences, combining the hinge loss function with the trace norm penalty (5) leads to the maximum margin matrix factorization (MMMF) approach proposed by Srebro et al. (2005), which can be rewritten as a semi-definite program. For the sake of efficiency, Rennie and Srebro (2005) proposed to add a constraint on the rank of the matrix, resulting in a non-convex problem that can nevertheless be handled efficiently by classical gradient descent techniques; in our setting, this corresponds to changing the trace norm penalty (5) by the penalty (7).

3.2 Multi-Task Learning

It may be the case that we have attributes only for objects $\mathbf{y}$ (or, similarly, only for users). In that case, for a finite number of users $\{\mathbf{x}_i\,, i = 1, \ldots, N\}$ organized in $n_{\mathcal{X}}$ groups, we aim to estimate a separate function on objects $f_i(\mathbf{y})$ for each of the $n_{\mathcal{X}}$ users i. Considering the estimation of each of these f_i's as a *learning task*, one can possibly learn all f_i's *simultaneously* using a *multi-task learning* approach.

In order to adapt our general framework to this scenario, it is natural to consider the attribute kernel $k_A^{\mathcal{Y}}$ for the objects, whose attributes are available, and the Dirac kernel $k_D^{\mathcal{X}}$ for the users, for which no attributes are used. Again the choice of the loss function depends on the precise task to be solved, and the spectral penalty function can be chosen to enforce some sharing of information between different tasks.

In particular, taking the rank penalty function (4) enforces a decomposition of the tasks (learning each f_i) into a limited number of factors. This results in a method for multi-task learning based on a low-rank representation of the predictor functions f_i. The resulting problem, however, is not convex due to the use of the non-convex rank penalty function. A natural alternative is then to replace the rank constraint by the trace norm penalty function (5), resulting in a convex optimization problem when the loss function is convex. Recently, a similar approach was independently proposed by Amit et al. (2007) in the context of multiclass classification and by Argyriou et al. (2008) for multi-task learning.

Alternatively, another strategy to enforce some constraints among the tasks is to constrain the variance of the different classifiers. Evgeniou et al. (2005) showed that this strategy can be formulated in the framework of support vector machines by considering a *multi-task kernel*, that is, a kernel $k_{multi-task}$ over the product space $\mathcal{X} \times \mathcal{Y}$ defined between any two user/object pairs $(\mathbf{x}, \mathbf{y})$ and $(\mathbf{x}', \mathbf{y}')$ by:

$$k_{multi-task}\left((\mathbf{x},\mathbf{y}),(\mathbf{x}',\mathbf{y}')\right) = \left(k_D^{\mathcal{X}}\left(\mathbf{x},\mathbf{x}'\right) + c\right) k_A^{\mathcal{Y}}\left(\mathbf{y},\mathbf{y}'\right) , \tag{9}$$

where $c > 0$ controls how the variance of the classifiers is constrained compared to the norm of each classifier. As explained in Appendix A, estimating a function over the product space $\mathcal{X} \times \mathcal{Y}$ by penalizing the RKHS norm of the kernel (9) is a particular case of our general framework, where we take the Hilbert-Schmidt norm (6) as spectral penalty function, and where the kernels between users and between objects are respectively $k_D^{\mathcal{X}}\left(\mathbf{x},\mathbf{x}'\right) + c$ and $k_A^{\mathcal{Y}}\left(\mathbf{y},\mathbf{y}'\right)$. When $c = 0$, that is, when we take a Dirac kernel for the users and an attribute kernel for the objects, then penalizing the Hilbert-Schmitt norm amounts to estimating independent models for each user, as explained by Evgeniou et al. (2005). Combining two Dirac kernels for users and objects, respectively, and penalizing the Hilbert-Schmitt norm would not be useful, since the optimal solution would be 0 all points other than training pairs. On the other hand, replacing the Hilbert-Schmidt norm defined by another penalty, such as the trace norm (5), would be an interesting extension when the kernels $k_D^{\mathcal{X}}\left(\mathbf{x},\mathbf{x}'\right) + c$ and $k_A^{\mathcal{Y}}\left(\mathbf{y},\mathbf{y}'\right)$ are used: this would constrain both the variance of the predictor functions f_i and their decomposition into a small number of factors, a potentially nice approach in some multi-task learning applications.

3.3 Pairwise Learning

When attributes are available for both users and objects then it is possible to use the attributes kernels for each. Combining this choice with the Hilbert-Schmidt penalty (6) results in classical machine learning algorithms (e.g., an SVM if the hinge loss is taken as the loss function) applied to the *tensor product* of $\mathcal{X}$ and $\mathcal{Y}$. This strategy is a classical approach to learn a function over pairs of points (see, e.g., Jacob and Vert, 2008). Replacing the Hilbert-Schmidt norm by another spectral penalty function, such as the trace norm, would result in new algorithms for learning low-rank functions over pairs.

3.4 Combining the Attribute and Dirac Kernels

As illustrated in the previous subsections, the setting of the application often determines the combination of kernels to be used for the users and the objects: typically, two Dirac kernels for the standard CF setting without attributes, one Dirac and one attributes kernel for multi-task problems, and two attributes kernels when attributes are available for both users and objects and one wishes to learn over pairs.

There are many situations, however, where the attributes available to describe the users and/or objects are certainly useful for the inference task, but on the other hand do not fully characterize the users and/or objects. For example, if we just know the age and gender of users, we would like to use this information to model their preferences, but would also like to allow for prediction of different preferences for different users even when they share the same age and gender. In our setting, this means that we may want to use the attributes kernel in order to use known attributes from the users and objects during inference, but also the Dirac kernel to incorporate the fact that different users and/or objects remain different even when they share many or all of their attributes.

This naturally leads us to consider the following convex combinations of Dirac and attributes kernels (Abernethy et al., 2006):

$$\begin{cases} k^{\mathcal{X}} = \eta k_{\mathrm{A}}^{\mathcal{X}} + (1-\eta) k_{\mathrm{D}}^{\mathcal{X}}, \\ k^{\mathcal{Y}} = \zeta k_{\mathrm{A}}^{\mathcal{Y}} + (1-\zeta) k_{\mathrm{D}}^{\mathcal{Y}}, \end{cases}$$

where $0 \leq \eta \leq 1$ and $0 \leq \zeta \leq 1$. These kernels interpolate between the Dirac kernels ($\eta = 0$ and $\zeta = 0$) and the attributes kernels ($\eta = 1$ and $\zeta = 1$). Combining this choice of kernels with, for example, the trace norm penalty function (5), allows us to continuously interpolate between different settings corresponding to different "corners" in the (η, ζ) square: standard CF with matrix completion at $(0,0)$, multi-task learning at $(0,1)$ and $(1,0)$, and learning over pairs at $(1,1)$. The extra degree of freedom created when η and ζ are allowed to vary continuously between 0 and 1 provides a principled way to optimally balance the influence of the attributes in the function estimation process.

Note that our representational framework encompasses simpler natural approaches to include attribute information for collaborative filtering: for example, one could consider completing matrices using matrices of the form $UV^\top + U_A R_A^\top + U_A S_A^\top$, where $UV^\top$ is a low-rank matrix to be optimized, U_A and V_A are the given attributes for the first and second domains, and R_A, S_A are parameters to be learned. This formulation corresponds to adding an unconstrained low-rank term $UV^\top$, and the simpler linear predictor from the concatenation of attributes $U_A R_A^\top + U_A S_A^\top$ (Jacob and Vert, 2008). Our approach implicitly adds a fourth cross-product term $U_A T V_A^\top$, where T is estimated from data. This exactly corresponds to imposing that the low rank matrix has a decomposition which includes U_A and V_A as columns. Our combination of Dirac and attribute kernels has the advantage of having specific weights η and ζ that control the trade-off between the constrained and unconstrained low-rank matrices.

4. Representer Theorems

We now present the key theoretical results of this paper and discuss how the general optimization problem (8) can be solved in practice. A first difficulty with this problem is that the optimization space $\{F \in \mathcal{B}_0(\mathcal{Y},\mathcal{X}) : \Omega(F) < \infty\}$ can be of infinite dimension. We note that this can occur even under a rank constraint, because the set $\{F \in \mathcal{B}_0(\mathcal{Y},\mathcal{X}) : \mathrm{rank}(F) \leq R\}$ is not included into any finite-dimensional linear subspace if $\mathcal{X}$ and $\mathcal{Y}$ have infinite dimensions. In this section, we show that the optimization problem (8) can be rephrased as a finite-dimensional problem, and propose practical algorithms to solve it in Section 5. While, as we show in Section 4.1, the reformulation of the problem as a finite-dimensional problem is a simple instance of the representer theorem when the Hilbert-Schmidt norm is used as a penalty function, we prove in Section 4.2 a generalized representer theorem that is valid with any spectral penalty function.

4.1 The Case of the Hilbert-Schmidt Penalty Function

In the particular case where the penalty function $\Omega(F)$ is the Hilbert-Schmidt norm (6), then the set $\{F \in \mathcal{B}_0(\mathcal{Y},\mathcal{X}) : \Omega(F) < \infty\}$ is the set of Hilbert-Schmidt operators. As recalled in Appendix A, this set is a Hilbert space isometric through (1) to the reproducing kernel Hilbert space $\mathcal{H}_\otimes$ of the kernel:

$$k_\otimes\left((\mathbf{x},\mathbf{x}'),(\mathbf{y},\mathbf{y}')\right) = \langle \mathbf{x},\mathbf{x}'\rangle_{\mathcal{X}} \langle \mathbf{y},\mathbf{y}'\rangle_{\mathcal{Y}},$$

and the isometry translates from F to f as:

$$\| f \|_{\mathcal{H}_\otimes}^2 = \| F \|^2 = \Omega(F).$$

As a result, in that case the problem (8) is equivalent to:

$$\min_{f \in \mathcal{H}_\otimes} R_N(f) + \lambda \| f \|_\otimes^2. \tag{10}$$

Therefore the representer theorem for optimization of empirical risks penalized by the RKHS norm (Aronszajn, 1950; Schölkopf et al., 2001) can be applied to show that the solution of (10) necessarily lives in the linear span of the training data. With our notations this translates into the following result:

Theorem 2 *If $\hat{F}$ is a solution of the problem:*

$$\min_{F \in \mathcal{B}_2(\mathcal{Y},\mathcal{X})} R_N(F) + \lambda \sum_{i=1}^{\infty} \sigma_i(F)^2,$$

then it is necessarily in the linear span of $\{\mathbf{x}_i \otimes \mathbf{y}_i : i = 1, \ldots, N\}$, that is, it can be written as:

$$\hat{F} = \sum_{i=1}^{N} \alpha_i \mathbf{x}_i \otimes \mathbf{y}_i, \tag{11}$$

for some $\alpha \in \mathbb{R}^N$.

For the sake of completeness, and to highlight why this result is specific to the Hilbert-Schmidt penalty function (6), we rephrase here, with our notations, the main arguments in the proof of Schölkopf et al. (2001). Any operator F in $\mathcal{B}_2(\mathcal{Y},\mathcal{X})$ can be decomposed as $F = F_S + F_\perp$, where F_S is the projection of F onto the linear span of $\{\mathbf{x}_i \otimes \mathbf{y}_i : i = 1,\ldots,N\}$. $F_\perp$ being orthogonal to each $\mathbf{x}_i \otimes \mathbf{y}_i$ in the training set, one easily gets $R_N(F) = R_N(F_S)$, while $\| F \|^2 = \| F_S \|^2 + \| F_\perp \|^2$ by the Pythagorean theorem. As a result a minimizer F of the objective function must be such that $F_\perp = 0$, that is, must be in the linear span of the training tensor products.

4.2 A Representer Theorem for General Spectral Penalty Functions

Let us now move on to the more general situation (8) where a general spectral function $\Omega(F)$ is used as regularization. Theorem 2 is usually not valid in such a case. Its proof breaks down because it is not true that $\Omega(F) = \Omega(F_S) + \Omega(F_\perp)$ for general Ω, or even that $\Omega(F) \geq \Omega(F_S)$.

The following theorem, whose proof is presented in Appendix B, can be seen as a generalized representer theorem. It shows that a solution of (8), if it exists, can be expanded over a finite basis of dimension $m_\mathcal{X} \times m_\mathcal{Y}$ (where $m_\mathcal{X}$ and $m_\mathcal{Y}$ are the underlying dimensions of the subspaces where the data lie), and that it can be found as the solution of a finite-dimensional optimization problem (with no convexity assumptions on the loss):

Theorem 3 *For any spectral penalty function* $\Omega : \mathcal{B}_0(\mathcal{Y},\mathcal{X}) \mapsto \mathbb{R} \cup \{+\infty\}$, *consider the optimization problem:*

$$\min_{F \in \mathcal{B}_0(\mathcal{Y},\mathcal{X}),} R_N(F) + \lambda \Omega(F) . \tag{12}$$

If the set of solutions is not empty, then there is a solution F in $\mathcal{X}_N \otimes \mathcal{Y}_N$, that is, there exists $\alpha \in \mathbb{R}^{m_\mathcal{X} \times m_\mathcal{Y}}$ such that:

$$F = \sum_{i=1}^{m_\mathcal{X}} \sum_{j=1}^{m_\mathcal{Y}} \alpha_{ij} \mathbf{u}_i \otimes \mathbf{v}_j , \tag{13}$$

where $(\mathbf{u}_1,\ldots,\mathbf{u}_{m_\mathcal{X}})$ and $(\mathbf{v}_1,\ldots,\mathbf{v}_{m_\mathcal{Y}})$ form orthonormal bases of $\mathcal{X}_N$ and $\mathcal{Y}_N$, respectively. Moreover, in that case the coefficients α can be found by solving the following finite-dimensional optimization problem:

$$\min_{\alpha \in \mathbb{R}^{m_\mathcal{X} \times m_\mathcal{Y}}} R_N\left(\operatorname{diag}\left(X \alpha Y^\top\right)\right) + \lambda \Omega(\alpha) , \tag{14}$$

where $\Omega(\alpha)$ refers to the spectral penalty function applied to the matrix α seen as an operator from $\mathbb{R}^{m_\mathcal{Y}}$ to $\mathbb{R}^{m_\mathcal{X}}$, and $X \in \mathbb{R}^{N \times m_\mathcal{X}}$ and $Y \in \mathbb{R}^{N \times m_\mathcal{X}}$ denote any matrices that satisfy $K = XX^\top$ and $G = YY^\top$ for the two $N \times N$ Gram matrices K and G defined by $K_{ij} = \langle \mathbf{x}_i, \mathbf{x}_j \rangle_\mathcal{X}$ and $G_{ij} = \langle \mathbf{y}_i, \mathbf{y}_j \rangle_\mathcal{Y}$, for $0 \leq i,j \leq N$.

This theorem shows that, as soon as a spectral penalty function is used to control the complexity of the compact operators, a solution can be searched in the finite-dimensional space $\mathcal{X}_N \otimes \mathcal{Y}_N$, which in practice boils down to an optimization problem over the set of matrices of size $m_\mathcal{X} \times m_\mathcal{Y}$. The dimension of this space might however be prohibitively large for real-world applications where, for example, tens of thousands of users are confronted to a database of thousands of objects. A convenient way to obtain an important decrease in complexity (at the expense of possibly losing convexity) is by constraining the rank of the operator through an adequate choice of a spectral penalty. Indeed, the set of non-zero singular components of F as an operator is equal to the set of non-zero singular values of α in (13) seen as a matrix. Consequently any constraint on the rank of F as an operator results in a constraint on α as a matrix, from which we deduce:

Corollary 4 *If, in Theorem 3, the spectral penalty function Ω is infinite on operators of rank larger than r (i.e., $\sigma_{r+1}(u) = +\infty$ for $u > 0$), then the matrix $\alpha \in \mathbb{R}^{m_{\mathcal{X}} \times m_{\mathcal{Y}}}$ in (13) has rank at most r.*

As a result, if a rank constraint $\text{rank}(F) \leq r$ is added to the optimization problem then the representer theorem still holds but the dimension of the parameter α becomes $r \times \left(m_{\mathcal{X}} + m_{\mathcal{Y}}\right)$ instead of $m_{\mathcal{X}} \times m_{\mathcal{Y}}$, which is usually beneficial. We note, however, that when a rank constraint is added to the Hilbert-Schmidt norm penalty, then the classical representer Theorem 2 and the expansion of the solution over N vectors (11) are not valid anymore, only Theorem 3 and the expansion (13) can be used. More importantly, this will likely have algorithmic/efficiency consequences.

5. Algorithms

In this section we explain how the optimization problem (14) can be solved in practice. We first consider a general formulation, then we specialize to the situation where many **x**'s and many **y**'s are identical; that is, we are in a matrix completion setting where it may be advantageous to consider other formulations that take into account some group structure explicitly.

5.1 Convex Dual of Spectral Regularization

When the loss is convex, we can derive the convex dual problem, which can be helpful for actually solving the optimization problem. As in the classical representer theorems, this could also provide an alternative proof of the representer theorem in that particular situation.

For all $i = 1, \ldots, N$, we let denote $\psi_i(v_i) = \ell(v_i, t_i)$ the loss corresponding to predicting v_i for the i-th data point. We now assume that each ψ_i is convex (this is usually met in practice). Following Bach et al. (2005), we let $\psi_i^*(\alpha_i)$ denote its Fenchel conjugate defined as $\psi_i^*(\alpha_i) = \max_{v_i \in \mathbb{R}} \alpha_i v_i - \psi_i(v_i)$. Minimizers of the optimization problem defining the conjugate function are often referred to as Fenchel duals to α_i (Boyd and Vandenberghe, 2003). In particular, we have the following classical examples:

- *Least-squares regression*: we have $\psi_i(v_i) = \frac{1}{2}(t_i - v_i)^2$ and $\psi_i^*(\alpha_i) = \frac{1}{2}\alpha_i^2 + \alpha_i t_i$.
- *Logistic regression*: we have $\psi_i(v_i) = \log(1 + \exp(-y_i v_i))$, where $y_i \in \{-1, 1\}$, and $\psi_i^*(\alpha_i) = (1 + \alpha_i t_i)\log(1 + \alpha_i t_i) - \alpha_i t_i \log(-\alpha_i t_i)$ if $\alpha_i t_i \in (-1, 0)$, $+\infty$ otherwise.

We also assume that the spectral regularization is such that for all $i \in \mathbb{N}$, $s_i = s$, where s is a *convex* function such that $s(0) = 0$. In this situation, we have $\Omega(A) = \sum_{i \in \mathbb{N}} s(\sigma_i(A))$. We can also define a Fenchel conjugate for $\Omega(A)$, which is also a spectral function $\Omega^*(B) = \sum_{i \in \mathbb{N}} s^*(\sigma_i(B))$ (Lewis and Sendov, 2002).[1]

Some special cases of interest for $s(\sigma)$ are:

1. Note that results on functions of eigenvalues of symmetric matrices can be extended to functions of singular values of rectangular matrices by using the equivalence between the singular value decomposition of A and the eigenvalue decomposition of $\begin{pmatrix} 0 & A \\ A^\top & 0 \end{pmatrix}$ (Stewart and Sun, 1990).

- $s(\sigma) = |\sigma|$ leads to the trace norm and then $s^*(\tau) = 0$ if $|\tau|$ is less than 1, and $+\infty$ otherwise.
- $s(\sigma) = \frac{1}{2}\sigma^2$ leads to the Frobenius/Hilbert Schmidt norm and then $s^*(\tau) = \frac{1}{2}\tau^2$.
- $s(\sigma) = \varepsilon \log(1 + e^{\sigma/\varepsilon}) + \varepsilon \log(1 + e^{-\sigma/\varepsilon})$ is a smooth approximation of $|\sigma|$, which becomes tighter when ε is closer to zero. We have: $s^*(\tau) = \frac{1}{\varepsilon}(1+\tau)\log(1+\tau) + \frac{1}{\varepsilon}(1-\tau)\log(1-\tau)$. Moreover, $s'(\sigma) = \tau \Leftrightarrow (s^*)'(\tau) = \sigma = \frac{1}{\varepsilon}\log\frac{1+\tau}{1-\tau}$.

Once the representer theorem has been applied, our optimization problem can be rewritten in the *primal* form in (14):

$$\min_{\alpha \in \mathbb{R}^{m_X \times m_Y}} \sum_{i=1}^{N} \psi_i((X\alpha Y^\top)_{ii}) + \lambda\Omega(\alpha). \tag{15}$$

We can now form the Lagrangian, associated with added constraints $v = \operatorname{diag}(X\alpha Y^\top)$ and corresponding Lagrange multiplier $\beta \in \mathbb{R}^N$:

$$\mathcal{L}(v, \alpha, \beta) = \sum_{i=1}^{N} \psi_i(v_i) - \sum_{i=1}^{N} \beta_i (v_i - (X\alpha Y^\top)_{ii}) + \lambda\Omega(\alpha),$$

and minimize with respect to v and W to obtain the *dual* problem, which is to maximize:

$$-\sum_{i=1}^{N} \psi_i^*(\beta_i) - \lambda\Omega^*\left(-\frac{1}{\lambda} X^\top \operatorname{Diag}(\beta) Y\right). \tag{16}$$

Once the optimal dual variable β is found (there are as many of those as there are observations), then we can go back to α (which may or may not be of smaller size), by Fenchel duality, that is, α is among the Fenchel duals of $-\frac{1}{\lambda}X^\top \operatorname{Diag}(\beta)Y$. Thus, when the function s is differentiable and strictly convex (which implies that the Fenchel dual is unique), then we obtain the primal variables α in closed form from the dual variables β. When s is not differentiable, for example, for the trace norm then, following Amit et al. (2007), we can find the primal variables by noting that once β is known, the singular vectors of α are known and we can find the singular values by solving a reduced convex optimization problem.

5.1.1 COMPUTATIONAL COMPLEXITY

Note that for optimization, we have two strategies, following the same two strategies in regular kernel methods: using the primal problem in Eq. (15) of dimension $m_{\mathcal{X}} m_{\mathcal{Y}} \leq n_{\mathcal{X}} n_{\mathcal{Y}}$ (the actual dimension of the underlying data) or using the dual problem in Eq. (16) of dimension N (the number of ratings). The choice between those two formulations is problem dependent.

5.2 Collaborative Filtering

In the presence of (many) identical columns and rows, which is often the case in collaborative filtering situations, the kernel matrices K and L have some columns (and thus rows) which are identical. In such cases, it is computationally more desirable to instead consider the kernel matrices (with their square-root decompositions) $\tilde{K} = \tilde{X}\tilde{X}^\top$ and $\tilde{L} = \tilde{Y}\tilde{Y}^\top$ as the kernel matrices for all distinct elements of $\mathcal{X}$ and $\mathcal{Y}$ (let $n_{\mathcal{X}}$ and $n_{\mathcal{Y}}$ be their sizes). Then each observation $(\mathbf{x}_i, \mathbf{y}_i, t_i)$ corresponds to a pair of indices $(a(i), b(i))$ in $\{1, \ldots, n_{\mathcal{X}}\} \times \{1, \ldots, n_{\mathcal{Y}}\}$, and the primal problem becomes:

$$\min_{\alpha \in \mathbb{R}^{m_{\mathcal{X}} \times m_{\mathcal{Y}}}} \sum_{i=1}^{n} \psi_i(\delta_{a(i)}^\top \tilde{X} \alpha \tilde{Y}^\top \delta_{b(i)}) + \lambda \Omega(\alpha),$$

where δ_u is a vector with only zeroes except at position u. The dual function is

$$-\sum_{i=1}^{N} \psi_i^*(\beta_i) - \lambda \Omega^* \left(-\frac{1}{\lambda} \tilde{X}^\top \sum_{i=1}^{N} \beta_i \delta_{a(i)} \delta_{b(i)}^\top \tilde{Y} \right).$$

Similar to usual kernel machines and the general case presented above, using the primal or the dual formulation for optimization depends on the number of available ratings N compared to the ranks $m_{\mathcal{X}}$ and $m_{\mathcal{Y}}$ of the kernel matrices $\tilde{K}$ and $\tilde{L}$. Indeed, the number of variables in the primal formulation is $m_{\mathcal{X}} m_{\mathcal{Y}}$, while in the dual formulation it is N.

5.3 Low-rank Constrained Problem

We approximate the spectral norm by an infinitely differentiable spectral function. Since we consider in this paper only infinitely differentiable loss functions, our problem is that of minimizing an infinitely differentiable convex function $G(W)$ over rectangular matrices of size $p \times q$ for certain integers p and q. As a result of our spectral regularization, we hope to obtain (approximately) low-rank matrices. In this context, it has proved advantageous to consider low-rank decompositions of the form $W = UV^\top$ where U and V have $m < \min\{p,q\}$ columns (Burer and Monteiro, 2005; Burer and Choi, 2006). Specifically, instead of optimizing a low-rank W directly, we can optimize U and V jointly in training. Burer and Monteiro (2005) have shown that if $m = \min\{p,q\}$ then the non-convex problem of minimizing $G(UV^\top)$ with respect to U and $V^\top$ has no local minima.

We now prove a stronger result in the context of twice differentiable functions, namely that if the global optimum of G has rank $r < \min\{p,q\}$, then the low-rank constrained problem with rank $r+1$ (or any larger rank, for that matter) has no local minimum and its global minimum corresponds to the global minimum of G. The following theorem makes this precise (see Appendix C for proof).

Proposition 5 *Let G be a twice differentiable convex function on matrices of size $p \times q$ with compact level sets. Let $m > 1$ and $(U,V) \in \mathbb{R}^{p \times m} \times \mathbb{R}^{q \times m}$ a local optimum of the function $H : \mathbb{R}^{p \times m} \times \mathbb{R}^{q \times m} \mapsto \mathbb{R}$ defined by $H(U,V) = G(UV^\top)$, that is, U is such that $\nabla H(U,V) = 0$ and the Hessian of H at (U,V) is positive semi-definite. If U or V is rank deficient, then $N = UV^\top$ is a global minimum of G, that is, $\nabla G(N) = 0$.*

The previous proposition shows that if we have a local minimum for the rank-m problem and if the solution is rank deficient, then we have a solution of the global optimization problem. This naturally leads to a sequence of reduced problems of increasing dimension m, smaller than $r+1$, where r is the rank of the global optimum. However, the number of iterations of each of the local minimizations and the final rank m cannot be bounded a priori in general.

Note that using a low-rank representation to solve the trace-norm regularized problem leads to a non-convex minimization problem with no local minima, while simply using the low-rank representation *without* the trace norm penalty and potentially with a Frobenius norm penalty, may lead to local minima. That is, instead of Eq. (14) with the trace norm, we consider the following formulation:

$$\min_{\alpha\in\mathbb{R}^{m_{\mathcal{X}}\times r},\ \beta\in\mathbb{R}^{m_{\mathcal{Y}}\times r}} R_N\left(\mathrm{diag}\left(X\alpha\beta^\top Y^\top\right)\right)+\lambda\sum_{q=1}^{r}\|\alpha(:,k)\|^2\|\beta(:,k)\|^2,$$

where $\alpha(:,k)$ and $\beta(:,k)$ are the k-th columns of α and β. In the simulation section, we compare the two approaches on a synthetic example, and show that the convex formulation solved through a sequence of non-convex formulations leads to better predictive performance.

5.4 Kernel Learning for Spectral Functions

In our collaborative filtering context, there are two potentially useful sources of kernel learning: learning the attribute kernels, or learning the weights η and ζ between Dirac kernels and attribute kernels. In this section, we show how multiple kernel learning (MKL) (Lanckriet et al., 2004; Bach et al., 2004) may be extended to spectral regularization.

We first prove a computationally useful fact about our particular objective function:

Proposition 6 *The dual solution of the optimization problem in Eq. (16) depends only on the matrix $K\otimes G$.*

Proof It suffices to show that for all matrices B, then the positive singular values of $X^\top BY$ only depend on $K\otimes G$. The largest singular value is defined as the maximum of $a^\top X^\top BYb$ over unit norm vectors a and b. By a change of variable, it is equivalent to maximize $\frac{(X^\top\tilde{a})X^\top BY(Y^\top\tilde{b})}{\|X^\top\tilde{a}\|\|Y^\top\tilde{b}\|} = \frac{\mathrm{vec}(\tilde{b}\tilde{a}^\top)(K\otimes G)\mathrm{vec}(B)}{\mathrm{vec}(\tilde{b}\tilde{a}^\top)^\top(K\otimes G)\mathrm{vec}(\tilde{b}\tilde{a}^\top)}$ with respect to $\tilde{a}$ and $\tilde{b}$ (Golub and Loan, 1996). Thus the largest positive singular value is indeed a function of $K\otimes G$. Results for other singular values may be obtained similarly. ■

This shows that the natural kernel matrix to be learned in our context is the Kronecker product $K\otimes G$. We thus follow Lanckriet et al. (2004) and consider M kernel matrices $K_1,\dots,K_M$ for $\mathcal{X}$ and M kernel matrices $G_1,\dots,G_M$ for $\mathcal{Y}$. One possibility could be to follow Lanckriet et al. (2004) and to learn a convex combination of the matrices $K_k\otimes G_k$ by minimizing with respect to the combination weights the optimal value of the problem in Eq. (16). However, unlike the usual Hilbert norm regularization, this does not lead to a convex problem in general. We thus focus on the alternative formulation of the MKL problem (Bach et al., 2004): we consider the sum of the predictor functions associated with each of the individual kernel pairs (K_k,G_k) and penalize by the sum of the norms.

That is, if we let denote $X_1,\dots,X_M$ and $Y_1,\dots,Y_M$ the respective square roots of matrices $K_1,\dots,K_M$ and $G_1,\dots,G_M$, we look for predictor functions which are sums of the M possible atomic predictor functions, and we penalize by the sum of spectral functions, to obtain the following optimization problem:

$$\min_{\forall k,\alpha_k\in\mathbb{R}^{m_x^k\times m_y^k}}\sum_{i=1}^{n}\psi_i\left(\sum_{k=1}^{M}(X_k\alpha_kY_k^\top)_{ii}\right)+\lambda\sum_{k=1}^{M}\Omega(\alpha_k).$$

We form the Lagrangian:

$$\mathcal{L}(v,\alpha_1,\dots,\alpha_M,\beta)=\sum_{i=1}^{n}\psi_i(v_i)-\sum_{i=1}^{N}\beta_i(v_i-\sum_{k=1}^{M}(X_k\alpha_kY_k^\top)_{ii})+\lambda\sum_{k=1}^{M}\Omega(\alpha_k),$$

and minimize w.r.t. v and $\alpha_1, \ldots, \alpha_M$ to obtain the dual problem, which is to maximize

$$-\sum_u \psi_i^*(\beta_i) - \sum_k \lambda \Omega^* \left(-\frac{1}{\lambda} X_k^\top \operatorname{Diag}(\beta) Y_k \right).$$

In the case of the trace norm, we obtain support kernels (Bach et al., 2004), that is, only a sparse combination of matrices ends up being used. Note that in the dual formulation, there is only one β to optimize, and thus it is preferable to use the dual formulation rather than the primal formulation.

6. Experiments

In this Section we present several experimental findings for the algorithms and methods discussed above. Much of the present work was motivated by the problem of collaborative filtering and we therefore focus solely within this domain. As discussed in Section 3, by using operator estimation and spectral regularization as a framework for CF, we may use potentially more information to predict preferences. Our primary goal now is to show that, as one would hope, such capabilities do improve prediction accuracy.

6.1 Data Sets and Metrics

We present several plots created by experimenting on synthetic data. This artificial data set was generated as follows: (1) sample i.i.d. multivariate features for x of dimension 6, (2) generate i.i.d. multivariate features for y of dimension 6 as well, (3) sample z from a random bilinear form in x and y plus some noise, (4) restrict the observed feature space to only 3 features for both x and y. Since part of the data is discarded, the label cannot be perfectly predicted by the known features. On the other hand, since we keep some of them, knowing and using these attributes should work better than not using them. In other words, we expect that setting η and ζ to be values other than 0 or 1 should provide better performance.

We also experimented with the well-known MovieLens 100k data set from the GroupLens Research Group at the University of Minnesota. This data set consists of ratings of 1682 movies by 943 users. Each user provided a rating, in the form of a score from $\{1,2,3,4,5\}$, for a small subset of the movies. Each user rated at least 20 movies, and the total number of ratings available is exactly 100,000, averaging about 105 per user. This data set was rather appropriate as it included attribute information for both the movies and the users. Each movie was labeled with at least one among 19 genres (e.g., action or adventure), while the users' attributes included age, gender, and an occupation among a list of 21 occupations (e.g., administrator or artist). We converted the users' age attribute to a set of binary features identifying one of five age categories.

All test set accuracies are measured as the root mean squared error averaged over 10-fold cross validations. In particular, we focus on the comparisons of intermediate values of η and ζ, compared to the four "corners" of the $\eta/\zeta-$parameter space:

- $\eta = 0, \zeta = 0$: matrix completion
- $\eta = 0, \zeta = 1$ and $\eta = 1, \zeta = 0$: multi-task learning on users or objects
- $\eta = 1, \zeta = 1$: pairwise learning

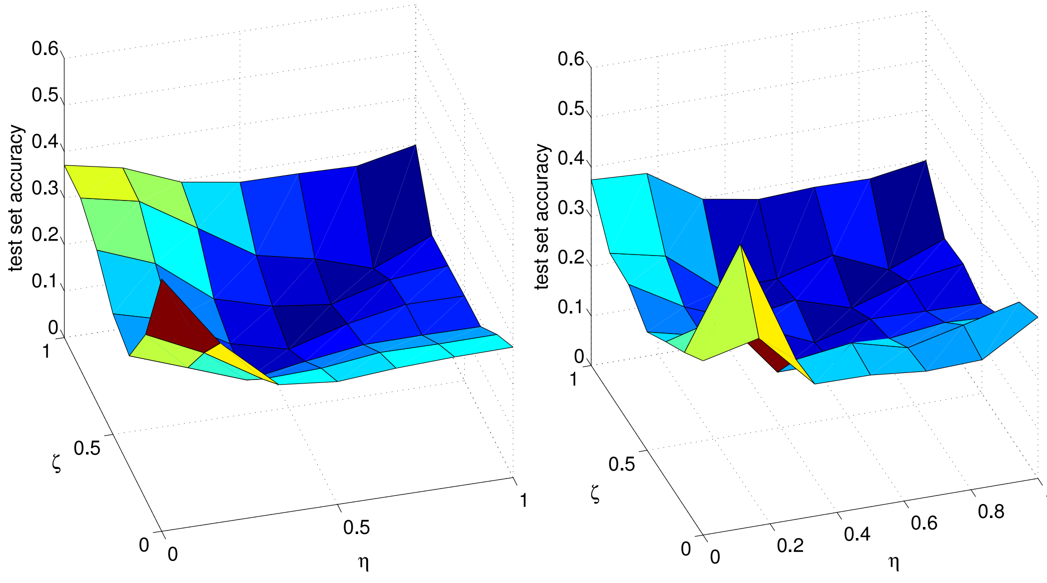


Figure 1: Comparison between two spectral penalties: the trace norm (left) and the Frobenius norm (right), each with an additional fixed rank constraint as described in Section 5.3. Each surface plot displays performance values over a range of η and ζ values, all obtained using the synthetic data set. The minimal value achieved by the trace norm is 0.1222 and the one achieved by the rank constraint is 0.1540.

6.2 Results

We now summarize a number of experimental results for several of the aforementioned algorithms.

6.2.1 TRACENORM VERSUS LOW-RANK

In Figure 1, we present two performance plots over the η/ζ parameter space, both obtained using the synthetic data set. The left plot displays the results when using the trace norm spectral penalty. Here we used the low rank decomposition formulation described in Section 5.3 which (by Proposition 5) has no local minima. The plot on the right uses the same rank-constrained formulation, but with a Frobenius norm penalty instead. The trace norm constrained algorithm performs slightly better. Moreover, best predictive performance is achieved in both cases in the middle of the square and not at any of the four corners.

6.2.2 KERNEL LEARNING

In Figure 2, we show the test set accuracy as a function of the regularization parameter, when we use the kernels corresponding to the four corners as the four basis kernels. We can see that we recover similar performance (error of 0.14 instead of 0.12) than by searching over all η and ζ's. The same algorithm could also be used to learn kernels on the attributes.

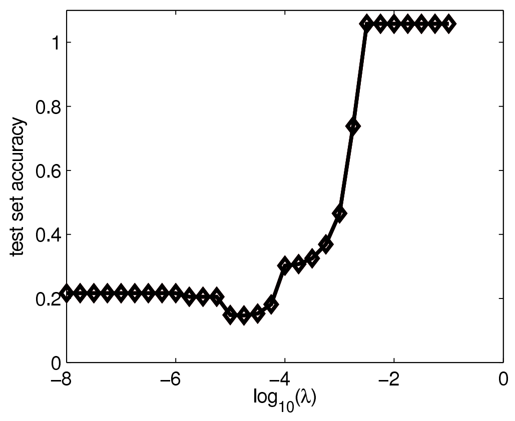


Figure 2: Learning the kernel: test set accuracy vs. regularization parameter. Minimum value is 0.14.

6.2.3 PERFORMANCE ON MOVIELENS DATA

Figure 3 shows the predictive accuracy in RMSE on the MovieLens data set, obtained by 10-fold cross-validation. The heat plot provides some insight on the relative value, for both movies and users, of the given attribute kernels versus the simple identity kernels. The corners have higher values than some of the values inside the square, showing that the best balance between attribute and Dirac kernels is achieved for η, ζ well inside the interior of $[0,1] \times [0,1]$.

7. Conclusions

We have presented a method for solving a generalized matrix completion problem where we have attributes describing the matrix dimensions. The problem is formalized as the problem of inferring a linear compact operator between two general Hilbert spaces, which generalizes the classical finite-dimensional matrix completion problem. We introduced the notion of spectral regularization for operators, which generalizes various spectral penalizations for matrices, and we proved a general representer theorem for this setting. Various approaches, such as standard low rank matrix completion, are special cases of our method. This framework is particularly relevant for CF applications where attributes are available for users and/or objects, and preliminary experiments confirm the benefits of our method.

An interesting direction of future research is to explore further the multi-task learning algorithm we obtained with low-rank constraint, and to study the possibility to derive on-line implementations that may better fit the need for large-scale applications where training data are continuously increasing. On the theoretical side, a better understanding of the effects of norm and rank regularizations and their interaction would be of considerable interest.

Acknowledgments

Francis Bach and Jean Phillipe-Vert are supported by ANR grant ANR-07-BLAN-0311-03.

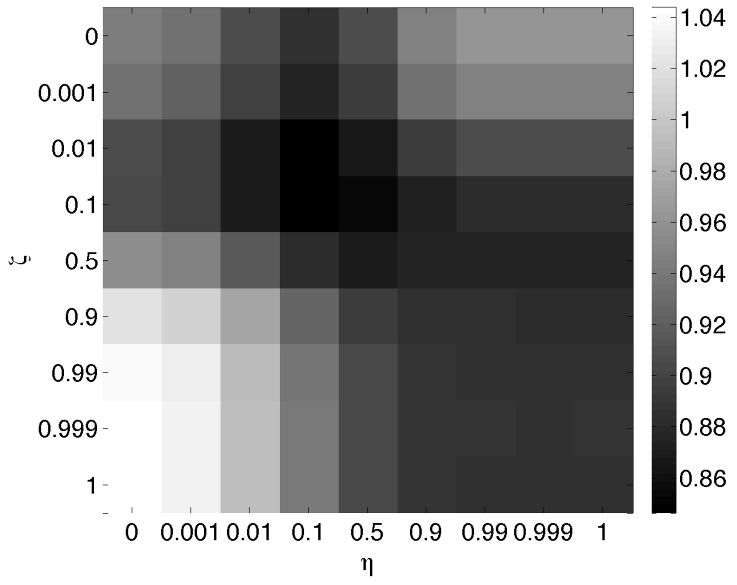


Figure 3: A heat plot of performance for a range of kernel parameter choices, η and ζ, using the MovieLens data set.

Appendix A. Compact Operators on Hilbert Spaces

In this appendix, we recall basic definitions and properties of Hilbert space operators. We refer the interested reader to general books (Brezis, 1980; Berlinet and Thomas-Agnan, 2003) for more details.

Let $\mathcal{X}$ and $\mathcal{Y}$ be two Hilbert spaces, with respective inner products denoted by $\langle \mathbf{x}, \mathbf{x}' \rangle_{\mathcal{X}}$ and $\langle \mathbf{y}, \mathbf{y}' \rangle_{\mathcal{Y}}$ for $\mathbf{x}, \mathbf{x}' \in \mathcal{X}$ and $\mathbf{y}, \mathbf{y}' \in \mathcal{Y}$. We denote by $\mathcal{B}(\mathcal{Y}, \mathcal{X})$ the set of bounded operators from $\mathcal{X}$ to $\mathcal{Y}$, that is, of continuous linear mappings from $\mathcal{Y}$ to $\mathcal{X}$. For any two elements $(\mathbf{x}, \mathbf{y})$ in $\mathcal{X} \times \mathcal{Y}$, we denote by $\mathbf{x} \otimes \mathbf{y}$ their *tensor product*, that is, the linear operator from $\mathcal{Y}$ to $\mathcal{X}$ defined by:

$$\forall \mathbf{h} \in \mathcal{Y}, \quad (\mathbf{x} \otimes \mathbf{y})\, \mathbf{h} = \langle \mathbf{y}, \mathbf{h} \rangle_{\mathcal{Y}} \mathbf{x}.$$

We denote by $\mathcal{B}_0(\mathcal{Y}, \mathcal{X})$ the set of *compact* linear operators from $\mathcal{Y}$ to $\mathcal{X}$, that is, the set of linear operators that map the unit ball of $\mathcal{Y}$ to a relatively compact set of $\mathcal{X}$. Alternatively, they can also be defined as the limit of finite rank operators.

When $\mathcal{X}$ and $\mathcal{Y}$ have finite dimensions, then $\mathcal{B}_0(\mathcal{Y}, \mathcal{X})$ is simply the set of linear mappings from $\mathcal{Y}$ to $\mathcal{X}$, which can be represented by the set of matrices of dimensions $\dim(\mathcal{X}) \times \dim(\mathcal{Y})$. In that case the tensor product $\mathbf{x} \otimes \mathbf{y}$ is represented by the matrix $\mathbf{x}\mathbf{y}^\top$, where $\mathbf{y}^\top$ denotes the transpose of $\mathbf{y}$.

For general Hilbert spaces $\mathcal{X}$ and $\mathcal{Y}$, any compact linear operator $F \in \mathcal{B}_0(\mathcal{Y}, \mathcal{X})$ admits a *spectral decomposition*:

$$F = \sum_{i=1}^{\infty} \sigma_i \mathbf{u}_i \otimes \mathbf{v}_i\,. \tag{17}$$

Here the the *singular values* $(\sigma_i)_{i\in\mathbb{N}}$ form a sequence of non-negative real numbers such that $\lim_{i\to\infty}\sigma_i = 0$, and $(\mathbf{u}_i)_{i\in\mathbb{N}}$ and $(\mathbf{v}_i)_{i\in\mathbb{N}}$ form orthonormal families in $\mathcal{X}$ and $\mathcal{Y}$, respectively. Although the vectors $(\mathbf{u}_i)_{i\in\mathbb{N}}$ and $(\mathbf{v}_i)_{i\in\mathbb{N}}$ in (17) are not uniquely defined for a given operator F, the set of singular values is uniquely defined. By convention we denote by $\sigma_1(F), \sigma_2(F), \ldots$, the successive singular values of F ranked by decreasing order. The *rank* of F is the number $\operatorname{rank}(F) \in \mathbb{N}\cup\{+\infty\}$ of strictly positive singular values.

We now describe three subclasses of compact operators of particular relevance in the rest of this paper.

- The set of operators with finite rank is denoted $\mathcal{B}_F(\mathcal{Y},\mathcal{X})$.
- The operators $F \in \mathcal{B}_0(\mathcal{Y},\mathcal{X})$ that satisfy:
$$\sum_{i=1}^{\infty}\sigma_i(F)^2 < \infty$$
are called *Hilbert-Schmidt* operators. They form a Hilbert space, denoted $\mathcal{B}_2(\mathcal{Y},\mathcal{X})$, with inner product $\langle\cdot,\cdot\rangle_{\mathcal{X}\otimes\mathcal{Y}}$ between basic tensor products given by:
$$\langle \mathbf{x}\otimes\mathbf{y}, \mathbf{x}'\otimes\mathbf{y}'\rangle_{\mathcal{X}\otimes\mathcal{Y}} = \langle \mathbf{x},\mathbf{x}'\rangle_{\mathcal{X}}\langle \mathbf{y},\mathbf{y}'\rangle_{\mathcal{Y}}\,. \tag{18}$$
In particular, the Hilbert-Schmidt norm of an operator in $\mathcal{B}_2(\mathcal{Y},\mathcal{X})$ is given by:
$$\|F\|_2 = \left(\sum_{i=1}^{\infty}\sigma_i(F)^2\right)^{\frac{1}{2}}.$$
Another useful characterization of Hilbert-Schmidt operators is the following. Each linear operator $F:\mathcal{Y}\to\mathcal{X}$ uniquely defines a bilinear function $f_H:\mathcal{X}\times\mathcal{Y}\to\mathbb{R}$ by
$$f(\mathbf{x},\mathbf{y}) = \langle \mathbf{x}, F\mathbf{y}\rangle_{\mathcal{X}}\,.$$
The set of functions f_F associated to the Hilbert-Schmidt operators forms itself a Hilbert space of functions $\mathcal{X}\times\mathcal{Y}\to\mathbb{R}$, which is the reproducing kernel Hilbert space of the product kernel defined for $((\mathbf{x},\mathbf{y}),(\mathbf{x}',\mathbf{y}'))\in(\mathcal{X}\times\mathcal{Y})^2$ by
$$k_{\otimes}\left((\mathbf{x},\mathbf{y}),(\mathbf{x}',\mathbf{y}')\right) = \langle \mathbf{x},\mathbf{x}'\rangle_{\mathcal{X}}\langle \mathbf{y},\mathbf{y}'\rangle_{\mathcal{Y}}\,.$$
- The operators $F \in \mathcal{B}_0(\mathcal{Y},\mathcal{X})$ that satisfy:
$$\sum_{i=1}^{\infty}\sigma_i(F) < \infty$$
are called *trace-class* operators. The set of trace-class operators is denoted $\mathcal{B}_1(\mathcal{Y},\mathcal{X})$. The *trace norm* of an operator $F\in\mathcal{B}_1(\mathcal{Y},\mathcal{X})$ is given by:
$$\|F\|_1 = \sum_{i=1}^{\infty}\sigma_i(F)\,.$$

Obviously the following ordering exists among these various classes of operators:
$$\mathcal{B}_F(\mathcal{Y},\mathcal{X}) \subset \mathcal{B}_1(\mathcal{Y},\mathcal{X}) \subset \mathcal{B}_2(\mathcal{Y},\mathcal{X}) \subset \mathcal{B}_0(\mathcal{Y},\mathcal{X}) \subset \mathcal{B}(\mathcal{Y},\mathcal{X})\,,$$
and all inclusions are equalities if $\mathcal{X}$ and $\mathcal{Y}$ have finite dimensions.

Appendix B. Proof of Theorem 3

We start with a general result about the decrease of singular values for compact operators composed with projection:

Lemma 7 *Let $\mathcal{G}$ and $\mathcal{H}$ be two Hilbert spaces, H a compact linear subspace of $\mathcal{H}$, and Π_H denote the orthogonal projection onto H. Then for any compact operator $F : \mathcal{G} \mapsto \mathcal{H}$ it holds that:*

$$\forall i \geq 1\,, \quad \sigma_i(\Pi_H F) \leq \sigma_i(F)\,.$$

Proof We use the classical characterization of the i-th singular value:

$$\sigma_i(F) = \max_{V \in \mathcal{V}_i(\mathcal{G})} \min_{\mathbf{x} \in V, \|\mathbf{x}\|_{\mathcal{G}}=1} \|F\mathbf{x}\|_{\mathcal{H}}\,,$$

where $\mathcal{V}_i(\mathcal{G})$ denotes the set of all linear subspaces of $\mathcal{G}$ of dimension i. Now, observing that for any $\mathbf{x}$ we have $\|\Pi_H F\mathbf{x}\|_{\mathcal{H}} \leq \|F\mathbf{x}\|_{\mathcal{H}}$ proves the Lemma. ■

Given a training set of patterns $(\mathbf{x}_i, \mathbf{y}_i)_{i=1,\ldots,N} \in \mathcal{X} \times \mathcal{Y}$, remember that we denote by $\mathcal{X}_N$ and $\mathcal{Y}_N$ the linear subspaces of $\mathcal{X}$ and $\mathcal{Y}$ spanned by the training patterns $\{\mathbf{x}_i\,, i = 1, \ldots, N\}$ and $\{\mathbf{y}_i\,, i = 1, \ldots, N\}$, respectively. For any operator $F \in \mathcal{B}_0(\mathcal{Y}, \mathcal{X})$, let us now consider the operator $G = \Pi_{\mathcal{X}_N} F \Pi_{\mathcal{Y}_N}$. By construction, F and G agree on the training patterns, in the sense that for $i = 1, \ldots, N$:

$$\langle \mathbf{x}_i, G\mathbf{y}_i \rangle_{\mathcal{X}} = \left\langle \mathbf{x}_i, \Pi_{\mathcal{X}_N} F \Pi_{\mathcal{Y}_N} \mathbf{y}_i \right\rangle_{\mathcal{X}} = \left\langle \Pi_{\mathcal{X}_N} \mathbf{x}_i, F \Pi_{\mathcal{Y}_N} \mathbf{y}_i \right\rangle_{\mathcal{X}} = \langle \mathbf{x}_i, F\mathbf{y}_i \rangle_{\mathcal{X}}\,.$$

Therefore F and G have the same empirical risk:

$$R_N(F) = R_N(G)\,. \tag{19}$$

Now, by denoting F^* the adjoint operator, we can use Lemma 7 and the fact that the singular values of an operator and its adjoint are the same to obtain, for any $i \geq 1$:

$$\begin{aligned} \sigma_i(G) &= \sigma_i(\Pi_{\mathcal{X}_N} F \Pi_{\mathcal{Y}_N}) \\ &\leq \sigma_i(F \Pi_{\mathcal{Y}_N}) \\ &= \sigma_i(\Pi_{\mathcal{Y}_N} F^*) \\ &\leq \sigma_i(F^*) \\ &= \sigma_i(F). \end{aligned}$$

This implies that the spectral penalty term satisfies $\Omega(G) \leq \Omega(F)$. Combined with (19), this shows that if F is a solution to (12), then $G = \Pi_{\mathcal{X}_N} F \Pi_{\mathcal{Y}_N}$ is also a solution. Observing that $G \in \mathcal{X}_N \otimes \mathcal{Y}_N$ concludes the proof of the first part of Theorem 3, resulting in (13).

We have now reduced the optimization problem in $\mathcal{B}_0(\mathcal{Y}, \mathcal{X})$ to a finite-dimensional optimization over the matrix α of size $m_{\mathcal{X}} \times m_{\mathcal{Y}}$. Let us now rephrase the optimization problem in this finite-dimensional space.

Let us first consider the spectral penalty term $\Omega(F)$. Given the decomposition (13), the non-zero singular values of F as an operator are exactly the non-zero singular values of α as a matrix, as soon as $(\mathbf{u}_1, \ldots, \mathbf{u}_{m_{\mathcal{X}}})$ and $\left(\mathbf{v}_1, \ldots, \mathbf{v}_{m_{\mathcal{Y}}}\right)$ form *orthonormal* bases of $\mathcal{X}_N$ and $\mathcal{Y}_N$, respectively. In order to be able to express the empirical risk $R_N(F)$ we must however consider a decomposition of F over the training patterns, as:

$$F = \sum_{i=1}^{N} \sum_{j=1}^{N} \gamma_{ij} \mathbf{x}_i \otimes \mathbf{y}_j \,. \tag{20}$$

In order to express the singular values from this expression let us introduce the *Gram matrices* K and G of the training patterns, that is, the $N \times N$ matrices defined for $i, j = 1, \ldots, N$ by:

$$K_{ij} = \left\langle \mathbf{x}_i, \mathbf{x}_j \right\rangle_{\mathcal{X}} \,, \quad G_{ij} = \left\langle \mathbf{y}_i, \mathbf{y}_j \right\rangle_{\mathcal{Y}} \,.$$

We note that by definition the ranks of K and G are respectively $m_{\mathcal{X}}$ and $m_{\mathcal{Y}}$. Let us now factorize these two matrices as $K = XX^\top$ and $G = YY^\top$, where $X \in \mathbb{R}^{N \times m_{\mathcal{X}}}$ and $Y \in \mathbb{R}^{N \times m_{\mathcal{Y}}}$ are any square roots, for example, obtained by kernel PCA or incomplete Cholesky decomposition (Fine and Scheinberg, 2001; Bach and Jordan, 2005). The matrices X and Y provide a representation of the pattern in two orthonormal bases which we denote by $(\mathbf{u}_1, \ldots, \mathbf{u}_{m_{\mathcal{X}}})$ and $\left(\mathbf{v}_1, \ldots, \mathbf{v}_{m_{\mathcal{Y}}}\right)$. In particular we have, for any $i, j \in 1, \ldots, N$:

$$\mathbf{x}_i \otimes \mathbf{y}_j = \sum_{l=1}^{m_{\mathcal{X}}} \sum_{m=1}^{m_{\mathcal{Y}}} X_{il} Y_{jm} \mathbf{u}_l \otimes \mathbf{v}_m \,,$$

from which we deduce:

$$F = \sum_{l=1}^{m_{\mathcal{X}}} \sum_{m=1}^{m_{\mathcal{Y}}} \left(\sum_{i=1}^{N} \sum_{j=1}^{N} X_{il} Y_{jm} \gamma_{ij} \right) \mathbf{u}_l \otimes \mathbf{v}_m \,.$$

Comparing this expression to (13) we deduce that:

$$\alpha = X^\top \gamma Y \,.$$

The empirical error $R_N(F)$ is a function of $f(\mathbf{x}_l, \mathbf{y}_l)$ for $l = 1, \ldots, N$. From (20), we see that:

$$f(\mathbf{x}_l, \mathbf{y}_l) = \sum_{i=1}^{N} \sum_{j=1}^{N} \gamma_{ij} K_{il} G_{lj} \,,$$

and therefore the vector of predictions $F_N = (f(\mathbf{x}_l, \mathbf{y}_l))_{l=1,\ldots,N} \in \mathbb{R}^N$ can be rewritten as:

$$F_N = \operatorname{diag}(K\gamma G) = \operatorname{diag}\left(X\alpha Y^\top\right) \,.$$

We can now replace the empirical risk $R_N(F_N)$ by $R_N\left(\operatorname{diag}\left(X\alpha Y^\top\right)\right)$ and the penalty $\Omega(F)$ by $\Omega(\alpha)$ to deduce the optimization problem (14) from (12), which concludes the proof of Theorem 3.

Appendix C. Proof of Proposition 5

Since the function has compact level sets, we may assume that we are restricted to an open bounded subset of $\mathbb{R}^{p \times q}$ where the second and first derivatives are uniformly bounded. We let denote $C > 0$ a common upper bound of all derivatives. The gradient of the function H is equal to $\nabla H = \binom{\nabla G^\top U}{\nabla G \; V}$, while the Hessian of H is the following quadratic form:

$$\nabla^2 H[(dU, dV), (dU, dV)] = 2 \operatorname{tr} dV^\top \nabla G dU + \nabla^2 G[U dV^\top + dU V^\top, U dV^\top + dU V^\top].$$

Without loss of generality, we may assume that the last columns of U and V are equal to zero (this can be done by rotation of U or V). The zero gradient assumption implies that $\nabla G^\top U = 0$ and $\nabla G V = 0$. While if we take dU and dV with the first $m-1$ columns equal to zero, and last columns equal to arbitrary u and v, then the second term in the Hessian is equal to zero. The positivity of the first term implies that for all u and v, $v^\top \nabla G u \geq 0$, that is, the gradient of G at $N = UV^\top$ is equal to zero, and thus we get a stationary point and thus a global minimum of G.

References

J. Abernethy, F. Bach, T. Evgeniou, and J.-P. Vert. Low-rank matrix factorization with attributes. Technical Report N24/06/MM, Ecole des Mines de Paris, 2006.

Y. Amit, M. Fink, N. Srebro, and S. Ullman. Uncovering shared structures in multiclass classification. In Z. Ghahramani, editor, *ICML '07: Proceedings of the 24th International Conference on Machine Learning*, pages 17–24, New York, NY, USA, 2007. ACM.

A. Argyriou, T. Evgeniou, and M. Pontil. Convex multi-task feature learning. *Mach. Learn.*, 73(3): 243–272, 2008.

N. Aronszajn. Theory of reproducing kernels. *Trans. Am. Math. Soc.*, 68:337 – 404, 1950.

F. R. Bach. Consistency of trace norm minimization. *J. Mach. Learn. Res.*, 9:1019–1048, 2008.

F. R. Bach and M. I. Jordan. Predictive low-rank decomposition for kernel methods. In *ICML '05: Proceedings of the 22nd International Conference on Machine Learning*, pages 33–40, New York, NY, USA, 2005. ACM.

F. R. Bach, G. R. G. Lanckriet, and M. I. Jordan. Multiple kernel learning, conic duality, and the SMO algorithm. In *ICML '04: Proceedings of the Twenty-first International Conference on Machine Learning*, page 6, New York, NY, USA, 2004. ACM.

F. R. Bach, R. Thibaux, and M. I. Jordan. Computing regularization paths for learning multiple kernels. In L. K. Saul, Y. Weiss, and L. Bottou, editors, *Adv. Neural. Inform. Process Syst. 17*, pages 73–80, Cambridge, MA, 2005. MIT Press.

A. Berlinet and C. Thomas-Agnan. *Reproducing Kernel Hilbert Spaces in Probability and Statistics*. Kluwer Academic Publishers, 2003.

S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge Univ. Press, 2003.

J. S. Breese, D. Heckerman, and C. Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In *14th Conference on Uncertainty in Artificial Intelligence*, pages 43–52, Madison, W.I., 1998. Morgan Kaufman.

H. Brezis. *Analyse Fonctionnelle*. Masson, 1980.

S. A. Burer and C. Choi. Computational enhancements in low-rank semidefinite programming. *Optimization Methods and Software*, 21:493–512, 2006.

S. A. Burer and R. D. C. Monteiro. Local minima and convergence in low-rank semidefinite programming. *Mathematical Programming*, 103:427–444, 2005.

T. Evgeniou, C. Micchelli, and M. Pontil. Learning multiple tasks with kernel methods. *J. Mach. Learn. Res.*, 6:615–637, 2005.

M. Fazel, H. Hindi, and S. Boyd. A rank minimization heuristic with application to minimum order system approximation. In *Proceedings of the 2001 American Control Conference*, volume 6, pages 4734–4739, 2001.

S. Fine and K. Scheinberg. Efficient SVM training using low-rank kernel representations. *J. Mach. Learn. Res.*, 2:243–264, 2001.

G. H. Golub and C. F. Van Loan. *Matrix Computations*. Johns Hopkins University Press, 1996.

D. Heckerman, D. M. Chickering, C. Meek, R. Rounthwaite, and C. Kadie. Dependency networks for inference, collaborative filtering, and data visualization. *J. Mach. Learn. Res.*, 1:49–75, 2000.

L. Jacob and J.-P. Vert. Efficient peptide-MHC-I binding prediction for alleles with few known binders. *Bioinformatics*, 24(3):358–366, Feb 2008.

G. R. G. Lanckriet, N. Cristianini, L. El Ghaoui, P. Bartlett, and M. I. Jordan. Learning the kernel matrix with semidefinite programming. *J. Mach. Learn. Res.*, 5:27–72, 2004.

A. S. Lewis and H. S. Sendov. Twice differentiable spectral functions. *SIAM J. Mat. Anal. App.*, 23 (2):368–386, 2002.

J. D. M. Rennie and N. Srebro. Fast maximum margin matrix factorization for collaborative prediction. In *Proceedings of the 22nd International Conference on Machine Learning*, pages 713–719, New York, NY, USA, 2005. ACM Press.

R. Salakhutdinov, A. Mnih, and G. Hinton. Restricted boltzmann machines for collaborative filtering. In *ICML '07: Proceedings of the 24th International Conference on Machine Learning*, pages 791–798, New York, NY, USA, 2007. ACM.

B. Schölkopf, R. Herbrich, and A. J. Smola. A generalized representer theorem. In *Proceedings of the 14th Annual Conference on Computational Learning Theory*, volume 2011 of *Lecture Notes in Computer Science*, pages 416–426, Berlin / Heidelberg, 2001. Springer.

J. Shawe-Taylor and N. Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, 2004.

N. Srebro and T. Jaakkola. Weighted low-rank approximations. In T. Fawcett and N. Mishra, editors, *Proceedings of the Twentieth International Conference on Machine Learning*, pages 720–727. AAAI Press, 2003.

N. Srebro, J. D. M. Rennie, and T. S. Jaakkola. Maximum-margin matrix factorization. In L. K. Saul, Y. Weiss, and L. Bottou, editors, *Adv. Neural. Inform. Process Syst. 17*, pages 1329–1336, Cambridge, MA, 2005. MIT Press.

G. W. Stewart and J. Sun. *Matrix Perturbation Theory*. Academic Press, 1990.

Journal of Machine Learning Research 10 (2009) 827-856 Submitted 9/07; Revised 8/08; Published 4/09

Consistency and Localizability

Alon Zakai ALON.ZAKAI@MAIL.HUJI.AC.IL
Interdisciplinary Center for Neural Computation
Hebrew University of Jerusalem
Jerusalem, Israel 91904

Ya'acov Ritov YAACOV.RITOV@HUJI.AC.IL
Department of Statistics
Interdisciplinary Center for Neural Computation and Center for the Study of Rationality
Hebrew University of Jerusalem
Jerusalem, Israel 91905

Editor: Gabor Lugosi

Abstract

We show that all consistent learning methods—that is, that asymptotically achieve the lowest possible expected loss for any distribution on (X,Y)—are necessarily localizable, by which we mean that they do not significantly change their response at a particular point when we show them only the part of the training set that is close to that point. This is true in particular for methods that appear to be defined in a non-local manner, such as support vector machines in classification and least-squares estimators in regression. Aside from showing that consistency implies a specific form of localizability, we also show that consistency is logically equivalent to the combination of two properties: (1) a form of localizability, and (2) that the method's global mean (over the entire X distribution) correctly estimates the true mean. Consistency can therefore be seen as comprised of two aspects, one local and one global.

Keywords: consistency, local learning, regression, classification

1. Introduction

In a supervised learning problem we are given an i.i.d sample $S_n = \{(x_i, y_i)\}_{i=1..n}$ of size n from some distribution P; then a new pair (x,y) is drawn from the same P and our goal is to predict y when shown only x. Our prediction (also called estimate, or guess) of y is written $f(S_n,x)$, some function that depends on the training set and the point at which we estimate (note that it is slightly atypical to have both the training set and point as inputs to f, but this will be very convenient in our setting). We call f a *learning method*; in the context of regression we will also use the term *estimator*, and in the context of classification we will use the term *classifier*. In both of these settings, if f achieves the lowest possible expected loss as $n \to \infty$, for every distribution, then we call f *consistent* (we will formalize all of these definitions later on; for now we just sketch the general ideas). Consistent estimators are of obvious interest due to their capability to learn without knowing in advance anything about the actual distribution.

When we look at the learning methods known to be consistent, we can separate them into two general types. In the first of these we have methods that are defined in a local manner, for example, the k-nearest-neighbor (k-NN) classifier (Stone, 1977; Devroye et al., 1996). The k-NN classifier guesses the class of a point x based on its nearest neighbors in the training set, thus, this classifier behaves in a 'local' way: only close-by points affect the estimate. More generally, by a locally-behaving method we mean one that, given a training set S_n and a point x at which to estimate the value of y, in some way treats the close-by part of the training set as the most important. A second type of consistent learning method is defined in a global manner, for example, support vector machines (see Vapnik, 1998; Steinwart, 2002, for a description and proof of consistency, respectively). It is not clear from the definition of support vector machines whether they behave locally or not: The separating hyperplane is determined based on the entire training set S_n, and furthermore does not depend on the specific point x at which we classify, perhaps leading us to expect that support vector machines do *not* behave locally. Thus, on an intuitive level we might think that some consistent methods behave locally and some do not.

This intuition also appears relevant when we consider regression: The k-NN regression estimator appears to behave locally, while on the other hand support vector regression (see, e.g., Smola and Schoelkopf, 1998), kernel ridge regression (Saunders et al., 1998), etc., seem not to have that property. Another example is that of orthogonal series estimation, that is, of using a weighted sum of fixed harmonic functions (Lugosi and Zeger, 1995); this method appears to not behave locally both because the harmonic functions are non-local and because the coefficients are determined in a way based on all of the data.

Despite the intuition that some consistent methods might not behave locally, we will see that in fact *all* of them necessarily behave in that manner. As mentioned before, we already know that some locally-behaving methods are consistent, since some are in fact defined in a local manner, for example, k-NN. What we will see is that all other consistent methods must *also* behave locally. In classification, this implies that, in particular, (properly regularized) support vector machines and boosting (Freund and Schapire, 1999; Zhang, 2004; Bartlett and Traskin, 2007) must behave locally, despite being defined in a way that appears global. In the area of regression, our results show that neural network estimators, orthogonal series estimators, etc., must behave locally if they are consistent, again, despite their being defined in a way that does not indicate such behavior.

In the rest of this introductory section we will present a summary of our approach and results as well as background regarding related work. While doing so we focus on regression problems since that setting allows for simpler and clearer definitions. For the same reasons we will also focus on regression in the main part of this paper; in a later section we will show how to apply our results to classification.

Our goal in regression is to estimate $f^*(x) = E(y|x)$, that is, the expected value of y conditioned on x, or the regression of y on x. Our hope is that $f(S_n,x)$ is close to $f^*(x)$. We say that f is consistent on a distribution P iff

$$L_n(f) \equiv E\left|f(S_n,x) - f^*(x)\right| \xrightarrow[n\to\infty]{} 0$$

where the expected value is taken over training sets S_n and observations x both distributed according to P (which is suppressed in the notation). If a method is consistent on all P then we call it consistent (this is sometimes called *universal consistency*). Note that there are stronger notions of consistency, such as requiring that the loss converge to 0 with probability 1 (see, e.g., Gyorfi et al., 2002), but we will focus on the loss as just described. Note also that the L_n loss is 'global' in that we average over all x, which makes it all the more interesting to see whether methods that minimize it must end up behaving locally.

The notion of local behavior that we will consider will be called *localizability*, and it entails that the method returns a similar estimate for x when shown S_n in comparison to the estimate it would have returned when shown only the part of S_n that is close to x. In other words, if we define

$$S_n(x,r) = \{(x_i,y_i) \in S_n \,:\, ||x_i - x|| \leq r\}$$

then a localizable method has the property that

$$f(S_n,x) \approx f(S_n(x,r),x)$$

for some small $r > 0$ (the formal definitions of all of these concepts will be given in later sections). More specifically, for any sequence $R_n \searrow 0$ we can see $g(S_n,x) = f\left(S_n\big(x,R_n\big),x\right)$ as a 'localization' of f, since it applies f to the close-by part of the training set for the particular point at which we estimate. In other words, a localizable method is one that behaves similarly to a localization of itself. (Note the convenience of the $f(S_n,x)$ notation here, that is, of seeing f as a function of both S_n and x.)

Why is the concept of localizability of interest? The main motivation for us is that, as we will see later, consistency implies a form of localizability. That is, even an estimator defined in what seems to be a global manner, for example, by minimizing a global loss of the general form

$$\frac{1}{n}\sum_{i=1..n} l\big(f(x_i),y_i\big) + \lambda c(f)$$

where l is, for example, least-squares, and c is (optional) complexity penalization—then even such an estimator must be in some sense localizable, if it is consistent. Thus, our first motivation is to point out that not only locally-defined methods like k-NN behave locally, but also all other consistent methods as well.

Aside from this main motivation for investigating localizability, another reason is that it allows us to answer questions such as, "What might happen if we localize a support vector machine?" That is, we can apply a support vector machine (or some other useful method) to only the close-by part of the training set, perhaps motivated by the fact that training on this smaller set is more computationally efficient, at least if all we need is to generate estimates at a small number of points. If support vector machines are localizable, then we in fact know that such an approach can be consistent; and if they are not localizable, then we may end up with a non-consistent method with poor performance. Thus, localizability can have practical applications.

Note that one can consider other ways to define local behaviour than localizability. In one such approach, we can evaluate the behavior of a method when altering the far-off part of the training set, as opposed to removing it (which is what we do with localizability). Work along those lines (Zakai and Ritov, 2008) arrives at similar conclusions to the ones presented here. Comparing the two approaches, localizability has the advantage of relevance to practical applications, as mentioned in the previous paragraph.

Previous work related to localizability has been done in the context of learning methods that work by minimizing a loss function: We can 'localize' the loss function by re-weighting it so that close-by points are more influential; this has been investigated in the context of Empirical Risk Minimization (ERM; Vapnik, 1998) (Bottou and Vapnik, 1992; Vapnik and Bottou, 1993), as well as in the specific case of linear regression (see, e.g., Cleveland and Loader, 1995, and references therein); this approach has also lead to various applications (Atkeson et al., 1997). In this paper we differ from these approaches in that we work in a more general context: Our approach is applicable to all learning methods, and not just those that are based on minimizing a loss function that can be re-weighted. Another difference is that we consider consistency in the sense of asymptotically arriving at the lowest possible loss achievable by any measurable function, and not in the sense of minimizing the loss within a set of finite VC dimension.

Another related work is that of Bengio et al. (2006), in which it was shown that kernel machines behave locally, in the sense of requiring a large number of examples in order to learn complex functions (because each local area must be learned separately). Our approach differs from this work in the way we define local behavior, and in that we are interested in all (consistent) learning methods, not just kernel machines. However, our conclusion is in agreement with theirs, that even methods that may appear to be global like support vector machines in fact behave locally.

We now sketch our main result, which is that consistency is logically equivalent to the combination of two properties (which will be given later, in Definitions 2 and 5): *Uniform Approximate Localizability* (UAL), which is a form of localizability, and *Weak Consistency in Mean* (WCM), which deals with the mean $Ef(S_n,x)$ estimating the true mean $Ef^*(x)$ reasonably well, where the expected values are taken over S_n and x. It will be easy to see that the UAL and WCM properties are 'independent' in the sense that neither implies the other, and therefore we can see consistency as comprised of two independent aspects, which might be presented as

$$\textbf{Consistency} \quad \Longleftrightarrow \quad \textbf{UAL} \oplus \textbf{WCM}.$$

This can be seen as describing consistency in terms of local (UAL) and global (WCM) aspects (WCM is 'global' in the sense of only comparing scalar values averaged over x).

Note that there are two issues here which might be surprising, the first of which has already been mentioned—that all consistent methods must behave locally. The second important issue is that WCM is sufficient, when combined with UAL, to imply consistency. That is, if our goal is to be consistent,

$$E|f(S_n,x) - f^*(x)| \longrightarrow 0 \tag{1}$$

then it is interesting that all that is needed in addition to behaving locally is a property close to

$$|Ef(S_n,x) - Ef^*(x)| \longrightarrow 0.$$

Note that the latter condition is very weak. For example, it is fulfilled by $g(S_n,x) = \frac{1}{n}\sum_i y_i$, the simple empirical mean of the y values in the sample S_n (ignoring the xs completely), since $\frac{1}{n}\sum_i y_i \longrightarrow Ey = Ef^*(x)$ a.s., where Ey is the mean of y. On the other hand, g does not fulfill the stronger property (1) except on trivial distributions.

Our main result, which has just been described, will also be generalized to settings other than that of methods consistent on the set of all distributions. This will lead to peculiar consequences: Consider, for example, the following two sets of distributions:

$$\mathbb{P}_1 = \left\{P \,:\, y = f^*(x) + \varepsilon \,,\, E\varepsilon = 0 \,,\, \varepsilon \text{ is independent of } x\right\} \quad , \quad \mathbb{P}_2 = \left\{P \,:\, Ey = 0\right\}$$

(note that $\mathbb{P}_1$ is simply regression with additive noise). It turns out that a method consistent on $\mathbb{P}_1$ must behave locally on that set (just as with the set of all distributions), but the same is not true for $\mathbb{P}_2$, where a consistent method does not necessarily behave locally. The reason for this will be explained later.

The rest of this work is as follows. In Section 2 we present the formal setting and other preliminary matters. In Section 3 we present definitions of local properties (and specifically UAL) as well as some results concerning them. In Section 4 we define the global concepts that we need, specifically WCM. In Section 5 we present our main result, the equivalence of consistency to the combination of UAL and WCM. In Section 6 we extend our results to various sets of distributions. In Section 7 we use results from previous sections in order to derive consequences for classification. In Section 8 we summarize our results and discuss some directions for future work. Finally, proofs of our results appear in the Appendices.

2. Preliminaries

We now complete the description of the formal setting in which we work, as well as lay out notation useful later. Most of this section deals with the context of regression; details specific to classification will appear in Section 7.

We consider distributions P on (X,Y) where $X \subset \mathbb{R}^d, Y \subset \mathbb{R}$. We assume that X,Y are bounded, $\sup_{x\in X} ||x||, \sup_{y\in Y} |y| \leq M_1$ for some $M_1 > 0$ which is the same for all distributions. Thus, when we say 'all distributions' we mean all distributions bounded by the same value of M_1. We also assume that our learning methods return bounded responses, $\forall S,x \quad |f(S,x)| \leq M_2$.[1] Let M be a constant fulfilling $M \geq M_1, M_2$. Importantly, note that while these boundedness assumptions are non-trivial in the context of regression, they do not limit us when we consider classification, as we will see in Section 7.

Note that we wrote $f(S,x)$ instead of $f(S_n,x)$ in the previous paragraph. The reason is that n will always denote the size of the original training set, which in turn will always be written as S_n. Since we will also apply f to to other training sets (in particular, subsets of S_n), for clarity of notation we will therefore write S for a general (finite) set of pairs (x_i,y_i) and define learning methods via $f(S,x)$.

Formally speaking, a learning method $f(S,x)$ is defined as a sequence of measurable functions $\{f_k\}_{k\in\mathbb{N}}$, where each f_k is a function on training sets of size k, that is,

$$f_k : (X\times Y)^k \times X \longrightarrow Y.$$

For brevity, we will continue to write f instead of f_k since which f_k is used is determined by the size of the training set that we pass to f, that is, $f(S,x) = f_{|S|}(S,x)$. For example, we will often denote by $\widetilde{S}$ a subset of the original training set S_n. Then in an expression of the form $f(\widetilde{S},x)$ the actual function used is f_m, where $m = |\widetilde{S}|$, and in this example we expect to have $m \leq n$ (where, as mentioned before, n is the size of the original training set S_n).

For any distribution P on (X,Y), we write $f^*(x) = E(y|x)$, as already mentioned, and we denote the marginal distribution on X by μ. We write f^*_P, μ_P instead of f^*, μ to make explicit the dependence on P when necessary. Denote by $\mathrm{supp}_X(P) = \mathrm{supp}(\mu_P)$ the support of μ_P. The measure of sets $B \subseteq X$ will be written in the form $\mu(B)$.

1. Note that this is a minor assumption since for most methods we have $\sup_x |f(S,x)| \leq C\cdot \max_i |y_i|$ for some $C > 0$, and the y_i values are already assumed to be bounded. If this does not hold, then we might in any case want to consider enforcing boundedness based on the sample, that is, to truncate values larger in absolute value than $\max_i |y_i|$, and in doing so perhaps improve performance. Finally, recall that we are concerned with consistent methods, that is, that behave similarly to f^* in the limit, and f^* is bounded.

We denote random variables by, for example, $(x,y) \sim P$ and $S_n \sim P$ where in the latter case we intend a random i.i.d sample of n elements from the distribution P. We will often abbreviate and write $x,y \sim P$ instead of $(x,y) \sim P$; also, we will write $x \sim P$ where we mean $x \sim \mu_P$. To prevent confusion we always use x and y to indicate a pair (x,y) sampled from P.

We will write the mean and variance of random variables v as $E(v) = Ev$ and $\sigma^2(v)$, respectively. More generally, expected values will be denoted by $E_{v \sim V} H(v) = E_v H(v)$ where v is a random variable distributed according to V and $H(v)$ is some function of v; we may write $EH(v)$ when the random variables are clear from the context. The conditional expected value of $H(v)$ given w will be written in the form $E_{v|w}(H(v))$.

We will work mainly with the $\mathcal{L}_1$ loss, which we can now write formally as

$$L_{n,P}(f) \equiv E_{S_n, x \sim P} \left| f(S_n,x) - f^*_P(x) \right|,$$

or, more briefly,

$$L_n(f) \equiv E_{S_n,x} \left| f(S_n,x) - f^*(x) \right|.$$

Note that for purposes of consistency (i.e., $L_n(f) \to 0$) all $\mathcal{L}_p$ losses are equivalent, since

$$\forall 0 < p < q \qquad E|z|^p \le (E|z|^q)^{p/q} \le (2M)^{p(q-p)/q} (E|z|^p)^{p/q}$$

where $z = f(S_n,x) - f^*(x)$. Thus, if one $\mathcal{L}_p$ loss converges to 0, so do all the others. Hence our results apply to all $\mathcal{L}_p$ norms; we work mainly with the $\mathcal{L}_1$ norm for convenience.

For any set $B \subseteq X$, denote by P_B the conditioning of P on B, that is, the conditioning of μ_P on B (and leaving unchanged the behavior of y given x). Denote the ball of radius r around x by $B_{x,r} = \{x' \in \mathbb{R}^d \ : \ ||x - x'|| \le r\}$. Let $P_{x,r} = P_{B_{x,r}}$.

Finally, we mention two useful conventions. Note that we defined consistency on a single distribution, and then consistency in general as the property of being consistent on all distributions. More specifically, for any property **A** that can hold for a method f on particular distributions, we say that f has property **A** on a *set* of distributions $\mathbb{P}$ when f has property **A** on all $P \in \mathbb{P}$. We also say that f has property **A** (without specifying P or $\mathbb{P}$) when it has property **A** on the set of *all* distributions (with bounded support). This convention will be used for consistency as well as for UAL and other properties.

Similarly, when we start by defining a property **A** on the set of all distributions, we then use the convention that f has property **A** on a set $\mathbb{P}$ when we simply replace $\forall P$ with $\forall P \in \mathbb{P}$. This convention will be used with the WCM property.

3. Local Behavior

In this section we consider local properties of learning methods. We will present a series of definitions, leading up to a definition of Uniform Approximate Localizability (UAL).

We start with some introductory definitions. For any two learning methods f, g, we say that they are **mutually consistent** iff

$$D_n(f,g) \equiv E_{S_n,x} \left| f(S_n,x) - g(S_n,x) \right| \underset{n \to \infty}{\longrightarrow} 0.$$

That is, f and g are mutually consistent if they behave asymptotically similarly according to a distance metric D_n. The term ‘mutual consistency’ is used since $D_n(f, f^*) = L_n(f)$ (where we can formally define $f^*(S,x) = f^*(x)$), that is, being mutually consistent with f^* is equivalent to being consistent. Thus, mutual consistency is a natural extension of consistency. Note that D_n obeys the triangle inequality,

$$\forall f,g,h \qquad D_n(f,g) \leq D_n(f,h) + D_n(h,g)$$

which will be convenient later.

We now define some useful notation for the topic of locality. For any training set S and $r \geq 0$, we call

$$S(x,r) = \{(x_i,y_i) \in S \,:\, ||x_i - x|| \leq r\}$$

a **local training set** for x within S, of radius r. For every learning method f and $r \geq 0$, let

$$f|_r(S,x) = f(S(x,r),x).$$

Note that, as mentioned previously, if $f = \{f_k\}_{k\in\mathbb{N}}$ then in the expression $f(S_n(x,r),x)$ (where S_n is, as always, the original training set of size n), we are passing $S_n(x,r),x$ to f_m, where $m = |S_n(x,r)|$, which will in general be smaller than $n = |S_n|$. Thus, formally speaking we might write

$$f|_r(S,x) = f_{|S(x,r)|}(S(x,r),x)$$

but for simplicity we will continue to drop the lower index on f. Note that $f|_r$ can also be formally defined as a series of functions $\{f|_{r,k}\}_{k\in\mathbb{N}}$, but again, for simplicity we avoid this.

In words, $f|_r$ is a learning method that results from forcing f to only work on local training sets of radius r around x, when estimating the value at x. For example, if f is a linear regression estimator then we can see $f|_r$ as performing local linear regression (Cleveland and Loader, 1995).

Continuing in our definitions, for any sequence $\{R_k\}_{k\in\mathbb{N}}, R_k \geq 0, R_k \to 0$, we call $f|_{\{R_k\}}$ a **local version**, or a **localization** of f; by this notation, we mean

$$f|_{\{R_k\}}(S,x) = f\big(S\big(x,R_{|S|}\big),x\big)$$

—that is, which R_k is used from the sequence $\{R_k\}$ depends on the size of the training set passed to $f|_{\{R_k\}}$. In particular, for the original training set S_n we have

$$f|_{\{R_k\}}(S_n,x) = f\big(S_n\big(x,R_n\big),x\big).$$

Note that, as a consequence, local versions are indeed 'local' in the limit since we have $R_n \to 0$.

We can now define one form of local behavior: Call a method f **localizable** on a distribution P iff there exists a local version $f|_{\{R_k\}}$ of f with which f is mutually consistent, that is,

$$D_n\left(f|_{\{R_k\}}, f\right) \underset{n\to\infty}{\longrightarrow} 0.$$

Thus, a localizable method is one that is similar, in the sense of mutual consistency, to a local version of it, which implies that it gives similar results when seeing the entire training set versus only the local part of it; we can localize the method without changing the estimates significantly. Note once more that the requirement $R_n \to 0$ is what makes this definition truly define local behavior.

We will also need a notion of a method that, when localized, is consistent. Call a method f **locally consistent** on a distribution P iff there exists a local version $f|_{\{R_k\}}$ of f which is consistent,

$$L_n\left(f|_{\{R_k\}}\right) = D_n\left(f|_{\{R_k\}}, f^*\right) \underset{n\to\infty}{\longrightarrow} 0.$$

That is, there is a way to localize f so that it becomes consistent.

Are all consistent learning methods localizable and locally consistent? Consider the latter property: It appears as if any consistent method must be locally consistent, since a consistent method can successfully 'learn' given any underlying distribution, and when we localize such a method we are in effect applying that same useful behavior in every local area. Thus, it seems reasonable to expect a localization of a consistent method to be consistent as well, and if this were true then it might have useful practical applications, as mentioned in the introduction. Yet this intuition turns out to be false, and a similar failure occurs for localizability:

Proposition 1 *A learning method exists which is consistent but neither localizable nor locally consistent.*

The proof of Proposition 1 (appearing in Appendix A) is mainly technical, and consists in constructing a method $f = \{f_k\}$ which becomes less smooth as n rises, and asymptotically considerably less smooth than the true f^*. That is, the issue is that we have required merely that the functions f_k be measurable, and it turns out that without additional assumptions they can behave erratically in a manner that renders f not locally consistent. The specific example that we construct in the proof involves functions f_k that behave oddly on an area close to Ex but otherwise perform well. It is then possible to show that the 'problematic area' near Ex can be large enough so that all local versions of the method behave poorly, but small enough so that the original method is consistent.

Now, the counterexample constructed in the proof of Proposition 1 might rightfully be called a 'fringe case'. Yet, it suffices to show that not all consistent methods are localizable, contrary perhaps to intuition. There is therefore the question of what to do. One solution to this matter is to work with a property stronger than consistency, one that includes an additional smoothness requirement. The disadvantage of such an approach is that we cannot immediately derive consequences for the various methods known to be consistent, unless we also prove that they have the stronger property.

Instead, the approach that we will follow is to define more complex notions of local behavior which can be used to arrive at properties equivalent to consistency. Thus, one major goal of this paper is to arrive at suitable definitions for the topic of local behavior, that on the one hand capture the intuition correctly and on the other allow useful results to be proven. The simple definitions given before fail in the second matter; we will now give definitions that remedy that problem.

In order to formulate our improved definitions we first require some preparation. For any $r, q \geq 0$ and distribution P, let

$$\bar{f}|_r^q(S,x) = E_{x' \sim P_{x,q \cdot r}} f(S(x,r),x'). \tag{2}$$

Compared to $f|_r$, $\bar{f}|_r^q$ adds a smoothing operation performed around the x at which we estimate. Note that if $q = 0$ then we interpret the expected value as a delta function and we get $\bar{f}|_r^0 = f|_r$. Note also that we require the actual unknown distribution P in the definition of $\bar{f}|_r^q$, that is, $\bar{f}|_r^q$ cannot be directly implemented in practice—$\bar{f}|_r^q$ is a construction useful mainly for theoretical purposes. However, we can implement an approximate version of $\bar{f}|_r^q(S,x)$ by replacing the true expectation with the empirical one. We will return to this matter later.

We define the following set of sequences:

$$\mathcal{T} = \Big\{ \{T_k\} \ : \ T_k \searrow 0 \ \text{strictly} \Big\}.$$

For any sequence $T = \{T_k\}$, let $\mathbf{T} = \{T_k \ : \ k \in \mathbb{N}\}$, that is, the set containing the elements in the sequence. For any such sequence $T = \{T_k\}$, we then define the set of its infinite subsequences and selection functions on them by

$$\mathcal{R}(T) = \Big\{ R = \{R_k\} \quad : \quad \mathbf{R} \subseteq \mathbf{T} \,,\, R_k \searrow 0 \Big\},$$
$$\mathcal{Q}(T) = \Big\{ Q : \mathbf{T} \to \mathbf{T} \quad : \quad Q(T_k) \searrow 0 \Big\}.$$

For any $T \in \mathcal{T}$ and $\{R_k\} \in \mathcal{R}(T)$, $Q \in \mathcal{Q}(T)$, we call $\bar{f}|_{\{R_k\}}^{\{Q(R_k)\}}$ a **smoothed local version** of f; by this notation, we mean to replace the q, r values in (2) in an appropriate manner, that is,

$$\bar{f}|_{\{R_k\}}^{\{Q(R_k)\}}(S,x) = E_{x' \sim P_{x,Q(R_{|S|}) \cdot R_{|S|}}} f\big(S(x, R_{|S|}), x'\big).$$

Note that on the original training set S_n we get

$$\bar{f}|_{\{R_k\}}^{\{Q(R_k)\}}(S_n,x) = E_{x' \sim P_{x,Q(R_n) \cdot R_n}} f\big(S(x, R_n), x'\big).$$

We now elaborate on these definitions. T is the set of possible values that $Q(R_n), R_n$ can take; these values must approach 0 as our goal is to consider local behavior. $\mathcal{R}(T)$ contains sequences of radii of local training sets; we require that $R_n \searrow 0$, as we are interested in behavior on local training sets with radius descending to 0, that is, that become truly 'local' asymptotically. $\mathcal{Q}(T)$ contains functions that become small when T_k is small; the values $Q(R_n)$ determine radii on which to smooth, via $Q(R_n) \cdot R_n$. Since $Q(R_n) \cdot R_n = o(R_n)$, the smoothing is done on radii much smaller (asymptotically negligibly small) than the radii of the local training sets R_n, and therefore this is a minor operation. In conclusion, a smoothed local version is similar to a local version, but adds an averaging operation on small radii.

We now start with our main definitions. The idea behind them is not overly complex, but their description is necessarily somewhat technical.

Definition 2 *Call a learning method f* ***Uniformly Approximately Localizable (UAL)*** *iff*

$$\begin{aligned}
&\forall P \quad \forall T \in \mathcal{T} \\
&\exists Q \in \mathcal{Q}(T) \\
&\forall Q' \in \mathcal{Q}(T) \,,\, Q' \geq Q \\
&\exists \{R_k\} \in \mathcal{R}(T) \\
&\forall \{R'_k\} \in \mathcal{R}(T) \,,\, \{R'_k\} \geq \{R_k\} \\
&\qquad D_n\left(\bar{f}|_{\{R'_k\}}^{\{Q'(R'_k)\}}, f\right) \underset{n\to\infty}{\longrightarrow} 0.
\end{aligned}$$

(Here the expression $\{R'_k\} \geq \{R_k\}$ simply implies an inequality for the entire series, that is, for all k. $Q' \geq Q$ implies $Q'(T_k) \geq Q(T_k)$ for all k.)

In essence, a UAL learning method is one for whom, for any choice of T, *all* large-enough choices of Q and $\{R_k\}$ are suitable in order to get similar behavior between f and a smoothed local version of f. That is, if we take $\{R_k\}$ and Q slowly enough to 0 then we get local behavior. Note that in any case taking $\{R_k\}$ to 0 very quickly is problematic since we may get empty local training sets, that is, $S_n(x, R_n) = \emptyset$.

Concerning our choice of name for this definition, 'uniformly' appears in the title 'uniformly approximately local' due to the requirement for all large-enough choices of $\{R_k\}, Q$ to be relevant, and 'approximately' appears because we allow smoothed local versions and not just local versions.

Both of these changes from the original definition of localizability are present in order to prevent odd counterexamples. A concrete counterexample was shown in Proposition 1 to make clear the need for smoothing; we do not present one in full for the uniformity requirement in order to save space.

Note that we only consider $\{R_k\}, Q$ taking values in some fixed T, and that while T is arbitrary it does need to be determined in advance. The issue is that if we instead allow all the values $[0,\infty)$ to appear in $\{R_k\}$ and Q then, due to this being an uncountable set, it is not clear to the authors if additional conditions are not required to prove our main results in that case. In any event, a countable set of possible values is of sufficient interest for any practical learning-theoretical purpose, and as already mentioned the actual set of possible values can be chosen in whatever manner is desired.

We also need a definition parallel to that of local consistency, as follows.

Definition 3 *Let **Uniform Approximate Local Consistency (UALC)** be the property defined exactly the same as UAL, except for replacing the last condition with*

$$L_n\left(\bar{f}|_{\{R'_k\}}^{\{Q'(R'_k)\}}\right) = D_n\left(\bar{f}|_{\{R'_k\}}^{\{Q'(R'_k)\}}, f^*\right) \underset{n\to\infty}{\longrightarrow} 0.$$

A UALC method is one whose smoothed local versions are consistent for any large-enough choice of $Q, \{R_k\}$.

We now mention some properties of UAL and UALC, noting first that they are independent, in that each can exist without the other. Consider the following two methods:

$$f_y(S,x) = \frac{1}{|S|}\sum_{i=1..|S|} y_i, \qquad\qquad f_0(S,x) = 0 \tag{3}$$

f_y (called thus because it considers only the y values) is UALC since a local version of it is simply a kernel estimator, using the 'window kernel' $k(x) = 1\{||x|| \le 1\}$, and thus consistent, for any $\{R_k\}$ fulfilling $nR_n^d \to \infty$ (Devroye and Wagner, 1980); $\{R_k\}$ acts as the bandwidth parameter of a kernel estimator. (Note that smoothing has no effect, as the guess does not depend on x.) f_y, however, clearly cannot be UAL (e.g., consider the simple example of x uniform on $[-1,1]$ and $y = \text{sign}(x)$); neither is it consistent. Turning to f_0, this method is clearly UAL but it is neither UALC nor consistent.

Our first main result is that consistency is equivalent to the combination of UAL and UALC:

Theorem 4 *A learning method is consistent iff it is both UAL and UALC.*

In light of the independence of UAL and UALC, mentioned before, we can summarize this as

$$\textbf{Consistency} \quad \Longleftrightarrow \quad \textbf{UAL} \oplus \textbf{UALC}.$$

The proof of $\Leftarrow$ is immediate: Pick any T. We derive some $Q, \tilde{Q}$ from the appropriate $\exists$ clauses of the definitions of UAL and UALC, respectively; let $Q' \in \mathcal{Q}(T)$ fulfill $Q' \ge Q, \tilde{Q}$. Given Q', we can then derive some $\{R_k\}, \{\tilde{R}_k\}$ from the appropriate $\exists$ clauses of UAL and UALC; let $\{R'_k\} \in \mathcal{R}(T)$ fulfill $R'_k \ge R_k, \tilde{R}_k$. It is then clear that

$$D_n\left(\bar{f}|_{\{R'_k\}}^{\{Q'(R'_k)\}}, f^*\right), D_n\left(\bar{f}|_{\{R'_k\}}^{\{Q'(R'_k)\}}, f\right) \longrightarrow 0$$

due to UALC and UAL. By the triangle inequality we get

$$D_n(f,f^*) \le D_n\left(f, \bar{f}|_{\{R'_k\}}^{\{Q'(R'_k)\}}\right) + D_n\left(\bar{f}|_{\{R'_k\}}^{\{Q'(R'_k)\}}, f^*\right) \longrightarrow 0 \tag{4}$$

that is, consistency. Thus, it is fairly immediate from the definitions of UAL and UALC that together they suffice for consistency. What is interesting is that they are equivalent to it. Instead of proving that fact directly, the remainder of the proof of Theorem 4 will be a corollary of the results in Section 5.

4. Global Properties

In this section we define global properties that will be useful in the next section, where we present our main results.

First we need some preliminary definitions. We define the means of f, f^* in the natural way,

$$E_n(f) \equiv E_{n,P}(f) \equiv E_{S_n,x} f(S_n,x),$$

$$E(f^*) \equiv E_P(f^*) \equiv E_x f^*(x) = E_x E(y|x) = Ey$$

the latter expression which is just the global mean of y. We also want to consider the Mean Absolute Deviation (MAD) of f and f^*,

$$\text{MAD}_n(f) \equiv \text{MAD}_{n,P}(f) \equiv E_{S_n,x}\left|f(S_n,x) - E_n(f)\right|,$$

$$\text{MAD}(f^*) \equiv \text{MAD}_P(f^*) \equiv E_x\left|f^*(x) - E(f^*)\right|.$$

We can now define the first version of the global property of interest to us: We say that f is **consistent in mean** iff

$$\forall P \qquad \lim_{n\to\infty} |E_n(f) - E(f^*)| \;=\; \lim_{n\to\infty} |\text{MAD}_n(f) - \text{MAD}(f^*)| \;=\; 0.$$

A consistent in mean learning method is required to correctly estimate $E(f^*)$ and $\text{MAD}(f^*)$; that is, we require that the global behavior of f, averaged over x, be asymptotically equal to that of f^*. Note that we are only interested here in two scalar values which represent global averages of the behavior of f, f^*.

Consistency in mean is obviously a weaker property than requiring that, on average, f behave similarly to f^* on every x separately—that is, consistency—since

$$|E_n(f) - E(f^*)| = |E_{S_n,x} f(S_n,x) - E_x f^*(x)| \le E_{S_n,x}\left|f(S_n,x) - f^*(x)\right|. \tag{5}$$

By consistency the RHS converges to 0, and therefore so does the LHS. Consider now the MAD:

$$\begin{aligned}
\text{MAD}_n(f) &= E_{S_n,x}|f(S_n,x) - E_n(f)| \\
&= E_{S_n,x}|f(S_n,x) - f^*(x) + f^*(x) - E(f^*) + E(f^*) - E_n(f)| \\
&\le E_{S_n,x}|f(S_n,x) - f^*(x)| + E_x|f^*(x) - E(f^*)| + |E(f^*) - E_n(f)| \\
&\le D_n(f,f^*) + \text{MAD}(f^*) + |E(f^*) - E_n(f)|.
\end{aligned}$$

Of the last three expressions, the first converges to 0 by consistency, and the third by (5) (which was implied by consistency). Similarly,

$$\mathrm{MAD}_n(f) \geq \mathrm{MAD}(f^*) - D_n(f,f^*) - |E(f^*) - E_n(f)|$$

showing that $|\mathrm{MAD}_n(f) - \mathrm{MAD}(f^*)| \to 0$. Thus, unsurprisingly, consistency implies consistency in mean.

It turns out that a weaker property than consistency in mean is sufficient for our purposes:

Definition 5 *We say that f is* ***Weakly Consistent in Mean (WCM)*** *iff there exists a function H : $\mathbb{R} \to \mathbb{R}$, $H(0) = 0$, $\lim_{t\to 0} H(t) = 0$, for which*

$$\forall P \qquad \limsup_{n\to\infty} \left|E_n(f) - E(f^*)\right| \,,\, \limsup_{n\to\infty} MAD_n(f) \ \leq \ H\big(MAD(f^*)\big).$$

(Note that the same H is used for all P.) A WCM learning method is required only to do 'reasonably' well in estimating the global properties of the distribution, in a way that depends on the MAD, that is, on the difficulty: We only require that performance be good when the learning task is overall quite easy, in the sense of $f^*(x)$ being almost constant. Note that when $H(\mathrm{MAD}(f^*)) \geq 2M$ we require nothing of f for such f^* (since $|f|, |f^*| \leq M$), and also that for small $\mathrm{MAD}(f^*)$ we may allow the MAD of f to be significantly larger than that of f^* (consider, for example, $H(t) = c \cdot (\sqrt{t} + t)$ for large $c > 0$). Note also that we take the lim sups, that is, we do not even require that the limits exist (except in the trivial case where $\mathrm{MAD}(f^*) = 0$).

To see the justification for the adjective 'weak', note first that consistency in mean immediately implies WCM, using $H(t) = t$. Second, recall the example hinted at in the introduction: $f_y(S,x) = \frac{1}{n}\sum_i y_i$ is clearly not consistent, since it ignores the xs, nor is it consistent in mean, since

$$\mathrm{MAD}_n(f_y) = E_{S_n,x}|f_y(S_n,x) - E_n(f_y)| = E_{y_1,\ldots,y_n}\left|\frac{1}{n}\sum_i y_i - Ey\right| \longrightarrow 0.$$

However, f_y is WCM, since $E_n(f_y) \to E(f^*)$ and, as just mentioned, f_y's MAD converges to 0. (In fact, f_y is WCM with $H \equiv 0$, that is, in the strongest sense. That is, there are even 'weaker' methods that are WCM.)

5. Main Result

In this section we present our main result, the logical equivalence of consistency to the combination of the UAL and WCM properties. This will be arrived at during the course of proving the final step of Theorem 4.

The logical relationships between the concepts of consistency, UAL, UALC and WCM are shown in graph form in Figure 1. Note that we already saw that consistency implies WCM (since consistency implies consistency in mean, which implies WCM), and that UAL combined with UALC implies consistency (shown immediately after the statement of Theorem 4). Instead of proving directly that consistency implies UAL and UALC, we will prove that WCM implies UALC, and hence that consistency implies UALC. Consistency combined with UALC, in turn, implies UAL, since

$$D_n\left(f, \bar{f}|_{\{R_k\}}^{\{Q(R_k)\}}\right) \leq D_n\left(f,f^*\right) + D_n\left(\bar{f}|_{\{R_k\}}^{\{Q(R_k)\}}, f^*\right) \to 0$$

similarly to (4). Thus, in order to complete the picture sketched in Figure 1, it remains to show that

$$\begin{tikzcd} \textbf{Consistency} \arrow[r, shift left] \arrow[drrr] & (\ \textbf{UAL} \arrow[l, shift left] & + & \textbf{UALC}\ ) \\ & & & \textbf{WCM} \arrow[u] \end{tikzcd}$$

Figure 1: Logical relations between consistency, Uniform Approximate Localizability (UAL), Uniform Approximate Local Consistency (UALC) and Weak Consistency in Mean (WCM). That is, consistency is equivalent to the combination of UAL and UALC; consistency implies WCM; and WCM implies UALC.

Lemma 6 *A learning method that is WCM is UALC.*

The proof of Lemma 6 is the most complex of our results; we now sketch it very briefly (the full proof appears in Appendix B). The initial idea is to consider $S_n(x,r)$ as $|S_n(x,r)|$ points derived from $P_{x,r}$. The expected loss of a smoothed local version can then be seen to be approximately equal to a mean of expected losses over $P_{x,r}$ for various x. The expected losses over $P_{x,r}$ can, in turn, be bounded by the MADs of $f^*_{P_{x,r}}$ and f as well as the difference between their means; we bound the latter two using the WCM property. Finally, we construct in a recursive manner Q and $\{R_k\}$ that fulfill the requirements of UALC, using some results from measure theory regarding the asymptotic behavior of $f^*_{P_{x,r}}$.

Based on our results thus far, as summarized in Figure 1, it is obvious that we can also conclude the following:

Corollary 7 *A learning method is consistent iff it is both UAL and WCM.*

This can be stated as

$$\textbf{Consistency} \quad \Longleftrightarrow \quad \textbf{UAL} \oplus \textbf{WCM}$$

since just like UAL and UALC, UAL and WCM are clearly independent in that neither implies the other, in fact, the same two examples seen in (3) apply here: f_y has already been mentioned to be WCM, while clearly it is not UAL, whereas f_0 is UAL but not WCM.

6. Localizable Sets of Distributions

Thus far we have been concerned with consistency in the sense of the set of all (appropriately bounded) distributions; this is a very general case. However, many specific types of learning problems consider more limited sets of distributions. Our goal in this section is to apply our results to such problems. In addition, this section will provide some of the tools used in Section 7 to derive results for classification.

This section relies on the following definition:

Definition 8 *We call a set of distributions $\mathbb{P}$ **localizable** iff*

$$P \in \mathbb{P} \qquad \Longrightarrow \qquad \forall r \geq 0, x \in \text{supp}_X(P) \quad P_{x,r} \in \mathbb{P}.$$

That is, a localizable set of distributions contains all conditionings of its distributions onto small balls. Simple inspection of the proof of Lemma 6 reveals that it holds true on localizable sets of distributions, and not just on the set of all distributions, simply because we apply the WCM property of f only on distributions $P_{x,r}$. Consequently, it is easy to see that our results—specifically, Theorem 4 and Corollary 7—apply to localizable sets in general, and not just to the set of all distributions bounded by some constant $M > 0$ (which is just one type of example of a localizable set). Note that the requirement that a set of distributions be localizable is in a sense the minimal requirement we would expect, since if f is consistent on a distribution P but not on some $P_{x,r}$ then there is no reason to expect f to be UALC. We will say more about this matter in Section 8.

Are all standard learning problems defined (perhaps implicitly) on localizable sets of distributions? The answer is no; for example, if we assume that $Ey = 0$, which implies $E_x f^*(x) = 0$, then this is clearly not necessarily preserved when we consider some $P_{x,r}$. However, the converse is true for many standard setups in statistics and machine learning. Specific examples include the following, to all of which our results apply (assuming the boundedness assumption is upheld):

1. $y = f^*(x) + \varepsilon$, ε is independent of x , $E\varepsilon = 0$

   This is the standard regression model with additive noise (and random design) appearing in statistics. It is clearly a setup that implicitly works on a localizable set since $P_{x,r}$ retains the property that $y = f^*(x) + \varepsilon$.

2. As a subcase of the previous example, we can assume that $f^* \in \mathcal{F}$, where $\mathcal{F}$ is the set of all continuous functions, or alternatively some 'smoothness class', for example, Lipschitz-continuous functions, etc. Note, however, that if the density of the marginal distribution on X is assumed to be bounded then this is no longer a localizable set.

3. $P\big(y = f^*(x)\big) = 1$, $\forall S,x \;\; f(S,x), f^*(x) \in \{-1,+1\}$

   This is a noiseless classification problem, or set-estimation problem with non-overlapping sets, since

   $$P\big(f(S_n,x) \neq y\big) = E_{S_n,(x,y)} 1\{f(S_n,x) \neq y\} = \frac{1}{2} E_{S_n,x} \big|f(S_n,x) - f^*(x)\big| = \frac{1}{2} L_n(f).$$

   That is, the 0-1 loss used in classification is equal to (half) the $\mathcal{L}_1$ loss in this case. Note, however, that we assume $f(S,x) \in \{-1,+1\}$, and smoothed local versions do not have this property; for them the equality between the 0-1 and $\mathcal{L}_1$ losses is not valid. If we are willing to use the $\mathcal{L}_1$ norm for classification, however, then our results apply here.

   In the next section we will see a way to derive results for the 0-1 loss, as well as allow noisy distributions, that is, the standard classification setup. This will require somewhat different definitions than those used for regression.

4. $x = \phi(z)$ for some random variable z, where $\phi : R^d \to R^D$ is smooth and $D > d$.

   We conclude with this final somewhat more complex example in order to show how localizable sets can be present even in settings where we might not expect them. In the setting described here, the original data (z,y) lies on some low-dimensional space, but we observe $(x,y) = (\phi(z),y)$, which lies on a low-dimensional manifold inside a high-dimensional space. This sort of setting is considered in the manifold-learning field in unsupervised learning (see, e.g., Roweis and Saul, 2000; Belkin and Niyogi, 2003); recently such methods have been applied to supervised learning (see, e.g., Kouropteva et al., 2003; Li et al., 2005). Note also the relevance to the kernel trick, in which a kernel implicitly defines such a transformation ϕ.

The assumption of $x = \phi(z)$ (and that ϕ is smooth) is a nontrivial requirement; however, it is easy to see that if we consider the set of all ϕ and z then this set is a localizable set of distributions.

7. Classification

As mentioned in the previous section, noiseless classification in the $\mathcal{L}_1$ norm can be dealt with using the results we have seen thus far, but this is quite limiting. Therefore in this section we consider the standard case of classification, using the natural 0-1 loss, and allowing for the possibility of noise.

In classification (also known as pattern recognition; Devroye et al., 1996) we consider only distributions for whom $Y = \{-1,+1\}$; when we say 'all distributions' in this section we mean only distributions of this sort. Note that such distributions form a localizable set, and thus we might expect our results to apply to them. Note also that $f^*(x) = E(y|x) = 2\eta(x) - 1$ where $\eta(x) = P(y = 1|x)$, the conditional probability. We call a learning method c a **classifier** iff $\forall S,x\ \ c(S,x) \in \{-1,+1\}$. We will use c,d, etc., to denote classifiers, to differentiate them from general learning methods, which we generally denote f,g.

In classification the natural loss is the 0-1,

$$R_{0-1}(c) = P\left(c(S_n,x) \neq y\right) = E_{S_n,(x,y)} 1\{c(S_n,x) \neq y\}$$

and the minimal (Bayesian) loss is

$$R^*_{0-1} = \inf_h E_{x,y} 1\{h(x) \neq y\}$$

where the infimum is taken over all measurable h. Denote the optimal (Bayesian) classification rule by

$$c^*(x) = \operatorname{sign}(f^*(x)) = \operatorname{sign}(2\eta(x) - 1)$$

which clearly minimizes R_{0-1}. We are interested in the relative loss $\Delta R_{0-1}(c) = R_{0-1}(c) - R^*_{0-1}$. This is well-known to be equal to (half the value of)

$$\widetilde{L}_n(c) \equiv E_{S_n,x}|c(S_n,x) - c^*(x)| \cdot |2\eta(x) - 1| = E_{S_n,x}|c(S_n,x) - c^*(x)| \cdot |f^*(x)|.$$

That is, we have the usual $\mathcal{L}_1$ loss but it is weighted according to the distance of $\eta(x)$ from $1/2$. Another difference is that we compare c to $c^* = \operatorname{sign}(f^*)$ and not to f^*. These differences between $\widetilde{L}_n$ and the loss L_n we considered in the main part of this work prevent an immediate application of our results. We will therefore present alternative definitions that will allow us to get around this problem.

First, we define mutual consistency in the context of classification, which we will call **mutual classification-consistency** (or, briefly, **mutual *C*-consistency**), using

$$\widetilde{D}_n(c,d) = E_{S_n,x}|c(S_n,x) - d(S_n,x)| \cdot |2\eta(x) - 1| \longrightarrow 0$$

—that is, we simply add the same weighting as in $\widetilde{L}_n$. This leads to defining **classification-consistency** (or, briefly, ***C*-consistency**) on a distribution P as

$$\widetilde{L}_n(c) = \widetilde{D}_n(c,c^*) = E_{S_n,x}|c(S_n,x) - c^*(x)| \cdot |2\eta(x) - 1| \longrightarrow 0$$

(denoting $c^*(S,x) = c^*(x)$). Note that, in a similar way as in regression, C-consistency is a specific case of mutual C-consistency.

A change needs to be made to smoothed local versions to ensure that they remain classifiers, since $\bar{c}|_r^q$ can return values not in $\{-1,+1\}$. Define, therefore,

$$\widetilde{c}|_r^q(S,x) = \operatorname{sign}\left(\bar{c}|_r^q(S,x)\right) = \operatorname{sign}\left(E_{x' \sim P_{x,qr}} c(S(x,r),x')\right).$$

When we talk of smoothed local versions in the context of classification, we intend $\widetilde{c}|_r^q$.

We can now define a version of UAL for the context of classification, in the following natural way. Let ***C*-UAL** be the same as UAL, but replace D_n with $\widetilde{D}_n$, f with c and $\bar{f}|_r^q$ with $\widetilde{c}|_r^q$.

Instead of proving results 'from scratch' for the definitions given in this section, we can build upon the previous ones, using the following general technique. Let f_{ker} be a consistent Nadaraya-Watson kernel estimator using the window kernel. Define

$$f_{|\cdot|}(S,x) = |f_{\text{ker}}(S,x)|.$$

For any classifier c, define a learning method

$$f_c(S,x) = c(S,x) f_{|\cdot|}(S,x).$$

We can see $f_c(S,x)$ as an estimate of $f^*(x)$, using (in a plug-in manner) $c(S,x)$ to estimate $\operatorname{sign}(f^*(x))$ and $f_{|\cdot|}(S,x)$ to estimate $|f^*(x)|$.

We will now use our results on regression for estimators f_c in order to arrive at conclusions for classifiers c. Note that $|f_c(S,x)| \leq 1$ (since f_{ker} uses the window kernel, it is bounded by the largest $|y_i|$ in the training set, which is 1). Given the additional fact that in classification we have $Y = \{-1,+1\}$, we can conclude that our boundedness assumption on learning methods is of no consequence to our treatment of classification, that is, to get completely general results for classification we need only rely on bounded regression with $M = 1$.

To arrive at results, we will need some minor facts:

Lemma 9 *The following hold true for every P, c, d:*

1. $D_n(f_c, f_d) \to 0 \iff \widetilde{D}_n(c,d) \to 0.$
2. $D_n(f_c, f^*) \to 0 \iff \widetilde{D}_n(c,c^*) \to 0.$
3. *f_c is UALC* $\Rightarrow$ $D_n\left((\bar{f}_c)|_{\{R_k\}}^{\{Q(R_k)\}}, f_{\widetilde{c}|_{\{R_k\}}^{\{Q(R_k)\}}}\right) \to 0$
   for large-enough $Q, \{R_k\}$ in the sense appearing in the definitions UAL,UALC.

The first part of the lemma indicates that the C-mutual consistency of any c, d are linked to the mutual consistency of f_c, f_d; the second does the same for C-consistency and consistency. The third part of the lemma shows that if f_c is UALC then smoothed local versions of f_c are asymptotically equivalent to $f_{c'}$, where c' is a smoothed local version in the sense of classification of c (in other words, we can either smooth f_c or first smooth c and then apply f; the result is similar). A proof of all parts of the lemma appears in Appendix E.

We can now present an example of deriving results for classification from those for regression. Assume that c is C-consistent on some localizable set $\mathbb{P}$. Then f_c is consistent on $\mathbb{P}$ by part 2 of

Lemma 9, hence by Theorem 4 (and what we have seen regarding localizable sets in Section 6) f_c is UAL on $\mathbb{P}$, that is, for every $P \in \mathbb{P}$ we have

$$D_n\left(f_c, (\bar{f_c})|_{\{R_k\}}^{\{Q(R_k)\}}\right) \to 0$$

for large-enough Q, R_k in the sense of the definition of UAL. f_c is also UALC on $\mathbb{P}$, and therefore for large enough Q, R_k we have

$$D_n\left((\bar{f_c})|_{\{R_k\}}^{\{Q(R_k)\}}, f_{\tilde{c}|_{\{R_k\}}^{\{Q(R_k)\}}}\right) \to 0$$

by part 3 of Lemma 9, and therefore we conclude by the triangle inequality that (again, for large enough Q, R_k)

$$D_n\left(f_c, f_{\tilde{c}|_{\{R_k\}}^{\{Q(R_k)\}}}\right) \to 0.$$

Hence, by part 1 of Lemma 9, c and $\tilde{c}|_{\{R_k\}}^{\{Q(R_k)\}}$ are mutually C-consistent for large-enough Q, R_k, that is, c is C-UAL on P. Since this was for every $P \in \mathbb{P}$, we conclude that c is C-UAL on $\mathbb{P}$. Since the entire argument was for an arbitrary localizable set $\mathbb{P}$, we conclude that

Theorem 10 *A C-consistent classifier on a localizable set $\mathbb{P}$ is C-UAL on $\mathbb{P}$.*

In particular, since the set of all distributions (having $Y = \{-1, +1\}$) is localizable, we get

Corollary 11 *A C-consistent classifier is C-UAL.*

In a similar way we can define in the context of classification concepts parallel to UALC and WCM, and prove corresponding results (we do not go into details to avoid repetition).

8. Concluding Remarks

Our analysis of consistency has led to the following result: Consistent learning methods must have two properties, first, that they behave locally (UAL); second, that their mean must not be far from estimating the true mean (WCM). Only a learning method having these two independent properties is consistent, and vice versa; their combination is logically equivalent to consistency.

To further elaborate on this result, note that UAL is clearly a local property, and WCM a global one. Thus, we can see consistency as comprised of two aspects, one local and one global. Note also that the global property, WCM, is a trivial consequence of consistency, and therefore what is worth noting about this result is that consistency implies local behavior. We can then ask, in an informal manner at least, why is this so?

As noted in the introduction, the loss L_n that we considered is a global one, in that we average over x, and hence it does not seem to directly imply local behavior. What does seem to be the crux of the matter is the requirement to perform well on *all* distributions, or more generally on a localizable set; note that the term 'localizable' here is a giveaway. Indeed, if we have a method that is consistent on a *non*-localizable set of distributions, it may not behave locally. As a simple example (but complex enough for the underlying issues to be evident), let us assume we work on distributions having $Ey = 0$ (which was mentioned in Section 6 as being non-localizable). We might consider the following approach on this problem: Let f be some general learning method known to be useful on other data sets; specifically, assume that f is consistent. We can then calibrate its output so that it returns an empirical mean of 0, which makes sense since we know that $Ey = 0$. That is, let

$$g(S,x) = f(S,x) - \frac{1}{|S|} \sum_{i=1..|S|} f(S,x_i)$$

(alternatively, we might calibrate by multiplying by an appropriate scalar, etc.). Then, while g is consistent on the set of distributions with $Ey = 0$ (using the consistency of f on all distributions), g does not behave locally. This can be seen, for example, by considering g on a distribution with x uniform on $[-1,1]$ and $y = \text{sign}(x)$. This distribution fulfills $Ey = 0$, and g is consistent on it, but neither UAL nor UALC, since smoothed local versions of g in fact return values tending towards 0.

Thus, in summary, the fact that the set of all distributions is localizable is what causes consistency to imply local behavior. If we are concerned about that fact (e.g., because we suspect local behavior might lead to the curse of dimensionality; Bengio et al., 2006), then we must do away with consistency on the set of all distributions and instead talk about consistency on a more limited set, one which is not localizable. However, part of the reason why nonparametric methods often outperform parametric ones on real-world data is precisely because they make as few as possible assumptions about the unknown distribution. Consequently, we may find that local behavior is hard to avoid.

We now turn to directions for future work. One such direction is to apply our results towards proving the consistency of learning methods not yet known to be consistent. It appears that in many cases proving the WCM property should not be difficult; hence, what remains is to prove UAL. While not necessarily a simple property to show, it may in some cases be easier than proving consistency directly.

An additional area for possible future work is empirical investigation, as follows. Since a consistent method is necessarily UALC, we can consider using a smoothed local version of the original method, since if chosen appropriately it is also consistent.[2] Now, if for example we have a large training set and only a few points at which to estimate, then by training on the local parts of the original training set we may save time. Of course, there is no guarantee that this will be beneficial on a particular problem, since consistency is an asymptotic property. Future empirical work might therefore examine real-world data sets in detail to see how the performance of local versions of a method compare to the original.

Acknowledgments

We thank the anonymous reviewers of a previous version of this paper for their comments. We also acknowledge support for this project from the Israel Science Foundation as well as partial support from NSF grant DMS-0605236.

2. Note that we might not need smoothing in practice, since the smoothing radius becomes negligibly small, $Q(R_k)R_k = o(R_k)$. That is, local versions might behave similarly enough to smoothed local versions for practical purposes.

Appendix A. Proof of Proposition 1

For any sample S, denote the mean and standard deviation (of the xs) by

$$\hat{E}(S) = \frac{1}{|S|} \sum_{i=1..|S|} x_i \qquad , \qquad \hat{\sigma}(S) = \sqrt{\frac{1}{|S|} \sum_{i=1..|S|} \left(x_i - \hat{E}(S)\right)^2}.$$

Now, start with a consistent method g, which is *translation-invariant* in the sense that

$$\forall S, x, b \qquad g(S,x) = g(S+b, x+b)$$

where $S+b = \{(x_i+b, y_i) \ : \ (x_i, y_i) \in S\}$ (for example, we can take g to be a kernel estimator).

Define, for any $q > 0$,

$$f^q(S,x) = \begin{cases} g(S,x) & x \notin B_{\hat{E}(S),q} \text{ or } x \in \{x_i \ : \ (x_i,y_i) \in S\} \\ 0 & \text{otherwise} \end{cases}$$

and let

$$Q(S) = \frac{\hat{\sigma}(S)}{\log(|S|)}.$$

Finally, let

$$f(S,x) = f^{Q(S)}(S,x).$$

We will first show that f is consistent; then we will show that f is neither locally consistent nor localizable. A brief overview of the proof is as follows: As $n \to \infty$, we get that $Q(S_n) \approx \frac{\text{const}}{\log(n)}$. This means that f is forced to return 0 on an area with vanishing radius, and hence f behaves like g on an area with measure rising to 1, and is consistent. On the other hand, when we consider a local version, we get that $Q(S_n(x,R_n)) \approx O\left(\frac{R_n}{\log(m)}\right)$, where m is the size of $S_n(x,R_n)$. We also get that x, the point at which we are estimating, is at distance $\approx O\left(\frac{R_n}{\sqrt{m}}\right)$ from $\hat{E}(S_n(x,R_n))$, a distance which is asymptotically smaller than Q. Hence x will tend to be in the area on which f is forced to return 0, making f neither locally consistent nor localizable.

We now start with the formal proof, first showing that f is consistent. Since X is bounded, $\hat{\sigma}(S_n)$ is bounded, and hence $Q(S_n) = \frac{\hat{\sigma}(S_n)}{\log(n)} \to 0$. We first assume that there is not a point mass on Ex, which implies $\mu\left(B_{\hat{E}(S_n),Q(S_n)}\right) \to 0$ almost surely, which follows from the fact that, since $\hat{E}(S_n) \to Ex$ a.s. (by the LLN) and $Q(S_n) \to 0$, we must have $\limsup_n B_{\hat{E}(S_n),Q(S_n)} \subseteq \{Ex\}$ with probability 1. We can then use the consistency of g to see that

$$\begin{aligned}
&E_{S_n} E_x |f(S_n,x) - f^*(x)| \\
&\quad \leq E_{S_n} E_x 1\{x \notin B_{\hat{E}(S_n),Q(S_n)} \text{ or } x \in \{x_i \ : \ (x_i,y_i) \in S_n\}\} |g(S_n,x) - f^*(x)| \\
&\qquad + E_{S_n} 2M\mu\left(B_{\hat{E}(S_n),Q(S_n)} \bigcap \{x_i \ : \ (x_i,y_i) \in S_n\}^C\right) \\
&\quad \leq E_{S_n}\left[E_x |g(S_n,x) - f^*(x)| + 2M\mu\left(B_{\hat{E}(S_n),Q(S_n)}\right)\right] \\
&\quad \to 0
\end{aligned}$$

which proves that f is consistent. Consider now the case where there is a point mass on Ex. Recall that $\limsup_n B_{\hat{E}(S_n),Q(S_n)} \subseteq \{Ex\}$ a.s., and note that due to the point mass on Ex, we have $P\big(Ex \in \{x_i \,:\, (x_i,y_i) \in S_n\}\big) \to 1$. Because of these two facts, we get

$$E_{S_n}\mu\left(B_{\hat{E}(S_n),Q(S_n)} \bigcap \{x_i \,:\, (x_i,y_i) \in S_n\}^C\right) \to 0$$

hence once more f is consistent by the consistency of g, by a similar argument as before.

We will now show that f is not locally consistent. For this, it is sufficient to show a single distribution on which $L_n\big(f|_{\{R_k\}}\big)$ does not converge to 0, for any sequence $R_k \to 0$. Take $X = [0,1]$ (higher-dimensional cases can be proved similarly), let μ be uniform on X, and let $y = +1$ with probability $(1 + f^*(x))/2$, and otherwise $y = -1$ (which makes sense if $f^*(x) \in [-1,+1]$). Define

$$f^*(x) = \begin{cases} +1/2 & x \in \left[0, \frac{1}{2}\right] \\ -1/2 & x \in \left(\frac{1}{2}, 1\right] \end{cases}.$$

Fix some $x \in (0,1)$. Note that μ has no point masses, so $P(x \in \{x_i \,:\, (x_i,y_i) \in S_n\}) = 0$, and the relevant condition in the definition of f^q is of no consequence. Now, for large enough n (that is, small enough R_n) we have $x \in [R_n, 1 - R_n]$; we will now focus on that case.

Denote $\widetilde{S} = S_n(x,R_n)$ and $m = m(n,x,R_n) = |\widetilde{S}|$. Notice that m is the sum of i.i.d Bernoulli variables, and that

$$Em = 2nR_n \quad , \quad \sigma^2(m) \le 2nR_n \tag{6}$$

(recall that $Em, \sigma^2(m)$ denote the expected value and variance of the random variable m, respectively). Now, consider the case in which $nR_n \not\to \infty$. Then there is some bounded subsequence, $k_nR_{k_n} \le K$ for all n. Then (using Chebyshev) clearly m is less than $4k_nR_{k_n}$ with non-vanishing probability on this subsequence. Since $R_k \to 0$ then in order for f to be consistent we must, for large enough n, discriminate between the two possibilities $f^*(x) = +1/2, f^*(x) = -1/2$ in a way that does not depend upon x (due to the translation-invariance of f, which stems from the translation-invariance of g). But discriminating between the two cases with arbitrarily small error cannot be done with a bounded sample size, hence the loss cannot go to 0. Thus, we conclude that if $nR_n \not\to \infty$ then f is not locally consistent.

Consider now the other case, of $nR_n \to \infty$. We start by formulating bounds for m, $|x - \hat{E}(\widetilde{S})|$, and $\hat{\sigma}^2(\widetilde{S})$.

- m: Using Bernstein's Inequality and (6), we get

$$P(|m - 2nR_n| \ge t) \le 2\exp\left(-\frac{1}{2}\frac{t^2}{2nR_n + t/3}\right).$$

  Picking $t = t_n = nR_n$, we get

$$P(|m - 2nR_n| \ge nR_n) \le 2\exp\left(-\frac{1}{6}nR_n\right). \tag{7}$$

  Note that this bound converges to 0 since $nR_n \to \infty$.

- $|x-\hat{E}(\widetilde{S})|$: Clearly x is the mean of P_{x,R_n}, and also the mean of the individual observations in $\widetilde{S}$, as they are distributed i.i.d as P_{x,R_n}. Note that for every $x_i \in \widetilde{S}$ (i.e., taken from the distribution P_{x,R_n}) we have $\sigma^2(x_i) = \frac{R_n^2}{3}$. Then, by Hoeffding's Inequality,

$$P\left(\left|x-\hat{E}(\widetilde{S})\right| \geq t\right) \leq E_m\left\{2\exp\left(-\frac{mt^2}{2R_n^2}\right)\right\}.$$

Picking $t = t_n = R_n\sqrt{\frac{\log(nR_n)}{nR_n}}$, we get

$$\begin{aligned}P\left(\left|x-\hat{E}(\widetilde{S})\right| \geq R_n\sqrt{\frac{\log(nR_n)}{nR_n}}\right) &\leq E_m\left\{2\exp\left(-\frac{1}{2}\frac{m}{nR_n}\log(nR_n)\right)\right\} \qquad (8)\\ &\leq 2\exp\left(-\frac{1}{2}\log(nR_n)\right)+4\exp\left(-\frac{1}{6}nR_n\right)\\ &\to 0\end{aligned}$$

using the bound for m above (7).

- A final bound that we will need relates to $\hat{\sigma}^2(\widetilde{S})$. Note that for any sample S and point x,

$$\hat{\sigma}^2(S) = \frac{1}{|S|}\sum_i (x_i-x)^2 - (x-\hat{E}(S))^2$$

Consider the first expression on the RHS. On our (sub)sample $\widetilde{S}$, we have for every $x_i \in \widetilde{S}$ that $E(x_i-x)^2 = \frac{R_n^2}{3}$, and note that $(x_i-x)^2 \leq R_n^2$. Then, by Hoeffding's Inequality,

$$P\left(\left|\frac{1}{m}\sum_{i=1}^{m}(x_i-x)^2 - \frac{R_n^2}{3}\right| \geq t\right) \leq E_m 2\exp\left(-\frac{mt^2}{2R_n^4}\right).$$

Picking $t = t_n = R_n^2\sqrt{\frac{\log(nR_n)}{nR_n}}$, we get

$$\begin{aligned}&P\left(\left|\frac{1}{m}\sum_{i=1}^{m}(x_i-x)^2 - \frac{R_n^2}{3}\right| \geq R_n^2\sqrt{\frac{\log(nR_n)}{nR_n}}\right)\\ &\quad\leq E_m 2\exp\left(-\frac{1}{2}\frac{m}{nR_n}\log(nR_n)\right)\\ &\quad\leq 2\exp\left(-\frac{1}{2}\log(nR_n)\right)+4\exp\left(-\frac{1}{6}nR_n\right)\\ &\quad\to 0\end{aligned}$$

similarly as before. Combined with the bound from before for $|x-\hat{E}(\widetilde{S})|$, we get, for large enough n,

$$\begin{aligned}
&P\left(\hat{\sigma}^2(\widetilde{S}) \leq \frac{1}{12}R_n^2\right) \\
&\quad = P\left(\hat{\sigma}^2(\widetilde{S}) \leq \frac{1}{12}R_n^2\ ,\ \left(x-\hat{E}(\widetilde{S})\right)^2 < R_n^2\frac{\log(nR_n)}{nR_n}\right) \\
&\qquad + P\left(\hat{\sigma}^2(\widetilde{S}) \leq \frac{1}{12}R_n^2\ ,\ \left(x-\hat{E}(\widetilde{S})\right)^2 \geq R_n^2\frac{\log(nR_n)}{nR_n}\right) \\
&\quad \leq P\left(\frac{1}{m}\sum_{i=1}^{m}(x_i-x)^2 \leq \frac{1}{6}R_n^2\right) + P\left(\left|x-\hat{E}(\widetilde{S})\right| \geq R_n\sqrt{\frac{\log(nR_n)}{nR_n}}\right) \\
&\quad \to 0.
\end{aligned}$$

We now have all the bounds we need. Using our results for $m=|\widetilde{S}|$ and $\hat{\sigma}^2(\widetilde{S})$ together, we get

$$\begin{aligned}
&\lim_{n\to\infty} P\left(\left|x-\hat{E}(\widetilde{S})\right| \geq Q(\widetilde{S})\right) \\
&\quad = \lim_{n\to\infty} P\left(\left|x-\hat{E}(\widetilde{S})\right| \geq \frac{\hat{\sigma}(\widetilde{S})}{\log(m)}\right) \\
&\quad = \lim_{n\to\infty} P\left(\left|x-\hat{E}(\widetilde{S})\right| \geq \frac{\hat{\sigma}(\widetilde{S})}{\log(m)}\ ,\ |m-2nR_n| \leq nR_n\ , \hat{\sigma}^2(\widetilde{S}) \geq \frac{1}{12}R_n^2\right) \\
&\quad \leq \lim_{n\to\infty} P\left(\left|x-\hat{E}(\widetilde{S})\right| \geq \frac{R_n}{\sqrt{12}\log(3nR_n)}\right) \\
&\quad \to 0
\end{aligned}$$

where to reach the very last line we used what we know about $|x-\hat{E}(\widetilde{S})|$, as appearing in (8). Thus, for every $x \in \mathrm{supp}_X(P)$ we have $P\left(x \in B_{\hat{E}(\widetilde{S}),Q(\widetilde{S})}\right) \to 1$.

Finally, we can see that

$$\begin{aligned}
E_{S_n}\left|f|_{\{R_k\}}(S_n,x)-f^*(x)\right| &= E_{S_n}\left|f(\widetilde{S},x)-f^*(x)\right| \\
&= E_{S_n}1\{x \in B_{\hat{E}(\widetilde{S}),Q(\widetilde{S})}\}\left|f(\widetilde{S},x)-f^*(x)\right| \\
&\quad + E_{S_n}1\{x \notin B_{\hat{E}(\widetilde{S}),Q(\widetilde{S})}\}\left|f(\widetilde{S},x)-f^*(x)\right| \\
&\geq E_{S_n}1\{x \in B_{\hat{E}(\widetilde{S}),Q(\widetilde{S})}\}\left|f^*(x)\right| \\
&= \frac{1}{2}E_{S_n}1\{x \in B_{\hat{E}(\widetilde{S}),Q(\widetilde{S})}\} \\
&= \frac{1}{2}P\left(x \in B_{\hat{E}(\widetilde{S}),Q(\widetilde{S})}\right) \\
&\to \frac{1}{2}
\end{aligned}$$

where we used that $f^*(x) \in \{-1/2, +1/2\}$. Taking the expected value over x, the dominated convergence theorem gives us

$$\lim_{n\to\infty} L_n\left(f|_{\{R_k\}}\right) \geq \frac{1}{2}.$$

Thus, $f|_{\{R_k\}}$ is not consistent, that is, f is not locally consistent (note that this is even by a relatively large constant factor).

Having shown that f is consistent but not locally consistent, we now show that in addition f cannot be localizable. This is immediate, since if f were localizable, then some sequence $R_k \to 0$ would exist for which $D_n\left(f, f|_{\{R_k\}}\right) \to 0$, and therefore

$$L_n\left(f|_{\{R_k\}}\right) = D_n\left(f|_{\{R_k\}}, f^*\right) \leq D_n\left(f|_{\{R_k\}}, f\right) + D_n(f, f^*) \to 0$$

by the localizability and consistency of f. But this result implies f is locally consistent, contradicting what we saw earlier.

Appendix B. Proof of Lemma 6

Define (as in the proof of Proposition 1) $\widetilde{S} = S_n(x,r)$, $m = m(n,x,r) = |S_n(x,r)|$, the size of the local training set. Note that we can see $\widetilde{S}$ as m points sampled from $P_{x,r}$. Then we get, for any $r, q > 0$,

$$\begin{aligned}
D_n\left(\bar{f}|_r^q, f^*\right) &= E_{S_n,x}\left|E_{x'\sim P_{x,qr}} f(S_n(x,r),x') - f^*(x)\right| \\
&= E_{x,m} E_{\widetilde{S}\sim P_{x,r} \mid m} \left|E_{x'\sim P_{x,qr}} f(\widetilde{S},x') - f^*(x)\right| \\
&\leq E_{x,m} E_{\widetilde{S}\sim P_{x,r} \mid m} E_{x'\sim P_{x,qr}} \left|f(\widetilde{S},x') - f^*(x)\right| \\
&\leq E_{x,m} E_{\widetilde{S}\sim P_{x,r} \mid m\,,\, x'\sim P_{x,qr}} \left|f(\widetilde{S},x') - f^*(x')\right| \\
&\quad + E_{x,x'\sim P_{x,qr}} \left|f^*(x') - f^*(x)\right|.
\end{aligned}$$

We now start to consider the limit behavior of these expressions when we replace $r > 0$ with $\{R_k\} \in \mathcal{R}(T)$ and $q > 0$ with $\{Q(R_k)\}, Q \in \mathcal{Q}(T)$, where $T \in \mathcal{T}$ is arbitrary. First, for any such $\{R_k\}, Q$ we have

$$\lim_{n\to\infty} E_{x,x'\sim P_{x,Q(R_n)R_n}} \left|f^*(x') - f^*(x)\right| = 0$$

by the corollary to the following lemma (and since $R_n, Q(R_n) \to 0$):

Lemma 12 *[Devroye, 1981; Lemma 1.1] For any distribution P and measurable g, if $E_{x\sim P}|g(x)| < \infty$ then*

$$\lim_{r\to 0} E_{x'\sim P_{x,r}} g(x') = g(x)$$

for almost all x.

Corollary 13 *For any distribution P and measurable g, if $E_{x\sim P}|g(x)| < \infty$ then*

$$\lim_{r\to 0} E_{x'\sim P_{x,r}} \left|g(x') - g(x)\right| = 0$$

for almost all x.

Write $\widetilde{S} = S_n(x, R_n)$ (a minor abuse of our notation, as we also write $\widetilde{S} = S_n(x, r)$, but r is always a placeholder for R_n in any case). Then

$$\limsup_{n\to\infty} D_n\left(\bar{f}|_{\{R_k\}}^{\{Q(R_k)\}}, f^*\right) \le \limsup_{n\to\infty} E_{x,m} E_{\widetilde{S}\sim P_{x,R_n}\,|\,m\,,\,x'\sim P_{x,Q(R_n)R_n}} \left|f(\widetilde{S},x') - f^*(x')\right|.$$

To simplify notation for this last expression, define, for any $x \in \operatorname{supp}_X(P), n \in \mathbb{N}, r, q \in T$,

$$C(x,n,r,q) = E_m E_{\widetilde{S}\sim P_{x,r}\,|\,m\,,\,x'\sim P_{x,qr}} \left|f(\widetilde{S},x') - f^*(x')\right|$$

and

$$C(n,r,q) = E_x C(x,n,r,q).$$

It is therefore our goal to show that $C(n,R_n,Q(R_n)) \to 0$ for appropriate $\{R_k\}, Q$. Towards that end, we consider the limit $\limsup_{n\to\infty} C(n,r,q)$, for fixed r,q. We have, for every $x \in \operatorname{supp}_X(P)$,

$$\begin{aligned}
&C(x,n,r,q)\\
&= E_m E_{\widetilde{S}\sim P_{x,r}\,|\,m\,,\,x'\sim P_{x,qr}} \left|f(\widetilde{S},x') - f^*(x')\right|\\
&= \frac{\mu(B_{x,r})}{\mu(B_{x,qr})} E_m E_{\widetilde{S}\sim P_{x,r}\,|\,m\,,\,x'\sim P_{x,r}} \left|f(\widetilde{S},x') - f^*(x')\right| 1\left\{x' \in B_{x,qr}\right\}\\
&\le \frac{\mu(B_{x,r})}{\mu(B_{x,qr})} E_m E_{\widetilde{S},x'\sim P_{x,r}\,|\,m} \left|f(\widetilde{S},x') - f^*(x')\right|\\
&= \frac{\mu(B_{x,r})}{\mu(B_{x,qr})} E_m E_{\widetilde{S},x'\sim P_{x,r}\,|\,m} \left|f(\widetilde{S},x') - E_m(f) + E_m(f) - E(f^*) + E(f^*) - f^*(x')\right|\\
&\le \frac{\mu(B_{x,r})}{\mu(B_{x,qr})} E_m\left[\mathrm{MAD}_{m,P_{x,r}}(f) + |E_{m,P_{x,r}}(f) - E_{P_{x,r}}(f^*)| + \mathrm{MAD}_{P_{x,r}}(f^*)\right]
\end{aligned}$$

where the expected values $E_m(f), E(f^*)$ on the line before last are w.r.t $P_{x,r}$, and the expected values and MADs on the last two lines are all conditional on m.

Now, clearly $m \to \infty$ almost surely (since x is in the support of μ, and $r > 0$ is fixed, so there is a positive probability for an observation to fall within $B_{x,r}$). Hence, by the WCM property of f on $P_{x,r}$ (a *fixed* distribution in the current view),

$$\limsup_{n\to\infty} C(x,n,r,q) \le \frac{\mu(B_{x,r})}{\mu(B_{x,qr})}\left[\mathrm{MAD}_{P_{x,r}}(f^*) + 2H\left(\mathrm{MAD}_{P_{x,r}}(f^*)\right)\right].$$

Using this bound, we can then conclude that

$$\begin{aligned}
\limsup_{n\to\infty} C(n,r,q) &= \limsup_{n\to\infty} E_x C(x,n,r,q)\\
&\le E_x \limsup_{n\to\infty} C(x,n,r,q) \qquad (9)\\
&\le E_x \min\left\{2M\,,\ \frac{\mu(B_{x,r})}{\mu(B_{x,qr})}\left[\mathrm{MAD}_{P_{x,r}}(f^*) + 2H\left(\mathrm{MAD}_{P_{x,r}}(f^*)\right)\right]\right\}
\end{aligned}$$

using the Fatou-Lebesgue theorem for the first inequality, where we rely on the trivial fact that $C(x,n,r,q) \leq 2M$ based on our boundedness assumptions on f, f^*, and using those same boundedness assumptions for the second inequality as well. We define

$$C^*(r,q) = \limsup_{n\to\infty} C(n,r,q),$$
$$\bar{C}^*(r,q) = E_x \min\left\{2M\ ,\ \frac{\mu(B_{x,r})}{\mu(B_{x,qr})}\left[\mathrm{MAD}_{P_{x,r}(f^*)} + 2H\left(\mathrm{MAD}_{P_{x,r}}(f^*)\right)\right]\right\}.$$

Hence, based on (9) we have

$$C^*(r,q) \leq \bar{C}^*(r,q) \tag{10}$$

We will now need the following lemma:

Lemma 14 *[Devroye, 1981; see the proof of Lemma 2.2] For any measure μ, for almost every x,*

$$\lim_{r\to 0} \frac{r^d}{\mu(B_{x,r})} = c_x \quad , \quad |c_x| < \infty.$$

That is, the limit exists and is finite.

We now consider $\bar{C}^*$ for fixed q and varying r. By Lemma 14, we know that for almost every x,

$$\lim_{r\to 0} \frac{\mu(B_{x,r})}{\mu(B_{x,qr})} = \lim_{r\to 0} \frac{c_x r^d}{c_x q^d r^d} = \frac{1}{q^d} \tag{11}$$

and using Lemma 12 we can consider the MAD, as follows. First, by Lemma 12 we have, for almost every x,

$$\lim_{r\to 0} E_{x'\sim P_{x,r}} f^*(x') = f^*(x)$$

so, for almost every x,

$$\begin{aligned}
\lim_{r\to 0} \mathrm{MAD}_{P_{x,r}}(f^*) &= \lim_{r\to 0} E_{x'\sim P_{x,r}} |f^*(x') - E_{x''\sim P_{x,r}} f^*(x'')| \\
&\leq \lim_{r\to 0} E_{x'\sim P_{x,r}} |f^*(x') - f^*(x)| + |f^*(x) - E_{x''\sim P_{x,r}} f^*(x'')| \\
&= \lim_{r\to 0} E_{x'\sim P_{x,r}} |f^*(x') - f^*(x)| \\
&= 0
\end{aligned}$$

using Corollary 13 for the last equality. Combining this with (11), and using the properties of H, we get

$$\lim_{r\to 0} \bar{C}^*(r,q) = \lim_{r\to 0} E_x \min\left\{2M\ ,\ \frac{\mu(B_{x,r})}{\mu(B_{x,qr})}\left[\mathrm{MAD}_{P_{x,r}}(f^*) + 2H\left(\mathrm{MAD}_{P_{x,r}}(f^*)\right)\right]\right\} = 0$$

since we have convergence to 0 for almost every x in the expected value, and can apply the dominated convergence theorem (for which the bound of $2M$ is crucial). Consequently, due to (10) we have

$$\lim_{r\to 0} C^*(r,q) \leq \lim_{r\to 0} \bar{C}^*(r,q) = 0. \tag{12}$$

We now turn to defining $\{R_k\}$ and Q, using what we have seen thus far. Recall that $T = \{T_k\}$ is arbitrary and that $T_k \searrow 0$. Define in a recursive manner, for every $q \in \mathbf{T}$ (where recall that $\mathbf{T}$ is the set containing all the members in the sequence T),

$$\begin{aligned} &K(T_0) = 0, \\ &K(q) = \min\left\{ k \in \mathbb{N} \quad : \quad \begin{array}{c} \forall q' \in \mathbf{T}, q' > q \\ k > K(q') \end{array} \quad \text{and} \quad \begin{array}{c} \forall k' \geq k \, \forall q' \in \mathbf{T}, q' \geq q \\ C^*(T_{k'}, q') \leq q' \end{array} \right\} \end{aligned}$$

(since $T_k \searrow 0$, we start by defining K for the largest value in T, which is T_0, and then proceed to lower ones). Note that the clause regarding C^* is fulfillable by (12) for individual q', and since we consider a finite number of such q', we can take k large enough for them all. Note also that we ensure that $K(T_k)$ strictly increases, for which we rely on the fact that T_k strictly descends.

We now define Q:

$$\begin{aligned} Q(T_0), \ldots, Q(T_{K(T_1)}) &= T_0, \\ Q(T_{K(T_1)+1}), \ldots, Q(T_{K(T_2)}) &= T_1, \\ &\vdots \\ Q(T_{K(T_k)+1}), \ldots, Q(T_{K(T_{k+1})}) &= T_k, \\ &\vdots \end{aligned} \tag{13}$$

Then according to these definitions, for any $Q' \in \mathcal{Q}(T), Q' \geq Q$, and any (large enough) $k' \in \mathbb{N}$,

$$Q'(T_{k'}) \geq Q(T_{k'}) = T_k \quad \text{for some } k \text{ fulfilling} \quad k' > K(T_k)$$

and hence, for all large enough k',

$$C^*(T_{k'}, Q'(T_{k'})) \leq Q'(T_{k'}). \tag{14}$$

We now work towards defining $R = \{R_k\}$, which just as with Q will be done in two stages. First, we define in a recursive manner, for every $r, q \in \mathbf{T}$,

$$\begin{aligned} &N(T_0, T_0) = 0, \\ &N(r, q) = \\ &\min\left\{ \tilde{n} \in \mathbb{N} \quad : \quad \begin{array}{c} \forall r', q' \in \mathbf{T}, r' > r, q' \geq q \\ \tilde{n} > N(r', q') \end{array} \quad \text{and} \quad \begin{array}{c} \forall \tilde{n}' \geq \tilde{n} \, \forall r', q' \in \mathbf{T}, r' \geq r, q' \geq q \\ C(\tilde{n}', r', q') \leq 2C^*(r', q') \end{array} \right\}. \end{aligned}$$

Note that the clause regarding C, C^* is fulfillable by the definition of C^* as the $\limsup$ of C, and hence we can achieve it over a set of finite q, r as well. Note also that we ensure $N(r, q)$ strictly increases when r strictly descends (and q does not rise).

Define, for any $Q' \in \mathcal{Q}(T)$,

$$\begin{aligned} R_0, \ldots, R_{N(T_1, Q'(T_1))} &= T_0, \\ R_{N(T_1, Q'(T_1))+1}, \ldots, R_{N(T_2, Q'(T_2))} &= T_1, \\ &\vdots \\ R_{N(T_k, Q'(T_k))+1}, \ldots, R_{N(T_{k+1}, Q'(T_{k+1}))} &= T_k, \\ &\vdots \end{aligned}$$

Then, for any $R' \in \mathcal{R}(T), R' \geq R$, and if n is large enough,

$$R'_n \geq R_n = T_k \text{ for some } k \text{ fulfilling } \quad n > N(T_k, Q'(T_k)).$$

Note that $Q'(R'_n) \geq Q'(T_k)$. Thus, by the definition of $N(\cdot,\cdot)$,

$$C(n, R'_n, Q'(R'_n)) \leq 2C^*(R'_n, Q'(R'_n)).$$

If we now also assume that $Q' \geq Q$, where Q is as defined in (13), then by (14) we get

$$C(n, R'_n, Q'(R'_n)) \leq 2C^*(R'_n, Q'(R'_n)) \leq 2Q'(R'_n) \underset{n\to\infty}{\longrightarrow} 0$$

completing the proof.

Appendix C. Proof of Lemma 9

1. First, note that

$$\forall P \qquad E_{S_n,x} \left| f_{|\cdot|}(S_n,x) - |f^*(x)| \right| \leq E_{S_n,x} \left| f_{\text{ker}}(S_n,x) - f^*(x) \right| \longrightarrow 0. \tag{15}$$

Now, notice that for any P, c, d,

$$\begin{aligned}
D_n(f_c, f_d) &= E_{S_n,x} \left| c(S_n,x) f_{|\cdot|}(S_n,x) - d(S_n,x) f_{|\cdot|}(S_n,x) \right| \\
&= E_{S_n,x} \left| c(S_n,x) - d(S_n,x) \right| f_{|\cdot|}(S_n,x) \\
&= E_{S_n,x} \left| c(S_n,x) - d(S_n,x) \right| \left| 2\eta(x) - 1 \right| \\
&\quad + E_{S_n,x} \left| c(S_n,x) - d(S_n,x) \right| \left(f_{|\cdot|}(S_n,x) - |2\eta(x) - 1| \right) \\
&= \widetilde{D}_n(c,d) + E_{S_n,x} \left| c(S_n,x) - d(S_n,x) \right| \left(f_{|\cdot|}(S_n,x) - |2\eta(x) - 1| \right).
\end{aligned}$$

The last expression converges to 0 by (15) since

$$E_{S_n,x} \left| c(S_n,x) - d(S_n,x) \right| \left(f_{|\cdot|}(S_n,x) - |2\eta(x)-1| \right) \leq 2E_{S_n,x} \left| f_{|\cdot|}(S_n,x) - |2\eta(x)-1| \right| \to 0$$

which leads directly to the result we wanted.
2. By part 1 we know that $D_n(f_c, f_{c^*}) \to 0 \iff \widetilde{D}_n(c, c^*) \to 0$, so it suffices to prove that $\forall P \quad D_n(f_{c^*}, f^*) \to 0$, which can be shown using (15):

$$D_n(f_{c^*}, f^*) = E_{S_n,x} \left| c^*(x) f_{|\cdot|}(S_n,x) - f^*(x) \right| = E_{S_n,x} \left| f_{|\cdot|}(S_n,x) - |f^*(x)| \right| \to 0.$$

3. We consider $D_n\left((\bar{f}_c)|_r^q, f_{\bar{c}|_r^q} \right)$; later we will replace q, r with $\{Q(R_k)\}, \{R_k\}$. By the definitions,

$$\begin{aligned}
&D_n\left((\bar{f}_c)|_r^q, f_{\bar{c}|_r^q} \right) \\
&= E_{S_n,x} \left| E_{x' \sim P_{x,qr}} c(S_n(x,r),x') f_{|\cdot|}(S_n(x,r),x') - \operatorname{sign}\left(E_{x' \sim P_{x,qr}} c(S_n(x,r),x') \right) f_{|\cdot|}(S_n,x) \right| \\
&\leq E_{S_n,x} \left| E_{x' \sim P_{x,qr}} c(S_n(x,r),x') - \operatorname{sign}\left(E_{x' \sim P_{x,qr}} c(S_n(x,r),x') \right) \right| |f^*(x)| \\
&\quad + E_{S_n,x} \left| \operatorname{sign}\left(E_{x' \sim P_{x,qr}} c(S_n(x,r),x') \right) \left(f_{|\cdot|}(S_n,x) - |f^*(x)| \right) \right| \\
&\quad + E_{S_n,x} \left| E_{x' \sim P_{x,qr}} c(S_n(x,r),x') \left(f_{|\cdot|}(S_n(x,r),x') - |f^*(x)| \right) \right| \qquad (16) \\
&\leq E_{S_n,x} \left| E_{x' \sim P_{x,qr}} c(S_n(x,r),x') - c^*(x) \right| |f^*(x)| \\
&\quad + E_{S_n,x} \left| f_{|\cdot|}(S_n,x) - |f^*(x)| \right| \\
&\quad + E_{S_n,x} E_{x' \sim P_{x,qr}} \left| f_{|\cdot|}(S_n(x,r),x') - |f^*(x)| \right|
\end{aligned}$$

where we used the fact that $|a-\operatorname{sign}(a)| \leq |a-b|$ for any $b \in \{-1,+1\}$. We now consider the final three expressions separately, denoting them (1),(2),(3):

(1) :
$$\begin{aligned}
& E_{S_n,x}\left|E_{x'\sim P_{x,qr}} c(S_n(x,r),x') - c^*(x)\right| |f^*(x)| \\
&= E_{S_n,x}\left|E_{x'\sim P_{x,qr}} c(S_n(x,r),x')|f^*(x)| - f^*(x)\right| \\
&\leq E_{S_n,x}\left|E_{x'\sim P_{x,qr}} c(S_n(x,r),x')\left(|f^*(x)| - f_{|\cdot|}(S_n(x,r),x')\right)\right| \\
&\quad + E_{S_n,x}\left|E_{x'\sim P_{x,qr}} c(S_n(x,r),x') f_{|\cdot|}(S_n(x,r),x') - f^*(x)\right| \\
&\leq E_{S_n,x}E_{x'\sim P_{x,qr}}\left||f^*(x)| - f_{|\cdot|}(S_n(x,r),x')\right| \\
&\quad + E_{S_n,x}\left|E_{x'\sim P_{x,qr}} c(S_n(x,r),x') f_{|\cdot|}(S_n(x,r),x') - f^*(x)\right|.
\end{aligned}$$

Of the last two expressions, the first is equal to the last of the three expressions we arrived at in (16), so we can consider later on double the value of that expression instead of handling it here. Regarding the second, it is simply equal to $D_n\left((\bar{f}_c)|_r^q, f^*\right)$, which we know to converge to 0 for large-enough $Q,\{R_k\}$ since f_c is UALC on P.

(2) : This converges to 0 by (15).

(3) : Since

$$E_{S_n,x}E_{x'\sim P_{x,qr}}\left|f_{|\cdot|}(S_n(x,r),x') - |f^*(x)|\right| \leq E_{S_n,x}E_{x'\sim P_{x,qr}}\left|f_{\text{ker}}(S_n(x,r),x') - f^*(x)\right|$$

we can use the same techniques as in the proof of Lemma 6:

$$\begin{aligned}
& E_{S_n,x}E_{x'\sim P_{x,qr}}\left|f_{\text{ker}}(S_n(x,r),x') - f^*(x)\right| \\
&\leq E_{S_n,x}E_{x'\sim P_{x,qr}}\left|f_{\text{ker}}(S_n(x,r),x') - f^*(x')\right| \\
&\quad + E_x E_{x'\sim P_{x,qr}}\left|f^*(x') - f^*(x)\right| \\
&= E_x \frac{\mu(B_{x,r})}{\mu(B_{x,rq})} E_m E_{\widetilde{S}\sim P_{x,r}\,|\,m}\, E_{x'\sim P_{x,r}}\left|f_{\text{ker}}(\widetilde{S},x') - f^*(x')\right| 1\{x' \in B_{x,rq}\} \\
&\quad + E_x E_{x'\sim P_{x,qr}}\left|f^*(x') - f^*(x)\right| \\
&\leq E_x \frac{\mu(B_{x,r})}{\mu(B_{x,rq})} E_m E_{\widetilde{S},x'\sim P_{x,r}\,|\,m}\left|f_{\text{ker}}(\widetilde{S},x') - f^*(x')\right| \\
&\quad + E_x E_{x'\sim P_{x,qr}}\left|f^*(x') - f^*(x)\right|
\end{aligned}$$

where, as in previous proofs, $m = |S_n(x,r)|$. The expression before last converges to 0 for every $x \in \operatorname{supp}_X(P)$ and fixed $r,q>0$ by the consistency of f_{ker} on $P_{x,r}$ (using the fact that $m \to \infty$ a.s.), and therefore in a similar way as that performed in Lemma 6 we can see that the entire expression (with expected value over x) converges to 0 for slowly-enough descending $Q,\{R_k\}$. The final expression converges to 0 for any $qr \to 0$ by Corollary 13.

Thus, by replacing q,r with large-enough $Q,\{R_k\}$ we can see that the original expression converges to 0, as required.

References

C. G. Atkeson, A. W. Moore, and S. Schaal. Locally weighted learning. *Artificial Intelligence Review*, 11:11–73, 1997.

P. L. Bartlett and M. Traskin. Adaboost is consistent. In *Advances in Neural Information Processing Systems 19*, pages 105–112. MIT Press, Cambridge, MA, 2007.

M. Belkin and P. Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Computation*, 15(6):1373–1396, 2003.

Y. Bengio, O. Delalleau, and N. Le Roux. The curse of highly variable functions for local kernel machines. In *Advances in Neural Information Processing Systems 18*, pages 107–114. MIT Press, Cambridge, MA, 2006.

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

W. Cleveland and C. Loader. Smoothing by local regression: Principles and methods. Technical report, AT&T Bell Laboratories, Murray Hill, NY., 1995. Available at `http://citeseer.ist.psu.edu/194800.html`.

L. Devroye. On the almost everywhere convergence of nonparametric regression function estimates. *Annals of Statistics*, 9:1310–1319, 1981.

L. Devroye and T. J. Wagner. Distribution-free consistency results in nonparametric discrimination and regression function estimation. *Annals of Statistics*, 8(2):231–239, 1980.

L. Devroye, L. Gyorfi, and G. Lugosi. *A Probabilistic Theory of Pattern Recognition*. Springer-Verlag, New York, NY, 1996.

Y. Freund and R. Schapire. A short introduction to boosting. *J. Japan. Soc. for Artif. Intel.*, 14(5): 771–780, 1999.

L. Gyorfi, M. Kohler, A. Krzyzak, and H. Walk. *A Distribution-Free Theory of Nonparametric Regression*. Springer, Berlin, 2002.

O. Kouropteva, O. Okun, and M. Pietikine. Supervised locally linear embedding algorithm for pattern recognition. *Lecture Notes in Computer Science*, 2652:386–394, 2003.

H. Li, L. Teng, W. Chen, and I. Shen. Supervised learning on local tangent space. In L. Gyorfi, editor, *Lecture Notes in Computer Science*, pages 546–551. Springer-Verlag, 2005.

G. Lugosi and K. Zeger. Nonparametric estimation via empirical risk minimization. *IEEE Transactions on Information Theory*, 41(3):677–687, 1995.

S. Roweis and L. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–2326, 2000.

G. Saunders, A. Gammerman, and V. Vovk. Ridge regression learning algorithm in dual variables. In *Proc. 15th International Conf. on Machine Learning*, pages 515–521. Morgan Kaufmann, San Francisco, CA, 1998.

A. J. Smola and B. Schoelkopf. A tutorial on support vector regression, 1998. Available at `http://citeseer.ist.psu.edu/smola03tutorial.html`.

I. Steinwart. Support vector machines are universally consistent. *Journal of Complexity*, 18:768–791, 2002.

C. J. Stone. Consistent nonparametric regression. *Annals of Statistics*, 5:595–645, 1977.

V. N. Vapnik. *Statistical Learning Theory*. John Wiley and Sons, New York, NY, 1998.

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

A. Zakai and Y. Ritov. How local should a learning method be? In *21st Annual Conference on Learning Theory (COLT)*, pages 205–216, 2008.

T. Zhang. Statistical behavior and consistency of classification methods based on convex risk minimization. *Annals of Statistics*, 32(1):56–134, 2004.

Journal of Machine Learning Research 10 (2009) 857-882 Submitted 5/08; Revised 1/09; Published 4/09

Stable and Efficient Gaussian Process Calculations

Leslie Foster FOSTER@MATH.SJSU.EDU
Alex Waagen AWAAGEN@MAILBOLT.COM
Nabeela Aijaz NABBO_A@YAHOO.COM
Michael Hurley MHURLEY@GMAIL.COM
Apolonio Luis JPOLOROLU@GMAIL.COM
Joel Rinsky JOEL_RINSKY@YAHOO.COM
Chandrika Satyavolu CHANDRIKA_S84@YAHOO.COM
Department of Mathematics
San Jose State University
San Jose, CA 95192, USA

Michael J. Way MICHAEL.J.WAY@NASA.GOV
NASA Goddard Institute for Space Studies
New York, NY, 10025, USA

Paul Gazis PGAZIS@MAIL.ARC.NASA.GOV
Ashok Srivastava ASHOK@EMAIL.ARC.NASA.GOV
NASA Ames Research Center
Intelligent Systems Division, MS 269-4
Moffett Field, CA 94035, USA

Editor: Chris Williams

Abstract

The use of Gaussian processes can be an effective approach to prediction in a supervised learning environment. For large data sets, the standard Gaussian process approach requires solving very large systems of linear equations and approximations are required for the calculations to be practical. We will focus on the subset of regressors approximation technique. We will demonstrate that there can be numerical instabilities in a well known implementation of the technique. We discuss alternate implementations that have better numerical stability properties and can lead to better predictions. Our results will be illustrated by looking at an application involving prediction of galaxy redshift from broadband spectrum data.

Keywords: Gaussian processes, low rank approximations, numerical stability, photometric redshift, subset of regressors method

1. Introduction

The use of Gaussian processes can be an effective approach to prediction in a supervised learning environment. For large data sets, the standard Gaussian process approach requires solving very large systems of linear equations and approximations are required for the calculations to be practical. We will focus on the subset of regressors technique which involves low rank approximations. The goal of the paper is to describe techniques that are fast—requiring $O(nm^2)$ operations where n is the number of data points available for training and m is the rank of a low rank approximation—and have good numerical stability properties in the sense that the growth of computer arithmetic errors is limited.

The paper begins with a review of Gaussian processes and the subset of regressors approach. We then show that implementation of the subset of regressors method using normal equations can be inaccurate due to computer arithmetic errors. A key contribution of the paper is a discussion of alternative implementations of the subset of regressors technique that have improved numerical stability. Another valuable contribution of the paper is a discussion of how pivoting can be incorporated in the subset of regressors approach to further enhance numerical stability. We discuss the algorithm of Lucas (2004, pp. 4-5) for construction of a partial Cholesky factorization with pivoting and emphasize that with this algorithm the flop count, including subset selection, of the subset of regressors calculations is $O(nm^2)$.

In Section 2 we provide background about using Gaussian processes to facilitate prediction. In Section 3 we discuss how low rank approximations lead to the subset of regressors approach. In Section 4 we describe why a commonly used implementation of this technique may suffer from numerical instabilities and in Section 5 we propose two alternative implementations that have better numerical stability properties. In Section 6 we address the subset selection problem and indicate that a solution to this problem can enhance numerical stability. In Section 7 we discuss tools that aid in the choice of rank in the low rank approximation. In Section 8 we illustrate that the numerical stability issues addressed in Section 4 can lead to unacceptably large growth of computer arithmetic errors in an important application involving prediction of galaxy redshift from broadband spectrum data. Our alternative implementations of the subset of regressors method overcome these difficulties. Also in Section 8 we discuss code, available at `http://dashlink.arc.nasa.gov/algorithm/stableGP`, that implements our ideas. Finally, in Section 9 we summarize our results.

2. Gaussian Processes

Supervised learning is the problem of learning input-output mappings using empirical data. We will assume that a training data set is known consisting of a $n \times d$ matrix X of input measurements and a n by 1 vector y of output or target values. The task is to use the training data set to develop a model that can be used to make prediction with new data. We will assume the new data, called the testing data, is contained in an $n^* \times d$ matrix X^* of inputs. The $n^* \times 1$ vector y^* will represent the target values corresponding to X^*. The goal is to predict the value of y^* given X, y, and X^*.

In the Gaussian process approach the prediction of y^* involves selection of a covariance function $k(x,x')$, where x and x' are vectors with d components. It is required that the covariance function be positive semidefinite (Rasmussen and Williams, 2006, p. 80) which implies that the $n \times n$ covariance matrix K with entries $K_{ij} = k(x_i, x_j)$ where x_i and x_j are rows of X is symmetric positive semidefinite (SPS), so that $v^T K v \geq 0$ for any $n \times 1$ real column vector v. The covariance function can be used to construct K and also the n^* by n cross covariance matrix K^* where $K^*_{ij} = k(x^*_i, x_j)$ where x^*_i is the i^{th} row of X^*. The prediction $\widehat{y}^*$ for y^* is given by the Gaussian processes equation (Rasmussen and Williams, 2006, p. 17):

$$\widehat{y}^* = K^*(\lambda^2 I + K)^{-1} y. \tag{1}$$

The parameter λ in this equation represents the noise in the measurements of y and, in practice, it is often selected to improve the quality of the model (Rasmussen and Williams, 2006).

It is often not clear how to choose the covariance function k. There exist many different covariance functions that apply broadly to many cases. Potential covariance function choices include the squared exponential (sometimes called the radial basis function), Matern, rational quadratic, neural network, polynomial or other covariance functions (Rasmussen and Williams, 2006, pp. 79-102). Most of these covariance functions contain free parameters that need to be selected. Such parameters and λ in (1) are called hyperparameters. We will not focus on the choice of a covariance function or alternative methods for selection of hyperparameters in this paper. In the examples discussed in Section 8 we tried out a variety of covariance functions and selected the one that provided the best predictions. Hyperparameters were selected using the Matlab routine minimize (Rasmussen and Williams, 2006, pp. 112-116, 221) which finds a (local) maximum of the marginal likelihood function calculated using the training set data.

We should mention that the choice of the hyperparameter λ can affect the numerical stability of the Gaussian process calculations. Generally larger values of λ lead to reduced computer arithmetic errors but a large value of λ may be a poor theoretical choice—note that $\widehat{y}^* \to 0$ as $\lambda \to \infty$. One needs to select a value of λ that balances such competing errors. The choice of λ in Gaussian processes is closely related to the parameter choice in ridge regression in the statistics literature (Montgomery et al., 2006, pp. 344-355) and in the literature on regularization (Hansen, 1998, pp. 175-208). As mentioned above we select hyperparameters, including λ, using the routine minimize (Rasmussen and Williams, 2006, pp. 112-116, 221). This technique worked well for the practical example presented in Section 8 when used with our algorithms with improved numerical stability.

We should note that Gaussian process approach also leads to an equation for C the covariance matrix for the predictions in (1). If the $n^* \times n^*$ matrix K^{**} has entries $K^*_{ij} = k(x^*_i, x^*_j)$ then (Rasmussen and Williams, 2006, pp. 79-102):

$$C = K^{**} - K^*(\lambda I + K)^{-1} K^{*T}. \tag{2}$$

The superscript T indicates transpose. The pointwise variance of the predictions is diag(C), the diagonal of the $n^* \times n^*$ matrix C.

3. Low Rank Approximation: The Subset of Regressors Method

In (1) the matrix $(\lambda^2 I + K)$ is an n by n matrix that, in general, is dense (that is has few zero entries). Therefore for large n, for example $n \geq 10000$, it is not practical to solve (1) since the memory required to store K is $O(n^2)$ and the number of floating point operations required to solve (1) is $O(n^3)$. Therefore for large n it is useful to develop approximate solutions to (1).

To do this, for some $m < n$, we can partition the matrices K and K^* as follows:

$$K = \begin{pmatrix} K_{11} & K_{12} \\ K_{21} & K_{22} \end{pmatrix} = \begin{pmatrix} K_1 & K_2 \end{pmatrix}, \; K^* = \begin{pmatrix} K^*_1 & K^*_2 \end{pmatrix}. \tag{3}$$

Here K_{11} is $m \times m$, K_{21} is $(n-m) \times m$, $K_{12} = K_{21}^T$ is $m \times (n-m)$, K_{22} is $(n-m) \times (n-m)$, K_1 is $n \times m$, K_2 is $n \times (n-m)$, K^*_1 is $n^* \times m$ and K^*_2 is $n^* \times (n-m)$. Next we approximate K and K^* using

$$K \cong \widehat{K} \equiv K_1 K_{11}^{-1} K_1^T \tag{4}$$

and

$$K^* \cong \widehat{K}^* \equiv K^*_1 K_{11}^{-1} K_1^T$$

and in (1) we replace K with $\widehat{K}$ and K^* with $\widehat{K}^*$. Therefore

$$\widehat{y}^* \cong \widehat{y}^*_N \equiv \widehat{K}^* (\lambda^2 I + \widehat{K})^{-1} y =$$

$$K_1^* K_{11}^{-1} K_1^T (\lambda^2 I + K_1 K_{11}^{-1} K_1^T)^{-1} y =$$
$$K_1^* K_{11}^{-1} (\lambda^2 I + K_1^T K_1 K_{11}^{-1})^{-1} K_1^T y, \text{ so that}$$
$$\widehat{y}_N^* = K_1^* (\lambda^2 K_{11} + K_1^T K_1)^{-1} K_1^T y. \tag{5}$$

Equation (5) is called the subset of regressors method (Rasmussen and Williams, 2006, p. 176) and was proposed, for example, in Wahba (1990, p. 98) and Poggio and Girosi (1990, p. 1489). As we discuss in the next section the subscript N stands for normal equations. We refer to use of (5) as the SR-N approach.

If $m << n$ then (5) is substantially more efficient than (1). For large n the leading order term in the operation count for the calculations in (5) is nm^2 flops or floating point operations (where a floating point operation is either an addition, subtraction, multiplication or division), whereas the calculations in (1) require approximately $2n^3/3$ flops. If $n = 180,000$ and $m = 500$, as in an example discussed later, the solution to (1) requires approximately 4×10^{15} flops which is five order of magnitudes greater than the approximately 4×10^{10} flops required to solve (5). Furthermore, to use (1) one needs to calculate all $n^2 + nn^*$ elements of K and K^* whereas (5) requires that one calculate only the $nm + n^*m$ elements in K_1 and K_1^*. This also improves the efficiency of the calculations and will reduce the memory requirements dramatically.

We should add that if in Equation (2) we use the approximations (4), (3) and

$$K^{**} \cong \widehat{K}^{**} \equiv K_1^* K_{11}^{-1} K_1^{*T}$$

then, in (2) replacing K with $\widehat{K}$, K^* with $\widehat{K}^*$, K^{**} with $\widehat{K}^{**}$ and using algebra similar to that used in deriving (5), it follows that

$$C \cong \widehat{C}_N \equiv \lambda^2 K_1^* (\lambda^2 K_{11} + K_1^T K_1)^{-1} K_1^{*T}. \tag{6}$$

For an alternate derivation of (6) see Rasmussen and Williams (2006, p. 176). Also diag(C) $\cong$ diag($\widehat{C}_N$) so that diag($\widehat{C}_N$) provides approximations for the variance of the predictions.

4. Numerical Instability

The sensitivity of a problem measures the growth of errors in the answer to the problem relative to perturbations in the initial data to the problem, assuming that that there are no errors in the solution other than the errors in the initial data. A particular algorithm implementing a solution to the problem is numerically stable if the error in the answer calculated by the algorithm using finite precision arithmetic is closely related (a modest multiple of) the error predicted by the sensitivity of the problem. An algorithm is unstable if the error in the answer calculated by the algorithm is substantially greater than the error predicted by the sensitivity of the problem.

A straightforward implementation for the subset of regressors approximation using (5) has a potential numerical instability. To see this note that since K is SPS it follows that the $m \times m$ submatrix K_{11} is also. Therefore we can factor the matrix K_{11} with a Cholesky factorization (Golub and Van Loan, 1996, p. 148)

$$K_{11} = V_{11} V_{11}^T \tag{7}$$

where V_{11} is an $m \times m$ lower triangular matrix. Now let

$$A = \begin{pmatrix} K_1 \\ \lambda V_{11}^T \end{pmatrix} \text{ and } b = \begin{pmatrix} y \\ 0 \end{pmatrix}, \tag{8}$$

where 0 is an $m \times 1$ zero vector, A is an $(n+m) \times m$ matrix and b is a $(n+m) \times 1$ vector. Consider the least square problem:

$$\min_x ||Ax - b|| \tag{9}$$

where the norm is the usual Euclidean norm. The normal equations solution (Golub and Van Loan, 1996, p. 237) to this least squares problem is $x = (A^T A)^{-1} A^T b = (\lambda^2 V_{11} V_{11}^T + K_1^T K_1)^{-1} K_1^T y$ and so by (7)

$$x_N = (\lambda^2 K_{11} + K_1^T K_1)^{-1} K_1^T y. \tag{10}$$

Therefore the solution $\widehat{y}_N^*$ presented in (5) can also be written

$$\widehat{y}_N^* = K_1^* x_N. \tag{11}$$

The subscript N indicates the use of the normal equations solution to (8).

The potential difficulty with the above solution is that the intermediate result x_N is the solution to a least squares problem using the normal equation. It is well known that the use of the normal equation can, in some cases, introduce numerical instabilities and can be less accurate than alternative approaches. As discussed in Golub and Van Loan (1996, p. 236-245) the sensitivity of the least squares problem (9) is roughly proportional to $\text{cond}(A) + \rho_{LS} \text{cond}^2(A)$, where $\rho_{LS} = ||b - Ax||$ and $\text{cond}(A) = ||A|| ||(A^T A)^{-1} A^T||$ is the condition number of A. The problem with the normal equations solution to (9) is that the accuracy of the calculated solution is (almost always) proportional to $\text{cond}^2(A)$, the square of the condition number of A, whereas in the case that ρ_{LS} is small the sensitivity of the least squares problem is approximately $\text{cond}(A)$. To quote from Golub and Van Loan (1996, p. 245):

> We may conclude that if ρ_{LS} is small and $\text{cond}(A)$ is large, then the method of normal equations … will usually render a least squares solution that is less accurate than a stable QR approach.

We will discuss use of the stable QR approach and another alternative to the normal equations in the next section.

5. Improving Numerical Stability

The calculation of $\widehat{y}_N^*$ as given by (5) is equivalent to the solution to (9) and (11) using the normal Equations (10). We can reduce the computer arithmetic errors in the calculation of $\widehat{y}_N^*$ if we develop algorithms that avoid the use of the normal equations in the solution to (9) and (11) . We will present two alternative algorithms for solving (9) and (11). We should add that although these algorithms can have better numerical properties than use of (5), all the algorithms presented in this section are mathematically (in exact arithmetic) equivalent to (5).

5.1 The Subset of Regressors Using a QR Factorization

We first describe use of the QR factorization to solve (9). In this approach (Golub and Van Loan, 1996, p.239) one first factors $A = QR$ where Q is an $(n+m) \times m$ matrix with orthonormal columns and R is an $m \times m$ right triangular matrix. Then

$$x_Q = R^{-1} Q^T b = R^{-1} Q^T \begin{pmatrix} y \\ 0 \end{pmatrix} \tag{12}$$

so that

$$\widehat{y}_Q^* = K_1^* x_Q = K_1^* R^{-1} Q^T \begin{pmatrix} y \\ 0 \end{pmatrix}. \tag{13}$$

With the above algorithm $\widehat{y}^*$ can still be solved quickly. Assuming that the elements of K_1 and K_1^* have been calculated, and that $m << n$, then the approximate number of operations for the QR approach is $2nm^2$ flops. Therefore both the QR and normal equations approach require $O(nm^2)$ flops.

We should also note that we can use the QR factorization to reduce computer arithmetic errors in the computation of the approximate covariance matrix in (6). If we let

$$\widehat{C}_{QR} \equiv \lambda^2 (K_1^* R^{-1})(K_1^* R^{-1})^T$$

then mathematically (in exact arithmetic) $\widehat{C}_N$ and $\widehat{C}_{QR}$ are the same. However, for reasons similar to those discussed in Section 4, the computer arithmetic errors in $\widehat{C}_{QR}$ will usually be smaller than those in $\widehat{C}_N$, assuming, for example, that $\widehat{C}_N$ is computed using a Cholesky factorization of $\lambda^2 K_{11} + K_1^T K_1$.

We will refer to the subset of regressors method using the QR factorization as the SR-Q method. We should add that the use of a QR factorization in equations related to Gaussian process calculations is not new. For example Wahba (1990, p. 136) discusses using a QR factorization for cross validation calculations.

5.2 The V Method

If we assume the V_{11} is nonsingular we can define the $n \times m$ matrix V:

$$V = K_1 V_{11}^{-T} \tag{14}$$

where the superscript $-T$ indicates inverse transpose. Note that by (7) it follows that V is lower trapezoidal and that $V = \begin{pmatrix} V_{11} \\ V_{21} \end{pmatrix}$, where $V_{21} = K_{21} V_{11}^{-T}$. Substituting $K_1 = V V_{11}^T$ and (9) into (10) we get

$$x_V = V_{11}^{-T} (\lambda^2 I + V^T V)^{-1} V^T y \tag{15}$$

so that

$$\widehat{y}_V^* = K_1^* x_V = K_1^* V_{11}^{-T} (\lambda^2 I + V^T V)^{-1} V^T y. \tag{16}$$

We should note that this formulation of the subset of regressors method is not new. It is presented, for example, in Seeger et al. (2003) and Wahba (1990, p. 136) presents a formula closely related to (16). We will call the formula (16) for $\widehat{y}^*$ the V method. We should note, as will be seen in Section 6.2, that one can calculate V as part of a partial Cholesky factorization rather than using (14).

We will see in our numerical experiments and the theoretical analysis in Section 6 that the V method is intermediate in terms of growth of computer arithmetic errors between the normal equations and QR approach. Often, but not always, the accuracy of the V method is close to that of the QR approach.

Assuming that the elements of K_1 and K_1^* have been calculated, and that $m << n$, then the approximate number of operations for the V method is $2nm^2$ flops—approximately nm^2 flops to form V and another nm^2 flops to solve for x_V using (15). This is approximately the same as SR-Q and approximately twice the flop count for the SR-N method.

We can also compute the approximate covariance matrix with the V method approach:

$$\widehat{C}_V \equiv \lambda^2 K_1^* V_{11}^{-T} (\lambda^2 I + V^T V)^{-1} V_{11}^{-1} K_1^{*T}.$$

In exact arithmetic $\widehat{C}_N, \widehat{C}_{QR}$ and $\widehat{C}_V$ are identical but the computer arithmetic errors are often smaller in $\widehat{C}_V$ and $\widehat{C}_{QR}$ than $\widehat{C}_N$.

5.3 Examples Illustrating Stability Results

We present two sets of examples that illustrate some of the above remarks.

Example 1 *Let the $n \times n$ matrices K be of the form $K = UDU^T$ where U is a random orthogonal matrix (Stewart, 1980) and D is a diagonal matrix with diagonal entries $s_1 \geq s_2 \geq \ldots s_n \geq 0$. Therefore $s_1, s_2, \ldots, s_n$ are the singular values of K. We will choose a vector $w \in R^n$, where R^n is real n dimensional space, of the form $w = \begin{pmatrix} x \\ 0 \end{pmatrix}$ where $x \in R^m$ is a random vector and 0 indicates a zero vector with $(n-m)$ components. We let the target data be $y = Kw$. We will also assume for simplicity that $\lambda = 0$.*

Due to the structure of w each of x_N (10), x_Q (12), and x_V (15) will calculate x exactly in exact arithmetic. Therefore in finite precision arithmetic $||x - \dot{x}||$, with $\dot{x} = x_N, x_Q$ or x_V will be a measure of the computer arithmetic errors in the calculation.

We carried out an experiment $n = 100$, $m = 50$, $s_i = 10^{-(i-1)/5}$, $i = 1, 2, \ldots, m$, and $s_i = 10^{-10}$, $i = m+1, m+2, \ldots, n$ using a set of one hundred random matrices of this type. For this class of matrices the singular values of K vary between 1 and 10^{-10}, $cond(K) = 10^{10}$ and $cond(K_1) \cong 10^{10}$. The results are:

$\dot{x}$ =	x_N	x_V	x_Q
min $\|\|x-\dot{x}\|\|/\|\|x\|\|$	9.3×10^{-1}	5.1×10^{-7}	2.7×10^{-8}
mean $\|\|x-\dot{x}\|\|/\|\|x\|\|$	9.1×10^{0}	3.6×10^{-6}	1.2×10^{-7}
max $\|\|x-\dot{x}\|\|/\|\|x\|\|$	9.6×10^{1}	9.9×10^{-6}	4.5×10^{-7}

Table 1: Min, mean and max errors, $||x - \dot{x}||/||x||$, for 100 matrices and various methods.

For this set of matrices x_Q and x_V have small errors. However x_N has large errors due to its use of normal equations.

Example 2 *This example will illustrate that, although the V method often greatly improves upon the stability of the SR-N method, this is not always the case. For $0 < s \leq 1$ let $C = \begin{pmatrix} s^2 & 10s \\ 10s & 200 \end{pmatrix}$, let the 4×4 matrix $K = \begin{pmatrix} s^2C & 10sC \\ 10sC & 200C \end{pmatrix}$, let $x = \begin{pmatrix} 1/3 \\ 1/3 \end{pmatrix}$, $w = \begin{pmatrix} x \\ 0 \\ 0 \end{pmatrix}$, $\lambda = 0$ let $y = Kw$.*

Due to the structure of w we again have each of x_N, x_Q and x_V will calculate x exactly in exact arithmetic. However, in finite precision arithmetic the calculated values will not be exact. For this example for small s the errors in both x_N and x_V can be significantly larger than the errors in x_Q. For example if $s = 10^{-4}$ we get the following results:

$\dot{x}$ =	x_N	x_V	x_Q
$\|\|x-\dot{x}\|\|/\|\|x\|\|$	8.8×10^{-1}	2.1×10^{-1}	7.7×10^{-11}

Table 2: Errors $||x-\dot{x}||/||x||$ for a 4×4 matrices and various methods.

In Section 6 and Appendix A we will discuss the reason that the V method performs poorly in this example and show that the numerical instability illustrated in this example can be cured by interchanging the columns and rows of K appropriately. Also we should note that although difficulties like the one illustrated here are possible for the V method, experiments like those in Example 1 suggest that such difficulties are not likely. As we discuss in Section 6, the method performed well when we applied it to real world applications.

6. Pivoting and Subset Selection

In Section 5 we discussed low rank approximations to K which involved the first m columns of K. However one can select any subset of the columns to construct a low rank approximation. The choice of these columns or the "active" set is the subset selection problem. This problem has been addressed by, for example, Smola and Bartlett (2001), Seeger et al. (2003), Csato and Opper (2002) and Fine and Scheinberg (2001). The technique that we will use is the same as that in Fine and Scheinberg (2001). However we will focus on the effect of the resulting choice of the active set on the numerical stability of the resulting algorithm. This is a different motivation than the motivations in the above references.

6.1 The Singular Value Decomposition

To pursue this we will first discuss the singular value decomposition which, in a certain sense, produces an optimal low rank approximation to K. The singular value decomposition (SVD) of the symmetric semidefinite matrix K produces the factorization

$$K = UDU^T = \begin{pmatrix} U_1 & U_2 \end{pmatrix} \begin{pmatrix} D_1 & 0 \\ 0 & D_2 \end{pmatrix} \begin{pmatrix} U_1 & U_2 \end{pmatrix}^T$$

where U is an $n\times n$ orthogonal matrix, D is an $n\times n$ diagonal matrix whose diagonal entries $s_1 \geq s_2 \geq \ldots \geq s_n \geq 0$ are the singular values of K, U_1 is $n\times m, U_2$ is $n\times(n-m), D_1$ is an $n\times n$ diagonal matrix, and D_2 is an $(n-m)\times(n-m)$ diagonal matrix. We then can construct the truncated singular value decomposition (TSVD) low rank approximation to K:

$$\widehat{K}_{SVD} = U_1 D_1 U_1^T. \tag{17}$$

The TSVD approximation $\widehat{K}_{SVD}$ is the best low rank approximation (Golub and Van Loan, 1996, p. 72) to K in the sense that

$$\min_{\text{rank}(\widehat{K})=m} ||K-\widehat{K}|| = ||K-\widehat{K}_{SVD}|| = s_{m+1}. \tag{18}$$

Given an $n\times q$ matrix A with rank $m \leq \min(n,q)$ we will define (Björck, 1996, p. 28) the condition number of A to be $\text{cond}(A) = s_1/s_m$ where s_1 and s_m are singular values of A. This definition generalizes to singular matrices the definition of condition number that we used in Section 4 (where A had m columns). It then follows from (17) that

$$\text{cond}(\widehat{K}_{SVD}) = s_1/s_m \tag{19}$$

where s_1 and s_m are singular values of K (which are the same as the singular values of $\widehat{K}_{SVD}$). Thus the singular value decomposition provides two desirable properties:

- Equation (18) indicates that $\widehat{K}_{SVD}$ will be close to K, if there exists a rank m approximation that is close to K, and
- Equation (19) limits the condition number of $\widehat{K}_{SVD}$ which will limit the growth of computer arithmetic errors in the use of $\widehat{K}_{SVD}$.

However, for large n, it is not practical to calculate the SVD of K since the SVD requires $O(n^3)$ operations and is much more expensive than the algorithms described in Section 5 which require $O(nm^2)$ operations. We would like to construct an approximation that requires only $O(nm^2)$ operations and that produces low rank approximations with properties related to (18) and (19).

6.2 Cholesky Factorization with Pivoting

The algorithms describe in Sections 3 and 5 (which are mathematically but not numerically identical) do not satisfy relations related to (18) and (19) as is apparent from the following example.

Example 3 *For the matrix*

$$K = \begin{pmatrix} 1+\varepsilon & 1-\varepsilon & 0 \\ 1-\varepsilon & 1+\varepsilon & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

if we let $m = 2$ then by (4) and (17) we have

$$\widehat{K} = \begin{pmatrix} 1+\varepsilon & 1-\varepsilon & 0 \\ 1-\varepsilon & 1+\varepsilon & 0 \\ 0 & 0 & 0 \end{pmatrix} \textit{ and } \widehat{K}_{SVD} = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

so that, for small ε,

$$||K - \widehat{K}_{SVD}|| = 2\varepsilon \;<<\; 1 = ||K - \widehat{K}|| \textit{ and}$$
$$\textit{cond}(\widehat{K}_{SVD}) = 2 \;<<\; 1/\varepsilon = \textit{cond}(\widehat{K}).$$

For this example the low rank approximation $\widehat{K}$ has two problems: (1) it does not provide a good approximation to K even though a good low rank approximation exists and (2) the condition number of $\widehat{K}$ can be arbitrarily large which potentially could lead to a large growth of computer arithmetic errors.

To overcome the difficulties illustrated in this example we can use a Cholesky factorization, with pivoting, to insure that linearly independent columns and rows appear first. The Cholesky factorization with pivoting produces a decompostion

$$P^T K P = LL^T$$

where P is an $n \times n$ permutation matrix and L is an $n \times n$ lower triangular matrix. To produce our low rank approximations to the Gaussian process equations we do not need to factor all of K, rather it is sufficient to calculate a partial factorization that factors only m columns and rows of $P^T KP$. This is a partial Cholesky factorization with pivoting. If the pivoting is done using complete pivoting (that is the pivoting in the Cholesky factorization is equivalent to using complete pivoting in Gaussian elimination) then there are a variety of algorithms that determine the factorization (Higham, 2002, p. 202; Golub and Van Loan, 1996, p. 149; Lucas, 2004, pp. 4-5 and Fine and Scheinberg, 2001, p. 255). Here we will summarize the algorithm presented in Lucas (2004, pp. 4-5) since it is not as widely known as the algorithms in Higham (2002, p. 202) and Golub and Van Loan (1996, p. 149) and is more efficient in our context. The algorithm below is also the same as that in Fine and Scheinberg (2001, p. 255) except for the stopping criteria.

Algorithm 1: Algorithm for the partial Cholesky factorization

```
Data: an $n \times n$ symmetric positive semidefinite matrix $K$
    a stopping tolerance $tol \geq 0$
    the maximum rank, $max\_rank \leq n$, of the low rank approximation
Result: $m$, the rank of the low rank approximation
    an $n \times m$ partial Cholesky factor $V$
    a permutation vector $piv$
    Note: on completion the first $m$ rows and columns of $P^T KP - VV^T$ are
    zero, where $P$ is a permutation matrix with $P_{piv_i,i} = 1$, $i = 1, \ldots, n$
initialize:
    $d_i = K_{ii}$, $i = 1, \ldots, n$
    $K_{\max} = \max_{i=1,\ldots,n}(d_i)$
    $piv_i = i$, $i = 1, \ldots, n$
    $m = max\_rank$
for $j = 1$ to $max\_rank$ do
    $[d_{\max}, j_{\max}] = \max_{i=j,\ldots,n}(d_i)$
    where $j_{\max}$ is an index where the max is achieved
    if $d_{\max} \leq (tol)K_{\max}$ then
        $m = j - 1$ ;
        exit the algorithm ;
    end
    if $j_{\max} \neq j$ then
        switch elements $j$ and $j_{\max}$ of $piv$ and $d$
        for $i = j+1 : n$ let $u_i$ = element $i$ of column $j_{\max}$ of $P^T KP$
        switch rows $j$ and $j_{\max}$ of the current $n \times (j-1)$ matrix $V$
    end
    $V_{jj} = \sqrt{d_{\max}}$
    for $i = j+1$ to $n$ do
        $V_{ij} = (u_i - \sum_{k=1}^{j-1} V_{ik}V_{jk}) / V_{jj}$
        $d_i = d_i - V_{ij}^2$
    end
end
```

There are two choices of the stopping tolerance tol that have been suggested elsewhere. For the choice $tol = 0$ the algorithm will continue as long as the factorization determines that K is positive definite (numerically). This choice of tol is used in LINPACK's routine xCHDC (Dongarra et al., 1979) and also by Matlab's Cholesky factorization chol (which implements a Cholesky factorization without pivoting). The choice $tol = n \times \varepsilon$ where ε is machine precision is suggested in Lucas (2004, p. 5) and in Higham (2002). The best choice of tol will depend on the application.

There are a number of attractive properties of the partial Cholesky factorization.

- The number of floating point operations in the algorithm is approximately $nm^2 - 2m^3/3$ flops. The calculations to determine the pivoting require only $O(nm)$ flops.
- The algorithm accesses only the diagonal entries of K and elements from m columns of K.
- The storage requirement for the algorithm is approximately $n(m+2)$ floating point numbers plus storage for the integer vector piv and any storage needed to calculate entries in K.
- The accuracy and condition number of the low rank approximation to K produced by the algorithm is related to the accuracy and condition number of the low rank approximation produced by the singular value decomposition. In particular

Theorem 1 *Let the $n \times m$ matrix V be the partial Cholesky factor produced by Algorithm 1 and let*

$$\widehat{K}_P = PVV^T P^T. \tag{20}$$

Also let $\widehat{K}_{SVD}$ be the rank m approximation (17) produced by the singular value decomposition. Then

$$||K - \widehat{K}_P|| \leq c_1 \, ||K - \widehat{K}_{SVD}|| \text{ and} \tag{21}$$

$$\text{cond}(\widehat{K}_P) \leq c_2 \, \text{cond}(\widehat{K}_{SVD}) \text{ where} \tag{22}$$

$$c_1 \leq (n-m)4^m \text{ and } c_2 \leq (n-m)4^m. \tag{23}$$

Proof *The theorem follows from results in Gu and Eisenstat (1996) for the QR factorization with pivoting. First we consider a Cholesky factorization, without pivoting, of K so that $K = LL^T$ where L is and $n \times n$ lower triangular matrix. Let $\sigma_i(A)$ represent the i^{th} singular value of a matrix A. Then, making use of the singular value decomposition, it follows easily that $\sigma_i(K) = \sigma_i^2(L)$, $i = 1, \ldots, n$. Consider a QR factorization of L^T with standard column pivoting (Golub and Van Loan, 1996, p. 249-250) so that $QR = L^T P_1$. The permutation matrix P_1 produced by this QR factorization will be identical, in exact arithmetic, to the permutation matrix produced by the Cholesky factorization with pivoting applied to K (Dongarra et al., 1979, p. 9.26). In addition, the Cholesky factorization, with pivoting, of K is $P_1^T K P_1 = R^T R$, assuming the diagonal entries of R are chosen to be nonnegative (Dongarra et al., 1979, p. 9.2). Now we partition the Cholesky factorization:*

$$P_1^T K P_1 = \begin{pmatrix} R_{11}^T & 0 \\ R_{12}^T & R_{22}^T \end{pmatrix} \begin{pmatrix} R_{11} & R_{12} \\ 0 & R_{22} \end{pmatrix}. \tag{24}$$

It follows from Theorem 7.2 in Gu and Eisenstat (1996, p. 865) that

$$\sigma_1(R_{22}) \leq c_3 \, \sigma_{m+1}(L) \text{ and } \frac{1}{\sigma_m(R_{11})} \leq c_4 \frac{1}{\sigma_m(L)} \text{ where } c_3, c_4 \leq \sqrt{n-m} \; 2^m. \tag{25}$$

Now the first m steps of Cholesky factorization, with pivoting, of K will produce identical results to the m steps of the partial Cholesky factorization described in Algorithm 1. Let

$$V = \begin{pmatrix} V_{11} \\ V_{21} \end{pmatrix} \text{ and } R_1 = \begin{pmatrix} R_{11} & R_{12} \end{pmatrix} \text{ so that } R_1^T = \begin{pmatrix} R_{11}^T \\ R_{12}^T \end{pmatrix}. \tag{26}$$

In the (complete) Cholesky factorization with pivoting of K, after the first m steps of the algorithm additional pivoting will be restricted to the last $n-m$ rows and columns of P^TKP. Let P_2 be a $n \times n$ permutation matrix representing the pivoting in the last $n-m$ steps in the algorithm. Then it follows that

$$P_1 = PP_2,\ V_{11} = R_{11}^T \text{ and } R_1^T = P_2^T V.$$

Therefore

$$\widehat{K}_P = PVV^TP = P_1P_2^TVV^TP_2P_1^T = P_1R_1^TR_1P_1^T. \tag{27}$$

By (18), (24), (25), (26) and (27) we can conclude that

$$||K - \widehat{K}_P|| = ||R_{22}^TR_{22}|| = \sigma_1^2(R_{22}) \le c_3^2\, \sigma_{m+1}^2(L) = c_1\, \sigma_{m+1}(K) = c_1\, ||K - \widehat{K}_{SVD}||.$$

Also, by (25), (27) and the interlace theorem (Björck, 1996, p. 15)

$$\sigma_m(\widehat{K}_P) = \sigma_m^2(R_1) \ge \sigma_m^2(R_{11}) \ge \sigma_m^2(L)/c_4^2 = \sigma_m(K)/c_4^2. \tag{28}$$

Next by (27) and the interlace theorem

$$\sigma_1(\widehat{K}_P) = \sigma_1^2(R_1) \le \sigma_1^2(R) = \sigma_1(K). \tag{29}$$

Finally, (19), (28) and (29) imply that

$$cond(\widehat{K}_P) = \sigma_1(\widehat{K}_P)/\sigma_m(\widehat{K}_P) \le c_4^2\, \sigma_1(K)/\sigma_m(K) = c_2\, cond(\widehat{K}_{SVD}).$$

■

The bounds in (23) on c_1 and c_2 grow exponentially in m and in principle can be large for larger values of m. In practice this appears to be very uncommon. For example the constants c_3 and c_4 in (25) are closely related to $||W||$ where $W = R_{11}^{-1}R_{22}$ (Gu and Eisenstat, 1996, p. 865). Numerical experiments indicate the $||W||$ is almost always small in practice (typically less than 10) (Higham, 2002, p. 207 and Higham, 1990). Therefore $c_1 = c_3^2$ and $c_2 = c_4^2$ will not be large in practice. We should add that there are choices of the pivot matrices P in (20) which guarantee bounds on c_1 and c_2 that are polynomials in n and m rather than exponential in m as in (23) (Gu and Miranian, 2004). However algorithms that produce such pivot matrices are more expensive than Algorithm 1 and, in practice, usually do not lead to an improvement in accuracy.

Prior to applying one of the methods—SR-N, SR-V and SR-Q—from Sections 3 and 5 one can carry out a partial Cholesky factorization of K to determine the permutation matrix P, and apply the algorithms of Sections 3 and 5 using the matrices $\widetilde{K} \equiv P^TKP$, $\widetilde{K}^* = K^*P$ and the vector $\widetilde{y} = P^Ty$. If pivoting is used in this manner, we will call the algorithms SR-NP, SR-VP and SR-QP corresponding, respectively, to the algorithms SR-N, SR-V and SR-Q without pivoting.

Since the algorithms SR-N, SR-V and SR-Q are all mathematically (in exact arithmetic) equivalent, then by (4) in all these algorithms the low rank approximation to $\widetilde{K}$ is $\widetilde{K}_1\widetilde{K}_{11}^{-1}\widetilde{K}_1^T$ where $\widetilde{K}_1$ is the first m columns of $\widetilde{K}$ and $\widetilde{K}_{11}$ is the first m rows of $\widetilde{K}_1$. Therefore the low rank approximation to $K = P\widetilde{K}P^T$ would be

$$\widehat{K}_P = P\widetilde{K}_1\widetilde{K}_{11}\widetilde{K}_1^T P^T. \tag{30}$$

We then have

Theorem 2 *In exact arithmetic the matrices $\widehat{K}_P$ in (20) and (30) are the same.*
Proof *Let V be the factor produced by a partial Cholesky factorization, with pivoting, of K. Then, as mentioned in Algorithm 1, the first m columns and rows of $P^T KP - VV^T$ are zero. Since $\widetilde{K} = P^T KP$ it follows that $\widetilde{K}_{11} = V_{11}V_{11}^T$ and $\widetilde{K}_1 = VV_{11}^T$, where V_{11} is the $m \times m$ leading principle submatrix of V. Therefore that $VV^T = \widetilde{K}_1\widetilde{K}_{11}^{-1}\widetilde{K}_1^T$. We conclude $PVV^TP^T = P\widetilde{K}_1\widetilde{K}_{11}^{-1}\widetilde{K}_1^TP^T$.* ■

A key conclusion of Theorems 1 and 2 is that for the algorithms SR-NP, SR-VP and SR-QP which use pivoting, the low rank approximation $\widehat{K}_P$ to K has the desirable properties (21-23) which show that the accuracy and condition number of $\widehat{K}_P$ is comparable to the accuracy and condition number of the low rank approximation produced by the singular value decomposition. Therefore if m is small, difficulties such as those illustrated in Example 3 are not possible since for small m the bound $(n-m)4^m$ for c_1 and c_2 is not large. Furthermore, such difficulties are unlikely for large m since, as mentioned earlier, for large m, the values of c_1 and c_2 are, apparently, not large in practice.

For the algorithm SR-VP one does not need to calculate V using (14) since, as shown in the proof of Theorem 2, V is calculated by the partial Cholesky factorization. Using this fact the floating point operation counts of the six algorithms that we have discussed are:

method	no pivoting	pivoting
SR-N / SR-NP	nm^2	$2nm^2$
SR-V / SR-VP	$2nm^2$	$2nm^2$
SR-Q / SR-QP	$2nm^2$	$3nm^2$

Table 3: Approximate flop counts, for n and m large and $n >> m$, for various algorithms.

We should note that flop counts are only rough measures of actual run times since other factors, such as the time for memory access or the degree to which code uses Matlab primitives, can be significant factors. This is discussed further in Section 8.

Also we should note that all the algorithms listed in Table 3 require memory for $O(mn)$ numbers.

Another advantage of the use of pivoting is that if pivoting is included in the V method then for small examples such as Example 2 the potential numerical instability illustrated in Example 2 cannot occur. We illustrate this in the next example. In Appendix A we describe the reason that the SR-VP method is guaranteed to be numerically stable for small problems and why numerical instability is very unlikely for larger real world problems.

Example 4 *This example illustrates that if one includes pivoting in the V method then the numerical instability illustrated in Example 2 does not occur in the V method. As in Example 2 for* $0 < s \leq 1$

let $C = \begin{pmatrix} s^2 & 10s \\ 10s & 200 \end{pmatrix}$, *let the* 4×4 *matrix* $K = \begin{pmatrix} s^2C & 10sC \\ 10sC & 200C \end{pmatrix}$. *Now let* $x = \begin{pmatrix} 1/3 \\ 1/3 \end{pmatrix}$, $w = \begin{pmatrix} 0 \\ x_2 \\ 0 \\ x_1 \end{pmatrix}$, $\lambda = 0$ *let* $y = Kw$.

Due to the structure of w *(and since, in this example, a partial Cholesky factorization will move column 4 of* K *to the first column of* $\widetilde{K} = P^TKP$*) we again have each of* x_{NP}, x_{QP} *and* x_{VP} *will calculate* x *exactly in exact arithmetic. In finite precision arithmetic the calculated values will not be exact. For this example for small* s *the errors in both* x_{VP} *and* x_{QP} *are very small. For example if* $s = 10^{-4}$ *we get the results in Table 4.*

$\dot{x} \quad =$	x_{NP}	x_{VP}	x_{QP}
$\|\|x-\dot{x}\|\|/\|\|x\|\|$	1.7×10^{-1}	2.6×10^{-11}	9.7×10^{-12}

Table 4: Errors $||x-\dot{x}||/||x||$ for a 4×4 matrices and various methods.

Note that even with pivoting the error in the normal equations approach is large. With the normal equations approach the error in the calculated x *includes a term proportional to cond*$^2(K_1)$. *Even with pivoting cond*$^2(K_1)$ *can be large enough so the accuracy of the normal equations approach is poor.*

7. Rank Selection

In using low rank approximation the choice of rank will affect the accuracy of the approximation. It may be impractical to repeat the computations for a variety of different ranks and it is useful to have techniques to facilitate determination of the accuracy of a variety of low rank approximations.

We first consider the case that the true target values y^* corresponding to the testing data X^* are known. Then if $n^* < n$ the accuracy of the prediction for y^* can be calculated efficiently for all low rank approximations with rank less than a specified value m.

To illustrate this we first consider the QR implementation, (12) and (13), of the subset of regressors method. For the $(n+m)\times m$ matrix A in (8) let $A = QR$ where Q is an $(n+m)\times m$ matrix with orthonormal columns and R is an $m\times m$ upper triangular matrix and let $x = R^{-1}Q^Tb$, as in (12) (where we omit the subscript Q on x to simplify our notation). Then by (13) the predicted values of y^* are

$$\widehat{y}^* = K_1^* x$$

where K_1^* is the $n^*\times m$ matrix defined in (3).

Now for some i, $1\le i\le m$, consider the construction of a prediction for y^* using a rank i low rank approximation. Let $\widetilde{A}$ consist of the first i columns of A. It then follows from (9), (13) and the fact that the last $m-i$ rows of b and $\widetilde{A}$ are zero that the rank i prediction, which we call $\widetilde{y}^*$, for y^* is given by solving

$$\min_{\widetilde{x}} ||\widetilde{A}\,\widetilde{x} - b|| \text{ and letting}$$

$$\widetilde{y}^* = K_1^* \begin{pmatrix} \widetilde{x} \\ 0 \end{pmatrix} \tag{31}$$

where $\widetilde{x} \in R^i$ and the 0 in (31) indicates a vector of $m-i$ zeros. Since $A = QR$ it follows that $\widetilde{A} = Q\begin{pmatrix} \widetilde{R} \\ 0 \end{pmatrix}$ where the 0 here indicates $m-i$ rows of zeros. Therefore if $c =$ (the first i elements of $Q^T b$) it then follows (Golub and Van Loan, 1996, p. 239) that we can construct $\widetilde{x}$ using

$$\widetilde{x} = \widetilde{R}^{-1} c. \tag{32}$$

We can use (32) to construct predictions for y^* for every low rank approximation of rank less than or equal to m. To do this we let C be a $m \times m$ upper triangular matrix whose i^{th} column consists of the first i elements of $Q^T b$ and is zero otherwise. Let $\widetilde{Y}$ be the $n^* \times m$ matrix whose i^{th} columns consists of the prediction for y^* using a rank i approximation. Then, for the reasons described in the last paragraph,

$$\widetilde{Y} = K_1^* R^{-1} C. \tag{33}$$

If y^* is known (33) can be used to calculate, for example, the root mean square error of the prediction for y^* for all low rank approximations of rank less than or equal to m.

After the rank m low rank prediction for y^* is constructed, the above calculations require $O(m^3 + n^* m^2)$ floating point operations. If n^* is less than n, this is less than the $O(nm^2)$ operations required to construct the initial rank m prediction. Although we will not present the details here similar efficiencies are possible when using the normal equations approach or the V method.

If the true value y^* for the test set are not known, one can use the subset of regressors approach to estimate the known y values in the training set (by replacing K_1^* with K_1 in (11), (13) or (16)). Again one can calculate the accuracies in estimating y for every low rank approximation of rank less than a given rank m and this can be done relatively efficiently after the initial rank m low rank approximation is constructed. These accuracies will give some indication of the relative difference in using low rank approximations of different ranks.

Finally, we should note that our algorithms provide a limit on the largest rank that can be used. For example in SR-NP, SR-VP and SR-QP Algorithm 1 is used to determine the subset selection. Algorithm 1 returns a rank m where the factorization is stopped and m can be used as the maximum possible rank. For the SR-V and SR-Q algorithms a Cholesky factorization of K_{11} is required in (7). If Matlab's Cholesky routine chol is used for this factorization there is an option to stop the factorization when it is determined that K_{11} is not positive definite (numerically). The size of the factor that successfully factors a positive definite portion of K_{11} sets a limit on the rank that can be effectively used. Finally, SR-N and SR-NP require solving a system of Equations (5) involving the symmetric semidefinite system $\lambda^2 K_{11} + K_1^T K_1$. A good way to solve this system is to use Matlab's chol, which again has an option that can be used to determine a limit on the rank that can be effectively used. As discussed in the next section if these rank limits are exceeded then the calculated answers are often dominated by computer arithmetic errors and are not accurate.

8. Practical Example

In the Sloan Digital Sky Survey (York et al., 2000) broadband u, g, r, i, z photometric measurements will be made for 100s of million galaxies but only approximately 1 million galaxies will have careful spectroscopic measurements of redshifts. Therefore the estimation of redshift from broadband photometric measurements is important since it can lead to much better constraints on the formation and evolution of large-scale structured element in cosmological models (Way and Srivastava, 2006).

We illustrate our earlier remarks by using a training set of 180045 galaxies, each with five measured u, g, r, i, z broadband measurements. The training set consists of a 180045×5 matrix X of broadband measurements and the 180045×1 vector y with the corresponding redshifts. The testing set will consist of a 20229×5 matrix X^* of broadband measurements and the 20229×1 vector y^* of redshifts. This data is from the SDSS GOOD data set discussed in Way and Srivastava (2006).

To determine a good choice for a covariance function we calculated the root mean square (RMS) error for the prediction $\widehat{y}^*$ for y^* using the Matern (with parameter $\nu = 3/2$ and with parameter $\nu = 5/2$), squared exponential, rational quadratic, quadratic and neural network covariance functions from Rasmussen and Williams (2006, Chap. 4). As mentioned earlier we selected the hyperparameters for each covariance function using the Matlab routine minimize from Rasmussen and Williams (2006, pp. 112-116, 221). The covariance function which produced the smallest RMS error for the prediction of y^* was the neural network covariance function (Rasmussen and Williams, 2006, p. 91). For example, for low rank approximations of rank 500 with bootstrap resampling runs (described below) of size 100 the neural network median RMS error was .0204. The next smallest median RMS error was .0212 for the Matern covariance function with $\nu = 3/2$ and the largest median RMS error was .0248 for the quadratic covariance function. Therefore in the experiments below we will use the neural network covariance function.

To compare, experimentally, the efficiency of our implementations of the subset of regressors method we choose a training set size of 90023 (consistent with the bootstrap resampling runs describe below) and low rank approximations of rank $m = 150$ and $m = 1500$. On a computer with 2.2 GH Core Duo Intel processor we timed the SR-N, SR-V, SR-Q, SR-NP, SR-VP and SR-QP methods. On all the calculations in this section that use Algorithm 1 we set the stopping tolerance tol to 0. We ran each of the methods with the additional calculations required to determine the "history" of the accuracy of all low rank approximations less than the specified rank (either 150 or 1500) and also without these extra calculations. The results are summarized in Figure 1.

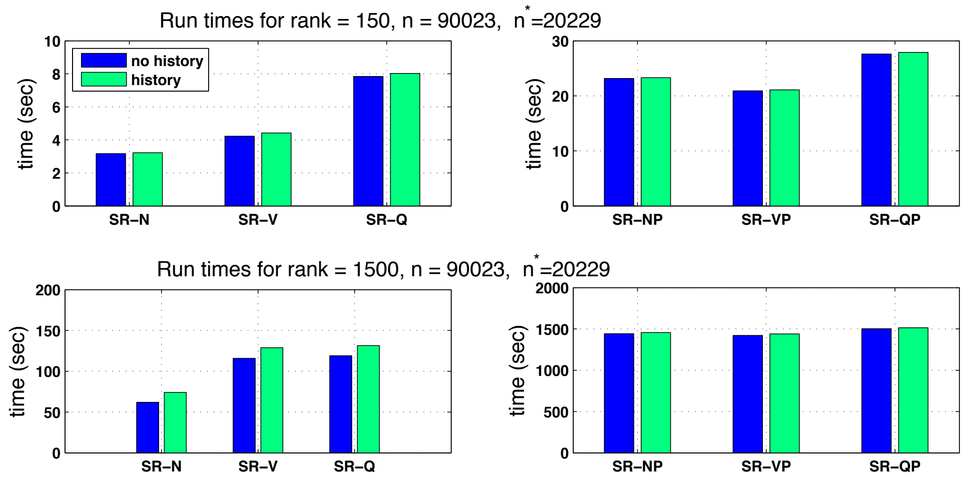


Figure 1: Comparison of run times for implementations of the subset of regressors method.

As can be seen in Figure 1, without pivoting the normal equations approach is the fastest, the QR factorization the slowest and the V method in between. With pivoting all the methods take similar amounts of time (the V method is slightly faster). The reason that all the methods require about the same time when using pivoting is that the code for SR-N, SR-V and SR-Q is written so that the key calculations are done almost entirely with Matlab primitives whereas our implementation of the partial Cholesky factorization contains loops written in Matlab code. The Matlab primitives make use of BLAS-3 (Anderson et al., 1999) routines and will make effective use of cache memory. Therefore, even though the big-O operation counts are similar, the partial Cholesky factorization takes longer to run than SR-N, SR-V or SR-Q and the partial Cholesky factorization dominates the run times in the SR-NP, SR-VP and SR-QP code. We should add that the times for the partial Cholesky factorization would be reduced if a partial Cholesky factorization with pivoting could be implemented using BLAS-3 operations. We are not aware of such an implementation. Finally, we should note that the calculations required to determine the accuracy of all low-order approximations adds only a modest amount to the run times.

To determine the accuracy of the algorithms for different choices of the training set we carried out bootstrap resampling (Efron and Tibshirani, 1993). For each of 100 samples we randomly selected half or 90023 of the 180045 galaxies in the original training set and used this smaller training set to predict the redshift for the 20229 galaxies in the testing set. We considered such resampling with replacement as well as without replacement. For SR-N, SR-V and SR-Q we selected the indices in the active set randomly. Following this we selected the hyperparameters using the minimize routine in Rasmussen and Williams (2006, pp. 112-116, 221). For SR-NP, SR-VP and SR-QP the active set was determined by the partial Cholesky factorization with pivoting. To illustrate the variation in the calculated accuracies, after carrying out a bootstrap resampling run we sorted the 100 RMS errors in increasing order and plotted these errors versus the sample number. The results for low rank approximations of rank 1500, using resampling without replacement, are pictured in Figure 2.

Note that mathematically (in exact arithmetic) SR-N, SR-V and SR-Q will produce identical results; as will SR-NP, SR-VP and SR-QP. Therefore the differences illustrated in Figure 2 between SR-N and SR-V or SR-Q and the differences between SR-NP and SR-VP or SR-QP are due to computer arithmetic and, in particular, the numerical instabilities in using a normal equations approach to solve the least squares problem (9). Also note that although pivoting reduces the numerical instability in using the normal equations approach, still in SR-NP the instability is evident for approximately half of the bootstrap resampling runs. Also we should remark that the $\widehat{y}^*$ predictions calculated using SR-V and SR-Q are essentially identical—they agree to at least seven significant digits in this example—as are the $\widehat{y}^*$ predictions calculated using SR-VP and SR-QP. Finally we should note that for this example the methods that avoid normal equation and use pivoting—SR-VP and SR-QP—are a small amount better than their counterparts, SR-V and SR-Q, that do not use pivoting.

As mentioned earlier, the parameter λ in the Gaussian process computations was selected while optimizing the hyperparameters using the routine minimize from Rasmussen and Williams (2006, pp. 112-116, 221). The values of λ varied over a small range, $.0176 \leq \lambda \leq .0214$, for the 100 samples illustrated in Figure 2. For our stable algorithms these values of λ were good values as can be seen by the accuracy of the results of SR-V, SR-VP, SR-Q and SR-QP pictured in Figures 2, 3 and 4. For SR-N and SR-NP we experimented with a variety of choices of λ but did not reliably achieve accurate predictions for any of our choices.

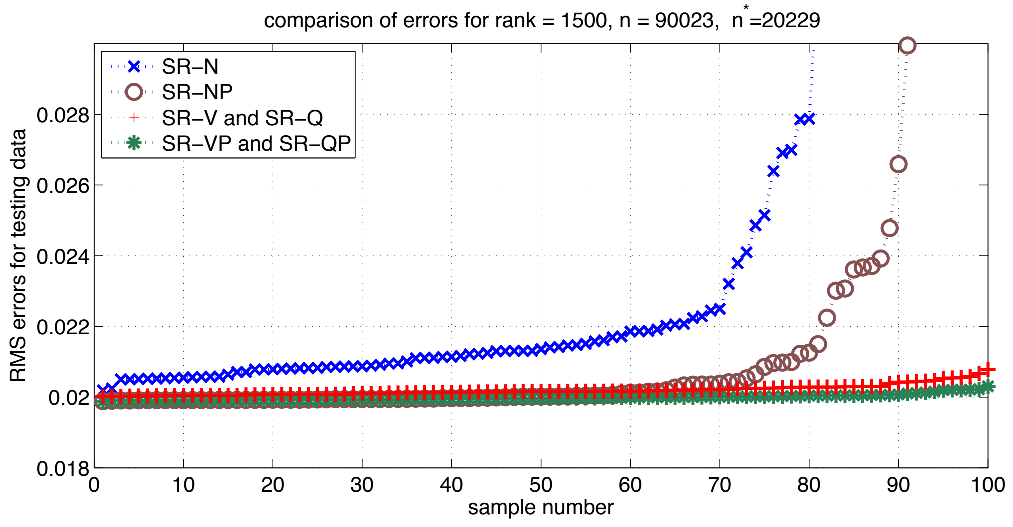


Figure 2: Bootstrap resampling: Comparison of RMS errors for implementations of the subset of regressors method.

We might also add that we tried other types of resampling. We obtained results similar to those illustrated in Figure 2 when using bootstrap resampling with replacement and also when we choose a number of galaxies in the sample size other than 90023.

We can also illustrate the ability to efficiently calculate the accuracy of low rank approximations lower than a specified rank. For the same runs picture in Figure 2 we calculated the mean RMS error of the 100 samples for each rank less than 1500 for each of the six implementations of the subset of regressors method. This is pictured in Figure 3.

As one increases the rank of the low rank approximation the condition number of the matrix A in (9) will tend to increase. This will increase the computer arithmetic errors in the calculated results. The ranks where significant computer arithmetic errors arise are illustrated in Figure 3 by the jumps in the mean errors calculated for the SR-N and SR-NP methods. The ranks where this occurs and the magnitude of the jumps is dependent on the particular data chosen for a bootstrap resampling run and will vary for different bootstrap resampling runs. For the SR-N method the ranks where numerical difficulties were first substantial varied between a rank of 46 to a rank of 839. For the SR-NP method the ranks where numerical difficulties were first substantial varied between ranks of 325 and 1479. For the SR-V, SR-Q, SR-VP and SR-QP methods we did not encounter significant numerical difficulties with these runs and the graphs for these methods smoothly decrease.

The SR-VP and SR-QP method, which use pivoting, are somewhat more accurate than the corresponding methods without pivoting after a rank of approximately 200 but prior to this SR-V and SR-Q are more accurate. Our motivation for subset selection using the Cholesky factorization with pivoting is based on controlling the condition number and improving numerical stability. For smaller ranks it appears that this choice of the active set is good but not optimal. Finally, we should note that Figure 3 indicates for the stable methods the mean RMS errors decrease rapidly for smaller ranks but are only slowly decreasing for larger ranks.

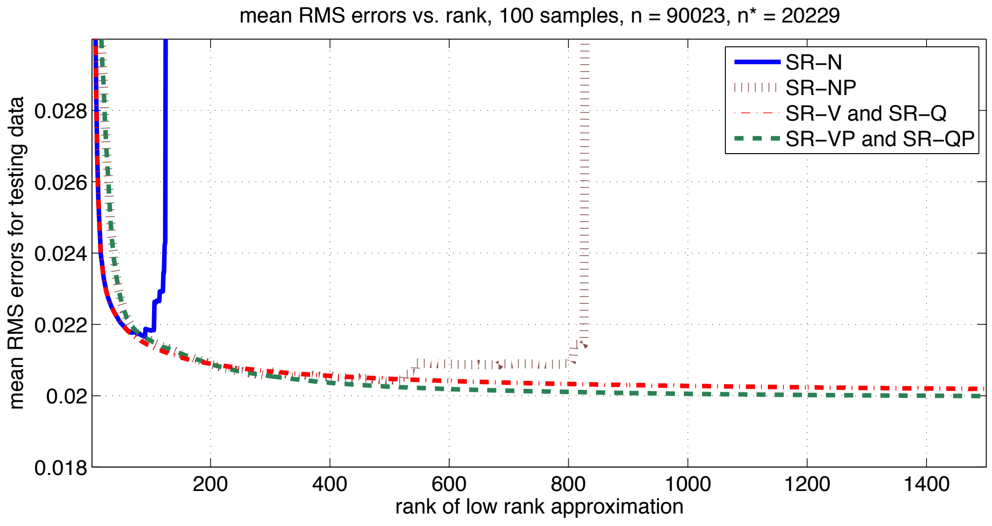


Figure 3: Mean RMS errors versus rank for implementations of the subset of regressors method.

As we mentioned earlier all of our algorithms may limit the rank so that the effective rank can be less than the desired rank. This did not occur on the above runs for SR-V, SR-Q, SR-VP or SR-QP but did occur for SR-N and SR-NP due to our use of the Cholesky factorization to solve the linear system (5). It is possible to solve the linear system in (5) using Gaussian elimination, rather than using a Cholesky factorization, for ranks up to 1500. However the Cholesky factorization in (5) will fail only if the matrix $\lambda^2 K_{11} + K_1^T K_1$ is very ill conditioned. In this case solving the system of equations in (5) by any method will be prone to large computer arithmetic errors. Indeed, for these runs, if we used Gaussian elimination to solve (5) for large ranks the errors became larger than when we limited the rank as we have described earlier. Also when the Cholesky factorization failed in the solution to (5) we tried perturbing K_{11} a small amount following a suggestion in the code provided with Rasmussen and Williams (2006). For our runs this did not improve the calculated results in a significant manner.

In Way and Srivastava (2006) there is a comparison of a variety of methods for predicting red-shift with data from the SLOAN digital sky survey. The methods compared in Way and Srivastava (2006) include linear regression, quadratic regression, artificial neural networks (label ANNz in Figure 4), E-model and Gaussian processes using a quadratic covariance function (labeled GP in Figure 4). In Figure 4 we have compared these methods with our predictions using the SR-VP and SR-QP implementations of the subset of regressors Gaussian processes method with a neural network covariance function. Other than the SR-VP and SR-QP predictions the results in Figure 4 are from Way and Srivastava (2006). As seen in Figure 4, in this example either SR-VP or SR-QP provides overall the best predictions. The E-model approach is also quite good.

We should add that in addition to the data set which was used to generate the results in Figures 1 to 4 we have also carried out experiments using other data sets described in Way and Srivastava (2006) (for example redshift prediction using photometry properties in addition to broadband measurements) and using the SARCOS robot arm inverse dynamics (Rasmussen and Williams, 2006; Vijayakumar et al., 2002). For the other redshift data sets significant computer arithmetic errors in

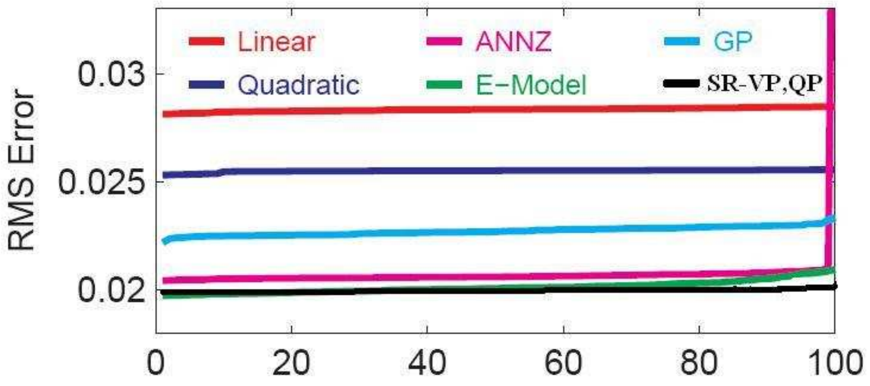


Figure 4: Bootstrap resampling: comparison of RMS errors for six methods of predicting redshift.

the predictions were common for the SR-N and SR-NP algorithms. For some data sets, for example the SARCOS robot arm, computer arithmetic errors were not significant and all the algorithms worked well. Also we might note that although prediction using Gaussian processes was more accurate than alternatives approaches in some cases, in other cases the E-model or artificial neural network approaches provided better accuracy.

Finally, we should note that Matlab code which implements the SR-N, SR-NP, SR-V,SR-VP, SR-Q and SR-QP methods and can produce graphs such as those in Figures 2 and 3 is available at `http://dashlink.arc.nasa.gov/algorithm/stableGP`. Our code makes use of the code from Rasmussen and Williams (2006, p. 221) and the syntax is modeled on that code. We should also note that Foster et al. (2008) and Cayco et al. (2006) discuss additional results related to redshift prediction.

9. Conclusions

An important conclusion of our results is that with the subset of regressors approach to Gaussian process calculations use of normal equations can be unstable and should, in some important practical examples, be avoided. We expect that this principle is also applicable to other approaches to Gaussian process calculations. For example when using approximations based on sparse Gaussian processes with pseudo-inputs (Snelson and Ghahramani, 2006) which is called the FITC approximation in the framework of Quinonero-Candela and Rasmussen (2005) the predicted values are calculated using

$$\widehat{y}^*_{FITC} = K_1^*(\lambda^2 K_{11} + K_1^T(\Lambda+I)^{-1}K_1)^{-1}K_1^T(\Lambda+I)^{-1}y.$$

where

$$\Lambda = diag(K - K_1 K_{11}^{-1} K_1^T)/\lambda^2.$$

Our results suggest that it may be more accurate to carry out these calculations using a QR factorization of $\begin{pmatrix} DK_1 \\ \lambda V_{11}^T \end{pmatrix}$ where $D = (\Lambda+I)^{-1/2}$ rather than, for example, using a Cholesky factorization of $\lambda^2 K_{11} + K_1^T(\Lambda+I)^{-1}K_1$.

To summarize our results, we have presented different implementations of the subset of regressors method for solving, approximately, the Gaussian process equations for prediction. An implementation of the subset of regressors method which uses the normal equations is the fastest approach but also can have poor numerical stability and unacceptable large growth of computer arithmetic errors. An implementation using orthogonal factorization is somewhat slower but in principle has better numerical stability properties. A third approach, which we call the V method, is intermediate between these other two approaches in terms of accuracy and stability. We can use the partial Cholesky factorization to select the active set prior to implementation of any of the above methods. This also will tend to reduce the growth of computer arithmetic errors and can, in some cases, improve the accuracy of the predictions. All of these implementations require $0(nm^2)$ operations where n is the number of data points in the training set and m is the size of the active set or the rank of the low rank approximation used. In this sense all these implementations are efficient and can be much faster than implementation of the full Gaussian process equations. Finally, we have illustrated these result with an important practical application—redshift prediction from broadband spectral measurements. Code implementing our algorithms is available at `http://dashlink.arc.nasa.gov/algorithm/stableGP`.

Acknowledgments

We would like to acknowledge support for this project from the Woodward Fund, Department of Mathematics, San Jose State University.

M.J.W. acknowledges funding received from the NASA Applied Information Systems Research Program and from the NASA Ames Research Center Director's Discretionary Fund. M.J.W also acknowledges Alex Szalay, Ani Thakar, Maria SanSebastien and especially Jim Gray for their help with the Sloan Digital Sky Survey.

A. N. Srivastava wishes to thank the NASA Aviation Safety Program, Integrated Vehicle Health Management Project for supporting this work.

Funding for the SDSS has been provided by the Alfred P. Sloan Foundation, the Participating Institutions, the National Aeronautics and Space Administration, the National Science Foundation, the U.S. Department of Energy, the Japanese Monbukagakusho, and the Max Planck Society. The SDSS Web site is `http://www.sdss.org/`.

The SDSS is managed by the Astrophysical Research Consortium for the Participating Institutions. The Participating Institutions are The University of Chicago, Fermilab, the Institute for Advanced Study, the Japan Participation Group, The Johns Hopkins University, Los Alamos National Laboratory, the Max-Planck-Institute for Astronomy, the Max-Planck-Institute for Astrophysics, New Mexico State University, University of Pittsburgh, Princeton University, the United States Naval Observatory, and the University of Washington.

Finally, we wish to thank the anonymous referees for carefully reading the manuscript and offering excellent suggestions.

Appendix A. Numerical Stability of SR-VP

Here we explain why, even though there is a potential numerical instability in SR-V, as illustrated in Example 2, this difficulty cannot occur with the SR-VP method for small problems and is very unlikely to occur for larger problems from real world applications.

Let P be the $n \times n$ permutation matrix determined by the partial Cholesky factorization with pivoting applied to K, let $\widetilde{K} = P^T K P$ and let $\widetilde{K}_1$ be the first m columns of $\widetilde{K}$. In the SR-VP method we apply Equations (14)-(16) to $\widetilde{K}$ and $\widetilde{K}_1$ rather than K and K_1.

We will begin by considering the special case where $\lambda = 0$ and later consider the more general case. In the case that $\lambda = 0$ the least square problem (9), with K_1 replaced by $\widetilde{K}_1$ since we are incorporating pivoting, is equivalent to

$$\min_x ||\widetilde{K}_1 x - y||.$$

and, by (15), we have

$$x = V_{11}^{-T}(V^T V)^{-1} V^T y. \tag{34}$$

where $\widetilde{K}_1 = V V_{11}^T$. There is a potential concern in using (34) since to construct x the linear system of equations

$$(V^T V)z = V^T y$$

must be solved. Forming $V^T V$ squares the condition number of V which, potentially could lead to the introduction of undesirable computer arithmetic errors. However we will argue that the matrix $B = V^T V$ is diagonally equivalent to a matrix that is guaranteed to be well conditioned for small problems and, in practice, is almost always well conditioned for larger problems. This will limit the growth of computer arithmetic errors. We should add that without pivoting one cannot prove such results, as is illustrated by Example 2.

Now V is formed by a partial Cholesky factorization with pivoting of the symmetric positive semidefinite matrix K. Since pivoting is included in the partial Cholesky factorization of the SPS matrix it follows, for each $i = 1, \ldots, m$, that the i^{th} diagonal entry of $\widetilde{K}_1$ is at least as large in magnitude as any off diagonal entry in row i or column i of $\widetilde{K}_1$ (Trefethen and Bau III, 1997, p. 176) and that the lower trapezoidal matrix V has the property that, for each $i = 1, \ldots, m$, the i^{th} diagonal entry in V is at least as large in magnitude as any entry in column i (Higham, 2002, p. 202). Therefore we can write V as $V = LD$ where D is an $m \times m$ diagonal matrix and L is an $n \times m$ lower trapezoidal matrix with all entries one or less in magnitude and with ones on the diagonal. Indeed this matrix L is identical to the lower trapezoidal matrix produced if Gaussian elimination with complete pivoting is applied to $\widetilde{K}_1$ (Higham, 2002, p. 202). Also since the pivoting has already been applied in forming $\widetilde{K}_1$ Gaussian elimination with complete pivoting will not pivot any entries in $\widetilde{K}_1$ and this implies that Gaussian with partial pivoting will not pivot any entries in $\widetilde{K}_1$ and will produce the same lower trapezoidal factor L. Now it follows from Higham (2002, p. 148) that

$$cond(L) \leq \sqrt{nm}\, 2^{m-1}$$

and therefore for n and m small, as in Example 2, L is well conditioned. More generally, according to Björck (1996, p. 73), if partial pivoting is used in the factorization of $\widetilde{K}_1$ then L is usually well conditioned and, indeed, the discussion in Trefethen and Bau III (1997, p. 169) indicates that, for matrices from applications and for random matrices $\widetilde{K}_1$, the matrix L is almost always well conditioned, in the sense, for example, that cond(L) is far from being exponentially large.

Thus V is a diagonal rescaling of a matrix L that is well conditioned in practice. Now define $U = DV_{11}^T$. It then follows from (34) that

$$x = U^{-1}(L^T L)^{-1} L^T y. \tag{35}$$

Equation (35) is precisely the Peters-Wilkinson method (Peters and Wilkinson, 1970 and Björck, 1996, p. 73) to the least square problem (34). Since L is usually well conditioned then the calculation of $(L^T L)^{-1} L^T y$ can be computed without substantial loss of accuracy and the calculation of x using (35) is more stable than using the normal equation solution to (34) (Björck, 1996, p. 73).

The SR-VP method uses (34) rather that (35). However, since V is a diagonal rescaling of L and U is a diagonal rescaling of V_{11}^T the SR-VP method will also have good numerical stability properties in practice. To demonstrate this we can write $V = LD_1D_2$ where the entries of the diagonal matrix D_1 are between 1 and 2 and where entries in D_2 are exact powers of 2. Since L will be well conditioned in practice then so is $W = LD_1$ (since $\text{cond}(LD_1) \leq \text{cond}(L)\text{cond}(D_1) \leq 2\,\text{cond}(L)$). Now, by (34), we have

$$x = (D_2V_{11})^{-T}(W^T W)^{-1} W^T y. \tag{36}$$

Since W is well conditioned in practice it follows, for the same reasons that (35) has good numerical stability, that (36) will have good numerical stability properties.

To finish the analysis of numerical stability of the SR-VP method in the case that $\lambda = 0$ note that since D_2 has entries that are exact powers of 2, it follows by the discussion in Higham (2002, p. 200) and Forsythe and Moler (1967, 37-39), for any computer using base 2 computer arithmetic, that the x calculated by (36) will be precisely the same, even in floating point arithmetic (as long as there is no overflow or underflow), as the x calculated by (34). Therefore we may conclude that in practice x calculated when using the SR-VP method will have good numerical stability properties and the SR-VP method will usually have smaller computer arithmetic errors than will the SR-N or SR-NP methods.

To consider the case that $\lambda \neq 0$ we note that in this case the condition number of $B = (\lambda^2 I + V^T V)$ will be important in solving

$$(\lambda^2 I + V^T V)z = V^T y.$$

However we have

Theorem 3 *For any* $\lambda \geq 0$, *cond*$(\lambda^2 I + V^T V) \leq$ *cond*$(V^T V)$.

Proof If $V^T V$ has eigenvalues $\alpha_1 \geq \alpha_2 \geq \ldots \geq \alpha_m \geq 0$ then the eigenvalues of $(\lambda^2 I + V^T V)$ are $(\lambda^2 + \alpha_i)$, $i = 1, \ldots, m$. Therefore *cond*$(V^T V) = \alpha_1/\alpha_m$ and *cond*$(\lambda^2 I + V^T V) = (\alpha_1 + \lambda^2)/(\alpha_m + \lambda^2)$. However it follows easily that $\alpha_1/\alpha_m \geq (\alpha_1 + \lambda^2)/(\alpha_m + \lambda^2)$. ■

Since *cond*$(\lambda^2 I + V^T V) \leq$ *cond*$(V^T V)$ we expect that solving $(\lambda^2 I + V^T V)z = V^T y$ with $\lambda \neq 0$ will be more accurate than solving this equation with $\lambda = 0$. Since we have argued that the error growth in solving this equation for $\lambda = 0$ should be limited we expect that this should also be true when $\lambda \neq 0$.

References

Ed Anderson, Zhaojun Bai, Christian H. Bischof, Susan Blackford, James W. Demmel, Jack J. Dongarra, Jeremy J. Du Croz, Anne Greenbaum, Sven J. Hammarling, Alan McKenney, and Danny C. Sorensen. *LAPACK Users' Guide*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, third edition, 1999. ISBN 0-89871-447-8.

Åke Björck. *Numerical Methods for Least Squares Problems*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1996. ISBN 0-89871-360-9.

Bem Cayco, Wasin So, Miranda Braselton, Kelley Cartwright, Michael Hurley, Maheen Khan, Miguel Rodriguez, David Shao, Jason Smith, Jimmy Ying, and Genti Zaimi. Camcos project – Fall 2006: Improved linear algebra methods for redshift computation from limited spectrum data. At www.math.sjsu.edu/˜foster/camcos07/redshift.html, 2006.

Lehel Csato and Manfred Opper. Sparse on-line gaussian processes. *Neural Computation*, 14: 641–668, 2002.

Jack J. Dongarra, James R. Bunch, Cleve B. Moler, and G. W. Stewart. *LINPACK Users' Guide*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1979. ISBN 0-89871-172-X.

Bradley Efron and Robert J. Tibshirani. *An Introduction to the Bootsrap*. Chapman and Hall, New York, 1993.

Shai Fine and Katya Scheinberg. Efficient svm training using low-rank kernel representations. *J. of Machine Learning Research*, 2:243–264, 2001.

George E. Forsythe and Cleve B. Moler. *Computer Solution of Linear Algebraic Systems*. Prentice-Hall, Englewood Cliffs, NJ, USA, 1967.

Leslie Foster, Alex Waagen, Nabeela Aijaz, Michael Hurley, Apolo Luis, Joel Rinsky, Chandrika Satyavolu, Ashok Srivastava, Paul Gazis, and Michael Way. Improved linear algebra methods for redshift computation from limited spectrum data - II. NASA Technical Report NASA/TM-2008-214571, NASA Ames Research Center, Moffett Field, CA, 2008. Available at ntrs.nasa.gov and at www.math.sjsu.edu/˜foster/camcos07/redshift.html.

Gene H. Golub and Charles F. Van Loan. *Matrix Computations*. Johns Hopkins University Press, Baltimore, MD, USA, third edition, 1996. ISBN 0-8018-5413-X, 0-8018-5414-8.

Ming Gu and Stanley C. Eisenstat. Efficient algorithm for computing a strong rank-revealing qr factorization. *SIAM J. Sci. Comput.*, 17(4):848–869, 1996.

Ming Gu and Luiza Miranian. Strong rank-revealing Cholesky factorization. *Electronic Transactions on Numerical Analysis*, 17:76–92, 2004.

Per Christian Hansen. *Rank-Deficient and Discrete Ill-Posed Problems*. SIAM, Philadelphia, PA, USA, 1998.

Nicholas J. Higham. Analysis of the Cholesky decomposition of a semi-definite matrix. In M. G. Cox and S. J. Hammarling, editors, *Reliable Numerical Computation*, pages 161–185. Oxford University Press, 1990.

Nicholas J. Higham. *Accuracy and Stability of Numerical Algorithms*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, second edition, 2002. ISBN 0-89871-521-0.

Craig Lucas. LAPACK-style codes for level 2 and 3 pivoted cholesky factorizations. Numerical Analysis Report No. 442, Manchester Centre for Computational Mathematics, Manchester, England, 2004. LAPACK Working Note 161.

Douglas C. Montgomery, Elizabeth A. Peck, and G. Geoffrey Vining. *Introduction to Linear Regression Analysis*. John Wiley and Sons, Hoboken, NJ, USA, fourth edition, 2006.

Gwen Peters and James H. Wilkinson. The least squares problem and pseudo-inverses. *Comput. J.*, 13(3):309–316, 1970.

Tomaso Poggio and Fedirico Girosi. Networks for approximation and learning. *Proceedings of IEEE*, 78:1481–1497, 1990.

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

Carl E. Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning*. MIT Press, Cambridge, Massachusetts, 2006.

Matthias Seeger, Christopher Williams, and Neil D. Lawrence. Fast forward selection to speed up sparse gaussian process regression. In C. M. Bishop and B. J. Frey, editors, *Proceedings of the Ninth International Workshop on Artificial Intelligence and Statistics*, San Francisco, 2003. Morgan Kaufmann.

Alex J. Smola and Peter Bartlett. Sparse greedy gaussian process regression. In T. Leen, T. Diettrich, and V. Tresp, editors, *Advances in Neural Information Processing Systems 13*, pages 619–625. MIT Press, 2001.

Edward Snelson and Zoubin Ghahramani. Sparse gaussian process using pseudo-inputs. In Y. Weiss, B. Scholkpf, and J. Platt, editors, *Advances in Neural Information Processing Systems 18*, pages 1257–1264. MIT Press, 2006.

G. W. Stewart. The efficient generation of random orthogonal matrices with an application to condition estimators. *SIAM J. Numer. Anal.*, 17(3):403–409, 1980.

Lloyd N. Trefethen and David Bau III. *Numerical Linear Algebra*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1997. ISBN 0-89871-361-7.

Sethu Vijayakumar, Aaron DSouza, Tomohiro Shibata, Jorg Conradt, and Stefan Schaal. Statistical learning for humanoid robots. *Autonomous Robot*, 12:55–69, 2002.

Grace Wahba. *Spline Models for Observation Data*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1990.

Michael J. Way and Ashok Srivastava. Novel methods for predicting photometric redshifts from broadband data using virtual sensors. *The Astrophysical Journal*, 647:102–115, 2006.

Donald G. York, J. Adelman, and John E. Anderson et. al. The sloan digital sky survey: Technical summary. *The Astronomical Journal*, 120:1579–1587, 2000.

Journal of Machine Learning Research 10 (2009) 883-906 Submitted 4/08; Revised 1/09; Published 4/09

Estimation of Sparse Binary Pairwise Markov Networks using Pseudo-likelihoods

Holger Höfling HHOEFLIN@GMAIL.COM
Department of Statistics
Stanford University
Stanford, CA 94305, USA

Robert Tibshirani TIBS@STANFORD.EDU
Depts. of Health, Research & Policy and Statistics
Stanford University
Stanford, CA 94305, USA

Editor: Michael Jordan

Abstract

We consider the problems of estimating the parameters as well as the structure of binary-valued Markov networks. For maximizing the penalized log-likelihood, we implement an approximate procedure based on the pseudo-likelihood of Besag (1975) and generalize it to a fast exact algorithm. The exact algorithm starts with the pseudo-likelihood solution and then adjusts the pseudo-likelihood criterion so that each additional iterations moves it closer to the exact solution. Our results show that this procedure is faster than the competing exact method proposed by Lee, Ganapathi, and Koller (2006a). However, we also find that the approximate pseudo-likelihood as well as the approaches of Wainwright et al. (2006), when implemented using the coordinate descent procedure of Friedman, Hastie, and Tibshirani (2008b), are much faster than the exact methods, and only slightly less accurate.

Keywords: Markov networks, logistic regression, L_1 penalty, model selection, Binary variables

1. Introduction

In recent years a number of authors have proposed the estimation of sparse undirected graphical models for continuous as well as discrete data through the use of L_1 (lasso) regularization. For continuous data, one assumes that the observations have a multivariate Gaussian distribution with mean μ and covariance matrix Σ. Then an L_1 penalty is applied to $\Theta = \Sigma^{-1}$. That is, the penalized log-likelihood $\ell(\Theta) - \rho||\Theta||_1$ is maximized, where ℓ is the Gaussian log-likelihood, $||\Theta||_1$ is the sum of the absolute values of the elements of Θ and $\rho \geq 0$ is a user-defined tuning parameter. Several papers proposing estimation procedures for this Gaussian model have been published. Meinshausen and Bühlmann (2006) develop a lasso based method for estimating the graph structure and give theoretical consistency results. Yuan and Lin (2007), Banerjee et al. (2008) and Dahl et al. (2008) as well as Friedman, Hastie, and Tibshirani (2008a) propose algorithms for solving this penalized log-likelihood with the procedure in Friedman, Hastie, and Tibshirani (2008a), the *graphical lasso*, being especially fast and efficient.

Here we focus on estimation of networks of discrete, more specifically binary-valued, units with pairwise interactions. These are a special class of Markov networks. The use of L_1 penalties for this

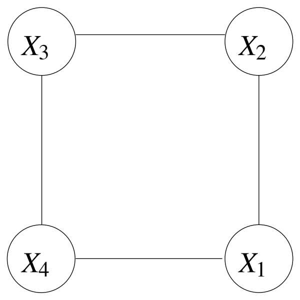


Figure 1: A simple graph for illustration.

x_1	x_2	x_3	x_4
1	1	1	0
1	1	0	1
0	0	1	0
1	1	0	1
1	1	1	0
1	1	0	1
0	0	1	0
1	1	0	1
0	1	1	0
1	1	0	1

Table 1: Sample data for graph of Figure 1

special class as well as more general Markov networks was proposed by Lee, Ganapathi, and Koller (2006a). This problem is more difficult than the continuous Gaussian version because of the need to compute the first and second moments under the model, which are derivatives of the log-partition function. Figure 1 shows an example, and Table 1 shows some sample data from this graph. Given this data and a model for binary graphical data (detailed later), we would like to a) infer a structure something like that of Figure 1, and b) estimate the link parameters itself. For the data in Table 1, Figure 2 shows the path of solutions for varying penalty parameter. Most edges for L_1-norm ≤ 2 are correctly identified as in the graph in Figure 1. However, edge $(1,3)$ is absent in the true model but included in the L_1 penalized model relatively early.

Our main focus in this paper is to develop and implement fast approximate and exact procedures for solving this class of L_1-penalized binary pairwise Markov networks and compare the accuracy and speed of these to other methods proposed by Lee, Ganapathi, and Koller (2006a) and Wainwright et al. (2006). Here, by "exact" procedures we refer to algorithms that find the exact maximizer of the penalized log-likelihood of the model whereas "approximate" procedures only find an approximate solution.

In Section 2 we describe the Ising model as well as the details of the competing methods of Lee, Ganapathi, and Koller (2006a) and Wainwright et al. (2006). Section 3 then introduces the basic pseudo-likelihood model and outlines the computational approach to increase the speed of the algorithm. The pseudo-likelihood is a very interesting and fast approximate method which has

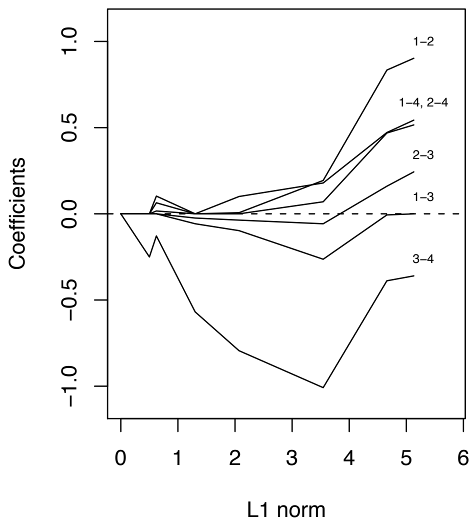


Figure 2: Toy example: profiles of estimated edge parameters as the penalty parameter is varied.

the added advantage that it can be used as a building block to a new algorithm for maximizing the penalized log-likelihood exactly. The adjustments necessary to achieve this are described in Section 4. Finally, Section 5 discusses the results of the simulations with respect to speed and accuracy of the competing algorithms.

2. The Model and Competing Methods

In this section we briefly outline the competing methods for maximizing the penalized log-likelihood. Apart from the method proposed in Lee, Ganapathi, and Koller (2006a), which we already mentioned above, we also discuss a very simple solution that was presented in Wainwright et al. (2006). We first describe the underlying model in more detail. Consider data generated under the Ising model

$$p(\mathbf{x},\Theta) = \exp\left[\sum_{s\in V}\theta_s x_s + \sum_{(s,t)\in E}\theta_{st}x_s x_t - \Psi(\Theta)\right].$$

for a single observation $\mathbf{x} = (x_1,\dots,x_p)^T \in \{0,1\}^p$ and model parameters θ_s and θ_{st} for $s,t \in V = \{1,\dots,p\}$. Here, V denotes the vertices and E the edges of a graph. $\Psi(\Theta)$ is the normalization constant, which is also known as the log-partition function. By setting $\theta_{st} = 0$ for $(s,t) \notin E$, and using the fact that $\mathbf{x}$ is a binary vector we can write the log-likelihood as

$$l(\mathbf{x},\Theta) = \log p(\mathbf{x},\Theta) = \sum_{s\geq t\geq 1}^{p}\theta_{st}x_s x_t - \Psi(\Theta)$$

where Θ is a symmetric $p \times p$ matrix with $\theta_{ss} = \theta_s$ for $s = 1, \dots, p$. Note that for notational convenience, we do not distinguish between θ_{st} and θ_{ts} and therefore we enforce symmetry of Θ although the log-likelihood only uses the lower-triangular matrix of Θ. For the L_1-penalty let $\mathbf{R}$ be a $p \times p$ lower triangular matrix of penalty parameters. The penalized log-likelihood for all N observations is

$$\sum_{s \geq t \geq 1}^{p} (\mathbf{X}^T\mathbf{X})_{st}\theta_{st} - N\Psi(\Theta) - N||\mathbf{R} * \Theta||_1$$

where $*$ denotes component-wise multiplication.

The algorithms that we will discuss in the following sections could be generalized to more general categorical variables or higher order interaction terms than those included in the Ising model. However, as we will see, solving these problems exactly is already computationally demanding in the pairwise binary case so that we chose not to adapt the algorithms to these more general settings.

2.1 The Lee, Ganapathi, and Koller (2006a) Method

The penalized log-likelihood is a concave function, so standard convex programming techniques can be used to maximize it. The main difficulty is that the log-partition function $\Psi(\Theta)$ is the sum over 2^p elements, and therefore it is computationally expensive to calculate Ψ or its derivatives in general. However, for sparse matrices Θ, algorithms such as the junction tree algorithm exist that can calculate Ψ and its derivatives efficiently. Therefore, it is especially important to maintain sparsity of Θ for any optimization method. Lee, Ganapathi, and Koller (2006a) achieve this by optimizing the penalized log-likelihood only over a set F of *active variables* which they gradually enlarge until an optimality criterion is satisfied.

To be more precise, they start out with a set of active variables $F = F_0$ (e.g., the diagonal of Θ if it is unpenalized). Using either conjugate gradients or BFGS, the penalized log-likelihood is maximized over the set of variables F. Then one of the currently inactive variables is selected by the *grafting* procedure (see Perkins et al., 2003) and added to the set F. These steps are repeated until grafting does not add any more features. The algorithm can be used for more general Markov networks, but for ease of implementation they choose to work with binary random variables and we do the same as well.

Their procedure provides an exact solution to the problem when the junction tree algorithm is used to calculate $\Psi(\Theta)$ and its derivatives. In their implementation, however, they used loopy belief propagation, which is faster on denser matrices, but only provides approximate results. In their method as well as ours, any procedure to evaluate $\Psi(\Theta)$ can be "plugged in" without any further changes to the rest of the algorithm; we decided to evaluate the speed and performance of only the exact algorithms. The relative performance of an approximate method using loopy belief propagation would likely be similar. They also provide a proof that under certain assumptions and using the L_1 regularized log-likelihood, it is possible to recover the true expected log-likelihood up to an error ε.

2.2 The Wainwright et al. (2006) Method

Wainwright et al. (2006) propose estimation of the Markov network by applying a separate L_1-penalized logistic regression to each of the p variables on the remaining variables. For every $s \in V$ regress x_s onto $\mathbf{x}_{\setminus s} = (x_1, \dots, x_{s-1}, x_{s+1}, \dots, x_p)^T$. Let the $p \times p$ matrix $\tilde{\Theta}$ denote the estimate of Θ.

Then set $\tilde{\theta}_{ss} = \beta_0$, the intercept of the logistic regression, and $\tilde{\theta}_{st} = \beta_t$, where β_t is the coefficient associated with x_t in the regression.

In the outline above, we assumed that Θ is a symmetric matrix. However, due to the way it was constructed, $\tilde{\Theta}$ is not necessarily symmetric. We investigate two methods for symmetrizing $\tilde{\Theta}$. The first way is to define Θ as

$$\theta_{st} = \theta_{ts} = \begin{cases} \tilde{\theta}_{st} & \text{if } |\tilde{\theta}_{st}| > |\tilde{\theta}_{ts}| \\ \tilde{\theta}_{ts} & \text{if } |\tilde{\theta}_{st}| \leq |\tilde{\theta}_{ts}| \end{cases}$$

which we call “Wainwright-max”. Similarly, “Wainwright-min” is defined by

$$\theta_{st} = \theta_{ts} = \begin{cases} \tilde{\theta}_{st} & \text{if } |\tilde{\theta}_{st}| < |\tilde{\theta}_{ts}| \\ \tilde{\theta}_{ts} & \text{if } |\tilde{\theta}_{st}| \geq |\tilde{\theta}_{ts}| \end{cases}.$$

Wainwright et al. (2006) mainly intended their method to be used in order to estimate the presence or absence of an edge in the underlying graph of the model. They show that under certain assumptions, their method correctly identifies the non-zero edges in a Markov graph, as $N \to \infty$ even for increasing number of parameters p or neighborhood sizes of the graph d, as long as N grows more quickly than $d^3 \log p$ (see Wainwright et al., 2008). Due to the simplicity of the method it is obvious that it could also be used for parameter estimation itself and here we will compare its performance in these cases to the pseudo-likelihood approach proposed below and the exact solution of the penalized log-likelihood. Furthermore, as an important part of this article is the comparison of the speeds of the underlying algorithms, we implement their method, using the fast coordinate descent algorithm for logistic regression with a lasso penalty (see Friedman, Hastie, and Tibshirani, 2008b).

3. Pseudo-likelihood Model

In this section we first introduce an approximate method to infer the structure of the graph that is based on pseudo-likelihoods (Besag, 1975). As we will see in the simulations section, the results are very close to the exact solution of the penalized log-likelihood. In the next section, we use the pseudo-likelihood model to design a very fast algorithm for finding an exact solution for the penalized Ising model.

The main computational problem in the Ising model is the complexity of the partition function. One possibility in this case is to solve an approximate version of the likelihood instead. Approaches of this kind have been proposed in various papers in the statistical literature before, for example the pseudo-likelihood approach of Besag (Besag, 1975) and the treatments of composite likelihoods in Lindsay (1998) and Cox and Reid (2004) among others. Here, we want to apply the approximation proposed in Besag (1975) to our problem. This approach is also related to the method of Wainwright et al. (2006), however instead of performing separate logistic regressions for every column of the parameter matrix Θ, the pseudo-likelihood approach here allows us to estimate all parameters at the same time. This way, the resulting matrix Θ is symmetric and no additional step like the max or min-rule described above is necessary. The log-pseudo-likelihood is then given by

$$\tilde{l}(\Theta|\mathbf{x}) = \sum_{s=1}^{p} \log p(x_s, \Theta|\mathbf{x}_{\setminus s})$$

with

$$\log p(x_s, \Theta|\mathbf{x}_{\setminus s}) = x_i(\theta_{ss} + \sum_{t \neq s} x_t \theta_{st}) - \Psi_s(\mathbf{x}, \Theta).$$

Here, $\Psi_s(\mathbf{x},\Theta)$ is the log-normalization constant when conditioning x_s on the other variables, which is exactly the same as in logistic regression with a logit link-function, that is,

$$\Psi_s(\mathbf{x},\Theta) = \log(1+\exp(\theta_{ss}+\sum_{t\neq s} x_t\theta_{st}))$$

where as above for notational convenience we set $\theta_{st} = \theta_{ts}$ for $t \neq s$. Putting all this together, the pseudo-likelihood for a single observation $\mathbf{x}$ is given by

$$\tilde{l}(\Theta|\mathbf{x}) = \sum_{s=1}^{p}\sum_{t=1}^{p} x_s x_t \theta_{st} - \sum_{s=1}^{p} \Psi_s(\mathbf{x},\Theta).$$

In the usual way, the pseudo-likelihood for all N observations is given by the sum of the pseudo-likelihood of the individual observations

$$\tilde{l}(\Theta|\mathbf{X}) = \sum_{k=1}^{N} \tilde{l}(\Theta|x_k)$$

where $\mathbf{x}_k$ is the kth row of matrix with observations $X \in \mathbb{R}^{N\times p}$.

As this is just a sum of logistic likelihoods, the pseudo-likelihood is a concave function and therefore the L_1-penalized pseudo-likelihood

$$\sum_{k=1}^{N} \tilde{l}(\Theta|\mathbf{x}_k) - N||\mathbf{S}*\Theta||_1$$

is concave as well. Here $\mathbf{S} = 2\mathbf{R} - diag(\mathbf{R})$ and is chosen to be roughly equivalent to the penalty terms in the penalized log-likelihood. The penalty term is doubled on the off-diagonal, as the derivative of the pseudo-likelihood on the off-diagonal is roughly twice as large as the derivative of the log-likelihood (see Equation 1).

3.1 Basic Optimization Algorithm

Due to its simple structure, a wide range of standard convex programming techniques can be used to solve this problem, although the non-differentiability of the L_1 penalty poses a problem. Here, we want to use a local quadratic approximation to the pseudo-likelihood. As the number of variables is $p(p+1)/2$, the Hessian could get very large, we restrict our quadratic approximation to have a diagonal Hessian.

In order to construct the approximation, we need the first and second derivative of $\tilde{l}$ w.r.t θ_{st}. These are

$$\frac{\partial\tilde{l}}{\partial\theta_{st}} = 2(\mathbf{X}^T\mathbf{X})_{st} - (\hat{\mathbf{p}}_s^T\mathbf{X})_t - (\hat{\mathbf{p}}_t^T\mathbf{X})_s \quad s\neq t, \tag{1}$$

$$\frac{\partial\tilde{l}}{\partial\theta_{ss}} = (\mathbf{X}^T\mathbf{X})_{ss} - \sum_{k=1}^{N} \hat{p}_{sk}$$

where $1-\hat{p}_{sk} = 1/(1+\exp(\theta_{ss}+\sum_{s\neq t} X_{kt}\theta_{st}))$. The second derivatives are

$$\frac{\partial^2\tilde{l}}{(\partial\theta_{st})^2} = -(\mathbf{X}^T\text{diag}(\hat{\mathbf{p}}_s)\text{diag}(1-\hat{\mathbf{p}}_s)\mathbf{X})_{tt} - (\mathbf{X}^T\text{diag}(\hat{\mathbf{p}}_t)\text{diag}(1-\hat{\mathbf{p}}_t)\mathbf{X})_{ss} \quad s\neq t,$$

$$\frac{\partial^2\tilde{l}}{(\partial\theta_{ss})^2} = -\sum_{k=1}^{N} \hat{p}_{sk}(1-\hat{p}_{sk}).$$

Algorithm 1: Estimation for L_1 penalized pseudo-likelihood

```
if $\Theta^{(0)}$ not given then
    Set $\Theta^{(0)} = \text{diag}(\text{logit}(\hat{\mathbf{p}}^{(0)}))$ where $\hat{p}_s^{(0)} = \frac{1}{N}\sum_{k=1}^{N} x_{ks}$
end
Set k:=0;
while not converged do
    With current estimate $\Theta^{(k)}$, define local approximation $f_{\Theta^{(k)}}(\Theta)$ to $\tilde{l} - N||\mathbf{S} * \Theta||_1$;
    Find solution $\Theta^*$ of $f_{\Theta^{(k)}}(\Theta)$;
    Perform backtracking line search on the line from $\Theta^{(k)}$ to $\Theta^*$ to find $\Theta^{(k+1)}$;
    Set k:=k+1
end
```

Assume that at the k-th step the parameter estimate is $\Theta^{(k)}$. Then define the local approximation to $\tilde{l}(\Theta|\mathbf{X}) - N||\mathbf{S} * \Theta||_1$ as

$$f_{\Theta^{(k)}}(\Theta) = C + \sum_{s \geq t} \frac{\partial \tilde{l}}{\partial \theta_{st}}(\theta_{st} - \theta_{st}^{(k)}) + \frac{1}{2}\frac{\partial^2 \tilde{l}}{(\partial \theta_{st})^2}(\theta_{st} - \theta_{st}^{(k)})^2 - N||\mathbf{S} * \Theta||_1$$

where C is some constant. As stated above, this is just a quadratic approximation with linear term equal to the gradient of $\tilde{l}$ and a diagonal Hessian with diagonal elements equal to the diagonal of the Hessian of $\tilde{l}$. The main reasons for using this simple structure are that it keeps the computation complexity per iteration low and it is very easy to solve this L_1 penalized local approximation. Let $\tilde{\Theta}$ be the minimizer of the unpenalized $f_{\Theta^{(k)}}(\Theta)$, then

$$\tilde{\theta}_{st} = \theta_{st}^{(k)} - \frac{\frac{\partial \tilde{l}}{\partial \theta_{st}}}{\frac{\partial^2 \tilde{l}}{(\partial \theta_{st})^2}}.$$

As the Hessian is diagonal, the L_1-penalized solution Θ^* of $f_{\Theta^{(k)}}(\Theta)$ can be obtained by soft thresholding as

$$\theta_{st}^* = \text{sign}(\tilde{\theta}_{st})\left(\tilde{\theta}_{st} - s_{st}/\frac{\partial^2 \tilde{l}}{(\partial \theta_{st})^2}\right)_+.$$

Using Θ^*, the next step $\Theta^{(k+1)}$ can now be obtained by, for example, a backtracking line search. The whole algorithm can be seen in Algorithm 1 and a proof of convergence that closely follows Lee, Lee, Abbeel, and Ng (2006b) is given in the appendix.

3.2 Speed Improvements

In practice, there are several things that can be done to speed up the algorithm given above. First of all, as Θ will in general be sparse, all computations to calculate $\hat{\mathbf{p}}$ should exploit this sparse structure. However, the sparseness of Θ can also be used in another way:

3.2.1 USING ACTIVE VARIABLES

As the solutions are usually sparse, calculating the gradient for all variables in every step is wasteful. Most variables are zero and will not change from one iteration to the next. In order to be more

Algorithm 2: Pseudo-likelihood algorithm using active variables

```
if $\Theta^{(0)}$ not given then
    Set $\Theta^{(0)} = \text{diag}(\text{logit}(\hat{\mathbf{p}}^{(0)}))$ where $\hat{p}_s^{(0)} = \frac{1}{N}\sum_{k=1}^N x_{ks}$
end
Set k:=0;
Set $\mathcal{A} = \{(s,t) : s \geq t, \theta_{st} \neq 0\}$ as active variables;
repeat
    while not converged over variables in $\mathcal{A}$ do
        Find $\Theta^{(k+1)}$ using local approximation over variables in $\mathcal{A}$;
        Set k:=k+1;
    end
    Set $\mathcal{A} = \left\{(s,t) : \theta_{st} \neq 0 \text{ or } \left|\frac{\partial \tilde{l}}{\partial \theta_{st}}\right| > s_{st}\right\}$;
until $\mathcal{A}$ did not change ;
```

efficient, it is possible to move variables that are zero only once in a while. Several different kinds of methods have been proposed to exploit this situation, for example grafting (Perkins et al., 2003) and the implementation of the penalized logistic regression in Friedman, Hastie, and Tibshirani (2008b) among others. In our case here, we use an outer and an inner loop. The outer loop decides which variables are active. The inner loop then optimizes over only the active variables until convergence occurs. Active variables are those that are either non-zero, or that have a gradient large enough so that they would become non-zero in the next step. More details are given in Algorithm 2.

When using this method, convergence is still guaranteed. In the outer loop, the criterion chooses variables to be active that are either non-zero already or will be non-zero after one step of the local approximation over all variables. Therefore, if the active set stays the same, no variables would be moved in the next step as all active variables are already optimal and therefore, we have a solution over all variables. However, if the active set changes, the inner loop is guaranteed to improve the penalized pseudo-likelihood and find the optimum for the given set of active variables. As there are only a finite number of different active variables sets, the algorithm has to converge after a finite number of iterations of the outer loop.

3.2.2 SUBSTITUTING THE LINE SEARCH

Calculating the pseudo-likelihood is computationally expensive compared to the cost of the local approximation. Therefore, we save time by not performing a line search after every step. Instead, we fix a step size γ and skip the line search. However, with this method, the algorithm may diverge. In order to detect this, we calculate the penalized pseudo-likelihood every 100 iterations and check that we improved during the last 100 steps. If yes, the step size remains the same. If no, reset the variables to where they were 100 steps ago and divide the step size by 2. If the step size drops below a pre-specified threshold, we revert to the original line-search algorithm. This way, in most cases, we do not have to perform the line-search. However, the algorithm is still guaranteed to converge as it automatically detects convergence problems when the fixed step size is used and reverts to the line-search algorithm, which as stated above is guaranteed to converge.

4. Exact Solution Using Pseudo-likelihood

We now turn to our new method for optimizing the penalized log-likelihood. As the log-likelihood is a concave function, there are several standard methods that can be used for maximizing it, for example, gradient descent, BFGS and Newton's method among others. For each step of these methods, the gradient of the log-likelihood has to be calculated. This requires the evaluation of the partition function and its derivatives, which is computationally much more expensive than any other part of the algorithms. Taking this into considerations, the standard methods mentioned above have the following drawbacks:

Gradient descent: It can take many steps to converge and therefore require many evaluations of the partition function and its derivatives (using the junction tree algorithm). Also, it does not control the number of non-zero variables in intermediate steps well to which the runtime of the junction tree algorithm is very sensitive. Therefore, intermediate steps can take very long.

BFGS: Takes less steps than gradient descent, but similar to gradient descent, intermediate steps can have more non-zero variables than the solution. Thus, same as above, computations of intermediate steps can be slow.

Newton's method: In order to locally fit the quadratic function, the second derivatives of the log-likelihood are needed. Computing these is computationally prohibitive.

Lee, Ganapathi, and Koller (2006a) use the BFGS method only on a set of active variables onto which additional variables are "grafted" until convergence. This mitigates the problem of slow intermediate steps and makes using BFGS feasible. However, this comes at the expense of an increased total number of steps, as only one variable at a time is being grafted. Here, we want to use the pseudo-likelihood in a new algorithm for maximizing the penalized log-likelihood.

The functional form of the pseudo-likelihood is by its definition closely related to the real likelihood and in Section 5 we will see that its solutions are also very similar, indicating that it approximates the real likelihood reasonably well. Furthermore, we can also maximize the pseudo-likelihood very quickly. We want to leverage this by using a quasi-Newton method in which we fit a "tilted" pseudo-likelihood instead of a quadratic function. Specifically, among all functions of the form

$$f_{\Theta^{(k)}} = \frac{1}{2}\tilde{l} + \sum_{s \geq t} a_{st}(\theta_{st} - \theta_{st}^{(k)}) - \gamma \sum_{s \geq t} (\theta_{st} - \theta_{st}^{(k)})^2 - N||\mathbf{R} * \Theta||_1,$$

we fit the one that at $\Theta^{(k)}$ is a first order approximation to the penalized log-likelihood l. Essentially, $f_{\Theta^{(k)}}$ is an L_1-penalized pseudo-likelihood with an added linear term as well as a very simple quadratic term for some $\gamma > 0$. Here, a_{st} will be chosen so that the sub-gradient is equal to the penalized log-likelihood l at $\Theta^{(k)}$. The additional quadratic term with coefficient γ is only being included to ensure the existence of a global maximum. In practice, we can set $\gamma = 0$, unless a convergence problem occurs. In our simulations, $\gamma = 0$ always worked very well.

In particular, for the approximation at $\Theta^{(k)}$, choose a_{st} such that

$$\begin{aligned} f_{\Theta^{(k)}} = \frac{1}{2}\Bigg(& \tilde{l}(\Theta|\mathbf{X}) + \sum_{s>t}(\theta_{st}-\theta_{st}^{(k)})\left((\hat{\mathbf{p}}(\Theta^{(k)})_s^T\mathbf{X})_t + (\hat{\mathbf{p}}(\Theta^{(k)})_t^T\mathbf{X})_s - 2\cdot N\cdot w_{st}(\Theta^{(k)})\right) + \\ & + \sum_s(\theta_{ss}-\theta_{ss}^{(k)})\left(\sum_k \hat{p}_{sk}(\Theta^{(k)}) + (\mathbf{X}^T\mathbf{X})_{ss} - 2\cdot N\cdot w_{ss}(\Theta^{(k)})\right)\Bigg) - \\ & - \sum_{s\geq t}\gamma(\theta_{st}-\theta_{st}^{(k)})^2 - N||\mathbf{R}*\Theta||_1 \end{aligned}$$

where $\mathbf{W}$ is a matrix with elements $w_{st}(\Theta) = \frac{\partial\Psi}{\partial\theta_{st}}(\Theta) = \mathbb{E}_\Theta(x_s x_t)$ is the derivative of the partition function and $\hat{\mathbf{p}}_s$ is as defined in the pseudo-likelihood section.

For the algorithm, we need an initial parameter estimate $\Theta^{(0)}$, which we pick as follows: Let $Z(\theta) = \frac{e^\theta}{1+e^\theta}$ be the logistic function and let Z^{-1} denote its inverse. Then choose

$$\theta_{st}^{(0)} = \begin{cases} 0 & \text{if } s \neq t \\ Z^{-1}\left(\frac{1}{N}\sum_{k=1}^N x_{ks}\right) & \text{if } s = t \end{cases}.$$

In this case we than have $\hat{p}_{sk} = \frac{1}{N}\sum_{k=1}^N x_{ks} \quad \forall k$ and also

$$W_{st} = \begin{cases} \left(\frac{1}{N}\sum_{k=1}^N x_{ks}\right)\left(\frac{1}{N}\sum_{k=1}^N x_{kt}\right) & \text{if } s \neq t \\ \left(\frac{1}{N}\sum_{k=1}^N x_{ks}\right) & \text{if } s = t \end{cases}$$

and together with $\gamma = 0$ we then get that

$$f_{\Theta^{(0)}} = \frac{1}{2}\tilde{l}(\Theta|\mathbf{X}) - N||\mathbf{R}*\Theta||_1$$

and thus just a regular pseudo-likelihood step. The only slight difference is that in this case the penalty term on the diagonal is twice as large as in the pseudo-likelihood case presented above. However, in practice we recommend not to penalize the diagonal at all, so that this difference vanishes. Therefore, this algorithm is a natural extension of the pseudo-likelihood approach that starts out by performing a regular pseudo-likelihood calculation and then proceeds to converge to the solution by a series of adjusted pseudo-likelihood steps.

The algorithm for maximizing the penalized log-likelihood is now very similar to the one presented for maximizing the penalized pseudo-likelihood. Assume our current estimate is $\Theta^{(k)}$. Then approximate the log-likelihood locally by $f_{\Theta^{(k)}}$ and find the maximizer Θ^* of $f_{\Theta^{(k)}}$. This is essentially the pseudo-likelihood and the algorithm presented above can easily be adjusted to accommodate the additional terms. Now, using a line search on the line between $\Theta^{(k)}$ and Θ^*, find the next estimate $\Theta^{(k+1)}$. This algorithm is guaranteed to converge by the same argument as the pseudo-likelihood algorithm. The proof that $f_{\Theta^{(k)}}$ approximates l at $\Theta^{(k)}$ to first order can be found in Appendix B, which is a prerequisite for the convergence proof of the algorithm in Appendix A.

4.1 Speed Improvement

As for the pseudo-likelihood algorithm, we can again save computations by using the active variables technique presented above. Here, the savings in time are especially large due to a special feature of the junction tree algorithm that we use to calculate the derivatives of the partition function.

Algorithm 3: Likelihood algorithm using active variables

```
if $\Theta^{(0)}$ not given then
    Set $\Theta^{(0)} = \mathrm{diag}(\mathrm{logit}^{-1}(\hat{\mathbf{p}}^{(0)}))$ where $\hat{p}_s^{(0)} = \frac{1}{N}\sum_{k=1}^{N} x_{ks}$
end
Set k:=0;
Set $\mathcal{A} = \{(s,t) : s \geq t, \theta_{st} \neq 0\}$ as active variables;
repeat
    while not converged over variables in $\mathcal{A}$ do
        Calculate $\mathbf{W}$ only over variables in $\mathcal{A}$;
        Find $\Theta^{(k+1)}$ using local approximation over variables in $\mathcal{A}$;
        Set k:=k+1;
    end
    Calculate the whole matrix $\mathbf{W}$;
    Set $\mathcal{A} = \left\{(s,t) : \theta_{st} \neq 0 \text{ or } \left|\frac{\partial l}{\partial \theta_{st}}\right| > r_{st}\right\}$;
until $\mathcal{A}$ did not change ;
```

In order to calculate derivatives with respect to non-zero variables, only one pass of the junction tree algorithm is necessary. However, p passes of the junction tree are needed in order to get the full matrix of derivatives $\mathbf{W}$. Therefore, depending on the size of p, using only active variables can be considerable faster. For details, see Algorithm 3.

4.2 Graphical Lasso for the Discrete Case

In the case of Gaussian Markov networks, the graphical lasso algorithm (see Friedman, Hastie, and Tibshirani, 2008a) is an very efficient method and implementation for solving the L_1-penalized log-likelihood. In order to leverage this speed, we extended the methodology to the binary case treated in this article. However, the resultant algorithm was not nearly as fast as expected. In the Gaussian case, the algorithm is very fast as the approach to update the parameter matrix Θ one row at a time allows for a closed form solution and efficient calculations. In the binary case on the other hand, the computational bottleneck is not all of the calculation involved in the update of Θ, but specifically by a large margin the evaluations of the partition function itself. Therefore, any fast algorithm for solving the penalized log-likelihood exactly has to use as few evaluations of the partition function as possible. The graphical lasso approach is thus not suitable for the binary case as it takes a lot of small steps towards the solution.

This observation also explains the improvement in speed of the pseudo-likelihood based exact algorithm over the specific methods proposed in Lee, Lee, Abbeel, and Ng (2006b). Standard convex optimization procedures often rely on either the first or second derivatives of the functions they seek to optimize. Newton-like procedures that use the second derivative often converge in very few steps, however these cannot be used here as it is prohibitively expensive to evaluate the second derivative of the partition function, even in small examples. Approaches like conjugate gradients of BFGS as proposed in Lee, Lee, Abbeel, and Ng (2006b) are somewhat less efficient and take more steps. This is where the advantage of using the pseudo-likelihood as a local approximation comes into play. It usually only takes very few steps to converge and is therefore faster than standard methods.

5. Simulation Results

In order to compare the performance of the estimation methods for sparse graphs described in this article as well as Lee, Ganapathi, and Koller (2006a) and Wainwright et al. (2006), we use simulated data and compare the speed as well as the accuracy of these methods.

5.1 Setup

Before simulating the data it is necessary to generate a sparse symmetric matrix Θ. First, the diagonal is drawn uniformly from the set $\{-0.5, 0, 0.5\}$. Then, using the average number of edges per node, upper-triangular elements of Θ are drawn uniformly at random to be non-zero. These non-zero elements are then set to either -0.5 or 0.5, again each uniformly. In order for Θ to by symmetric, the lower triangular matrix is set equal to the upper triangular matrix. The actual data is generated by Gibbs sampling using Θ as described above.

With respect to the penalty parameters that we use for the different methods, we always leave the diagonal unpenalized and all off-diagonal elements have the same parameter ρ, that is we set

$$r_{st} = \begin{cases} 0 & \text{if } s = t \\ \rho & \text{otherwise} \end{cases}$$

and the penalty term matrix for the pseudo-likelihood $\mathbf{S} = 2\mathbf{R} - \operatorname{diag}(\mathbf{R})$ as defined above. For the Wainwright-methods, the penalty parameter is ρ with no penalty on the intercept. Although the log-likelihood functions that are being penalized are somewhat different, this choice of parameters makes them perform roughly equivalent, as can be seen in Figure 3. The number of edges is plotted against the penalty parameter used and all methods behave very similar. However, in order not to confound some results by these slight differences of the effects of the penalty, all the following plots are with respect to the number of edges in the graph, not the penalty parameter itself.

5.2 Speed Comparison

First, we compare the speed of the four methods for the L_1-penalized model. We used an annealing schedule for Lee, Ganapathi, and Koller (2006a) to improve convergence as suggested in their article. Plots of the speeds of the exact methods can be seen in Figure 4 and the approximate methods are shown in Figure 5. Each plot shows the time the algorithm needed to converge versus the number of edges in the estimated graph. As can be seen, the pseudo-likelihood based exact algorithm described above is considerable faster than the one proposed in Lee, Ganapathi, and Koller (2006a). For the approximate algorithms, we can see that the p logistic regressions in the Wainwright et al. (2006) algorithm take roughly the same amount of time as the pseudo-likelihood algorithm presented above. This is not surprising due to the similarity of the optimization methods and any difference that can be observed in Figure 5 is mostly due to the specific implementations used. Furthermore, we would like to note that we decided to plot the speed against the number of edges in the graph instead of the penalty parameter, as the runtime of the algorithm is very closely related to the actual sparseness of the graph.

Overall, when comparing the computation times for the exact algorithms to the approximate algorithms, we can see that the approximate methods are orders of magnitude faster and do not suffer from an exponentially increasing computation time for decreasing sparsity as the exact methods. Therefore, if an exact solution is required, our exact algorithm is preferable. On the other hand, the

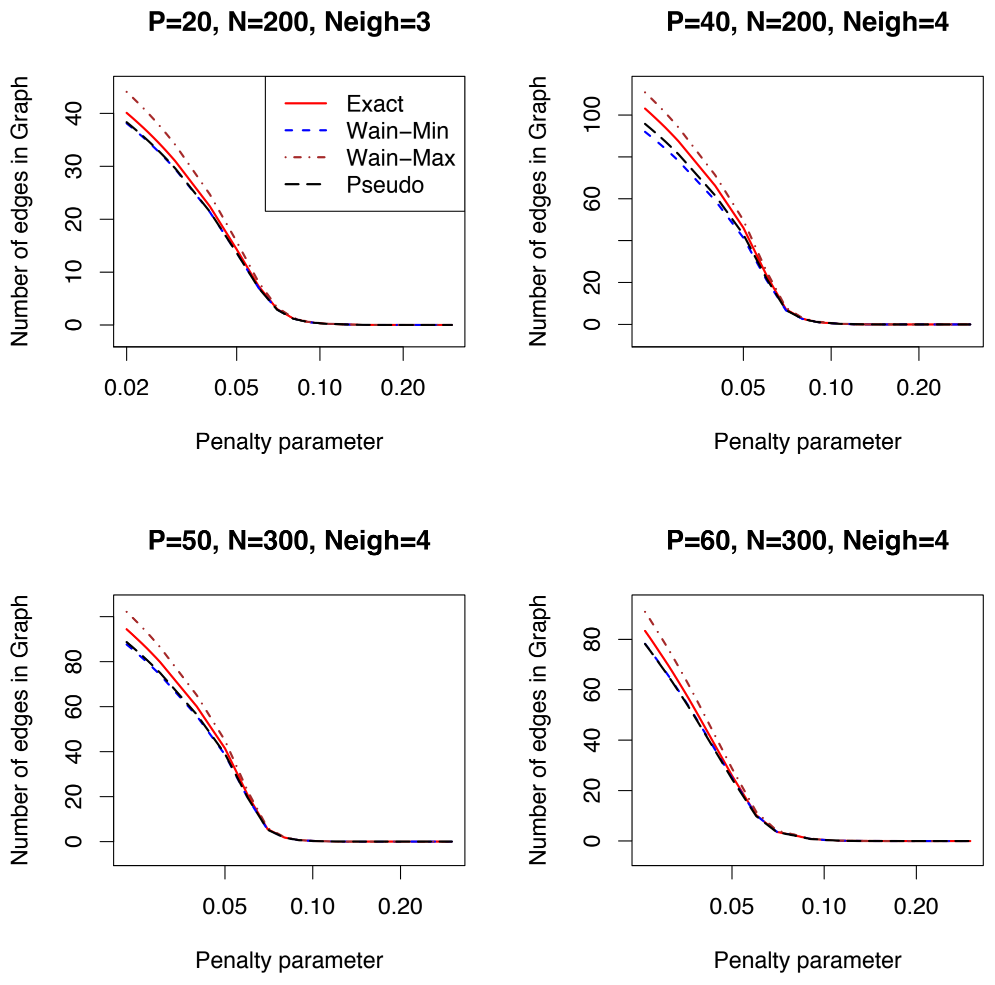


Figure 3: Number of edges in the graph vs. penalty parameter for different problem sizes, averaged over 20 simulations.

superior speed of the pseudo-likelihood and Wainwright et al. (2006) algorithm warrants a closer look at the trade-off with respect to accuracy.

5.3 Accuracy Comparisons

In this subsection we compare the algorithms mentioned above with respect to the accuracy with which they recover the original model. As both our L_1 penalized exact algorithm and the one by Lee, Ganapathi, and Koller (2006a) find the exact maximizer of the L_1-penalized log-likelihood, we only use our algorithm in the comparison. The other 3 methods we compare to are "Wainwright-min", "Wainwright-max" and the pseudo-likelihood.

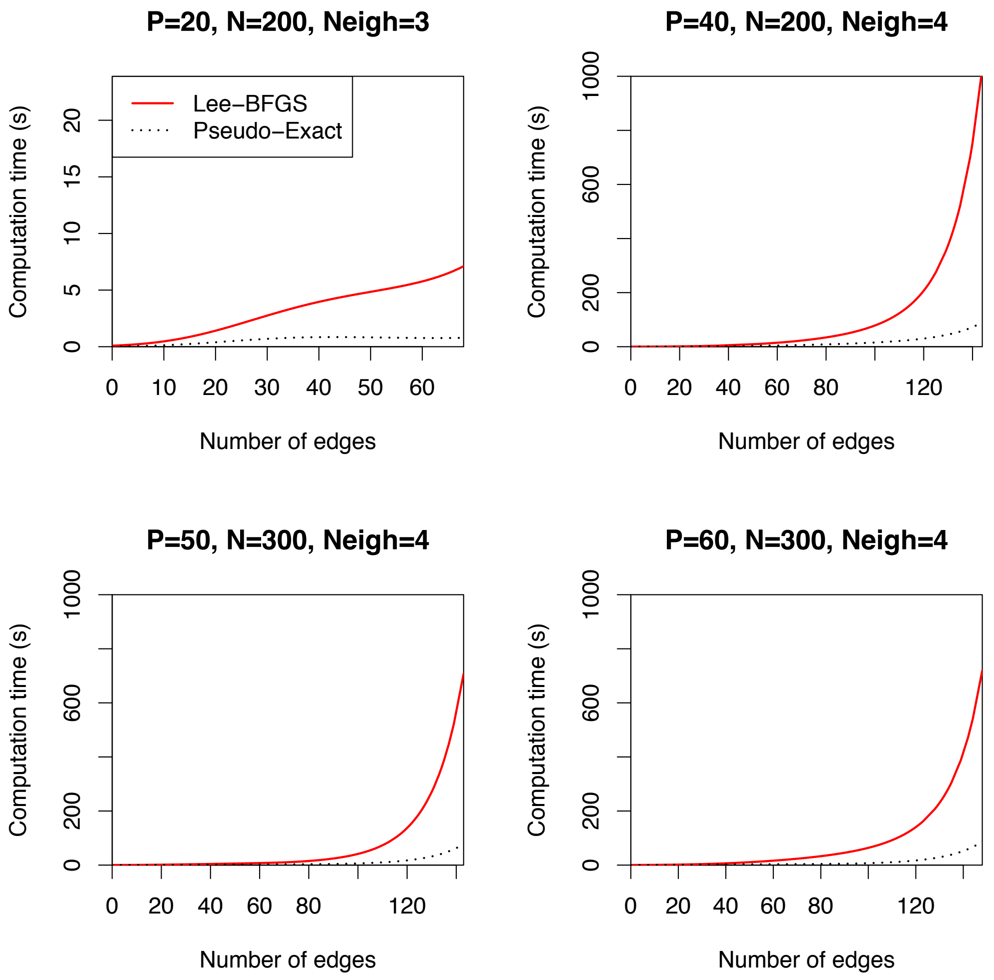


Figure 4: Computation time of the exact algorithms versus the number of non-zero elements in Θ. Values are averages over 20 simulation runs, along with ± 2 standard error curves. Also, p is the number of variables in the model, N the number of observations and *Neigh* is the average number of neighbors per node in the simulated data. Here, Pseudo-Exact refers to the the exact solution algorithm that uses adjusted pseudo-likelihoods as presented in Section 4.

First we investigate how closely the edges in the estimated graph correspond to edges in the true graph. In Figure 6, ROC curves are shown, plotting the false positive (FP) rates against true positive (TP) rates for edge identification, for various problem sizes. Note that only partial ROC curves are shown since our method cannot estimate non-sparse graphs due to very long computation times. Overall, we see that all approximate algorithms match the results of the exact solution very closely and in some of the plots, the curves even lie almost on top of each other.

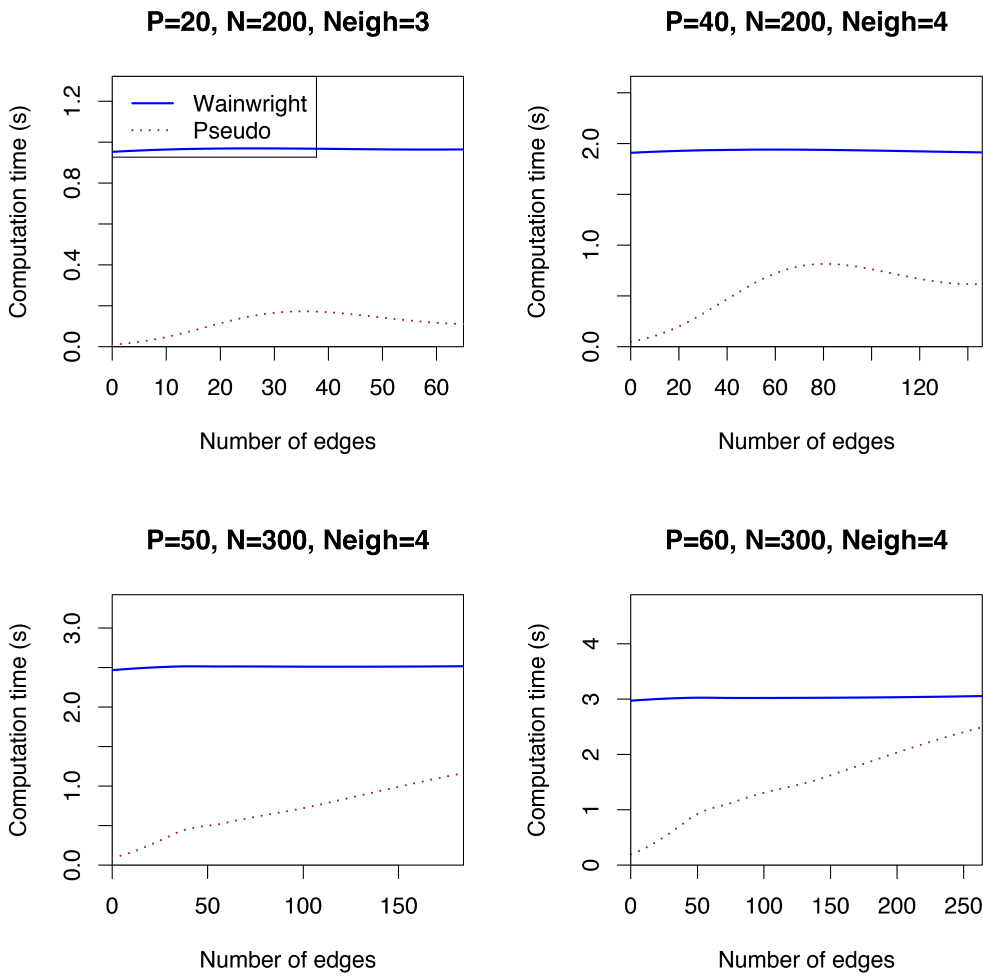


Figure 5: Computation time for the approximate algorithms versus the number of non-zero elements in Θ. Values are averages over 20 simulation runs, along with ± 2 standard error curves. Also, p is the number of variables in the model, N the number of observations and *Neigh* is the average number of neighbors per node in the simulated data.

Apart from the accuracy of edge identification, we also consider other statistics. The unpenalized log-likelihood is a measure of how well the estimated model fits the observed data (higher values are better). Again, the approximate solutions are all very close to the exact solution (see Figure 7) and the differences are always smaller than 2 standard deviations. In Figure 8, we plot the difference of the log-likelihood with respect to the exact solution. Also in this plot, no clear "winner" can be identified.

We also use the Kullback-Leibler divergence $D_{KL}(\mathbb{P}||\mathbb{Q})$, which is a measure of difference between a true probability distribution $\mathbb{P}$ and an arbitrary other distribution $\mathbb{Q}$. Here, for the true

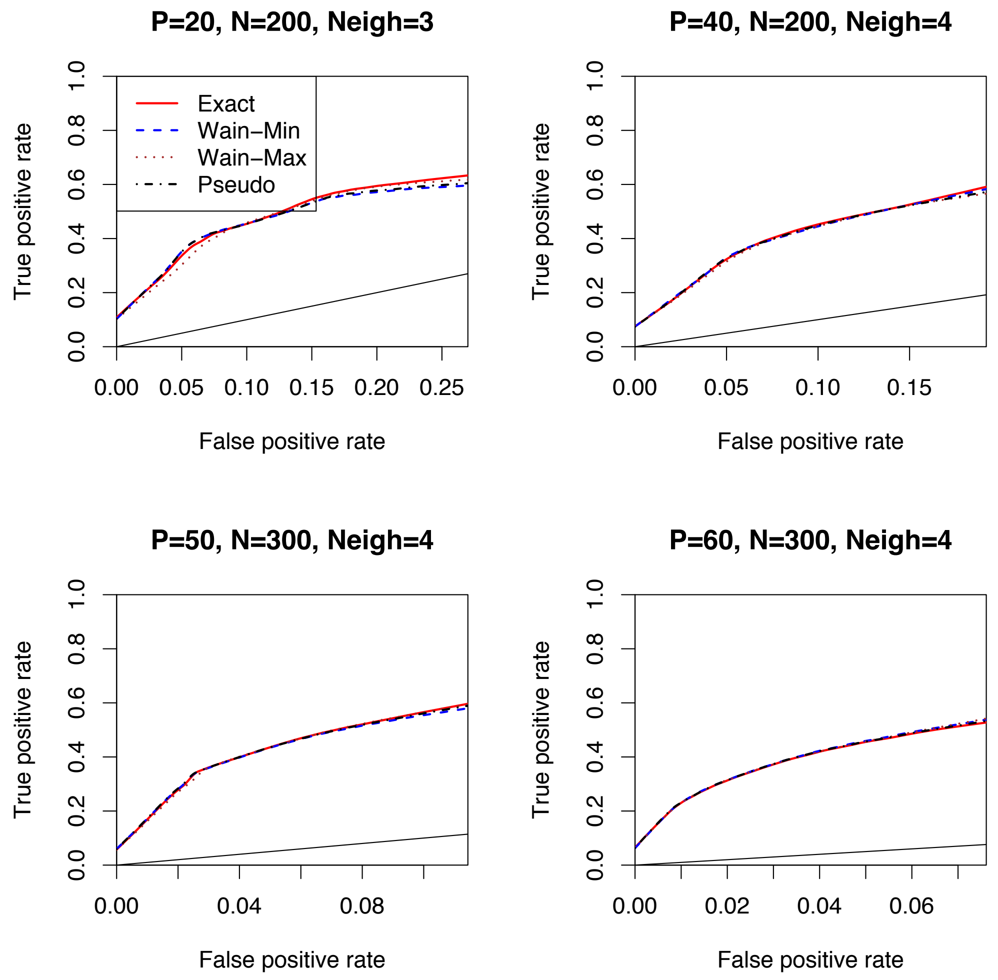


Figure 6: ROC curves: false positive versus true positive rate for edge identification. Values are averages over 20 simulation runs.

probability distribution we use the distribution of the binary Markov network using the true Θ_0-matrix that was used to generate the simulated data. The distribution $\mathbb{Q}$ in our case is the binary Markov network using the estimated $\hat{\Theta}$-matrix. We can compute $D_{KL}(\mathbb{P}||\mathbb{Q})$ as

$$\begin{aligned} D_{KL}(\mathbb{P}||\mathbb{Q}) &= \sum_x \mathbb{P}(x)\log\frac{\mathbb{P}(x)}{\mathbb{Q}(x)} = \Psi(\Theta_0) - \Psi(\hat{\Theta}) + \sum_x \mathbb{P}(x)\operatorname{tr}\left(xx^T(\hat{\Theta}-\Theta_0)\right) = \\ &= \Psi(\Theta_0) - \Psi(\hat{\Theta}) + \operatorname{tr}\left(\mathbb{E}_{\mathbb{P}}(xx^T)(\hat{\Theta}-\Theta_0)\right). \end{aligned}$$

If the distributions $\mathbb{P}$ and $\mathbb{Q}$ are identical, then $D_{KL}(\mathbb{P}||\mathbb{Q}) = 0$. In our simulations, the exact solution has lower KL-divergence than the other methods, however the differences are very small. For a plot of the KL-divergence against the number of edges in the model see Figure 9. Again, all approximate

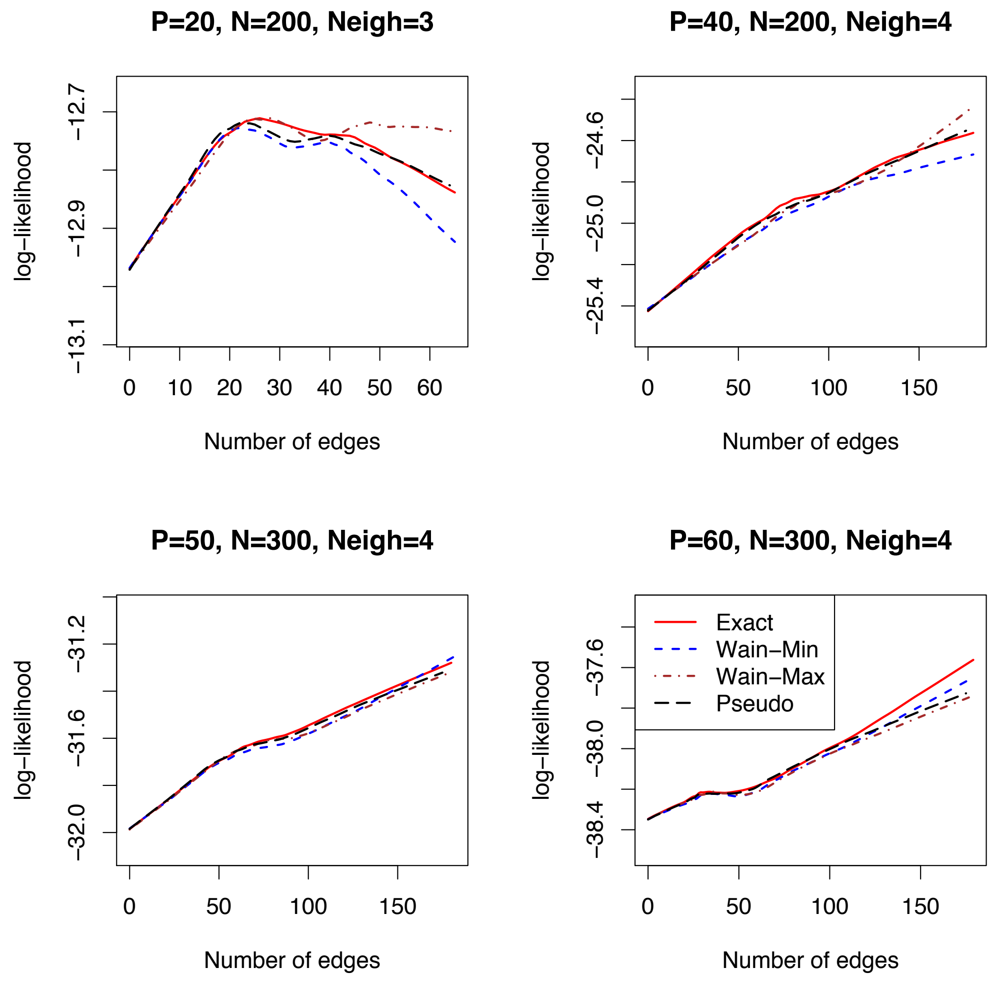


Figure 7: Log-likelihood of the estimated model vs number of edges in the graph for different problem sizes, averaged over 20 simulations.

methods match the exact solution very closely and any differences are well within the 2 standard deviation error band. In Figure 10 the differences of the KL-divergence of the approximate to the exact method can be seen. Again, all methods are very close with the pseudo-likelihood approach performing the best in this case.

6. Discussion

When we embarked on this work, our goal was to find a fast method for maximizing the L_1 penalized log-likelihood of binary-valued Markov networks. We succeeded in doing this, and found that the resulting procedure is faster than competing exact methods. However, in the course of this work,

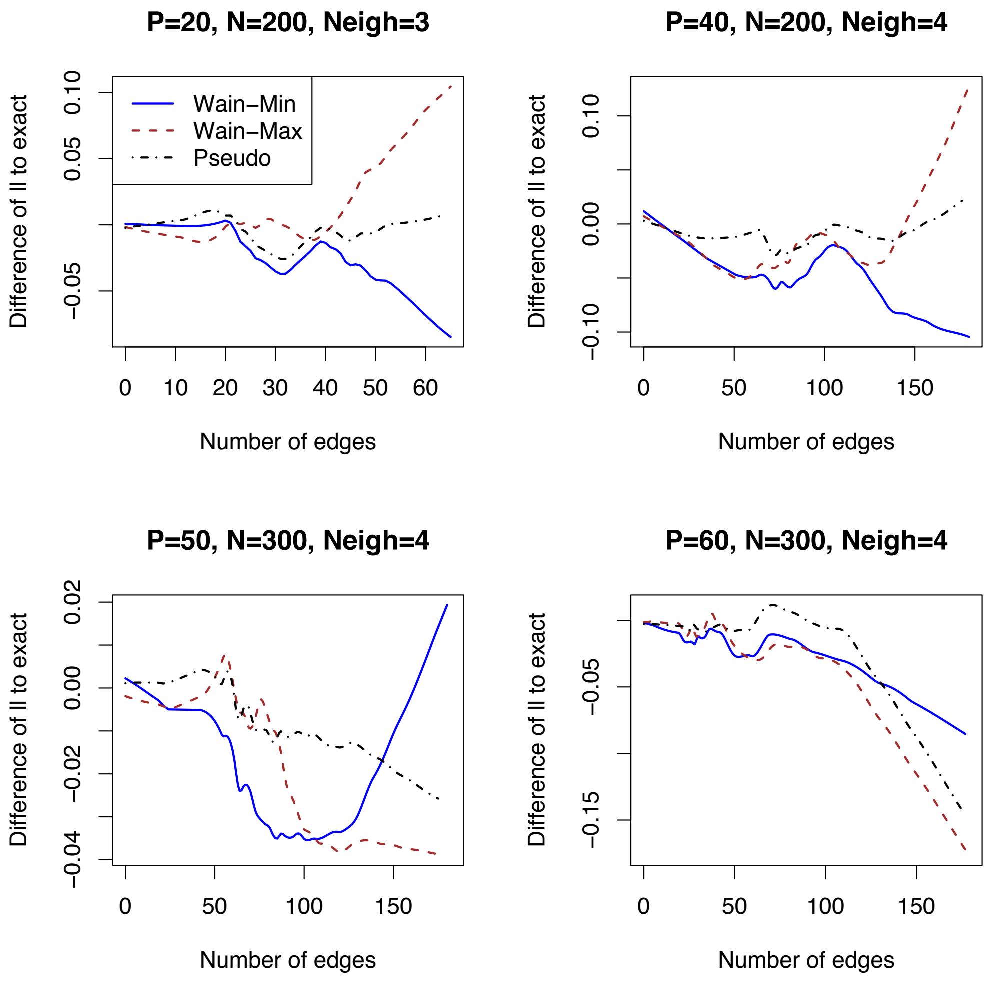


Figure 8: Difference of log-likelihood of the estimated model to the exact model vs number of edges in the graph for different problem sizes, averaged over 20 simulations.

we also learned something surprising: several approximate methods exist that are *much* faster and only slightly less accurate than the exact methods. In addition, when a dense solution is required, the exact methods become infeasible while the approximate methods can still be used. Our implementation of the methods of Wainwright et al. (2006) uses the fast coordinate descent procedure of Friedman, Hastie, and Tibshirani (2008b), a key to its speed. The pseudo-likelihood algorithm also uses similar techniques, which make it very fast as well. We conclude that the Wainwright and pseudo-likelihood methods should be seriously considered for computation in Markov networks.

In this article, we treated the case of pairwise Markov networks with a binary response variable. We think these methods can also be extended to more general cases. With respect to the response variables, a multinomial instead of a binary response could be used. In addition to this, it would

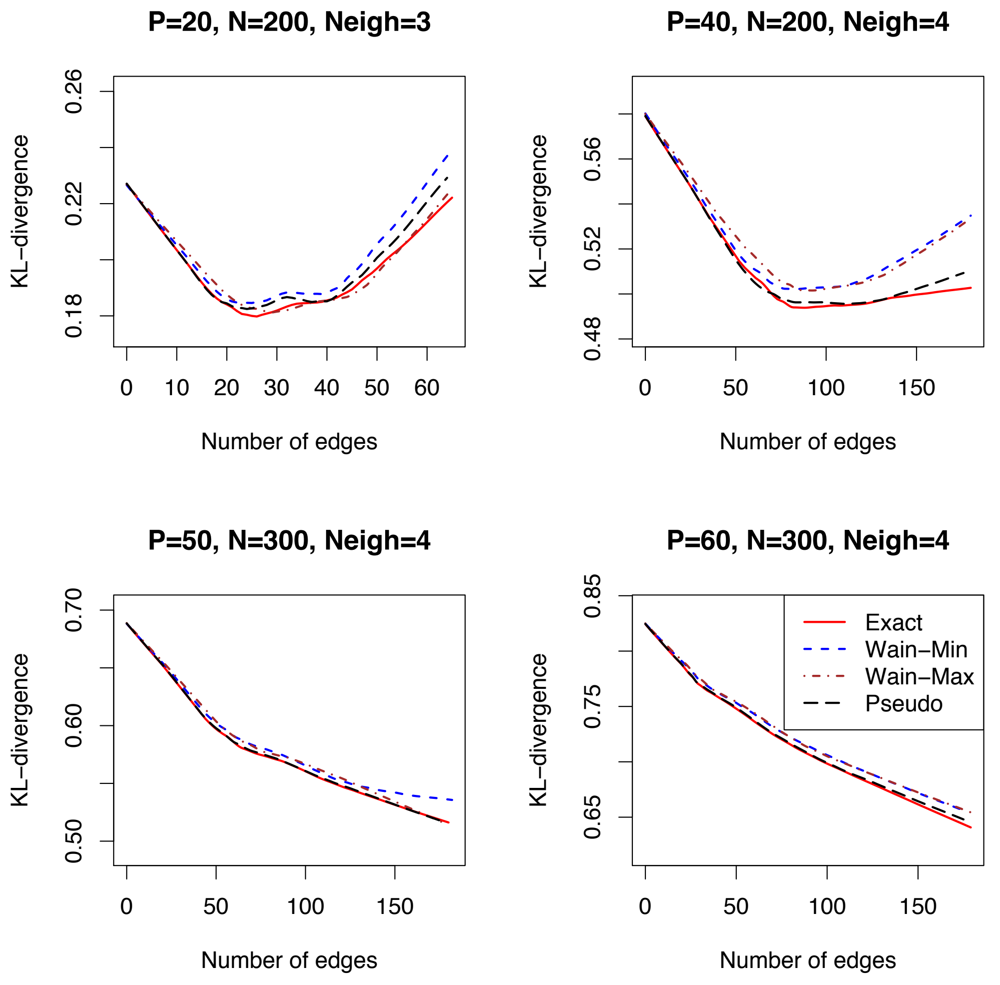


Figure 9: Kullback-Leibler divergence of the estimated model vs. number of edges in the graph for different problem sizes, averaged over 20 simulations.

also be possible to generalize the graph structure by introducing higher order interaction terms. Apart from these extensions, an interesting possibility for future work would also be to prove the theoretical results of Wainwright et al. (2006) for the pseudo-likelihood model. Furthermore, we believe that both the exact and fast approximate methods can also be applied to the learning of multilayer generative models, such as restricted Boltzmann machines (see Hinton, 2007).

An R language package for fitting sparse graphical models, both by exact and approximate methods, will be made available on the authors' websites.

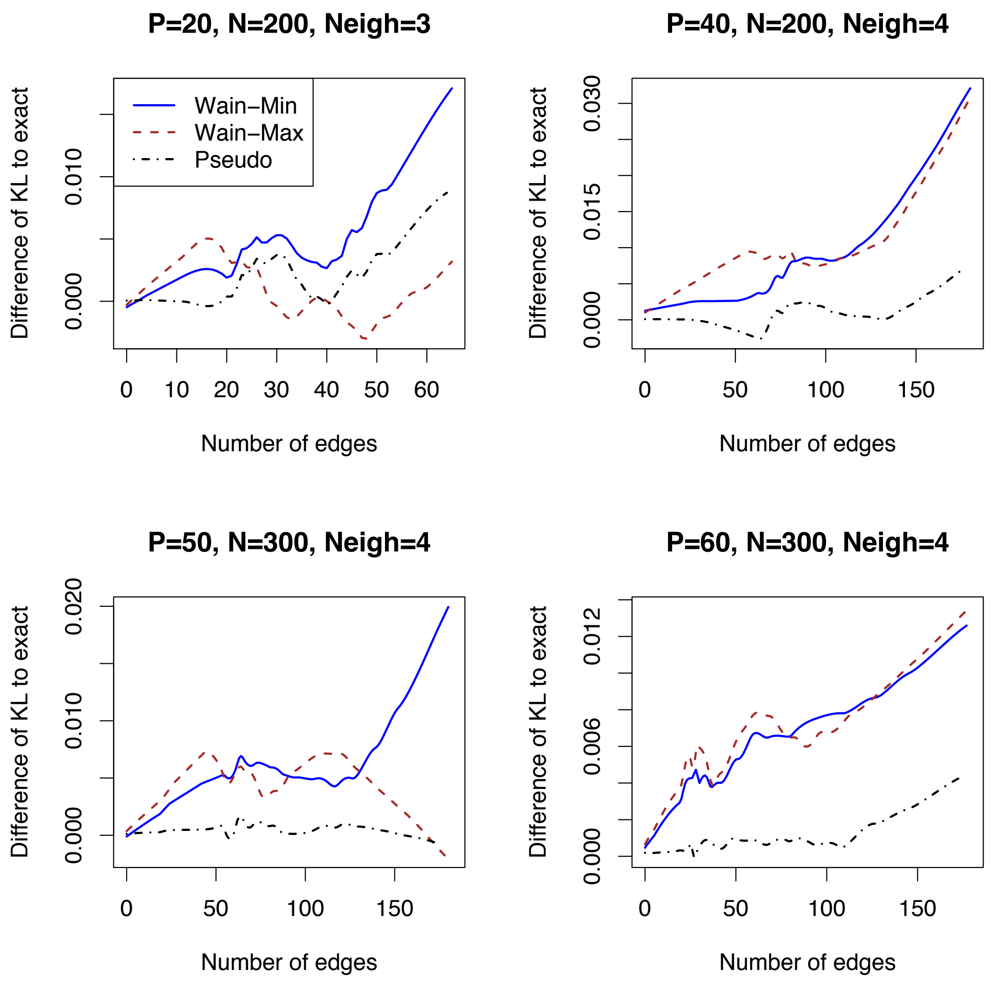


Figure 10: Difference of Kullback-Leibler divergence of the approximate methods to the exact solution vs. number of edges in the graph for different problem sizes, averaged over 20 simulations.

Acknowledgments

Hoefling was supported by a Stanford Graduate Fellowship. Tibshirani was partially supported by National Science Foundation Grant DMS-9971405 and National Institutes of Health Contract N01-HV-28183. We would like to thank Trevor Hastie and Jerome Friedman for code implementing the fast L_1 logistic regression and Daniela Witten for helpful comments on a draft of this article. We also want to thank the editor and two anonymous reviewers for their work and helpful reviews.

Appendix A. Proof of Convergence for Penalized Pseudo-likelihood and Penalized Log-likelihood Algorithms

Lee, Lee, Abbeel, and Ng (2006b) gives a proof of convergence for an algorithm that solves an L_1 constrained problem by quadratic approximation of the objective function. Here, we will follow this proof very closely and make few changes to accommodate that we are only using a first order approximation and are working with the Lagrangian form of the L_1 constrained problem instead of the standard form.

Assume that $g(\Theta)$ is a strictly convex function with a global minimum that we want to minimize. Furthermore, let $f_{\Theta_0}(\Theta)$ be a first order approximation of g at Θ_0 and assume that $f_{\Theta_0}(\Theta)$ is strictly convex, has a global optimum and is jointly continuous in (Θ_0, Θ). Here, by first order approximation at Θ_0, we mean that $f_{\Theta_0} - g$ is twice continuously differentiable with derivative 0 at Θ_0. Assume that our algorithm works as follows:

```
initialize $\Theta^{(0)}$;
Set k:=0;
**while** *not converged* **do**
    With current estimate $\Theta^{(k)}$, define local approximation $f_{\Theta^{(k)}}(\Theta)$ to $g$;
    Find solution $\Theta^*$ of $f_{\Theta^{(k)}}(\Theta)$;
    Perform backtracking line search on the line from $\Theta^{(k)}$ to $\Theta^*$ to find $\Theta^{(k+1)}$;
    Set k:=k+1;
**end**
```

Then $\Theta^{(k)}$ converges to the global minimizer of $g(\Theta)$. In order to show this, we first need the following lemma:

Lemma 1 *Let Θ_0 be any point that is not the global optimum. Then there is an open subset S_{Θ_0} and a constant K_{Θ_0} such that for every Φ_0 in S_{Θ_0} every iteration of the algorithm starting at Φ_0 will return a point Φ_1 that improves the objective by at least K_{Θ_0}, that is, $g(\Phi_1) \leq g(\Phi_0) - K_{\Theta_0}$.*

Proof First, let f_{Θ_0} be an approximation to g at Θ_0 with global optimum Θ_1. Then set

$$\delta = f_{\Theta_0}(\Theta_0) - f_{\Theta_0}(\Theta_1)$$

and we know that $\delta > 0$ as Θ_0 is not the global optimum. Now, there exists an $\varepsilon > 0$ such that for $\Phi_0 \in S_{\Theta_0} := \{\Theta : ||\Theta - \Theta_0||_2 < \varepsilon\}$ the following holds:

$$|g(\Theta_0) - g(\Phi_0)| \leq \frac{\delta}{8}$$

and

$$|f_{\Phi_0}(\Phi_1) - f_{\Theta_0}(\Theta_1)| \leq \frac{\delta}{4}$$

where Φ_1 is the global minimum of f_{Φ_0}. The existence of this ε follows from the continuity and convexity of f and g. Then

$$f_{\Phi_0}(\Phi_1) \leq f_{\Theta_0}(\Theta_1) + \frac{\delta}{4} \leq f_{\Theta_0}(\Theta_0) - \frac{3\delta}{4} = \\ = g(\Theta_0) - \frac{3\delta}{4} \leq g(\Phi_0) - \frac{\delta}{2}.$$

With step size $0 < t < 1$ in the line search, and using the previous result, it holds that

$$f_{\Phi_0}(\Phi_0 + t(\Phi_1 - \Phi_0)) \leq (1-t) f_{\Phi_0}(\Phi_0) + t f_{\Phi_0}(\Phi_1) \leq g(\Phi_0) - t\frac{\delta}{2}$$

For the next step observe that the minimizer Φ_1 of f_{Φ_0} is a continuous function of Φ_0 due to the convexity of f in the second argument and the continuity in both arguments. Then, as S_{Θ_0} is a compact set, there exists a compact set T_{Θ_0} with $\Phi_1(\Phi_0) \in T_{\Theta_0}$ for all $\Phi_0 \in S_{\Theta_0}$. Thus, as f_{Θ_0} is a first order approximation of g at Θ_0, there exists a C such that for all $\Theta \in T_{\Theta_0}$

$$g(\Theta) \leq f_{\Theta_0}(\Theta) + C||\Theta - \Theta_0||_2^2.$$

Therefore

$$g(\Phi_0 + t(\Phi_1 - \Phi_0)) \leq g(\Phi_0) - t\frac{\delta}{2} + Ct^2||\Phi_1 - \Phi_0||_2^2 \leq \\ \leq g(\Phi_0) - t\frac{\delta}{2} + t^2 CD^2$$

where D is the diameter of T_{Θ_0}. Now set $t^* = \min\left(1, \frac{\delta}{4CD^2}\right)$ and thus we know that it exists a Φ^* such that

$$g(\Phi^*) \leq g(\Phi_0) - t^*\frac{\delta}{2} + t^{*2}CD^2.$$

Setting $K_{\Theta_0} = t^*\frac{\delta}{2} - t^{*2}CD^2 > 0$ now finishes the proof. ■

Now, using the lemma the rest of the proof is again very similar as in Lee, Lee, Abbeel, and Ng (2006b) and we only repeat it here for completeness.

Theorem 1 *The algorithm converges in a finite number of steps.*

Proof Pick $\delta > 0$ arbitrary. Let Θ^* the global optimum and Θ_0 the starting point of the algorithm. Then there exists a compact set K such that $g(\Theta) > g(\Theta_0)$ for every $\Theta \notin K$. Define $P_\delta = \{\Theta : ||\Theta - \Theta^*|| \geq \delta\} \cap K$. We will show convergence by showing that the algorithm can only spend a finite number of steps in P_δ. For every Θ in P_δ there exists an open set S_Θ. So

$$P_\delta \subseteq \cup_{\Theta \in P_\delta} S_\Theta$$

As P_δ is compact, Heine-Borel guarantees that there is a finite set Q_δ such that

$$P_\delta \subseteq \cup_{\Theta \in Q_\delta} S_\Theta.$$

Furthermore, as Q_δ is finite, define

$$C_\delta = \min_{\Theta \in Q_\delta} K_\Theta.$$

As the lemma guarantees that every step of the algorithm inside P_δ improves the objective by at least C_δ and a global optimum exists by assumption, the algorithm can at most spend a finite number of steps in P_δ. Therefore, the algorithm has to converge in a finite number of steps. ■

For the penalized pseudo-likelihood algorithm, by definition of the approximation it is evident that it is a first order approximation. The situation for the penalized log-likelihood algorithm is a little more complicated and it will be shown in the next section of the appendix that the proposed approximation is to first order and therefore satisfies the assumptions of the proof.

Appendix B. First Order Approximation of Log-likelihood

In Section 4, we defined a function $f_{\Theta^{(k)}}$ to calculate the next estimate $\Theta^{(k+1)}$. The convergence proof in Appendix A requires that $f_{\Theta^{(k)}}$ is a first order approximation of the objective $l(\Theta|\mathbf{X}) - N||\mathbf{R} * \Theta||_1$. Here, we want to show that this is in fact the case. For this, we need to show that $f_{\Theta^{(k)}} - l(\Theta|\mathbf{X}) + N||\mathbf{R} * \Theta||_1$ is twice continuously differentiable with derivative 0 at $\Theta^{(k)}$.

First, inserting $f_{\Theta^{(k)}}$ from Section 4 yields

$$\begin{aligned}
d_{\Theta^{(k)}} &= f_{\Theta^{(k)}} - l(\Theta|\mathbf{X}) + N||\mathbf{R} * \Theta||_1 = \\
&= \frac{1}{2}\Bigg(\tilde{l}(\Theta|\mathbf{X}) + \sum_{s>t}(\theta_{st} - \theta_{st}^{(k)})\left((\hat{\mathbf{p}}(\Theta^{(k)})_s^T \mathbf{X})_t + (\hat{\mathbf{p}}(\Theta^{(k)})_t^T \mathbf{X})_s - 2 \cdot N \cdot w_{st}(\Theta^{(k)})\right) + \\
&\quad + \sum_s (\theta_{ss} - \theta_{ss}^{(k)})\left(\sum_k \hat{p}_{sk}(\Theta^{(k)}) + (\mathbf{X}^T\mathbf{X})_{ss} - 2 \cdot N \cdot w_{ss}(\Theta^{(k)})\right)\Bigg) - \\
&\quad - \sum_{s \geq t} \gamma(\theta_{st} - \theta_{st}^{(k)})^2 - l(\Theta|\mathbf{X})
\end{aligned}$$

which has derivative

$$\begin{aligned}
2\frac{\partial d_{\Theta^{(k)}}}{\partial \theta_{st}} &= 2(\mathbf{X}^T\mathbf{X})_{st} - (\hat{\mathbf{p}}(\Theta)_s^T \mathbf{X})_t - (\hat{\mathbf{p}}(\Theta)_t^T \mathbf{X})_s + (\hat{\mathbf{p}}(\Theta^{(k)})_s^T \mathbf{X})_t + (\hat{\mathbf{p}}(\Theta^{(k)})_t^T \mathbf{X})_s - \\
&\quad - 2 \cdot N \cdot w_{st}(\Theta^{(k)}) - 4\gamma(\theta_{st} - \theta_{st}^{(k)}) - 2(\mathbf{X}^T\mathbf{X})_{st} + 2 \cdot N \cdot w_{st}(\Theta)
\end{aligned}$$

for $s \neq t$ and

$$\begin{aligned}
2\frac{\partial d_{\Theta^{(k)}}}{\partial \theta_{st}} &= (\mathbf{X}^T\mathbf{X})_{ss} - \sum_k \hat{p}_{sk}(\Theta) + \sum_k \hat{p}_{sk}(\Theta^{(k)}) + (\mathbf{X}^T\mathbf{X})_{ss} - 2 \cdot N \cdot w_{ss}(\Theta^{(k)}) - \\
&\quad - 4\gamma(\theta_{ss} - \theta_{ss}^{(k)}) - 2(\mathbf{X}^T\mathbf{X})_{ss} + 2 \cdot N \cdot w_{ss}(\Theta)
\end{aligned}$$

for $s = t$. These are clearly continuous and differentiable. Furthermore, inserting $\Theta = \Theta^{(k)}$ yields that the derivative is 0. Therefore, $f_{\Theta^{(k)}}$ is a first order approximation and our proof holds.

References

O. Banerjee, L. El Ghaoui, and A. d'Aspremont. Model selection through sparse maximum likelihood estimation. *Journal of Machine Learning Research*, 9:485–516, 2008.

J. Besag. Statistical analysis of non-lattice data. In *Proceedings of the Twenty-First National Conference on Artificial Intelligence (AAAI-06)*, volume 24, pages 179–195, 1975.

D. R. Cox and N. Reid. A note on pseudolikelihood constructed from marginal densities. *Biometrika*, 91:729–737, 2004.

J. Dahl, L. Vandenberghe, and V. Roychowdhury. Covariance selection for non-chordal graphs via chordal embedding. Optimization Methods and Software, 2008.

J. Friedman, T. Hastie, and R. Tibshirani. Sparse inverse covariance estimation with the graphical lasso. *Biostatistics*, 9:432–441, 2008a.

J. Friedman, T. Hastie, and R. Tibshirani. Regularization paths for generalized linear models via coordinate descent. *Technical Report, Stanford University*, 2008b. Submitted.

G. E. Hinton. Learning multiple layers of representation. *Trends in Cognitive Sciences*, 11:428–434, 2007.

S.-I. Lee, V. Ganapathi, and D. Koller. Efficient structure learning of Markov networks using L_1-regularization. In *Advances in Neural Information Processing Systems (NIPS 2006)*, 2006a.

S.-I. Lee, H. Lee, P. Abbeel, and A.Y. Ng. Efficient L_1 regularized logistic regression. In *Proceedings of the Twenty-First National Conference on Artificial Intelligence (AAAI-06)*, 2006b.

B. Lindsay. Composite likelihood methods. In *Contemporary Mathemtics*, volume 80, pages 221–239, 1998.

N. Meinshausen and P. Bühlmann. High dimensional graphs and variable selection with the lasso. *Annals of Statistics*, 34:1436–1462, 2006.

S. Perkins, K. Lacker, and j. Theiler. Grafting: Fast, incremental feature selection by gradient descent in function space. *Journal of Machine Learning Research*, 3:1333–1356, 2003.

M. J. Wainwright, P. Ravikumar, and J. Lafferty. High-dimensional graphical model selection using L_1-regularized logistic regression. In *Advances in Neural Information Processing Systems, Vancouver*, 2006.

M. J. Wainwright, P. Ravikumar, and J. Lafferty. High-dimensional graphical model selection using L_1-regularized logistic regression. Technical report, University of California, Berkeley, April 2008.

M. Yuan and Y. Lin. Model selection and estimation in the gaussian graphical model. *Biometrika*, 94(1):19–35, 2007.

Journal of Machine Learning Research 10 (2009) 907-929 Submitted 2/08; Revised 2/09; Published 4/09

Polynomial-Delay Enumeration of Monotonic Graph Classes

Jan Ramon JAN.RAMON@CS.KULEUVEN.BE
Siegfried Nijssen SIEGFRIED.NIJSSEN@CS.KULEUVEN.BE
K.U.Leuven, Dept. of Computer Science
Celestijnenlaan 200A, B-3001 Leuven

Editor: Stefan Wrobel

Abstract

Algorithms that list graphs such that no two listed graphs are isomorphic, are important building blocks of systems for mining and learning in graphs. Algorithms are already known that solve this problem efficiently for many classes of graphs of restricted topology, such as trees. In this article we introduce the concept of a dense augmentation schema, and introduce an algorithm that can be used to enumerate any class of graphs with polynomial delay, as long as the class of graphs can be described using a monotonic predicate operating on a dense augmentation schema. In practice this means that this is the first enumeration algorithm that can be applied theoretically efficiently in any frequent subgraph mining algorithm, and that this algorithm generalizes to situations beyond the standard frequent subgraph mining setting.

Keywords: graph mining, enumeration, monotonic graph classes

1. Introduction

Among the most prominent graph mining problems is the problem of finding frequent subgraphs in databases of small graphs of any topology. This is witnessed by the large number of algorithms that have been proposed for this task (Yan and Han, 2002; Borgelt and Berthold, 2002; Kuramochi and Karypis, 2004; Inokuchi et al., 2003; Inokuchi, 2004; Huan et al., 2003; Nijssen and Kok, 2004; Leskovec et al., 2006). A fundamental problem that is addressed in all these works is how to enumerate a set of graphs such that no two graphs in the enumerated set are isomorphic with each other. The main motivation for this focus is that if duplicates would not be avoided, these algorithms would access the data more often than necessary and produce results that are larger than required.

To avoid isomorphic graphs in their output, all these existing graph mining algorithms use a methodology based on canonical codes. A canonical code is a code that uniquely identifies a set of isomorphic graphs. To determine if a graph should be part of the output, its canonical code is computed, and, in some algorithms, compared with the canonical codes of graphs found before.

A fundamental problem with the canonical code based approach, however, is that we essentially need to solve the graph isomorphism problem: if we could compute the canonical code of any graph efficiently, we could compute the codes of two graphs to determine if they are isomorphic. The state of the art is that no polynomial algorithm for the graph isomorphism problem is known. Consequently, it can be shown that when the existing graph mining algorithms are enumerating candidate subgraphs, the *delay* between two enumerated graphs is exponential in the worst case (in terms of the size of the largest graph enumerated).

The contribution of this article is that we introduce a novel algorithm for enumerating graphs that does not use canonical codes, and incrementally maintains data structures that ensure that no two isomorphic graphs are listed. We show that in contrast to other algorithms for enumerating graphs, this new algorithm outputs many classes of graphs, including arbitrary connected graphs, with polynomial delay, which makes this algorithm theoretically more efficient than any other graph enumeration algorithm used in the graph mining literature.

It is important to note that our algorithm works for many classes of graphs. If we would restrict the topology of the graphs, for instance, to only those graphs that are trees, the enumeration problem of frequent graph mining is already known to be more efficiently solvable, and algorithms are known (Wright et al., 1986; Nakano and Uno, 2004) and used in practice (Chi et al., 2005; Horváth et al., 2006).

Even though the frequency constraint is the most popular constraint in the graph mining literature, other constraints have been studied as well. An important property of the frequency constraint is that it is monotonic w.r.t. to subgraph isomorphism: if a graph is frequent, all its subgraphs are also frequent. In our algorithm we exploit this property to maintain data structures incrementally. An interesting question is to what extent enumeration with polynomial delay is feasible when the graphs to enumerate are not monotonic under the subgraph isomorphism relation. To this aim, we developed the concept of an *augmentation schema*. The augmentation schema defines relations between graphs in the space of graphs to enumerate (in the simplest case, the subgraph isomorphism relation). We will show that enumeration with polynomial delay is possible as long as an augmentation schema satisfies certain conditions, and the graphs to enumerate can be specified using a monotonic predicate w.r.t. the augmentation schema. We will specify our algorithm in terms of such augmentation schemas. This makes our method general enough to be applied in settings beyond the traditional frequent subgraph mining setting, and allows us also to enumerate both connected and unconnected graphs. For instance, we can also enumerate hereditary classes of graphs with bounded degree; a class of graphs is called *hereditary* if it is monotonic under the *induced* subgraph relation, instead of the traditional subgraph relation.

The problem of graph enumeration has not only been studied in the graph mining literature. In particular, Goldberg showed in the early nineties that there is a polynomial delay algorithm to list all graphs (Goldberg, 1992). We will provide more details about this algorithm in Section 3, where we will show that this algorithm cannot be used to list graphs that satisfy a monotonic predicate, as required in a graph mining setting. Many algorithms exist for enumerating classes of graphs without taking into account isomorphisms, such as classes of graphs described by first order logic formulas (Goldberg, 1993) and edge-maximal graphs with bounded branchwidth (Paul et al., 2006); it is not known how to list these classes while taking into account isomorphisms. Heuristic implementations exist for enumerating graphs in general (McKay, 1998), but these do not guarantee polynomial delay.

Our algorithm uses similar ideas as the algorithm of Goldberg (1992). In particular, as our algorithm maintains a data structure incrementally, our algorithm requires that all computed subgraphs are stored. In the pattern mining setting, where we are interested in finding these graphs, this is a common assumption.

This article is the full version of a workshop abstract (Ramon and Nijssen, 2007). Compared to the workshop abstract, in this article (1) we show how our method extends to other classes of graphs than connected graphs and (2) we provide full details and proofs.

The article is organized as follows. In Section 2 we introduce the problem of subgraph mining. In Section 3 we show why the algorithm of Goldberg is too limited for applications in graph mining. In Section 4 we introduce the concept of augmentation schemas and formally define the enumeration problems that we are addressing. In Section 5 we state our results. In Section 6 we provide a short introduction to concepts in group theory which we need in Section 7, where we outline our algorithm; Section 8 concludes. The proofs of our claims are given in an appendix.

2. Motivation

The main motivation for our work is the problem of efficiently mining subgraphs under constraints. The most common such problem is the problem of mining frequent subgraphs in a database of small graphs. We will first give a formal definition of this problem.

A graph g is a tuple (V,E) where V is a set of vertices and $E \subseteq V \times V$ is a set of edges. We denote with $V(g)$ the set of vertices and with $E(g)$ the set of edges of a graph g. In this article we restrict ourselves to unlabeled, simple graphs (i.e., undirected, unweighted, no loops, no multiple edges between two nodes). It easy to lift these restrictions. In particular, in frequent subgraph mining it is usually assumed that graphs have labels. However, our discussion is simplified by assuming that we do not have labels; this is not a fundamental restriction of our methodology.

There are many ways in which one can restrict the topology of graphs. For instance, a *path* is a graph in which all nodes have degree 2, except two nodes, which have degree one. A *tree* is a connected graph with k nodes and $k-1$ edges. When we use the word *graph*, we refer to graphs that have no *apriori* restriction on their topology (except being unlabeled and simple).

Between two graphs we can define the graph isomorphism and the subgraph isomorphism relations. Our definitions are as usual in the literature: two graphs g_1 and g_2 are *isomorphic* iff there is a bijection $\varphi : V(g_1) \rightarrow V(g_2)$ such that $(v_1,v_2) \in E(g_1) \Leftrightarrow (\varphi(v_1),\varphi(v_2)) \in E(g_2)$. We denote this by $g_1 \simeq_\varphi g_2$, where φ is the bijection between the graphs. The bijection can be omitted if this is clear from the context.

A graph g_1 is *subgraph isomorphic* to g_2 iff there is a *subgraph* (V',E') with $V' \subseteq V(g_2)$ and $E' \subseteq E(g_2)$, such that g_1 is isomorphic with (V',E'). This is denoted with $g_1 \preceq_\varphi g_2$, where φ is the bijection between the nodes of g_1 and the subset of nodes of g_2. A subgraph (V',E') of g_2 is an *induced subgraph* if for all $v,v' \in V' : \{v,v'\} \in E(g_2) \leftrightarrow \{v,v'\} \in E'$; in other words, all edges in g_2 between nodes in V' are also present in E'. A graph g_1 is *induced subgraph isomorphic* to g_2 iff g_1 is isomorphic with an induced subgraph of g_2.

The graph isomorphism and *sub*graph isomorphism problems should not be confused with each other. The *sub*graph isomorphism problem is known to be NP complete, while the graph isomorphism problem is believed to be in a complexity class of its own. For both problems in general no polynomial algorithm is known (Köbler et al., 1993).

The problem of frequent subgraph mining can now be formalized as follows. Given is a database of graphs, $DB = \{g_1, g_2, \ldots, g_n\}$, and a threshold t. Then we are interested in finding all graphs g for which the support is higher than or equal to t. The *support* of a graph g is the number of graphs in DB with which g is subgraph isomorphic.

This frequent subgraph mining problem can be generalized by replacing the minimum frequency constraint with other predicates. For instance, a predicate could involve an additional maximum size constraint.

A predicate on graphs is called *monotonic* if all subgraphs of a graph that satisfies the predicate, will also all satisfy the predicate.[1] The support constraint and the maximum size constraint are examples of predicates that are monotonic under subgraph isomorphism.

The problem of constraint-based subgraph mining is closely related to the problem of frequent item set mining. Many algorithms have been developed to tackle the frequent item set mining problem, the most well-known being the APRIORI algorithm (Agrawal et al., 1996). Both frequent graph mining algorithms and frequent item set mining algorithms are considered to be constraint-based *pattern* mining algorithms. Constraint-based pattern mining algorithms look for patterns in a pattern language $\mathcal{L}$, and assume that these patterns are ordered using a partial order relation $\leq$ on $\mathcal{L}$. In the case of graph mining, $\leq$ is usually the subgraph isomorphism relation.

Many algorithms for pattern mining are level-wise (breadth-first) enumeration algorithms. These algorithms assume that the *size* of a pattern in the language is well-defined, and look for the patterns by listing them increasing in size. A high-level description of such an algorithm is given in Algorithm 1.

Algorithm 1 Level-Wise Pattern Miner

```
Require: A pattern language $\mathcal{L}$ and a monotonic predicate $p$
Ensure: output all $g \in \mathcal{L}$ with $p(g)$
1: $\mathcal{C}_1 \leftarrow$ patterns of size 1
2: $k \leftarrow 1$
3: while $\mathcal{C}_k \neq \emptyset$ do
4:     $\mathcal{F}_k \leftarrow \{g \in \mathcal{C}_k | p(g)\}$
5:     Generate $\mathcal{C}_{k+1}$ from $\mathcal{F}_k$
6:     $k \leftarrow k+1$
7: end while
8: Output $\bigcup_k \mathcal{F}_k$
```

In this algorithm, $\mathcal{F}_k$ contains the patterns of size k that satisfy the predicate. In line 4 it is determined which candidates of size k satisfy the predicate. In frequent pattern mining, this line requires access to the data, and can be most time consuming. It is therefore essential that $\mathcal{C}_k$ be as small as possible.

The main focus of this article is on the computation that needs to be performed in line 5. In this line new candidates should be generated. This generation should ensure the following:

- by repeatedly generating new candidates we should be able to enumerate all patterns in the pattern space, in our case the space of all unlabeled, simple graphs;
- to ensure that the algorithm is as efficient as possible, we should not insert two patterns in $\mathcal{C}_{k+1}$ that are equivalent with each other; in our case, we should avoid inserting two graphs that are isomorphic;
- we should not insert patterns in $\mathcal{C}_{k+1}$ for which we can know beforehand that p will not be true; in our case, we should exploit the monotonicity of p to avoid inserting graphs of which a subgraph is not included in $\mathcal{F}_k$.

1. We adopt here the terminology most common in graph theory. Some authors in the data mining literature use the term ‘anti-monotonic’.

In the graph mining setting, the second and third requirements are difficult, as the second requirement requires us to solve a graph isomorphism problem, and the third requirement involves a subgraph isomorphism problem.

Algorithm 1 has applications beyond traditional frequent subgraph mining. For instance, if we are interested in computing a decomposition graph kernel between two graphs which counts the number of *non-isomorphic* subgraphs that two graphs have in common, we could compute this kernel by providing algorithm 1 a database of two graphs as input and a threshold of $t = 2$. The size of the output is the desired kernel value.

Similarly, we could be interested in enumerating all *different* graphs that include one node in a network (Leskovec et al., 2006). In this case, the input of Algorithm 1 consists of one graph with all nodes up to a certain threshold distance from the node of interest, and the subgraph isomorphism should be restricted such that only bijections are considered in which at least one node in the pattern is mapped to the special node in the data.

In all cases, the essential problem of enumerating graphs without duplicates remains. Several algorithms have been proposed in the literature to address this graph enumeration problem. The main idea that has been employed, is that for every graph, we can compute a *canonical* code, that is, a code that is unique for all graphs that are isomorphic. The level-wise graph miners AGM (Inokuchi et al., 2003) and FSG (Kuramochi and Karypis, 2004) define a canonical code from adjacency matrices. Essentially, all subgraphs are stored in a data structure that is indexed according to this canonical code, and duplicates are avoided by computing for every candidate the canonical code. The approaches of AGM and FSG differ in their definition of *size*: AGM grows graphs by adding nodes, FSG by adding edges.

Other graph miners search depth-first, but their enumeration strategy can easily be modified for use in a level-wise algorithm (Yan and Han, 2002; Huan et al., 2003; Nijssen and Kok, 2004). Also these algorithms use a canonical code, but do not require the use of an indexed data structure. An algorithm for enumerating graphs surrounding a node in a network was proposed by Leskovec et al. (2006). Again, this algorithm used a canonical code.

Unfortunately, currently no polynomial algorithm is known to compute a canonical code; if one was known, we would be able to solve the graph isomorphism problem in polynomial time. Overall, this means that in all existing graph mining algorithms exponential time can be spent between two graphs that are inserted in the set of candidates.

Enumeration algorithms for which this is not the case, that is, algorithms that solve an enumeration problem such that between any two enumerated solutions polynomial time is spent (in terms of the largest enumerated solution), are known as algorithms with *polynomial delay*. To the best of our knowledge, all algorithms that have been proposed in the graph mining literature for enumerating graphs in general do not have polynomial delay. Only for restricted classes of graphs, such as trees and outerplanar graphs, algorithms with polynomial delay are known (Chi et al., 2005; Horváth et al., 2006).

However, the fact that graph isomorphism is not known to be polynomially computable, does not imply that graph enumeration cannot be solved with polynomial delay. Even though ignored in the data mining and machine learning literature, a polynomial algorithm for enumerating graphs does exist and was proposed by Goldberg (1992).

The problem with this algorithm is that it solves a rather simple enumeration problem: given a bound on the size of the graphs to enumerate, Goldberg's algorithm lists all graphs of this size with polynomial delay. In the case of data mining and machine learning, we are dealing with more complicated monotonic constraints that are data-dependent. We will show in the next section that we can create databases such that the set of graphs to enumerate does not fulfill the basic assumptions that need to be satisfied in Goldberg's algorithm. Even worse, we will see that this type of data is very common.

It should be stressed that this paper only studies the candidate generation of graph mining algorithms; it does not study the frequency evaluation. For general graphs the frequency evaluation also takes exponential time; a general frequent graph miner which uses our enumeration algorithm will still have exponential delay due to the fact that frequency evaluation is still exponential. This article proposes an improvement only of the candidate generation phase. The key insight is that we devised a graph enumeration algorithm which does not use canonical codes to perform this task.

3. Goldberg's Algorithm

In this section we briefly discuss the key points in the algorithm of Goldberg (1992), which shows why this algorithm cannot be used in a pattern mining setting.

Goldberg's algorithm aims at listing all graphs with n nodes, and makes a distinction between easy and hard graphs. Easy is a graph g that satisfies at least one of these two properties:

- g has a vertex with degree $n-1$, that is, at least one vertex is connected to all other vertices;
- g has only one vertex, say v, of maximum degree and $g-v$ is rigid, that is, the graph $g-v$ has only one isomorphism with itself (called the identity automorphism in Section 6).

An example of a graph that is never rigid, is a path.

Let $E(n)$ be the set of easy graphs with n nodes, and $U(n)$ the set of all graphs with n nodes, then it was shown by Goldberg that

$$2|E(n)| \geq |U(n)|.$$

This property implies that a large fraction of the graphs to enumerate are in fact easy. It was then shown that $E(n)$ can be listed with polynomial delay, and that $H(n) = U(n) \setminus E(n)$ can be listed in $O(n^4|U(n)|)$ time steps, where $|U(n)|$ is exponential in n but linear in the number of solutions. The main idea is then to interleave these two methods. The method which lists easy graphs, makes sure that the delay is polynomial. The other method is allowed to spend an exponential number of steps between consecutive graphs, but these steps are spread over several iterations of the method that lists easy graphs. Effectively this gives an algorithm with polynomial delay.

It is clear that this method fundamentally relies on the property that many graphs are 'easy'. This property does not hold for sets of graphs defined by a monotonic predicate. Let us illustrate this for the monotonic constraint that every node in a graph has a degree of at most three. If $n > 3$, it is easily seen that

- as the degree is at most 3, the number of graphs that contain a vertex that is connected to all other vertices is independent of n;
- every graph g that contains a single node v of maximum degree 3, consists, after removal of v, only of a set of (possibly unconnected) paths, hence $g-v$ is not rigid.

Consequently, $E(n)$ is a constant independent of n, while $U(n)$ grows with n. The condition of Goldberg's approach is therefore not satisfied for this class of graphs.

Moreover, to list all graphs in $H(n)$ in time $O(n^4|U(n)|)$, it is assumed that the average size of the automorphism groups of the elements of $U(n)$ is bounded. However, one can find subclasses for which this bound is not polynomial.

The most popular application of graph mining algorithms is in chemistry (Horváth et al., 2006). Most of the graphs in these databases have a degree bounded by four, and a majority of the subgraphs that need to be enumerated have a degree bounded by three. Thus, we do not believe that the conditions for Goldberg's method are satisfied in such data.

4. Problem Statement

Our problem setting has two parameters:

- an augmentation operator, which takes as input a graph, and outputs a set of augmentations of this graph, and whose closure, starting from a given set of graphs, describes a class of graphs;
- a predicate which restricts this class.

For instance, the augmentation operator can be used to describe the class of connected or unconnected graphs, while the boolean predicate can restrict this class further to those graphs that have bounded degree.

More formally, we will denote by $\mathcal{VE}$ the set that contains all pairs (r_V, r_E) where r_V is a set of vertices and r_E a set of edges (not necessarily between vertices in r_V). We will use set operators on elements of $\mathcal{VE}$ to denote the corresponding operations on their components, for example, $(r_V, r_E) \cup (r'_V, r'_E) = (r_V \cup r'_V, r_E \cup r'_E)$. Again, $V(r)$ and $E(r)$ describe the components of an element $r \in \mathcal{VE}$. An augmentation operator ρ^+ is a function that takes as input a graph, and outputs a set of descriptions of possible augmentations. This set is a subset of $\mathcal{VE}$. Every element $r \in \rho^+(g)$ describes a new graph $(V(g) \cup V(r), E(g) \cup E(r))$, abbreviated by $g + r$, that we call a *child* of g.

An example of an augmentation operator is

$$\rho_t^+(g) = \{(\{v_{new}\}, \{\{v, v_{new}\}\}) \mid v \in V(g)\};$$

where v_{new} is a new vertex (not belonging to $V(g)$). This operator adds a new vertex and connects it to an existing vertex. We can use this operator to describe the set of all (connected) trees. The minimal graph on which we apply the operator is in this case the graph with one node $\top_t = (\{v\}, \emptyset)$; in general, when we use one graph as the minimal element, we will denote this initial graph with $\top$.

The following operator enumerates all graphs:

$$\rho_a^+(g) = \{(\{v_{new}\}, \emptyset)\} \cup \{(\emptyset, \{\{v_1, v_2\}\}) | v_1, v_2 \in V(g) \wedge \{v_1, v_2\} \notin E(g)\}. \tag{1}$$

with $\top_a = (\emptyset, \emptyset)$, while the following allows for enumerating all connected graphs:

$$\rho_c^+(g) = \rho_t^+(g) \cup \{(\emptyset, \{\{v_1, v_2\}\}) | v_1, v_2 \in V(g) \wedge \{v_1, v_2\} \notin E(g)\}. \tag{2}$$

with $\top_c = (\{v\}, \emptyset)$.

As we can see in these examples, the vertices occurring in edges $E(r)$ do not have to occur in $V(r)$. Still it is useful to determine the entire set of vertices involved in an augmentation. For this we use the notation V^*, that is, $V^*(r) = V(r) \cup \{v \mid \exists e \in E(r) : v \in e\}$. Observe that the example operators output a number of augmentations that is bounded by a polynomial in the size of g, and that for each r, the size of the set $V^*(r)$ is bounded by a constant.

The class of graphs defined by taking the closure of the augmentation operator on the minimal element is denoted by $\mathcal{L}_{\rho^+}$ (which we shorten further to $\mathcal{L}_a$ and $\mathcal{L}_c$ for the classes defined by ρ_a^+ and ρ_c^+). The operator ρ^+ defines an ancestry relation between the graphs. This relation is a partial order.

The second parameter of our problem setting is a predicate p on graphs. The set of graphs g in $\mathcal{L}_{\rho^+}$ such that $p(g)$ is true is denoted by $\mathcal{L}_{\rho^+,p}$. We only consider predicates that cannot distinguish between isomorphic graphs, that is, if $g \simeq g'$ then $p(g) = p(g')$. We call a predicate *monotonic* w.r.t. an augmentation operator ρ^+ if for every graph $g \in \mathcal{L}_{\rho^+,p}$ it also holds that $g' \in \mathcal{L}_{\rho^+,p}$ for every g' that is an ancestor of g. For instance, the predicate that tests if a graph has bounded degree, is monotonic under ρ_a^+ as defined in (1).

In this article, we consider the following problem.

Problem 1 *Given are an augmentation operator ρ^+ and a predicate p which is monotonic w.r.t. ρ^+. Then, enumerate all elements in $\mathcal{L}_{\rho^+,p}$ such that exactly one representative of every equivalence class under isomorphism of $\mathcal{L}_{\rho^+,p}$ is enumerated.*

In the next section we determine a set of sufficient conditions on the augmentation operator and the monotonic predicate that have to be fulfilled in order to obtain an algorithm with polynomial delay.

5. Main Result

The augmentation operator that we introduced in the previous section, generates the children of a graph. Our algorithm relies on the existence of an operator which can inverse this operator. We call this operator a *reduction operator*. The reduction operator generates the *parents* of a graph.

Formally, the definition of a reduction operator is similar to that of an augmentation operator; the input of a reduction operator ρ^- is a single graph, its output consists of a subset of $\mathcal{VE}$. We call each element $r \in \rho^-(g)$ a reduction of g. It defines a graph $(V(g) \setminus V(r), E(g) \setminus E(r))$, which is abbreviated by $g - r$.

For instance, in the case of connected graphs, we can define the following reduction operator:

$$\begin{aligned}\rho_c^-(g) \quad = \quad & \{r \,|\, r = (\emptyset, \{\{v_1, v_2\}\}) \wedge \{v_1, v_2\} \in E(g) \wedge \{v_1, v_2\} \text{ is in a cycle}\} \\ & \{r \,|\, r = (\{v_1\}, \{\{v_1, v_2\}\}) \wedge \{v_1, v_2\} \in E(g) \wedge v_1 \text{ has degree } 1\} \end{aligned} \qquad (3)$$

Let $\mathcal{L}$ be a class of graphs. Then, an augmentation schema on $\mathcal{L}$ is a pair (ρ^+, ρ^-) of an augmentation operator ρ^+ and a reduction operator ρ^-, such that

- $\forall g \in \mathcal{L}, \forall r \in \rho^+(g) : g + r \in \mathcal{L} \wedge g \cap r = (\emptyset, \emptyset)$, that is, $\rho^+(g)$ contains augmentations that can be added to g to obtain a larger graph (child);
- $\forall g \in \mathcal{L}, \forall r \in \rho^-(g) : g - r \in \mathcal{L} \wedge r \subseteq g$, that is, $\rho^-(g)$ contains reductions that can be removed from g to obtain a parent;
- $\forall g \in \mathcal{L}, \forall r \in \rho^+(g) : r \in \rho^-(g + r)$, that is, the effects of the additions $r \in \rho^+(g)$ can be inverted by a deletion from $\rho^-(g + r)$;

- $\forall g \in \mathcal{L}, \forall r \in \rho^-(g) : \exists r' \in \rho^+(g-r), \exists \varphi : \big((g-r)+r' \simeq_\varphi g\big) \wedge \big(I_{g-r} \subseteq \varphi\big)$, that is, deletions $r \in \rho^-(g)$ can be inverted by additions from $\rho^+(g-r)$. Here $I_{g-r} = \{(v,v) \mid v \in V(g-r)\}$ is the identity permutation over the vertices of $g-r$;

- $\forall g_1, g_2 \in \mathcal{L} : g_1 \simeq_\varphi g_2 \Rightarrow \forall r \in \rho^+(g_1) : \varphi(r) \in \rho^+(g_2)$, that is, ρ^+ (and hence also $\rho^-(g)$) is invariant to isomorphisms.

Given a graph g and two reductions $r_1, r_2 \in \rho^-(g)$, we are interested in applying both r_1 and r_2 to g. However, sometimes this is not possible directly. Consider for instance the class of connected graphs $\mathcal{L}_c$, the graph $g = (\{1,2,3\},\{\{1,2\},\{2,3\},\{3,1\}\})$, and the reductions $r_1 = (\emptyset,\{\{1,2\}\})$ and $r_2 = (\emptyset,\{\{2,3\}\})$. We have $f_1 = g - r_1 = (\{1,2,3\},\{\{2,3\},\{3,1\}\})$. We cannot apply r_2 to f_1 as this would result in a graph which is not in $\mathcal{L}_c$ due to the isolated node 2; r_2 is not an allowed reduction in $g - r_1$. We can however map the reduction r_2 to a reduction that is allowed; instead of r_2 we use $(\{2\},\{\{2,3\}\})$, which is a valid deletion from f_1. This translation of r_2 to the context of $g - r_1$ for ρ_c^- is denoted by $r_2 \uparrow_c^g r_1$. More formally, for connected graphs $\mathcal{L}_c$, we define $r_2 \uparrow_c^g r_1$ to be equal to r_2, except in the case where $r_1 = (\emptyset,\{v,u_1\})$, $r_2 = (\emptyset,\{v,u_2\})$ and v has degree 2, in which case $r_2 \uparrow_c^g r_1 = r_2 \cup (\{v\},\emptyset)$. Then, $(g-r_1) - \big(r_2 \uparrow_c^g r_1\big) = (\{1,3\},\{\{1,3\}\})$.

We now more formally introduce the properties of these reduction translators. A reduction translator is an operator $\cdot \uparrow^{\cdot} \cdot$ mapping graphs g and reductions $r_1, r_2 \in \rho^-(g)$ to a new reduction $r_2 \uparrow^g r_1$ satisfying

$$\forall g, r_1, r_2 : r_2 \subseteq r_2 \uparrow^g r_1, \tag{4}$$

that is, after the translator is applied the reduction can only become larger,

$$\forall g, r_1, r_2 : r_1 \cap (r_2 \uparrow^g r_1) = (\emptyset,\emptyset), \tag{5}$$

that is, no element is removed twice, and

$$\forall g, r_1, r_2 : (r_2 \uparrow^g r_1) \cup r_1 = (r_1 \uparrow^g r_2) \cup r_2, \tag{6}$$

that is, reversing the order does not affect which vertices and edges are removed.

To allow for an efficient enumeration of graphs by our algorithm, we require that the augmentation schema of the class of graphs fulfills a *density* property. This property is based on the *graph of parents*. The graph of parents $GoP_{\rho^-,\uparrow}(g)$ of a graph g is defined as follows:

- $V(GoP_{\rho^-,\uparrow}(g)) = \rho^-(g)$, that is, every reduction for g corresponds to a vertex in its graph of parents.

- For any two $r_1, r_2 \in V(GoP_{\rho^-,\uparrow}(g))$, the graph $GoP_{\rho^-,\uparrow}(g)$ has edge $\{r_1, r_2\}$ iff $(r_1 \uparrow^g r_2) \in \rho^-(g - r_2)$ and $(r_2 \uparrow^g r_1) \in \rho^-(g - r_1)$.

The intuition is that there is an edge between two nodes in the graph of parents, iff it is possible to go from one parent to the other by first applying a reduction, and then applying an augmentation. An example is shown in Figure 1.

We say that $(\rho^+, \rho^-, \uparrow)$ is a dense augmentation schema for $\mathcal{L}$ iff

1. (ρ^+, ρ^-) is an augmentation schema for $\mathcal{L}$ and $\uparrow$ is a reduction translator.

2. for every non-minimal graph $g \in \mathcal{L}$ it holds that either the graph of parents $GoP_{\rho^-,\uparrow}(g)$ is connected, or $g - r$ is a minimal element of $\mathcal{L}$ for all $r \in \rho^-(g)$ (in most cases, $g - r = \top$).

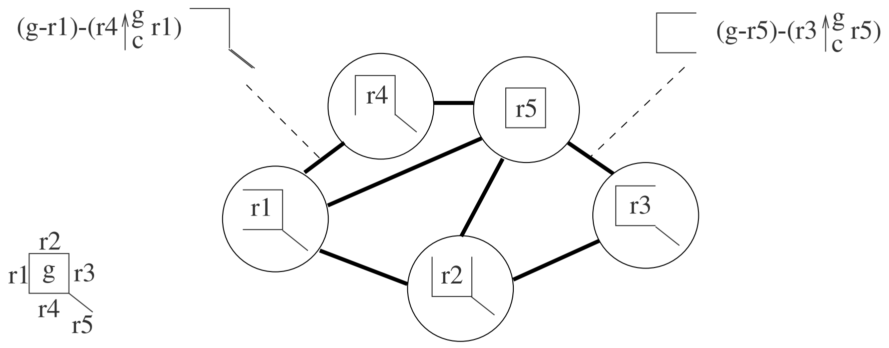

(g-r1)-(r4 $\uparrow^{g}_{c}$ r1)
(g-r5)-(r3 $\uparrow^{g}_{c}$ r5)
r4
r5
r1
r3
r2
r2
r1
g
r3
r4
r5

Figure 1: On the bottom-left, a connected graph g is drawn. In the middle, there is the graph of parents $GoP_{\rho_c^-,\uparrow_c}(g)$. For every of its vertices r_i, the corresponding $g - r_i$ of g is drawn. For the edges $\{r_1,r_4\}$ and $\{r_3,r_5\}$, the common (grand)parent is drawn. Note that there is no edge between for example r_3 and r_4 as $(g-r_3)-(r_4 \uparrow^g_c r_3)$ is not a connected graph.

Let us return to the examples of graph classes. For the class of all graphs $\mathcal{L}_a$, we define $r_1 \uparrow^g_a r_2 = r_1$. It is easy to show that $GoP_{\rho_a^-,\uparrow_a}(g)$ is a clique and that $(\rho_a^+,\rho_a^-,\uparrow_a)$ is a dense augmentation schema. We can also prove for connected graphs that $(\rho_c^+,\rho_c^-,\uparrow_c)$ is a dense augmentation schema (see the appendix for a proof).

The main result of this paper is the following:

Theorem 1 *Given an augmentation operator ρ^+ and a predicate p. We can solve problem 1 with polynomial delay if the following conditions are satisfied:*

- *there exists a reduction operator ρ^- and a reduction translator $\uparrow$ such that $(\rho^+,\rho^-,\uparrow)$ is a dense augmentation schema;*
- *the predicate p is monotonic w.r.t. ρ^+ over $\mathcal{L}_{\rho^+}$;*
- *ρ^+, ρ^- and p can be evaluated in polynomial time in their arguments;*
- *the number of vertices in $V^*(r)$ for every possible r resulting from ρ^+ and ρ^- is bounded by a constant.*

There are many classes of graphs for which these conditions are fulfilled. Below we give a non-exhaustive list of examples.

5.1 Monotonic Classes

Any predicate p that is monotonic w.r.t. edge and vertex deletion and that can be evaluated in polynomial time defines a class $\mathcal{L}_{\rho_a^+,p}$ that satisfies the conditions of Theorem 1, as $(\rho_a^+,\rho_a^-,\uparrow_a)$ is a dense augmentation schema, and all operations are polynomial. Hence, our algorithm can efficiently enumerate all these monotonic classes.

An interesting special case are the minor-closed classes of graphs. The minors of a graph can be obtained by deleting edges, vertices, or identifying adjacent vertices into a single new vertex that is adjacent to all vertices that were adjacent to any of the identified original vertices. A class is minor-closed if for any graph g all its minors are also included in the class. Some classes of graphs can be characterized by forbidden minors, that is, minors that none of the graphs are allowed to have. One example is the class of all planar graphs (the forbidden minors being the 5-clique K_5 and the complete bipartite graph $K_{3,3}$). It can be determined in polynomial time if a graph contains a given forbidden minor, and minor-closed classes of graphs are monotonic for ρ_a^+.

5.2 Connected Graphs

It is proven in the appendix that the augmentation schema $(\rho_c^+, \rho_c^-, \uparrow_c)$ is dense and its operators are polynomial. It follows that connected graphs can be enumerated with polynomial delay. Any predicate which is closed under edge and vertex deletion is also monotonic w.r.t. ρ_c^+. Therefore, for any monotonic predicate p, our algorithm can list all connected graphs satisfying p with polynomial delay.

5.3 Hereditary Classes with Bounded Degree

A predicate p is hereditary iff for every graph g_1 such that $p(g_1)$ holds, $p(g_2)$ holds for any *induced* subgraph g_2 of g_1. As pointed out in Section 2, an *induced* subgraph must contain all edges between a selected set of nodes in the original graph. A hereditary predicate is monotonic w.r.t. the following augmentation operator:

$$\rho_h^+(g) = \left\{ \left(\{v_{new}\}, \{\{v_{new}, v'\} | v' \in V\} \right) \;\middle|\; V \subseteq V(g) \right\},$$

where again v_{new} is a new vertex not in $V(g)$. The corresponding reduction operator should remove every single vertex as well as all edges emanating from it. It can be shown that the resulting augmentation schema is dense. However, the augmentation operator ρ_h^+ outputs a number of augmentations exponential in the size of the input. If we restrict ourselves to graphs with bounded degree, however, we can enumerate the resulting class with polynomial delay.

One example is the class of claw-free graphs. A graph is called claw-free if the graph

$$(\{v_1, v_2, v_3, v_4\}, \{\{v_1, v_2\}, \{v_1, v_3\}, \{v_1, v_4\}\})$$

is not one of its induced subgraphs. Claw-freeness is not a monotonic property in $\mathcal{L}_{\rho_a^+}$, as a subgraph of a graph without claws can contain a claw: for instance, consider the clique K_5; this graph does not have a claw as induced subgraph (all its induced subgraphs also being cliques), but many of its (ordinary) subgraphs are claws.

In general, claw-freeness is a hereditary property, as every induced subgraph of a graph without claws, is claw-free. To the best of our knowledge no polynomial algorithm is currently known for enumerating all claw-free graphs even if we do allow for duplicates (Goldberg, 1992). Our algorithm partially solves this problem by allowing for listing all claw-free graphs with bounded degree.

6. Automorphism Groups and Bases

In order to avoid enumerating duplicates, we will use some theory on automorphism groups. In this section, we will briefly review the necessary concepts.

An automorphism of a graph g is an isomorphism between g and itself. We will denote the identity automorphism with I_g. In Figure 2, a graph g^{ex} with 8 vertices is shown, together with

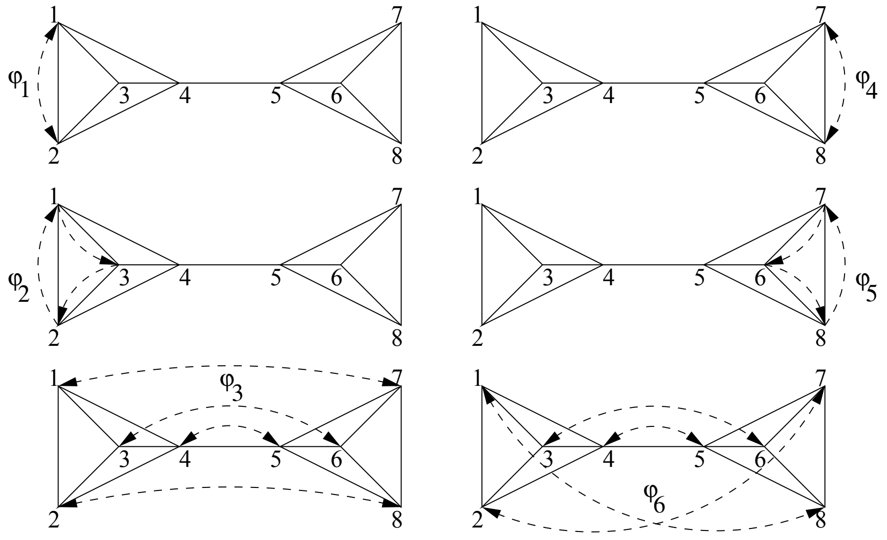


Figure 2: A graph g^{ex} and 6 of its automorphisms.

six of its automorphisms. An automorphism is a permutation of the vertices of g. The set of all automorphisms of g equipped with composition of permutations forms a permutation group acting on $V(g)$. This group is called the automorphism group of g, which we denote with $\mathcal{A}ut(g)$.

Let P be a permutation group and let $S \subseteq P$. We say S is a set of generators of P iff every element of P can be written as a composition of elements of S. We denote this fact with $P = < S >$. For example, in Figure 2, $\{\varphi_1, \varphi_2, \varphi_3, \varphi_5\}$ is a (non-minimal) set of generators of $\mathcal{A}ut(g^{ex})$. Automorphism φ_4 can be composed by $\varphi_4 = \varphi_3 \circ \varphi_1 \circ \varphi_3$. φ_6 can be composed as $\varphi_6 = \varphi_1 \circ \varphi_3 \circ \varphi_1$.

Let P be a permutation group acting on V and let $v \in V$. The stabilizer of v in P is the subgroup $P_v = \{\varphi \in P \mid \varphi(v) = v\}$. Consider for example the automorphism group $P = \mathcal{A}ut(C_5)$ on the 5-cycle graph $C_5 = (\{v_1, v_2, v_3, v_4, v_5\}, \{\{v_1, v_2\}, \{v_2, v_3\}, \{v_3, v_4\}, \{v_4, v_5\}\})$. This group has 10 elements, each of which can be decomposed by (possibly) a mirror permutation $\{(v_1, v_1), (v_2, v_5), (v_5, v_2), (v_3, v_4), (v_4, v_3)\}$ and rotations (0, 1 or more applications of $\{(v_1, v_2), (v_2, v_3), (v_3, v_4), (v_4, v_5), (v_5, v_1)\}$). For any vertex $v \in C_5$, the stabilizer P_v contains precisely 2 elements. E.g., P_{v_3} contains the identity permutation and the mirror $\{(v_3, v_3), (v_4, v_2), (v_2, v_4), (v_1, v_5), (v_5, v_1)\}$

One can apply the definition of stabilizers recursively; we will denote with $P_{v_1,\ldots,v_k}$ the stabilizer of v_k in $P_{v_1,\ldots,v_{k-1}}$. A sequence of points $B = [v_1 \ldots v_n]$ of V is called a *base* for permutation group P iff $P_{v_1,\ldots,v_{n-1}}$ only contains the identity permutation. For example, in Figure 2, $B_1 = [3,1,6,8]$ is a base for g^{ex}. Indeed, only the identity automorphism $I_{g^{ex}}$ leaves all four vertices 1, 3, 6 and 8 fixed. Also $B_2 = [3,1,6,8,2,4,5,7]$ is a base for g^{ex}.

Let P be a permutation group, B a base of P and $S \subseteq P$. S is called a strong generating set related to $B = \{v_1, \ldots, v_k\}$ if it contains generators for all permutations in $P_{v_1,\ldots,v_l}$ for $1 \leq l \leq k$. We will use the abbreviation BSGS for a pair (B, S) where B is a base and S is a strong generating set related to B.

Consider the base $B_1 = [3,1,6,8]$ for the group $\mathcal{A}ut(g^{ex})$ in Figure 2. We can construct a strong generating set by first choosing generators for $\mathcal{A}ut(g^{ex})_{3,1,6,8}$, then choosing generators for $\mathcal{A}ut(g^{ex})_{3,1}$, $\mathcal{A}ut(g^{ex})_{3,1,6}$ and so on until we have generators for $\mathcal{A}ut(g^{ex})$. Each time, we use the generators of the subgroup and extend them to a set of generators for the larger group. In our example, $\{\varphi_4\}$ is a set of generators for $\mathcal{A}ut(g^{ex})_{3,1,6}$, Next, $\{\varphi_4, \varphi_5\}$ is a set of generators for $\mathcal{A}ut(g^{ex})_{3,1}$ and $\{\varphi_4, \varphi_5, \varphi_1\}$ is a set of generators for $\mathcal{A}ut(g^{ex})_3$. Finally, $\{\varphi_4, \varphi_5, \varphi_1, \varphi_3, \varphi_2\}$ is a set of generators for $\mathcal{A}ut(g^{ex})$ and therefore a strong generating set for B.

A BSGS can represent a permutation group acting on n elements using only $O(n\log(n))$ generators. For example, even though in Figure 2, the automorphism group $\mathcal{A}ut(g^{ex})$ contains $2*3*2*3*2 = 72$ elements, only 5 generators are needed to represent it. Moreover, an $O(n^5)$ algorithm exists to transform a BSGS into a BSGS with a different base (Butler, 1991). In Figure 2, consider the BSGS $([3,1,6,8], \{\varphi_4, \varphi_5, \varphi_1, \varphi_3, \varphi_2\})$ and suppose we want a strongly generating set for the base $[3,1,8,6]$. Then, we have to combine the generators of $\mathcal{A}ut(g^{ex})_{3,1,6}$ and $\mathcal{A}ut(g^{ex})_{3,1}$. One possible strongly generating set is $\{\varphi_4 \circ \varphi_5, \varphi_5, \varphi_1, \varphi_3, \varphi_2\}$.

In general, it is easy to see that we can reduce any strong generating set for a permutation group on n elements to at most $n(n-1)/2$ elements. Let $B = \{v_1, \ldots, v_k\}$ be a base ($k \leq n$), and let S_i ($1 \leq i \leq k$) be the subset of the strong generating set containing all generators fixing v_j for all $1 \leq j < i$ but mapping v_i on a different element. Now for all i from 1 to k we can do the following. As long as any S_i has more than $n-i$ elements, there are two permutations $p_1, p_2 \in S_i$ such that $p_1(v_i) = p_2(v_i)$, and we can remove p_2 from S_i and add $p_2 \circ p_1^{-1}$ to S_{i+1}. The permutation group generated by the new strong generating set $\cup_i S_i$ remains the same, as any permutation requiring p_2 in its construction can still be constructed with $p_2 \circ p_1^{-1} \circ p_1$. After performing such replacements until all S_i contain at most $n-i$ elements, we have a strong generating set of size at most $n(n-1)/2$.

An important property is that given a strong generating set for some base, one can efficiently compute a strong generating set for another base. A basic step in such base change is to interchange two vertices, that is, given a strong generating set for $B = \{v_1 \ldots v_n\}$, find a strong generating set for $B' = \{v_1 \ldots v_{l-1}, v_{l+1}, v_l, v_{l+2} \ldots v_n\}$ for some l with $1 \leq l < n$. Let S be a strong generating set with $S = \cup_{i=1}^n S_i$ where S_i contains the generators fixing v_j ($1 \leq j < i$) and not fixing v_i. The only part of S that should be changed when swapping v_l and v_{l+1} in the base are S_l and S_{l+1}. Let $S' = S'_l \cup S'_{l+1} \cup \bigcup_{i \notin \{l,l+1\}} S_i$ be the strong generating set for the new base B', where S'_l fixes v_i ($1 \leq i < l$) and S'_{l+1} fixes v_i ($1 \leq i < l$) and v_{l+1}. One can construct S'_i by ensuring it contains permutations that map v_{i+1} on all its possible images. These images can be found with a so-called reachability graph. This means one starts with a set of possible images $I = \{v_{i+1}\}$, and as long as there is a permutation in $\cup_{i=l}^n S_i$ which maps an element $v \in I$ on an element $v' \notin I$, one adds v' to I; in the mean time, for each element v' one maintains a corresponding permutation. After constructing S'_l, one can construct S'_{l+1} by starting with $S'_{l+1} = S_l \cup S_{l+1}$ and then removing any permutations from it that are redundant. In this way, it is possible to find in polynomial time a strong generating set for a base in which two vertices have been swapped. By iterating this procedure, it is possible to efficiently find a strong generating set for any new base. Note that we here only described one naive strategy. In the literature, much more advanced algorithms have been proposed, which allow to perform these operations significantly more efficiently.

7. Algorithm Outline

Our algorithm enumerates the graph ordered by size, that is, no graph will be output before all its ancestors under ρ^+ are listed. Indeed, we can show that if $(\rho^+, \rho^-, \uparrow)$ is a dense augmentation schema, every graph has a unique size, that is, there is a function *size* that maps every graph to an integer such that for every g in $\mathcal{L}_{\rho^+}$ it holds that $r \in \rho^+(g)$ implies that $size(g+r) = size(g) + 1$. This results allows us to order the graphs that we need to enumerate level-wise.

Superficially, the idea is then as follows. We maintain graphs that we are enumerating in a queue. Initially, this queue contains the graph $\top$. Repeatedly, we pop a graph from the queue, apply the augmentation operator, and those children which are not equivalent to any graph in the queue, are pushed in the queue. Due to the level-wise enumeration, equivalent graphs must be in the queue. We need to address two issues:

- how do we avoid that we insert a child which is equivalent to a child of another graph?
- how do we avoid that we insert two children of the same graph that are equivalent with each other?

To make these computations possible in polynomial time, we also keep all parents of the graphs in the queue in memory. For each graph, both those in the queue and their parents, we store the following information:

- a representation for the graph g
- for each augmentation (and reduction) r of g, we store r together with an isomorphism mapping $\varphi = aug(g,r)$ (resp. $\varphi = red(g,r)$) such that $\varphi(g+r)$ (resp. $\varphi(g-r)$) is the stored representative of the isomorphism class of $g+r$ (resp. $g-r$). Until we assign the $aug(g,r)$ and $red(g,r)$ variables with a value, we will assume them initialized $aug(g,r) =' ?'$ and $red(g,r) =' ?'$. If $p(g+r)$ is false, with p the monotonic predicate of interest, we will assign $aug(g,r)$ the value nil.
- a base and strong generating set (BSGS) $bsgs(g)$ of the automorphism group $\mathcal{A}ut(g)$.

We output a graph when we pop it from the queue. Furthermore, we determine the BSGS at that point. The BSGS allows us to compute for each pair of augmentations of a graph if they result in equivalent graphs, and thus allows us to avoid two equivalent children of the same graph from being pushed in the queue.

To avoid that two different graphs insert children in the queue that are equivalent, we make sure that the first parent of a child marks at least one augmentation in each of the other parents of the child. When this alternative parent is popped from the queue later, it can use this information to avoid pushing this child and all its equivalent children.

To prove that this procedure is polynomial, we need to show that we can compute the BSGS in polynomial time, and that we can find all parents of a child in polynomial time. Let us start with this second point.

One of the conditions of Theorem 1 is that the augmentation schema is dense, that is, that the graph of parents is connected; hence we can compute a spanning tree for the graph of parents. If we create a child, we know the parent that generated this child; by traversing the spanning tree starting from the reduction achieving this parent, we can traverse all possible reductions. At the same time, for every step we take in this spanning tree, we can determine a corresponding stored parent: every step in the graph of parents corresponds to a reduction followed by an augmentation, for which we have stored associated permutations that point to stored representatives.

To compute the BSGS the key observation is that we can incrementally compute the BSGS of a graph from the BSGS of one of its parents, in a similar way to the algorithm of Goldberg (1992). Given a child and its parent, we perform a base change for the parent, such that we obtain permutations in which the vertices contained in the augmentation are stabilized. This gives generators for all vertices in the child, except those contained in the augmentation. By traversing all parents (as computed when the child was created), we can determine permutations and images for these vertices as well. The resulting set of generators can be reduced to a BSGS in polynomial time.

Details of this algorithm, including optimizations and proofs of complexity, can be found in the appendix.

8. Conclusions

We introduced an algorithm for listing graphs that makes sure that no two equivalent graphs are being output. We showed that for a well-defined set of conditions on a class of graphs to enumerate, this algorithm is correct and achieves polynomial delay. Classes of graphs that can be enumerated with low run-time complexity are connected graphs, planar graphs, minor closed graphs, monotonic classes of graphs in general, and hereditary classes with bounded degree. To the best of our knowledge, this is the first algorithm to be general enough to be able to list this range of classes efficiently, and for several of these classes no polynomial delay algorithm was presented before. In the appendix, we show that our algorithm runs with delay $O(n^5)$ for the class of all graphs, which is an improvement over the known method of Goldberg (1992), which achieves a delay of $O(n^6)$, where n is the number of vertices in the largest graph that is listed.

Most pattern mining algorithms consist of a candidate generation part and an interestingness evaluation part. This work contributes to the theory of pattern mining by providing a polynomial-delay algorithm for the first of these two common tasks. Also, our algorithm can be used as a generic candidate pattern generator for a wide range of algorithms for mining structured patterns, avoiding the need to research specialized canonical forms and enumeration strategies.

In contrast to other graph pattern mining systems, our algorithm provides at the same time a data structure in which one can look up a pattern in polynomial time. Indeed: given a pattern g, one can construct g from the empty graph by a sequence of augmentations, and then follow the augmentation pointers through the data structure. Efficient lookup could be very useful when the set of patterns resulting from a pattern mining step is queried by the user or by algorithms taking this set of patterns as input.

Even though its run-time complexity is favorable, the space complexity of our algorithm is an issue. The space required is polynomial in the size of the output; given that the number of listed graphs can be exponential, the storage requirements for large classes of graphs can be exponential. However, in those applications where the listed graphs need to be stored anyway, such as applications in data analysis, this drawback is of minor concern. Moreover, we know of no other general approaches that obtain a better space complexity.

As future work, we conjecture that the run-time complexity of our algorithm can be reduced further, at least to a delay of $O(n^4)$ for the class of all graphs and the class of connected graphs, through the definition of a canonical representation over the graphs. Furthermore, we hope to reduce the space requirements of our algorithm in problem settings where not all listed graphs are required to be stored, and plan to implement it in concrete data mining systems. Finally, relieving the constraint of bounded degree for hereditary classes remains an interesting problem.

Acknowledgments

Jan Ramon and Siegfried Nijssen are post-doctoral fellows of the Fund for Scientific Research of Flanders (FWO-Vlaanderen). Siegfried Nijssen was also supported by the European Commission under the 6th Framework Programme, project "Inductive Querying", contract number FP6-516169, and under the 7th Framework Programme FP7-ICT-2007-C FET-Open, project "Bison", contract number BISON-211898. We are grateful to Maurice Bruynooghe for proof-reading our final version.

Appendix A. Algorithmic Details and Proofs

In this appendix we provide details of our algorithm and proofs for our results.

A.1 Dense Augmentation Schemas

An important property of a dense augmentation schema is the following:

Lemma 2 *Let $\mathcal{L}$ be a class of graphs, and let $(\rho^+, \rho^-, \uparrow)$ be a dense augmentation schema for $\mathcal{L}$. Then, there exists a function size : $\mathcal{L} \rightarrow N$ such that $\forall g \in \mathcal{L}, \forall r \in \rho^+(g) : size(g+r) = size(g)+1$.*

Proof We will say that a graph g can be constructed from $\top$ in n steps if there exists a sequence $\top = g_0, g_1, \ldots, g_n = g$ such that $g_{i+1} = g_i + r_i$ for some $r_i \in \rho^+(g_i)$ for $i = 0 \ldots n-1$.

We first show that there is no graph g which both can be constructed from $\top$ in n_1 steps and can be constructed from $\top$ in n_2 steps for two distinct numbers n_1 and n_2.

Assume that such a graph g exists, and consider a minimal such graph g (according to the order induced by the augmentation schema). Then, there exists a parent $g - r_1'$ of g which can be constructed from $\top$ in $n_1 - 1$ steps, and a parent $g - r_2'$ of g which can be constructed from $\top$ in $n_2 - 1$ steps. As the graph of parents $GoP_{\rho^-,\uparrow}(g)$ of g is connected, there must exist two $r_1, r_2 \in \rho^-(g)$ such that (r_1, r_2) is an edge of $GoP_{\rho^-,\uparrow}(g)$, $g - r_1$ can be constructed from $\top$ in n_1' steps, $g - r_2$ can be constructed from $\top$ in n_2' steps and $n_1' \neq n_2'$. As (r_1, r_2) is an edge of $GoP_{\rho^-,\uparrow}(g)$, $(r_1 \uparrow^g r_2) \in \rho^-(g - r_2)$ and $(r_2 \uparrow^g r_1) \in \rho^-(g - r_1)$, there exists some graph p such that $p \simeq (g - r_2) - (r_1 \uparrow^g r_2)$ and $p \simeq (g - r_2) - (r_2 \uparrow^g r_1)$. Let n_p be the number of steps needed to construct p from $\top$. Then, remembering that augmentation schemas are isomorphism-invariant, we can conclude that both $g - r_1$ and $g - r_2$ can be constructed from $\top$ in $n_p + 1$ steps. Now as $n_1 \neq n_2$ either $n_1 - 1 \neq n_p + 1$ or $n_2 - 1 \neq n_p + 1$. Without loss of generality we can assume $n_1 \neq n_p + 1$. This means that $g - r_1$ can be constructed from $\top$ in both $n_p + 1$ and $n_1 - 1$ steps, which is a contradiction with the assumption that g is a minimal graph for which it holds that it can be constructed from $\top$ in two different numbers of steps.

Therefore, we conclude that no such graph exists. Hence, in order to obtain a function *size* satisfying the requirement, one can define $size(\top)$ to be 0 and for every g, $size(g)$ to be the number of steps in which g can be constructed from $\top$. ■

In Section 5 we stated the following.

Lemma 3 $(\rho_c^+, \rho_c^-, \uparrow_c)$ *is a dense augmentation schema.*

Proof We consider the different elements of the definition of a dense augmentation schema.

First, remember that (ρ_c^+, ρ_c^-) was defined by Equation (2) and (3) (see page 913 and 914). This is clearly an augmentation schema. Also, we defined $r_1 \uparrow_c^g r_2$ to be equal to r_1, except in the case where $r_1 = (\emptyset, \{v, u_1\})$, $r_2 = (\emptyset, \{v, u_2\})$ and v has degree 2. In that case $r_1 \uparrow_c^g r_2 = r_1 \cup (\{v\}, \{\})$. It is easy to see that $\uparrow_c$ satisfies Equations (4), (5) and (6), and hence is a reduction translator.

It remains to be shown that for every non-minimal graph $g \in \mathcal{L}_c$ it holds that either the graph of parents $GoP_{\rho_c^-, \uparrow_c}(g)$ is connected, or $g - r$ is the minimal element of $\mathcal{L}_c$ for all $r \in \rho^-(g)$.

Consider a connected graph g with at least 2 edges. We prove that $GoP_{\rho_c^-, \uparrow_c}(g)$ is connected. If g is a tree, then every reduction in $\rho^-(g)$ removes a leaf and the edge adjacent to it. By the definition of $\uparrow_c$, $r_1 \uparrow_c^g r_2 = r_1$ and $r_1 \in \rho_c^-(g - r_2)$ will hold for any distinct $r_1, r_2 \in \rho_c^-$. So if g is a tree, $GoP_{\rho_c^-, \uparrow_c}(g)$ is a clique.

If g is not a tree it contains at least one simple cycle C. We can partition $\rho_c^-(g)$ into two sets R_1 and R_2 such that R_1 contains all reductions removing an edge from the cycle C, and R_2 contains all other reductions. For any $r_1 \in R_1$ and $r_2 \in R_2$, it holds that removing r_2 from $g - r_1$ does not disconnect g and vice-versa. So such two r_1 and r_2 are adjacent in $GoP_{\rho_c^-, \uparrow_c}(g)$. Therefore, $GoP_{\rho_c^-, \uparrow_c}(g)$ is certainly connected if $\#R_2 \geq 1$. On the other hand, if $\#R_2 = 0$, g is a simple cycle. In that case, two reductions of R_1 are adjacent iff they remove two adjacent edges of the cycle. So in that case, $GoP_{\rho_c^-, \uparrow_c}(g)$ is a cycle and hence connected. ■

A.2 Algorithm

In this section we explain in more detail our algorithm. As stated in Section 7, for each graph, we store the following information:

- a representation for the graph g
- for each augmentation and reduction r of g, we store r together with an isomorphism mapping $aug(g,r)$ (resp. $red(g,r)$) that maps $g+r$ (resp. $g-r$), to the stored representative of their isomorphism class. In the algorithm, we use $aug(g,r)$ and $red(g,r)$ as functions that return the isomorphism. When the isomorphism is not yet computed, $aug(g,r)$ and $red(g,r)$ will return $'?'$. If $p(g+r)$ is false, $aug(g,r)$ returns the value *nil*.
- a base and strong generating set (BSGS) $bsgs(g)$ of the automorphism group $\mathcal{A}ut(g)$.

Algorithm 2 shows the high level algorithm. It repeatedly takes an unprocessed graph $g+r$ and processes it; $g+r$ must be minimal according to the *size* function from Lemma 2. A graph which has been processed remains in memory till it is no longer needed in the computation for one of its ancestors.

As shown by Algorithm 3, the processing of a graph g includes computing a BSGS and the automorphism group of g, computing reachability graphs of $\mathcal{A}ut(g)$ (line 3), finding isomorphic variants of children (lines 8-12), and examining all other children of g including constructing all $red(\cdot,\cdot)$ and some $aug(\cdot,\cdot)$ links (lines 13-24). We will describe each of the steps of the algorithm below.

Let us first consider what is known at the point PROCESS_GRAPH(g,r_0) is called. First, all graphs h for which $p(h)$ and $size(h) < size(g)$ have been processed and outputted, and the values for $aug(h,\cdot)$, $red(h,\cdot)$ and $bsgs(h)$ have been computed. Second, all graphs f for which $p(f)$ and

Algorithm 2 Highlevel algorithm

```
Require: a graph class $\mathcal{L}$, a monotonic predicate $p$ and a dense augmentation schema $(\rho^+, \rho^-, \uparrow)$
Ensure: output all $g \in \mathcal{L}$ with $p(g)$
1: $RefQueue \leftarrow \{(\top, (\emptyset, \emptyset))\}$
2: while $RefQueue \neq \emptyset$ do
3:     Let $(g, r) \in RefQueue$ such that $size(g + r)$ is minimal
4:     $RefQueue \leftarrow RefQueue \setminus \{(g, r)\}$
5:     PROCESS_GRAPH($g$,$r$)
6:     Output $g$
7: end while
```

$size(f) = size(g)$ have either entered $RefQueue$ or are fully processed. In any case, all values $red(f, \cdot)$ have been computed.

Let S_V denote the bound on the number of vertices and edges that can occur at most in any augmentation $r \in \rho^+(g)$ with $g \in \mathcal{L}$. Then we have the following lemma.

Lemma 4 *At line 2 of Algorithm 3 one can compute a BSGS for $\mathcal{Aut}(g)$ in time $O(|V(g)|^5 + |V(g)|^3 \cdot |\rho^-(g)| \cdot S_V!)$.*

Proof The case $g = \top$ is trivial, so we assume $g \neq \top$ and $r_0 \in \rho^-(g)$. A parent $g - r_0$ is known, and so is $bsgs(g - r_0)$ (as $size(g - r_0) = size(g) - 1$ and hence $g - r_0$ has been fully processed). By performing a base change, we can obtain a BSGS for the stabilizer $\mathcal{Aut}(g)_{V^*(r_0)}$ which fixes all elements in $V^*(r_0)$; please note that $V^*(r)$ contains all nodes involved in the reduction. Hence, we compute a base in which all nodes involved in the reduction are fixed. This is possible in time $O(|V(g)|^5)$. We can then extend the set of generators corresponding to this BSGS to a set of generators for $\mathcal{Aut}(g)$ by adding for every coset of $\mathcal{Aut}(g)_{V^*(r_0)}$ in $\mathcal{Aut}(g)$ one representative. Each such representative φ maps r_0 on some different $\varphi(r_0)$. Graphs $g - r_0$ and $g - \varphi(r_0)$ are isomorphic, which is reflected by $red(g, r_0)(g - r_0) = red(g, \varphi(r_0))(g - \varphi(r_0))$; here, $red(g, r_0)$ returns the stored permutation for reduction r_0 on graph g, which after application on the nodes and edges in graph $g - r_0$ yields the same graph as for the equivalent reduction $\varphi(r_0)$. One can therefore find all such representatives by iterating over all $r \in \rho^-(g)$, checking whether $red(g, r_0)(g - r_0) = red(g, r)(g - r)$ (all $red(g, \cdot)$ were computed earlier) and for all of these listing all possibilities to extend $red(g, r)^{-1} \circ aug(g, r_0)$ to an automorphism $(red(g, r)^{-1} \circ aug(g, r_0)) \cup \varphi_0$ (where $\varphi_0 : V(r) \to V(r_0)$). In total, the number of representatives of cosets of $\mathcal{Aut}(g)_{V^*(r_0)}$ is bounded by $|\rho^-(g)| \cdot S_V!$. One can eliminate the redundant automorphisms in this list in time $O(|V(g)|^3 \cdot |\rho^-(g)| \cdot S_V!)$. ■

Line 3 of Algorithm 3 computes reachability graphs Q_i, which are used in line 9 of the algorithm to determine if two augmentations yield isomorphic graphs. These graphs help to exploit the knowledge of the just computed BSGS $bsgs(g)$ of $\mathcal{Aut}(g)$. A reachability graph is computed for the number of nodes of graph g involved in the augmentation. In the case of (un)connected graphs, an augmentation involves either one or two nodes of graph g, and hence a reachability graph is used for $i = 1$ or $i = 2$. Q_i can be computed in time $O(i \cdot |V(g)|^i \cdot |bsgs(g)|)$. As for a reduced BSGS we have $|bsgs(g)| \leq |V(g)|^2$, line 3 can be performed in time $O(S_V \cdot |V(g)|^{S_V+2})$.

Algorithm 3 Processing one graph

```
1: procedure PROCESS_GRAPH($g$,$r_0$)
2:     $bsgs(g) \leftarrow$ Compute_BSGS$(g, r_0)$
3:     Ensure reachability graph $Q_i$ exists for $g$ where $i = |V^*(r_0) \cap V(g)|$,
       $V(Q_i) = (V(g))^i, E(Q_i) = \{(W, \varphi(W)) \mid W \in V(Q_i) \wedge \varphi \in bsgs(g)\}$
4:     $R_{iso+} \leftarrow \{r \in \rho^+(g) \mid aug(g,r) \neq '?'\}$
5:     $done \leftarrow$ FALSE
6:     while $not(done)$ do
7:         if $R_{iso+} \neq \emptyset$ then
8:             Let $r \in R_{iso+}$
9:             for all $r' \in \rho^+(g)$ s.t. $\exists\varphi : (\forall v \in V(g) : \varphi(v) \in V(g)) \wedge g + r \simeq_\varphi g + r'$ do
10:                $aug(g,r') \leftarrow aug(g,r) \circ \varphi^{-1}$
11:                $R_{iso+} \leftarrow R_{iso+} \setminus \{r'\}$
12:            end for
13:        else if $\exists r \in \rho^+(g) : aug(g,r) = '?'$ then
14:            $s = g + r$
15:            $(ok, s\_par\_list) \leftarrow$ SEARCH_PARENTS($g$,$r$,$s$)
16:            if $ok \wedge p(g)$ then
17:                for all $(r', \varphi') \in s\_par\_list$ do
18:                    $f \leftarrow \varphi'(s - r')$ ; $aug(f, \varphi'(r')) = \varphi'^{-1}$ ; $red(s,r') = \varphi'$
19:                $RefQueue = RefQueue \cup \{(s,r)\}$
20:            else
21:                for all $(r', \varphi') \in s\_par\_list$ do
22:                    $aug(\varphi'(s - r'), \varphi'(r')) \leftarrow nil$
23:            end if
24:            $R_{iso+} \leftarrow R_{iso+} \cup \{r\}$
25:        else
26:            $done \leftarrow$ TRUE
27:        end if
28:    end while
29: end procedure
```

It is possible that previous calls of PROCESS_GRAPH have assigned a value already to some of the $aug(g, \cdot)$, and line 4 collects these augmentations so that their isomorphic variants can be computed in lines 8-12.

Next, the while loop at line 6 runs until all $aug(g, \cdot)$ values have been computed. As soon as a new child is identified (either by previous calls to PROCESS_GRAPH (line 4) or by newly examined children (line 24) it is added to R_{iso+}, and in the next iteration all its isomorphic variants are computed. If R_{iso+} is empty, the $r \in \rho^+(g)$ for which $aug(g,r) = '?'$ are isomorphic to none of the r' for which $aug(g,r')$ has already been assigned a value and a new child is considered (lines 13-24).

We will first discuss the computation of isomorphic variants of an $r \in R_{iso+}$ in line 9. Recalling the definition, all r' are searched for which there is a mapping φ such that $(\forall v \in V(g) : \varphi(v) \in V(g)) \wedge g + r \simeq_\varphi g + r'$. One can do this as follows. First, use the reachability graph Q_i (with $i = |V^*(r) \cap V(g)|$) to find all possible images of $V^*(r) \cap V(g)$ under the automorphism group $\mathcal{A}ut(g)$.

Algorithm 4 search_parents

```
1: procedure SEARCH_PARENTS($g$, $r_0$, $s$)
2:     Construct a spanning tree $T$ for $GoP_{\rho^-,\uparrow}(s)$, the graph of parents of $s$
3:     $s\_par\_list \leftarrow \{(r_0, I_g)\}\}$
4:     Perform a depth-first search of $T$, starting at $r_0$
5:     for all edges $(r_1, r_2)$ visited during the depth first search do
6:         $\varphi \leftarrow$ GET_PARENT$(s, r_1, r_2)$
7:         if $\varphi = nil$ then return (FALSE,$s\_par\_list$)
8:         $s\_par\_list \leftarrow s\_par\_list \cup \{(r_2, \varphi)\}$
9:     end for
10:    return (TRUE,$s\_par\_list$)
11: end procedure
```

Algorithm 5 Get one parent

```
1: procedure GET_PARENT($s$,$r_1$,$r_2$)
2:     $\varphi_{f1} \leftarrow ext(s\_par\_list(s, r_1), I_s)$
3:     $f_1 \leftarrow \varphi_{f1}(s - r_1)$
4:     $\varphi_p \leftarrow ext(red(f_1, \varphi_{f1}(r_2 \uparrow^s r_1)), \varphi_{f1})$
5:     $p \leftarrow \varphi_p((s - r_1) - \varphi_{f1}(r_2 \uparrow^s r_1))$
6:     Let $r'_1 \in \rho^+(p)$ and $\varphi_* : V(\varphi_p(r_1 \uparrow^s r_2)) \rightarrow V(r'_1)$ such that $(I_p \cup \varphi_*)(\varphi_p(r_1 \uparrow^s r_2)) = r'_1$
7:     if $aug(p, r'_1) = nil$ then return $nil$
8:     else return $ext(aug(p, r'_1), (I_p \cup \varphi_*) \circ \varphi_p)$
9: end procedure
```

For each of these images of r, one can also compute one automorphism $\varphi_g \in \mathcal{A}ut(g)$ under which $\varphi_g(r)$ corresponds to the image, by following the edges of Q_i. Then, one can check for every $r' \in \rho^+(g)$ whether there is a mapping $\varphi_r : V(r) \rightarrow V(r')$ such that for $\varphi = \varphi_g \cup \varphi_r$ we have $\varphi(r) = r'$. Traversing Q_i and computing the automorphisms φ_g along the way is possible in time $O(|V(g)|^{i+1})$. As $i \leq S_V$ and we do this at most once for every $r \in \rho^+(g)$, the total time spent here is bounded by $O(|V(g)|^{S_V+1}|\rho^+(g)|)$.

Let us now consider the investigation of new children in lines 13-24 of Algorithm 3. For a particular child $s = g + r$, the algorithm first searches the parents $f = s - r'$ for all $r' \in \rho^-(s)$. As we explain below, if all parents of s satisfy the predicate p the algorithm is guaranteed to find all these parents. The $aug(f, r')$ variables have been assigned a value and depending on whether p holds for s itself, the $red(s, r')$ variables have been assigned a value and it is added to the data structures and to $RefQueue$.

Finding other parents of a proposed child $s = g + r$ is detailed in Algorithms 4 and 5. Before analysing this procedure and its consequences, we first explain some basic ideas. The key intuition that enables an efficient enumeration is that we can efficiently identify the (already enumerated) parents f_2 of the candidate graph s together with a suitable isomorphism mapping φ such that $s - r_2 \simeq_\varphi f_2$ for every reduction $r_2 \in \rho^-(s)$ by using the knowledge from Equation (6) that one can obtain $p = (s - r_1) - (r_2 \uparrow^s r_1)$ also by removing the parts in a different order: $p = (s - r_2) - (r_1 \uparrow^s r_2)$. Therefore, in Algorithm 5 we first remove some $r_1 \in \rho^-(s)$ to obtain a known parent f_1 of s, then go to a grand-parent p by removing a (translated) r_2 from f_1, and then go downwards

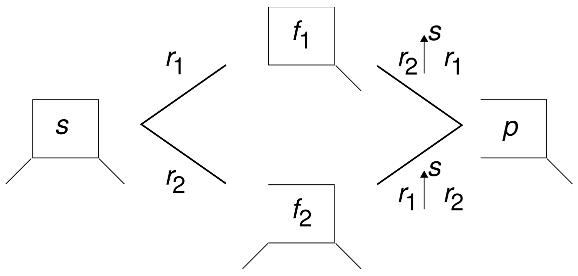


Figure 3: Searching for parents

again from p to the unknown parent f_2 (see Figure 3 for an illustration). In order to construct an isomorphism between $s - r_2$ and f_2, isomorphisms between $s - r_1$ and f_1, between $f_1 - (r_2 \uparrow^s r_1)$ and p and between $p + (r_1 \uparrow^s r_2)$ and f_2 are first extended to cover the full set of vertices of s (the $ext(\cdot,\cdot)$ function), and then composed (line 8).

In Algorithm 5 we have the following notation. Let φ_2 and φ_1 be bijections between sets of vertices. Then, $ext(\varphi_2, \varphi_1)$ is a bijection that maps any x for which $\varphi_1(x)$ is in the domain of φ_2 on $\varphi_2(\varphi_1(x))$ and maps any other x on a new vertex.

Now we return to the details of Algorithms 4 and 5. First, remember that for a dense augmentation schema, the graph of parents of a particular graph is connected. Therefore, it is possible in line 2 of Algorithms 4 to construct a spanning tree for it. Algorithm 4 attempts to construct in *s_par_list* a mapping from reductions $r \in \rho^-(s)$ to isomorphism mappings between the graphs $s - r$ and the representatives of their isomorphism class which were output before. We know already the isomorphism mapping between $s - r_0 = g$ and the representative of its isomorphism class, which is g itself (line 3). Now, if for some $r_1 \in \rho^-(s)$ we know an isomorphism mapping between $s - r_1$ and its representative f_1, and if for some other $r_2 \in \rho^-$, the graph $s - r_2$ satisfies p and has been listed earlier, then Algorithm 5 will provide us with an isomorphism mapping between $s - r_2$ and its representative f_2 as discussed above.

Then there are two possible cases. On the one hand, if there is a parent of s which did not fulfil the predicate p and hence was not listed earlier, we will not find that parent: line 7 of Algorithm 5 will detect this and return *nil*, which will cause also Algorithm 4 to return FALSE. On the other hand, if all parents of s fulfil predicate p, the search along the spanning tree will eventually provide an isomorphism mapping between s and the representatives of $s - r$ for all $r \in \rho^-(s)$.

The complexity of this search can be assessed as follows: Algorithm 5 is executed once for every $r' \in \rho^-(s)$ and contains operations on permutations that can be performed in time $O(|V(s)|)$. Assuming that $\uparrow$ can be performed efficiently, that a good data structure is built on $\rho^+(p)$ in order to be able to perform line 6 of Algorithm 5 efficiently, that $|\rho^-(s)|$ can be bounded by $O(|\rho^-(g)|)$ and that $|V(s)|$ can be bounded by $O(|V(g)|)$, we can therefore conclude that Algorithm 4 can be performed in time $O(|\rho^-(g)| \cdot |V(g)|)$. These assumptions are not very strong and hold for our two example augmentation schemas $(\rho_a^+, \rho_a^-, \uparrow_a)$ and $(\rho_c^+, \rho_c^-, \uparrow_c)$. Algorithm 4 is ran at most once for every $r \in \rho^+(g)$, for a total complexity of $O(|\rho^+(g)| \cdot |V(g)| \cdot |\rho^-(g)|)$.

In summary, this appendix has provided an informal proof of the following.

Theorem 5 *Under the assumptions stated in Theorem 1, Algorithm 2 correctly lists for every isomorphism class of graphs g for which $p(g) =$* TRUE *exactly one representative, and the time needed* *for outputting the next graph g is bounded by* $O(|V(g)|^5 + |V(g)|^3 \cdot |\rho^-(g)| \cdot S_V! + S_V \cdot |V(g)|^{S_V+2} + |V(g)|^{S_V+1}|\rho^+(g)| + |\rho^+(g)| \cdot |V(g)| \cdot |\rho^-(g)|)$.

This bound is polynomial for a constant S_V.

For example, for enumerating connected graphs with the schema $(\rho_c^+, \rho_c^-, \uparrow_c)$, $S_V = 2$. Therefore, this algorithm enumerates classes of connected graphs in time $O(|V(g)|^5)$ for each output graph g in the worst case. A similar result can be shown for enumerating (a monotonic subset of) all graphs with $(\rho_a^+, \rho_a^-, \uparrow_a)$.

Our conjecture is that this complexity can be improved further to $O(|V(g)|^4)$.

References

Rakesh Agrawal, Heikki Mannila, Ramakrishnan Srikant, Hannu Toivonen, and A. Inkeri Verkamo. Fast discovery of association rules. In *Advances in Knowledge Discovery and Data Mining*. AAAI/MIT Press, 1996.

Christian Borgelt and Michael R. Berthold. Mining molecular fragments: Finding relevant substructures of molecules. In *ICDM*, pages 51–58, 2002.

Gregory Butler. *Fundamental algorithms for permutation groups*, volume 559 of *Lecture Notes in Computer Science*. Springer-Verlag, 1991.

Yun Chi, Richard R. Muntz, Siegfried Nijssen, and Joost N. Kok. Frequent subtree mining - an overview. *Fundam. Inform.*, 66(1-2):161–198, 2005.

Leslie A. Goldberg. Efficient algorithms for listing unlabeled graphs. *Journal of Algorithms*, 13(1): 128–143, 1992.

Leslie A. Goldberg. Polynomial space polynomial delay algorithms for listing families of graphs. In *Proceedings of the Twenty-fifth Annual ACM Symposium on Theory of Computing (STOC'93)*, pages 218–225, New York, NY, USA, 1993. ACM Press.

Tamás Horváth, Jan Ramon, and Stefan Wrobel. Frequent subgraph mining in outerplanar graphs. In *Proceedings of the Twelfth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 197–206, Philadelphia, PA, August 2006.

Jun Huan, Wei Wang, and Jan Prins. Efficient mining of frequent subgraphs in the presence of isomorphism. In *ICDM*, pages 549–552. IEEE Computer Society, 2003.

Akihiro Inokuchi. Mining generalized substructures from a set of labeled graphs. In *ICDM*, pages 415–418. IEEE Computer Society, 2004.

Akihiro Inokuchi, Takashi Washio, and Hiroshi Motoda. Complete mining of frequent patterns from graphs: Mining graph data. *Machine Learning*, 50(3):321–354, 2003.

Johannes Köbler, Uwe Schöning, and Jacobo Torán. *The Graph Isomorphism Problem: Its Structural Complexity*. Birkhäuser, 1993.

Michihiro Kuramochi and George Karypis. An efficient algorithm for discovering frequent subgraphs. *IEEE Trans. Knowl. Data Eng.*, 16(9):1038–1051, 2004.

Jure Leskovec, Ajit Singh, and Jon M. Kleinberg. Patterns of influence in a recommendation network. In *Advances in Knowledge Discovery and Data Mining, 10th Pacific-Asia Conference (PAKDD)*, pages 380–389, 2006.

Brendan D. McKay. Isomorph-free exhaustive generation. *Journal of Algorithms*, 26:306–324, 1998.

Shin-ichi Nakano and Takeaki Uno. Constant time generation of trees with specified diameter. In Juraj Hromkovic, Manfred Nagl, and Bernhard Westfechtel, editors, *Proceedings of the 30th International Workshop on Graph Theoretical Concepts in Computer Science*, volume 3353 of *Lecture Notes in Computer Science*, pages 33–45. Springer-Verlag, 2004.

Siegfried Nijssen and Joost N. Kok. A quickstart in frequent structure mining can make a difference. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 647–652, 2004.

Christophe Paul, Andrzej Proskurowski, and Jan A. Telle. Generating graphs of bounded branchwidth. In *Proceedings of the 32nd International Workshop on Graph Theoretical Concepts in Computer Science*, volume 4271 of *Lecture Notes in Computer Science*, pages 206–216, 2006.

Jan Ramon and Siegfried Nijssen. General graph refinement with polynomial delay. In *Proceedings of the Workshop on Mining and Learning with Graphs (MLG'07)*, 2007.

Robert A. Wright, Bruce Richmond, Andrew Odlyzko, and Brendan D. McKay. Constant time generation of free trees. *SIAM Journal on Computing*, 15(2):540–548, 1986.

Xifeng Yan and Jiawei Han. gSpan: Graph-based substructure pattern mining. In *Proceedings of the 2002 IEEE International Conference on Data Mining (ICDM 2002)*, pages 721–724, Japan, 2002. IEEE Computer Society.

Journal of Machine Learning Research 10 (2009) 931-934 Submitted 9/08; Revised 1/09; Published 4/09

Java-ML: A Machine Learning Library

Thomas Abeel THOMAS.ABEEL@PSB.UGENT.BE
Yves Van de Peer YVES.VANDEPEER@PSB.UGENT.BE
Yvan Saeys YVAN.SAEYS@PSB.UGENT.BE
VIB Department of Plant Systems Biology
Ghent University
9000 Gent, Belgium

Editor: Sören Sonnenburg

Abstract

Java-ML is a collection of machine learning and data mining algorithms, which aims to be a readily usable and easily extensible API for both software developers and research scientists. The interfaces for each type of algorithm are kept simple and algorithms strictly follow their respective interface. Comparing different classifiers or clustering algorithms is therefore straightforward, and implementing new algorithms is also easy. The implementations of the algorithms are clearly written, properly documented and can thus be used as a reference. The library is written in Java and is available from http://java-ml.sourceforge.net/ under the GNU GPL license.

Keywords: open source, machine learning, data mining, java library, clustering, feature selection, classification

1. Introduction

Machine learning techniques are increasingly popular in research fields like bio- and chemo-informatics, text and web mining, as well as many other areas of research and industry. In this paper we present Java-ML: a cross-platform, open source machine learning library written in Java.

Several well-known data mining libraries already exist, including for example, Weka (Witten and Frank, 2005) and Yale/RapidMiner (Mierswa et al., 2006). These programs provide a user-friendly interface and are geared towards interactive use with the user. In contrast to these programs, Java-ML is oriented towards developers that want to use machine learning in their own programs. To this end, Java-ML interfaces are restricted to the essentials, and are very easy to understand. As a result, Java-ML facilitates a broad exploration of different models, is straightforward to integrate into your own source code, and can be easily extended.

Regarding the content of the library, Java-ML also has a different focus than the other libraries. Java-ML contains an extensive set of similarity based techniques, and offers state-of-the-art feature selection techniques. The large number of similarity functions allow for a broad set of clustering and instance based learning techniques, while the feature selection techniques are well suited to deal with high-dimensional domains, such as the ones often encountered in bioinformatics and biomedical applications.

Clustering
K-means-like (7)
Self organizing maps
Density based clustering (3)
Markov chain clustering
Cobweb
Cluster evaluation measures (15)

Classification
SVM (2)
Instance based learning (4)
Tree based methods (2)
Random Forests
Bagging

Feature selection
Entropy based methods (4)
Stepwise addition/removal (2)
SVM_RFE
Random forests
Ensemble feature selection

Data filters
Discretization
Normalization (2)
Missing values (3)
Instance manipulation (11)

Distance measures
Similarity measures (6)
Distance metrics (11)
Correlation measures (2)

Utilities
Cross-validation/evaluation
Data loading (ARFF and CSV)
Weka bridges (2)

Table 1: Overview of the main algorithms included in Java-ML. The number of algorithms for each category is shown in parentheses.

2. Description of the Library

In this section we first describe the software design of Java-ML, we then discuss how to integrate it in your program and finally we cover the documentation.

2.1 Structure of the Library

The library is built around two core interfaces: `Dataset` and `Instance`. These two interfaces have several implementations for different types of samples. The machine learning algorithms implement one of the following interfaces: `Clusterer`, `Classifier`, `FeatureScoring`, `FeatureRanking` or `FeatureSubsetSelection`. Distance, correlation and similarity measures implement the interface `DistanceMeasure`. These distance measures can be used in many algorithms to modify their behavior. Cluster evaluation measures are defined by the `ClusterEvaluation` interface. Manipulation filters either implement `InstanceFilter` or `DatasetFilter`, depending on the level they work on. All implementing classes for each of the interfaces are available from the API documentation that is available on the Java-ML website. Each of these interfaces provides one or two methods that are required to execute the algorithm on a particular data set. Several utility classes make it easy to load data from tab or comma separated files and from ARFF formatted files. An overview of the main algorithms included in Java-ML can be found in Table 1.

The library provides several algorithms that have not been made available before in a bundled form. In particular, clustering algorithms and the accompanying cluster evaluation measures are extensively represented. This includes the adaptive quality-based clustering algorithm, density based methods, self-organizing maps (both as clustering and classification algorithm) and numerous other well-known clustering algorithms. A large number of distance, similarity and correlation measures are included. Feature selection algorithms include traditional algorithms like symmetrical uncertainty, gain ratio, RELIEF, stepwise addition/removal, as well as a number of more recent methods (SVMRFE and random forest attribute evaluation). Also the recently introduced concept of ensemble feature selection techniques (Saeys et al., 2008) is incorporated in the library. We have also implemented a fast and simple random tree algorithm to cope with high dimensional, sparse and ambiguous data. Finally, we provide bridges for classification and clustering in Weka and libsvm (Fan et al., 2005).

2.2 Easy Integration in Your Own Source Code

Including Java-ML algorithms in your own source code is very simple. To illustrate this, we present here two short code fragments that demonstrate the ease to integrate the library. The following lines of code integrate a K-Means clustering algorithm in your own program.

```
Dataset data = FileHandler.loadDataset(new File("iris.data"), 4, ",");
Clusterer km = new KMeans();
Dataset[]clusters=km.cluster(data);
```

The first line uses the FileHandler utility to load data from the iris.data file. In this file, the class label is on the fourth position and the fields are separated by a comma. The second line constructs a new instance of the KMeans clustering algorithm with default values, in this case k=4. The third line uses the KMeans instance to cluster the data that we loaded in the first line. The resulting clusters will be returned as an array of data sets.

The following example illustrates how to perform a cross-validation experiment for a specific dataset and classifier.

```
Dataset data = FileHandler.loadDataset(new File("iris.data"), 4, ",");
Classifier knn = new KNearestNeighbors(5);
CrossValidation cv = new CrossValidation(knn);
Map<Object, PerformanceMeasure> p = cv.crossValidation(data);
```

First we load the iris data set, and construct a K-nearest neighbors classifier, which uses 5 neighbors to classify instances. In the next line, we initialize the cross-validation with our classifier. The last line runs the cross-validation on the loaded data. By default, a 10-fold cross validation will be performed. The result is returned in a map, which maps each class label to its corresponding PerformanceMeasure (`Map<Object,PerformanceMeasure>`). For classification problems, a performance measure is a wrapper around four values: (i) true positives, (ii) true negatives, (iii) false positives and (iv) false negatives. This class also provides a number of derivative measures such as accurracy, error rate, precision, recall and others. More advanced samples are available from the documentation pages on the Java-ML website.

2.3 Documentation

There are a number of resources for documentation about Java-ML. The source code itself is documented thoroughly, always up-to-date, and accessible from the web site through the API documentation. The web site additionally provides a number of tutorials with illustrated code samples for the most common tasks in Java-ML, covering the following topics: installing the library, introducing basic concepts, creating and loading data, creating algorithms and applying them to your data, and more advanced topics for people who would like to contribute to the library. Finally, all code samples as well as the PDF versions of the tutorials are also included in the Java-ML distribution itself.

3. Case Studies

The library described in this manuscript has been used in several studies. Here we highlight two applications which have been recently published.

Initially, the project focused on clustering algorithms and measures to evaluate the quality of a clustering. Our goal was to separate DNA sequences that are likely to contain a promoter (the controlling element of a gene) from other sequences, a well-known task in bioinformatics. The best results were obtained using a clustering algorithm based on self-organizing maps (Abeel et al., 2008).

More recently, the focus has shifted toward feature selection. More specifically, we are looking whether ensemble feature selection (combining different feature selectors) can improve the stability of feature selection in case of high-dimensional data sets with few samples. The improvements in stability were shown not to affect the prediction accuracy. This is ongoing research, but the first results are promising (Saeys et al., 2008).

Acknowledgments

We thank A. De Rijcke for his early contributions to Java-ML, as well as the anonymous reviewers for their valuable comments. TA is funded by IWT-Vlaanderen. YS would like to thank the Research Foundation-Flanders (FWO-Vlaanderen) for funding his research.

References

Thomas Abeel, Yvan Saeys, Pierre Rouzé, and Yves Van de Peer. ProSOM: Core promoter prediction based on unsupervised clustering of DNA physical profiles. *Bioinformatics*, 24(13):i24–i31, July 2008.

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

Ingo Mierswa, Michael Wurst, Ralf Klinkenberg, Martin Scholz, and Timm Euler. Yale: Rapid prototyping for complex data mining tasks. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-06)*, 2006.

Yvan Saeys, Thomas Abeel, and Yves Van de Peer. Robust feature selection using ensemble feature selection techniques. In *Proceedings of the ECML-PKDD conference 2008*, 2008.

Ian H. Witten and Eibe Frank. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, San Francisco, 2nd edition, 2005.

Journal of Machine Learning Research 10 (2009) 935-975 Submitted 8/08; Revised 1/09; Published 4/09

Nonextensive Information Theoretic Kernels on Measures*

André F. T. Martins† AFM@CS.CMU.EDU
Noah A. Smith NASMITH@CS.CMU.EDU
Eric P. Xing EPXING@CS.CMU.EDU
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA, USA

Pedro M. Q. Aguiar AGUIAR@ISR.IST.UTL.PT
Instituto de Sistemas e Robótica
Instituto Superior Técnico
Lisboa, Portugal

Mário A. T. Figueiredo MARIO.FIGUEIREDO@LX.IT.PT
Instituto de Telecomunicações
Instituto Superior Técnico
Lisboa, Portugal

Editor: Francis Bach

Abstract

Positive definite kernels on probability measures have been recently applied to classification problems involving text, images, and other types of structured data. Some of these kernels are related to classic information theoretic quantities, such as (Shannon's) mutual information and the Jensen-Shannon (JS) divergence. Meanwhile, there have been recent advances in nonextensive generalizations of Shannon's information theory. This paper bridges these two trends by introducing nonextensive information theoretic kernels on probability measures, based on new JS-type divergences. These new divergences result from extending the the two building blocks of the classical JS divergence: convexity and Shannon's entropy. The notion of convexity is extended to the wider concept of q-convexity, for which we prove a Jensen q-inequality. Based on this inequality, we introduce Jensen-Tsallis (JT) q-differences, a nonextensive generalization of the JS divergence, and define a k-th order JT q-difference between stochastic processes. We then define a new family of nonextensive mutual information kernels, which allow weights to be assigned to their arguments, and which includes the Boolean, JS, and linear kernels as particular cases. Nonextensive string kernels are also defined that generalize the p-spectrum kernel. We illustrate the performance of these kernels on text categorization tasks, in which documents are modeled both as bags of words and as sequences of characters.

Keywords: positive definite kernels, nonextensive information theory, Tsallis entropy, Jensen-Shannon divergence, string kernels

∗. Earlier versions of this work appeared in Martins et al. (2008a) and Martins et al. (2008b).
†. Also at Instituto de Telecomunicações, Instituto Superior Técnico, Lisboa, Portugal.

1. Introduction

In kernel-based machine learning (Schölkopf and Smola, 2002; Shawe-Taylor and Cristianini, 2004), there has been recent interest in defining kernels on probability distributions to tackle several problems involving structured data (Desobry et al., 2007; Moreno et al., 2004; Jebara et al., 2004; Hein and Bousquet, 2005; Lafferty and Lebanon, 2005; Cuturi et al., 2005). By defining a parametric family S containing the distributions from which the data points (in the input space X) are assumed to have been generated, and defining a map from X from S (e.g., via maximum likelihood estimation), a distribution in S may be fitted to each datum. Therefore, a kernel that is defined on $S \times S$ automatically induces a kernel on $X \times X$, through map composition. In text categorization, this framework appears as an alternative to the Euclidean geometry inherent to the usual bag-of-words representations. In fact, approaches that map data to statistical manifolds, equipped with well-motivated non-Euclidean metrics (Lafferty and Lebanon, 2005), often outperform support vector machine (SVM) classifiers with linear kernels (Joachims, 2002). Some of these kernels have a natural information theoretic interpretation, establishing a bridge between kernel methods and information theory (Cuturi et al., 2005; Hein and Bousquet, 2005).

The main goal of this paper is to widen that bridge; we do that by introducing a new class of kernels rooted in *nonextensive* information theory, which contains previous information theoretic kernels as particular elements. The Shannon and Rényi entropies (Shannon, 1948; Rényi, 1961) share the *extensivity* property: the joint entropy of a pair of independent random variables equals the sum of the individual entropies. Abandoning this property yields the so-called nonextensive entropies (Havrda and Charvát, 1967; Lindhard, 1974; Lindhard and Nielsen, 1971; Tsallis, 1988), which have raised great interest among physicists in modeling phenomena such as long-range interactions and multifractals, and in constructing nonextensive generalizations of Boltzmann-Gibbs statistical mechanics (Abe, 2006). Nonextensive entropies have also been recently used in signal/image processing (Li et al., 2006) and other areas (Gell-Mann and Tsallis, 2004). The so-called *Tsallis entropies* (Havrda and Charvát, 1967; Tsallis, 1988) form a parametric family of nonextensive entropies that includes the Shannon-Boltzmann-Gibbs entropy as a particular case. Nonextensive generalizations of information theory have been proposed (Furuichi, 2006).

Convexity and Jensen's inequality are key concepts underlying several central results of information theory, for example, the non-negativity of the *Kullback-Leibler (KL) divergence* (or *relative entropy*) (Kullback and Leibler, 1951). Jensen's inequality (Jensen, 1906) also underlies the *Jensen-Shannon (JS) divergence*, a symmetrized and smoothed version of the KL divergence (Lin and Wong, 1990; Lin, 1991), often used in statistics, machine learning, signal/image processing, and physics.

In this paper, we introduce new extensions of JS-type divergences by generalizing its two pillars: *convexity* and *Shannon's entropy*. These divergences are then used to define new information-theoretic kernels between probability distributions. More specifically, our main contributions are:

- The concept of *q-convexity*, generalizing that of convexity, for which we prove a *Jensen q-inequality*. The related concept of *Jensen q-differences*, which generalize Jensen differences, is also proposed. Based on these concepts, we introduce the *Jensen-Tsallis (JT) q-difference*, a nonextensive generalization of the JS divergence, which is also a "mutual information" in the sense of Furuichi (2006).

- Characterization of the JT q-difference, with respect to convexity and extrema, extending work by Burbea and Rao (1982) and by Lin (1991) for the JS divergence.

- Definition of k-th order joint and conditional JT q-differences for families of stochastic processes, and derivation of a chain rule.
- A broad family of (nonextensive information theoretic) positive definite kernels, interpretable as nonextensive mutual information kernels, ranging from the Boolean to the linear kernels, and including the JS kernel proposed by Hein and Bousquet (2005).
- A family of (nonextensive information theoretic) positive definite kernels between stochastic processes, subsuming well-known string kernels (e.g., the p-spectrum kernel) (Leslie et al., 2002).
- Extensions of results of Hein and Bousquet (2005) proving positive definiteness of kernels based on the unbalanced JS divergence. A connection between these new kernels and those studied by Fuglede (2005) and Hein and Bousquet (2005) is also established. In passing, we show that the parametrix approximation of the multinomial diffusion kernel introduced by Lafferty and Lebanon (2005) is *not* positive definite in general.

The paper is organized as follows. Section 2 reviews nonextensive entropies, with emphasis on the Tsallis case. Section 3 discusses Jensen differences and divergences. The concepts of q-differences and q-convexity are introduced in Section 4, where they are used to define and characterize some new divergence-type quantities. In Section 5, we define the Jensen-Tsallis q-difference and derive some of its properties; in that section, we also define k-th order Jensen-Tsallis q-differences for families of stochastic processes. The new family of entropic kernels is introduced and characterized in Section 6, which also introduces nonextensive kernels between stochastic processes. Experiments on text categorization are reported in Section 7. Section 8 concludes the paper and discusses future research.

2. Nonextensive Entropies and Tsallis Statistics

In this section, we start with a brief overview of nonextensive entropies. We then introduce the family of Tsallis entropies, and extend their domain to unnormalized measures.

2.1 Nonextensivity

In what follows, $\mathbb{R}_+$ denotes the nonnegative reals, $\mathbb{R}_{++}$ denotes the strictly positive reals, and

$$\Delta^{n-1} \triangleq \left\{ (x_1, \ldots, x_n) \in \mathbb{R}^n \mid \sum_{i=1}^{n} x_i = 1,\ \forall i\, x_i \geq 0 \right\}$$

denotes the $(n-1)$-dimensional simplex.

Inspired by the axiomatic formulation of Shannon's entropy (Khinchin, 1957; Shannon and Weaver, 1949), Suyari (2004) proposed an axiomatic framework for nonextensive entropies and a uniqueness theorem. Let $q \geq 0$ be a fixed scalar, called the *entropic index*. Suyari's axioms (Appendix A) determine a function $S_{q,\phi} : \Delta^{n-1} \to \mathbb{R}$ of the form

$$S_{q,\phi}(p_1, \ldots, p_n) = \begin{cases} \frac{k}{\phi(q)} \left(1 - \sum_{i=1}^{n} p_i^q\right) & \text{if } q \neq 1 \\ -k \sum_{i=1}^{n} p_i \ln p_i & \text{if } q = 1, \end{cases} \tag{1}$$

where k is a positive constant, and $\phi : \mathbb{R}_+ \to \mathbb{R}$ is a continuous function that satisfies the following three conditions: *(i)* $\phi(q)$ has the same sign as $q-1$; *(ii)* $\phi(q)$ vanishes if and only if $q = 1$; *(iii)* ϕ is differentiable in a neighborhood of 1 and $\phi'(1) = 1$.

Note that $S_{1,\phi} = \lim_{q\to 1} S_{q,\phi}$, thus $S_{q,\phi}(p_1, \dots, p_n)$, seen as a function of q, is continuous at $q = 1$. For any ϕ satisfying these conditions, $S_{q,\phi}$ has the *pseudoadditivity* property: for any two independent random variables A and B, with probability mass functions $p_A \in \Delta^{n_A-1}$ and $p_B \in \Delta^{n_B-1}$, respectively, consider the new random variable $A \otimes B$ defined by the joint distribution $p_A \otimes p_B \in \Delta^{n_A n_B-1}$; then,

$$S_{q,\phi}(A \otimes B) = S_{q,\phi}(A) + S_{q,\phi}(B) - \frac{\phi(q)}{k} S_{q,\phi}(A) S_{q,\phi}(B),$$

where we denote (as usual) $S_{q,\phi}(A) \triangleq S_{q,\phi}(p_A)$.

For $q = 1$, Suyari's axioms recover the Shannon-Boltzmann-Gibbs (SBG) entropy,

$$S_{1,\phi}(p_1, \dots, p_n) = H(p_1, \dots, p_n) = -k \sum_{i=1}^{n} p_i \ln p_i,$$

and pseudoadditivity turns into *additivity*, that is, $H(A \otimes B) = H(A) + H(B)$ holds.

Several proposals for ϕ have appeared in the literature (Havrda and Charvát, 1967; Daróczy, 1970; Tsallis, 1988). In this article, unless stated otherwise, we set $\phi(q) = q - 1$, which yields the *Tsallis entropy*:

$$S_q(p_1, \dots, p_n) = \frac{k}{q-1} \left(1 - \sum_{i=1}^{n} p_i^q \right). \tag{2}$$

To simplify, we let $k = 1$ and write the Tsallis entropy as

$$S_q(X) \triangleq S_q(p_1, \dots, p_n) = - \sum_{x \in X} p(x)^q \ln_q p(x), \tag{3}$$

where $\ln_q(x) \triangleq (x^{1-q} - 1)/(1-q)$ is the *q-logarithm function*, which satisfies $\ln_q(xy) = \ln_q(x) + x^{1-q} \ln_q(y)$ and $\ln_q(1/x) = -x^{q-1} \ln_q(x)$. This notation was introduced by Tsallis (1988).

2.2 Tsallis Entropies

Furuichi (2006) derived some information theoretic properties of Tsallis entropies. Tsallis *joint* and *conditional entropies* are defined, respectively, as

$$S_q(X,Y) \triangleq - \sum_{x,y} p(x,y)^q \ln_q p(x,y)$$

and

$$S_q(X|Y) \triangleq - \sum_{x,y} p(x,y)^q \ln_q p(x|y) = \sum_{y} p(y)^q S_q(X|y), \tag{4}$$

and the chain rule $S_q(X,Y) = S_q(X) + S_q(Y|X)$ holds.

For two probability mass functions $p_X, p_Y \in \Delta^n$, the *Tsallis relative entropy*, generalizing the KL divergence, is defined as

$$D_q(p_X \| p_Y) \triangleq - \sum_{x} p_X(x) \ln_q \frac{p_Y(x)}{p_X(x)}. \tag{5}$$

Finally, the *Tsallis mutual entropy* is defined as

$$I_q(X;Y) \triangleq S_q(X) - S_q(X|Y) = S_q(Y) - S_q(Y|X), \tag{6}$$

generalizing (for $q > 1$) Shannon's mutual information (Furuichi, 2006). In Section 5, we establish a relationship between Tsallis mutual entropy and a quantity called *Jensen-Tsallis q-difference*, generalizing the one between mutual information and the JS divergence (shown, e.g., by Grosse et al. 2002, and recalled below, in Section 3.2).

Furuichi (2006) also mentions an alternative generalization of Shannon's mutual information, defined as

$$\tilde{I}_q(X;Y) \triangleq D_q(p_{X,Y} \| p_X \otimes p_Y), \tag{7}$$

where $p_{X,Y}$ is the true joint probability mass function of (X,Y) and $p_X \otimes p_Y$ denotes their joint probability if they were independent. This alternative definition of a "Tsallis mutual entropy" has also been used by Lamberti and Majtey (2003); notice that $I_q(X;Y) \neq \tilde{I}_q(X;Y)$ in general, the case $q = 1$ being a notable exception. In Section 5, we show that this alternative definition also leads to a nonextensive analogue of the JS divergence.

2.3 Entropies of Measures and Denormalization Formulae

Throughout this paper, we consider functionals that extend the domain of the Shannon-Boltzmann-Gibbs and Tsallis entropies to include unnormalized measures. Although, as shown below, these functionals are completely characterized by their restriction to the normalized probability distributions, the denormalization expressions will play an important role in Section 6 to derive novel positive definite kernels inspired by mutual informations.

In order to keep generality, whenever possible we do not restrict to finite or countable sample spaces. Instead, we consider a measure space $(\mathcal{X}, \mathscr{M}, \nu)$ where $\mathcal{X}$ is Hausdorff and ν is a σ-finite Radon measure. We denote by $M_+(\mathcal{X})$ the set of *finite* Radon ν-absolutely continuous measures on $\mathcal{X}$, and by $M_+^1(\mathcal{X})$ the subset of those which are probability measures. For simplicity, we often identify each measure in $M_+(\mathcal{X})$ or $M_+^1(\mathcal{X})$ with its corresponding nonnegative density; this is legitimated by the Radon-Nikodym theorem, which guarantees the existence and uniqueness (up to equivalence within measure zero) of a density function $f : \mathcal{X} \to \mathbb{R}_+$. In the sequel, Lebesgue-Stieltjes integrals of the form $\int_{\mathcal{A}} f(x) d\nu(x)$ are often written as $\int_{\mathcal{A}} f$, or simply $\int f$, if $\mathcal{A} = \mathcal{X}$. Unless otherwise stated, ν is the Lebesgue-Borel measure, if $\mathcal{X} \subseteq \mathbb{R}^n$ and $\text{int}\mathcal{X} \neq \varnothing$, or the counting measure, if $\mathcal{X}$ is countable. In the latter case integrals can be seen as finite sums or infinite series.

Define $\overline{\mathbb{R}} \triangleq \mathbb{R} \cup \{-\infty, +\infty\}$. For some functional $G : M_+(\mathcal{X}) \to \overline{\mathbb{R}}$, let the set $M_+^G(\mathcal{X}) \triangleq \{f \in M_+(\mathcal{X}) : |G(f)| < \infty\}$ be its effective domain, and $M_+^{1,G}(\mathcal{X}) \triangleq M_+^G(\mathcal{X}) \cap M_+^1(\mathcal{X})$ be its subdomain of probability measures.

The following functional (Cuturi and Vert, 2005), extends the Shannon-Boltzmann-Gibbs entropy from $M_+^{1,H}(\mathcal{X})$ to the unnormalized measures in $M_+^H(\mathcal{X})$:

$$H(f) = -k \int f \ln f = \int \varphi_H \circ f, \tag{8}$$

where $k > 0$ is a constant, the function $\varphi_H : \mathbb{R}_+ \to \mathbb{R}$ is defined as

$$\varphi_H(y) = -k\, y \ln y,$$

and, as usual, $0 \ln 0 \triangleq 0$.

The generalized form of the KL divergence, often called *generalized I-divergence* (Csiszar, 1975), is a directed divergence between two measures $\mu_f, \mu_g \in M_+^H(\mathcal{X})$, such that μ_f is μ_g-absolutely continuous (denoted $\mu_f \ll \mu_g$). Let f and g be the densities associated with μ_f and μ_g, respectively. In terms of densities, this generalized KL divergence is

$$D(f,g) \quad = \quad k \int \left(g - f + f \ln \frac{f}{g} \right). \tag{9}$$

Let us now proceed similarly with the nonextensive entropies. For $q \geq 0$, let $M_+^{S_q}(\mathcal{X}) = \{f \in M_+(\mathcal{X}) : f^q \in M_+(\mathcal{X})\}$ for $q \neq 1$, and $M_+^{S_q}(\mathcal{X}) = M_+^H(\mathcal{X})$ for $q = 1$. The nonextensive counterpart of (8), defined on $M_+^{S_q}(\mathcal{X})$, is

$$S_q(f) = \int \varphi_q \circ f, \tag{10}$$

where $\varphi_q : \mathbb{R}_+ \to \mathbb{R}$ is given by

$$\varphi_q(y) = \begin{cases} \varphi_H(y) & \text{if } q = 1, \\ \frac{k}{\phi(q)}(y - y^q) & \text{if } q \neq 1, \end{cases} \tag{11}$$

and $\phi : \mathbb{R}_+ \to \mathbb{R}$ satisfies conditions *(i)-(iii)* stated following Equation (1). The Tsallis entropy is obtained for $\phi(q) = q - 1$,

$$S_q(f) = -k \int f^q \ln_q f. \tag{12}$$

Similarly, a nonextensive generalization of the generalized KL divergence (9) is

$$D_q(f,g) = -\frac{k}{\phi(q)} \int \left(qf + (1-q)g - f^q g^{1-q} \right),$$

for $q \neq 1$, and $D_1(f,g) \triangleq \lim_{q \to 1} D_q(f,g) = D(f,g)$.

Define $|f| \triangleq \int f = \mu_f(\mathcal{X})$. For $|f| = |g| = 1$, several particular cases are recovered: if $\phi(q) = 1 - 2^{1-q}$, then $D_q(f,g)$ is the Havrda-Charvát relative entropy (Havrda and Charvát, 1967; Daróczy, 1970); if $\phi(q) = q - 1$, then $D_q(f,g)$ is the Tsallis relative entropy (5); finally, if $\phi(q) = q(q-1)$, then $D_q(f,g)$ is the canonical α-divergence defined by Amari and Nagaoka (2001) in the realm of information geometry (with the reparameterization $\alpha = 2q - 1$ and assuming $q > 0$ so that $\phi(q) = q(q-1)$ conforms with the axioms).

Remark 1 *Both functionals S_q and D_q are completely determined by their restriction to the normalized measures. Indeed, the following equalities hold for any $c \in \mathbb{R}_{++}$ and $f, g \in M_+^{S_q}(\mathcal{X})$, with $\mu_f \ll \mu_g$:*

$$\begin{aligned} S_q(cf) &= c^q S_q(f) + |f| \varphi_q(c), \\ D_q(cf, cg) &= c D_q(f,g), \\ D_q(cf, g) &= c^q D_q(f,g) - q \varphi_q(c)|f| + \frac{k}{\phi(q)}(q-1)(1-c^q)|g|. \end{aligned} \tag{13}$$

For any $f \in M_+^{S_q}(\mathcal{X})$ and $g \in M_+^{S_q}(\mathcal{Y})$,

$$S_q(f \otimes g) = |g| S_q(f) + |f| S_q(g) - \frac{\phi(q)}{k} S_q(f) S_q(g).$$

If $|f| = |g| = 1$, we recover the pseudo-additivity property of nonextensive entropies:

$$S_q(f \otimes g) = S_q(f) + S_q(g) - \frac{\phi(q)}{k} S_q(f) S_q(g).$$

For $\phi(q) = q-1$, D_q is the Tsallis relative entropy and (13) *reduces to*

$$D_q(cf, g) = c^q D_q(f,g) - q\varphi_q(c)|f| + k(1-c^q)|g|.$$

By taking the limit $q \to 1$, we obtain the following formulae for H and D:

$$\begin{aligned} H(cf) &= cH(f) + |f|\, \varphi_H(c), \\ D(cf, cg) &= cD(f,g), \\ D(cf, g) &= cD(f,g) - |f|\, \varphi_H(c) + k\,(1-c)\,|g|. \end{aligned}$$

Consider $f \in M_+^H(\mathcal{X})$ and $g \in M_+^H(\mathcal{Y})$, and define $f \otimes g \in M_+^H(\mathcal{X} \times \mathcal{Y})$ as $(f \otimes g)(x,y) \triangleq f(x)g(y)$. Then,

$$H(f \otimes g) = |g|\, H(f) + |f|\, H(g).$$

If $|f| = |g| = 1$, we recover the additivity property of the Shannon-Boltzmann-Gibbs entropy, $H(f \otimes g) = H(f) + H(g)$.

3. Jensen Differences and Divergences

In this section, we review the concept of Jensen difference. We then discuss three particular cases: the Jensen-Shannon, Jensen-Rényi, and Jensen-Tsallis divergences.

3.1 The Jensen Difference

Jensen's inequality (Jensen, 1906) is at the heart of many important results in information theory. Let $E[.]$ denote the expectation operator. Jensen's inequality states that if Z is an integrable random variable taking values in a set $\mathcal{Z}$, and f is a measurable convex function defined on the convex hull of $\mathcal{Z}$, then

$$f(E[Z]) \leq E[f(Z)].$$

Burbea and Rao (1982) considered the scenario where $\mathcal{Z}$ is finite, and took $f \triangleq -H_\varphi$, where $H_\varphi : [a,b]^n \to \mathbb{R}$ is a concave function, called a *φ-entropy*, defined as

$$H_\varphi(z) \triangleq -\sum_{i=1}^{n} \varphi(z_i), \tag{14}$$

where $\varphi : [a,b] \to \mathbb{R}$ is convex. They studied the Jensen difference

$$J_\varphi^\pi(y_1, \ldots, y_m) \triangleq H_\varphi\left(\sum_{t=1}^{m} \pi_t\, y_t\right) - \sum_{t=1}^{m} \pi_t\, H_\varphi(y_t),$$

where $\pi \triangleq (\pi_1, \dots, \pi_m) \in \Delta^{m-1}$, and each $y_1, \dots, y_m \in [a,b]^n$.

We consider here a more general scenario, involving two measure sets $(\mathcal{X}, \mathscr{M}, \nu)$ and $(\mathcal{T}, \mathscr{T}, \tau)$, where the second is used to index the first.

Definition 2 *Let $\mu \triangleq (\mu_t)_{t\in\mathcal{T}} \in [M_+(\mathcal{X})]^{\mathcal{T}}$ be a family of finite Radon measures on $\mathcal{X}$, indexed by $\mathcal{T}$, and let $\omega \in M_+(\mathcal{T})$ be a finite Radon measure on $\mathcal{T}$. Define:*

$$J_\Psi^\omega(\mu) \quad \triangleq \quad \Psi\left(\int_{\mathcal{T}} \omega(t)\mu_t \, d\tau(t)\right) - \int_{\mathcal{T}} \omega(t)\Psi(\mu_t)\, d\tau(t) \tag{15}$$

where:

(i) Ψ *is a concave functional such that* $\operatorname{dom}\Psi \subseteq M_+(\mathcal{X})$*;*

(ii) $\omega(t)\mu_t(x)$ *is* τ*-integrable, for all* $x \in \mathcal{X}$*;*

(iii) $\int_{\mathcal{T}} \omega(t)\mu_t d\tau(t) \in \operatorname{dom}\Psi$*;*

(iv) $\mu_t \in \operatorname{dom}\Psi$*, for all* $t \in \mathcal{T}$*;*

(v) $\omega(t)\Psi(\mu_t)$ *is* τ*-integrable.*

If $\omega \in M_+^1(\mathcal{T})$*, we still call* (15) *a Jensen difference.*

In the following subsections, we consider several instances of Definition 2, leading to several Jensen-type divergences.

3.2 The Jensen-Shannon Divergence

Let p be a random probability distribution taking values in $\{p_t\}_{t\in\mathcal{T}}$ according to a distribution $\pi \in M_+^1(\mathcal{T})$. (In classification/estimation theory parlance, π is called the prior distribution and $p_t \triangleq p(.|t)$ the likelihood function.) Then, (15) becomes

$$J_\Psi^\pi(p) = \Psi(E[p]) - E[\Psi(p)], \tag{16}$$

where the expectations are with respect to π.

Let now $\Psi = H$, the Shannon-Boltzmann-Gibbs entropy. Consider the random variables T and X, taking values respectively in $\mathcal{T}$ and $\mathcal{X}$, with densities $\pi(t)$ and $p(x) \triangleq \int_{\mathcal{T}} p(x|t)\pi(t)$. Using standard notation of information theory (Cover and Thomas, 1991),

$$\begin{aligned} J^\pi(p) \triangleq J_H^\pi(p) &= H\left(\int_{\mathcal{T}} \pi(t)p_t\right) - \int_{\mathcal{T}} \pi(t)H(p_t) \\ &= H(X) - \int_{\mathcal{T}} \pi(t)H(X|T=t) \\ &= H(X) - H(X|T) \\ &= I(X;T), \end{aligned} \tag{17}$$

where $I(X;T)$ is the mutual information between X and T. (This relationship between JS divergence and mutual information was pointed out by Grosse et al. 2002.) Since $I(X;T)$ is also equal to the

KL divergence between the joint distribution and the product of the marginals (Cover and Thomas, 1991), we have

$$J^{\pi}(p) = H(E[p]) - E[H(p)] = E[D(p\|E[p])]. \tag{18}$$

When $\mathcal{X}$ and $\mathcal{T}$ are finite with $|\mathcal{T}| = m$, $J^{\pi}_{H}(p_1,\ldots,p_m)$ is called the *Jensen-Shannon (JS) divergence* of $p_1,\ldots,p_m$, with weights $\pi_1,\ldots,\pi_m$ (Burbea and Rao, 1982; Lin, 1991). Equality (18) allows two interpretations of the JS divergence:

- the Jensen difference of the Shannon entropy of p;
- the expected KL divergence from p to the expectation of p.

A remarkable fact is that $J^{\pi}(p) = \min_r E[D(p\|r)]$, that is, $r^* = E[p]$ is a minimizer of $E[D(p\|r)]$ with respect to r. It has been shown that this property together with Equality (18) characterize the so-called *Bregman divergences*: they hold not only for $\Psi = H$, but for any concave Ψ and the corresponding Bregman divergence, in which case J^{π}_{Ψ} is the *Bregman information* (Banerjee et al., 2005).

When $|\mathcal{T}| = 2$ and $\pi = (1/2, 1/2)$, p may be seen as a random distribution whose value on $\{p_1, p_2\}$ is chosen by tossing a fair coin. In this case, $J^{(1/2,1/2)}(p) = JS(p_1, p_2)$, where

$$\begin{aligned} JS(p_1,p_2) &\triangleq H\left(\frac{p_1+p_2}{2}\right) - \frac{H(p_1)+H(p_2)}{2} \\ &= \frac{1}{2}D\left(p_1\middle\|\frac{p_1+p_2}{2}\right) + \frac{1}{2}D\left(p_2\middle\|\frac{p_1+p_2}{2}\right), \end{aligned}$$

as introduced by Lin (1991). It has been shown that $\sqrt{JS}$ satisfies the triangle inequality (hence being a metric) and that, moreover, it is a Hilbertian metric[1] (Endres and Schindelin, 2003; Topsøe, 2000), which has motivated its use in kernel-based machine learning (Cuturi et al., 2005; Hein and Bousquet, 2005) (see Section 6).

3.3 The Jensen-Rényi Divergence

Consider again the scenario above (Section 3.2), with the Rényi q-entropy

$$R_q(p) = \frac{1}{1-q}\ln\int p^q$$

replacing the Shannon-Boltzmann-Gibbs entropy. It is worth noting that the Rényi and Tsallis q-entropies are monotonically related through $R_q(p) = \ln\left([1+(1-q)S_q(p)]^{\frac{1}{1-q}}\right)$, or, using the q-logarithm function,

$$S_q(p) = \ln_q \exp R_q(p).$$

The Rényi q-entropy is concave for $q \in [0,1)$ and has the Shannon-Boltzmann-Gibbs entropy as the limit when $q \to 1$. Letting $\Psi = R_q$, (16) becomes

$$J^{\pi}_{R_q}(p) = R_q(E[p]) - E[R_q(p)]. \tag{19}$$

1. A metric $d : \mathcal{X}\times\mathcal{X} \to \mathbb{R}$ is Hilbertian if there is some Hilbert space $\mathcal{H}$ and an isometry $f : \mathcal{X} \to \mathcal{H}$ such that $d^2(x,y) = \langle f(x) - f(y), f(x) - f(y)\rangle_{\mathcal{H}}$ holds for any $x, y \in \mathcal{X}$ (Hein and Bousquet, 2005).

Unlike in the JS divergence case, there is no counterpart of equality (18) based on the Rényi q-divergence

$$D_{R_q}(p_1\|p_2) = \frac{1}{q-1}\ln\int p_1^q\, p_2^{1-q}.$$

When $\mathcal{X}$ and $\mathcal{T}$ are finite, we call $J^{\pi}_{R_q}$ in (19) the *Jensen-Rényi (JR) divergence*. Furthermore, when $|\mathcal{T}| = 2$ and $\pi = (1/2, 1/2)$, we write $J^{\pi}_{R_q}(p) = JR_q(p_1, p_2)$, where

$$JR_q(p_1, p_2) = R_q\left(\frac{p_1+p_2}{2}\right) - \frac{R_q(p_1)+R_q(p_2)}{2}.$$

The JR divergence has been used in several signal/image processing applications, such as registration, segmentation, denoising, and classification (Ben-Hamza and Krim, 2003; He et al., 2003; Karakos et al., 2007). In Section 6, we show that the JR divergence is (like the JS divergence) a Hilbertian metric, which is relevant for its use in kernel-based machine learning.

3.4 The Jensen-Tsallis Divergence

Burbea and Rao (1982) have defined Jensen-type divergences of the form (16) based on the Tsallis q-entropy S_q, defined in (12). Like the Shannon-Boltzmann-Gibbs entropy, but unlike the Rényi entropies, the Tsallis q-entropy, for finite $\mathcal{T}$, is an instance of a φ-entropy (see Equation 14). Letting $\Psi = S_q$, (16) becomes

$$J^{\pi}_{S_q}(p) = S_q(E[p]) - E[S_q(p)]. \tag{20}$$

Again, as in Section 3.3, if we consider the Tsallis q-divergence,

$$D_q(p_1\|p_2) = \frac{1}{1-q}\left(1 - \int {p_1}^q\, {p_2}^{1-q}\right),$$

there is no counterpart of the Equality (18).

When $\mathcal{X}$ and $\mathcal{T}$ are finite, $J^{\pi}_{S_q}$ in (20) is called the *Jensen-Tsallis (JT) divergence* and it has also been applied in image processing (Ben-Hamza, 2006). Unlike the JS divergence, the JT divergence lacks an interpretation as a mutual information. Despite this, for $q \in [1,2]$, it exhibits joint convexity (Burbea and Rao, 1982). In the next section, we propose an alternative to the JT divergence which, among other features, is interpretable as a nonextensive mutual information (in the sense of Furuichi 2006) and is jointly convex, for $q \in [0,1]$.

4. q-Convexity and q-Differences

This section introduces a novel class of functions, termed *Jensen q-differences*, which generalize Jensen differences. Later (in Section 5), we will use these functions to define the *Jensen-Tsallis q-difference*, which we will propose as an alternative nonextensive generalization of the JS divergence, instead of the JT divergence discussed in Section 3.4. We begin by recalling the concept of q-expectation (Tsallis, 1988).

Definition 3 *The unnormalized* q-expectation *of a random variable* X*, with probability density* p*, is*

$$E_q[X] \triangleq \int x\, p(x)^q.$$

Of course, $q=1$ corresponds to the standard notion of expectation. For $q \neq 1$, the q-expectation does not match the intuitive meaning of average/expectation (e.g., $E_q[1] \neq 1$, in general). The q-expectation is a convenient concept in nonextensive information theory; for example, it yields a very compact form for the Tsallis entropy: $S_q(X) = -E_q[\ln_q p(X)]$.

4.1 q-Convexity

We now introduce the novel concept of q-convexity and use it to derive a set of results, namely the *Jensen q-inequality*.

Definition 4 *Let $q \in \mathbb{R}$ and $\mathcal{X}$ be a convex set. A function $f : \mathcal{X} \to \mathbb{R}$ is* q-convex *if for any $x, y \in \mathcal{X}$ and $\lambda \in [0,1]$,*

$$f(\lambda x + (1-\lambda)y) \leq \lambda^q f(x) + (1-\lambda)^q f(y). \tag{21}$$

If $-f$ is q-convex, f is said to be q-concave.

Of course, 1-convexity is the usual notion of convexity. Many properties of 1-convex functions do not have q-analogues. For example, for $q \neq 1$, any q-convex function must be either nonnegative (if $q < 1$) or nonpositive (if $q > 1$); this simple fact can be shown through *reductio ad absurdum* by setting $x = y$ in (21). However, other properties remain: the next proposition states the Jensen q-inequality.

Proposition 5 *If $f : \mathcal{X} \to \mathbb{R}$ is q-convex, then for any $n \in \mathbb{N}$, $x_1, \ldots, x_n \in \mathcal{X}$ and $\pi = (\pi_1, \ldots, \pi_n) \in \Delta^{n-1}$,*

$$f\left(\sum_{i=1}^{n} \pi_i x_i\right) \leq \sum_{i=1}^{n} \pi_i^q f(x_i).$$

Moreover, if f is continuous, the above still holds for countably many points $(x_i)_{i \in \mathbb{N}}$.

Proof In the finite case, the proof can be carried out by induction, as in the proof of the standard Jensen inequality (Cover and Thomas, 1991). Assuming that the inequality holds for $n \in \mathbb{N}$, then, from the definition of q-convexity, it will also hold for $n+1$:

$$\begin{aligned} f\left(\sum_{i=1}^{n+1} \pi_i x_i\right) &= f\left(\sum_{i=1}^{n} \pi_i x_i + \pi_{n+1} x_{n+1}\right) \\ &= f\left((1-\pi_{n+1}) \sum_{i=1}^{n} \pi'_i x_i + \pi_{n+1} x_{n+1}\right) \\ &\leq (1-\pi_{n+1})^q f\left(\sum_{i=1}^{n} \pi'_i x_i\right) + \pi_{n+1}^q f(x_{n+1}) \\ &\leq \sum_{i=1}^{n} \pi_i^q f(x_i) + \pi_{n+1}^q f(x_{n+1}) \;=\; \sum_{i=1}^{n+1} \pi_i^q f(x_i), \end{aligned}$$

where we used the fact that $\pi_{n+1} = 1 - \sum_{i=1}^{n} \pi_i$, and we defined $\pi'_i \triangleq \pi_i/(1-\pi_{n+1})$ (note that $\sum_{i=1}^{n} \pi'_i = 1$.) Furthermore, if f is continuous, it commutes with taking limits, thus

$$f\left(\sum_{i=1}^{\infty}\pi_i x_i\right) = f\left(\lim_{n\to\infty}\sum_{i=1}^{n}\pi_i x_i\right) = \lim_{n\to\infty} f\left(\sum_{i=1}^{n}\pi_i x_i\right) \le \lim_{n\to\infty}\sum_{i=1}^{n}\pi_i^q f(x_i) = \sum_{i=1}^{\infty}\pi_i^q f(x_i).$$

■

Proposition 6 *Let $f \ge 0$ and $q \ge r \ge 0$; then,*

$$f \text{ is } q\text{-convex} \quad \Rightarrow \quad f \text{ is } r\text{-convex} \tag{22}$$

$$f \text{ is } r\text{-concave} \quad \Rightarrow \quad f \text{ is } q\text{-concave}. \tag{23}$$

Proof Implication (22) results from

$$f(\lambda x + (1-\lambda)y) \ \le\ \lambda^q f(x) + (1-\lambda)^q f(y) \ \le\ \lambda^r f(x) + (1-\lambda)^r f(y),$$

where the first inequality states the q-convexity of f and the second one is valid because $f(x), f(y) \ge 0$ and $t^r \ge t^q \ge 0$, for any $t \in [0,1]$ and $q \ge r$. The proof of (23) is similar. ■

4.2 Jensen q-Differences

We now generalize Jensen differences, formalized in Definition 2, by introducing the concept of Jensen q-differences.

Definition 7 *Let $\mu \triangleq (\mu_t)_{t\in\mathcal{T}} \in [M_+(\mathcal{X})]^{\mathcal{T}}$ be a family of finite Radon measures on $\mathcal{X}$, indexed by $\mathcal{T}$, and let $\omega \in M_+(\mathcal{T})$ be a finite Radon measure on $\mathcal{T}$. For $q \ge 0$, define*

$$T^{\omega}_{q,\Psi}(\mu) \ \triangleq\ \Psi\left(\int_{\mathcal{T}} \omega(t)\,\mu_t\, d\tau(t)\right) - \int_{\mathcal{T}} \omega(t)^q\, \Psi(\mu_t)\, d\tau(t), \tag{24}$$

where:

(i) *Ψ is a concave functional such that* $\mathrm{dom}\,\Psi \subseteq M_+(\mathcal{X})$*;*

(ii) *$\omega(t)\mu_t(x)$ is τ-integrable for all $x \in \mathcal{X}$;*

(iii) *$\int_{\mathcal{T}} \omega(t)\mu_t\, d\tau(t) \in \mathrm{dom}\,\Psi$;*

(iv) *$\mu_t \in \mathrm{dom}\,\Psi$, for all $t \in \mathcal{T}$;*

(v) *$\omega(t)^q\, \Psi(\mu_t)$ is τ-integrable.*

If $\omega \in M^1_+(\mathcal{T})$, we call the function defined in (24) *a* Jensen q-difference.

Burbea and Rao (1982) established necessary and sufficient conditions on φ for the Jensen difference of a φ-entropy (see Equation 14) to be convex. The following proposition generalizes that result, extending it to Jensen q-differences.

Proposition 8 *Let $\mathcal{T}$ and $\mathcal{X}$ be finite sets, with $|\mathcal{T}| = m$ and $|\mathcal{X}| = n$, and let $\pi \in M^1_+(\mathcal{T})$. Let $\varphi : [0,1] \to \mathbb{R}$ be a function of class C^2 and consider the (φ-entropy, Burbea and Rao, 1982) function $\Psi : [0,1]^n \to \mathbb{R}$ defined as $\Psi(z) \triangleq -\sum_{i=1}^{n}\varphi(z_i)$. Then, the q-difference $T^{\pi}_{q,\Psi} : [0,1]^{nm} \to \mathbb{R}$ is convex if and only if φ is convex and $-1/\varphi''$ is $(2-q)$-convex.*

The proof is rather long, thus it is relegated to Appendix B.

5. The Jensen-Tsallis q-Difference

This section introduces the Jensen-Tsallis q-difference, a nonextensive generalization of the Jensen-Shannon divergence. After deriving some properties concerning the convexity and extrema of these functionals, we introduce the notion of joint and conditional Jensen-Tsallis q-difference, a contrast measure between stochastic processes. We end the section with a brief asymptotic analysis for the extensive case.

5.1 Definition

As in Section 3.2, let p be a random probability distribution taking values in $\{p_t\}_{t\in\mathcal{T}}$ according to a distribution $\pi \in M_+^1(\mathcal{T})$. Then, we may write

$$T^{\pi}_{q,\Psi}(p) = \Psi(E[p]) - E_q[\Psi(p)],$$

where the expectations are with respect to π. Hence Jensen q-differences may be seen as deformations of the standard Jensen differences (16), in which the second expectation is replaced by a q-expectation.

Let $\Psi = S_q$, the nonextensive Tsallis q-entropy. Introducing the random variables T and X, with values respectively in $\mathcal{T}$ and $\mathcal{X}$, with densities $\pi(t)$ and $p(x) \triangleq \int_{\mathcal{T}} p(x|t)\pi(t)$, we have (writing T^{π}_{q,S_q} simply as T^{π}_q)

$$\begin{aligned} T^{\pi}_q(p) &= S_q(E[p]) - E_q[S_q(p)] \\ &= S_q(X) - \int_{\mathcal{T}} \pi(t)^q S_q(X|T=t) \\ &= S_q(X) - S_q(X|T) \\ &= I_q(X;T), \end{aligned} \tag{25}$$

where $S_q(X|T)$ is the Tsallis conditional entropy (4), and $I_q(X;T)$ is the Tsallis mutual information (6), as defined by Furuichi (2006). Observe that (25) is a nonextensive analogue of (17). Since, in general, $I_q \neq \tilde{I}_q$ (see Equation 7), unless $q=1$ (in that case, $I_1 = \tilde{I}_1 = I$), there is no counterpart of (18) in terms of q-differences. Nevertheless, Lamberti and Majtey (2003) have proposed a non-logarithmic version of the JS divergence, which corresponds to using $\tilde{I}_q$ for the Tsallis mutual q-entropy (although this interpretation is not explicitly mentioned).

When $\mathcal{X}$ and $\mathcal{T}$ are finite with $|\mathcal{T}| = m$, we call the quantity $T^{\pi}_q(p_1,\ldots,p_m)$ the *Jensen-Tsallis (JT) q-difference* of $p_1,\ldots,p_m$ with weights $\pi_1,\ldots,\pi_m$. Although the JT q-difference is a generalization of the JS divergence, for $q \neq 1$, the term "divergence" would be misleading in this case, since T^{π}_q may take negative values (if $q<1$) and does not vanish in general if p is deterministic.

When $|\mathcal{T}| = 2$ and $\pi = (1/2, 1/2)$, define $T_q \triangleq T_q^{1/2,1/2}$,

$$T_q(p_1,p_2) = S_q\left(\frac{p_1+p_2}{2}\right) - \frac{S_q(p_1)+S_q(p_2)}{2^q}.$$

Notable cases arise for particular values of q:

- For $q=0$, $S_0(p) = -1 + \nu(\mathrm{supp}(p))$, where $\nu(\mathrm{supp}(p))$ denotes the measure of the support of p (recall that p is defined on the measure space $(\mathcal{X},\mathscr{M},\nu)$). For example, if $\mathcal{X}$ is finite and

ν is the counting measure, $\nu(\mathrm{supp}(p)) = \|p\|_0$ is the so-called *0-norm* (although it is not a norm) of vector p, that is, its number of nonzero components. The Jensen-Tsallis 0-difference is thus

$$\begin{aligned} T_0(p_1,p_2) &= -1+\nu\left(\mathrm{supp}\left(\frac{p_1+p_2}{2}\right)\right)+1-\nu(\mathrm{supp}(p_1))+1-\nu(\mathrm{supp}(p_2)) \\ &= 1+\nu(\mathrm{supp}(p_1)\cup\mathrm{supp}(p_2))-\nu(\mathrm{supp}(p_1))-\nu(\mathrm{supp}(p_2)) \\ &= 1-\nu(\mathrm{supp}(p_1)\cap\mathrm{supp}(p_2)); \qquad (26) \end{aligned}$$

if $\mathcal{X}$ is finite and ν is the counting measure, this becomes

$$T_0(p_1,p_2) = 1 - \|p_1 \odot p_2\|_0,$$

where $\odot$ denotes the Hadamard-Schur (i.e., elementwise) product. We call T_0 the *Boolean difference*.

- For $q = 1$, since $S_1(p) = H(p)$, T_1 is the JS divergence,

$$T_1(p_1,p_2) = JS(p_1,p_2).$$

- For $q = 2$, $S_2(p) = 1 - \langle p,p\rangle$, where $\langle a,b\rangle = \int_{\mathcal{X}} a(x)\,b(x)\,d\nu(x)$ is the inner product between a and b (which reduces to $\langle a,b\rangle = \sum_i a_i b_i$ if $\mathcal{X}$ is finite and ν is the counting measure). Consequently, the Tsallis 2-difference is

$$T_2(p_1,p_2) = \frac{1}{2} - \frac{1}{2}\langle p_1,p_2\rangle,$$

which we call the *linear difference*.

5.2 Properties of the JT q-Difference

This subsection presents results regarding convexity and extrema of the JT q-difference, for certain values of q, extending known properties of the JS divergence ($q = 1$). Some properties of the JS divergence are lost in the transition to nonextensivity; for example, while the former is nonnegative and vanishes if and only if all the distributions are identical, this is not true in general with the JT q-difference. Nonnegativity of the JT q-difference is only guaranteed if $q \geq 1$, which explains why some authors (e.g., Furuichi 2006) only consider values of $q \geq 1$, when looking for nonextensive analogues of Shannon's information theory. Moreover, unless $q = 1$, it is not generally true that $T_q^{\pi}(p,\ldots,p) = 0$ or even that $T_q^{\pi}(p,\ldots,p,p') \geq T_q^{\pi}(p,\ldots,p,p)$. For example, the solution of the optimization problem

$$\min_{p_1\in\Delta^n} T_q(p_1,p_2), \qquad (27)$$

is, in general, different from p_2, unless $q = 1$. Instead, this minimizer is closer to the uniform distribution if $q \in [0,1)$, and closer to a degenerate distribution for $q \in (1,2]$ (see Fig. 1). This is not so surprising: recall that $T_2(p_1,p_2) = \frac{1}{2} - \frac{1}{2}\langle p_1,p_2\rangle$; in this case, (27) becomes a linear program, and the solution is not $p_1^* = p_2$, but $p_1^* = \delta_j$, where $j = \arg\max_i p_{2i}$.

At this point, we should also remark that, when $\mathcal{X}$ is a finite set, the uniform distribution maximizes the Tsallis entropy for any $q \geq 0$, which is in fact one of the Suyari axioms underlying the Tsallis entropy (see Axiom A2 in Appendix A).

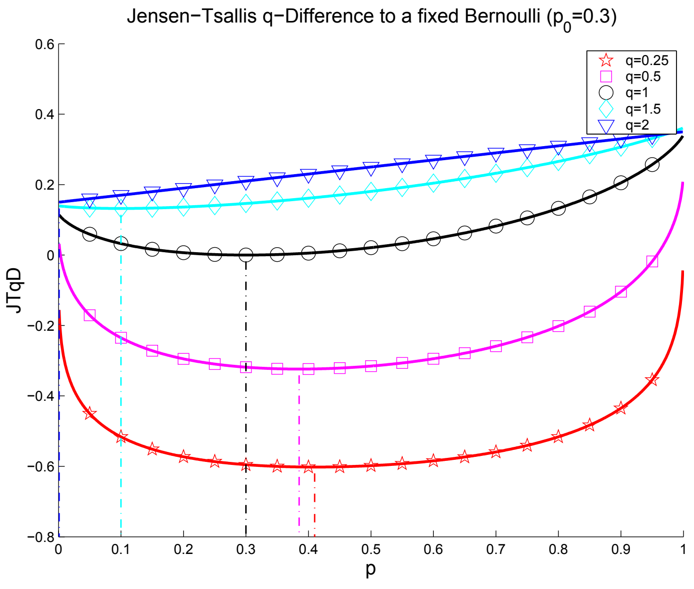


Figure 1: Jensen-Tsallis q-difference between two Bernoulli distributions, $p_1 = (0.3, 0.7)$ and $p_2 = (p, 1-p)$, for several values of the entropic index q. Observe that, for $q \in [0,1)$, the minimizer of the JT q-difference approaches the uniform distribution $(0.5, 0.5)$ as q approaches 0; for $q \in (1,2]$, this minimizer approaches the degenerate distribution, as $q \to 2$.

We start with the following corollary of Proposition 8, which establishes the joint convexity of the JT q-difference, for $q \in [0,1]$. (Interestingly, this "complements" the joint convexity of the JT divergence (20), for $q \in [1,2]$, proved by Burbea and Rao 1982.)

Corollary 9 *Let $\mathcal{T}$ and $\mathcal{X}$ be finite sets with cardinalities m and n, respectively. For $q \in [0,1]$, the JT q-difference is a jointly convex function on $M_+^{1,S_q}(\mathcal{X})$. Formally, let $\{p_t^{(i)}\}_{t\in\mathcal{T}}$, and $i = 1,\ldots,l$, be a collection of l sets of probability distributions on $\mathcal{X}$; then, for any $(\lambda_1,\ldots,\lambda_l) \in \Delta^{l-1}$,*

$$T_q^\pi\left(\sum_{i=1}^l \lambda_i p_1^{(i)}, \ldots, \sum_{i=1}^l \lambda_i p_m^{(i)}\right) \le \sum_{i=1}^l \lambda_i T_q^\pi(p_1^{(i)}, \ldots, p_m^{(i)}).$$

Proof Observe that the Tsallis entropy (3) of a probability distribution $p_t = \{p_{t1}, ..., p_{tn}\}$ can be written as

$$S_q(p_t) = -\sum_{i=1}^n \varphi(p_{ti}), \quad \text{where} \quad \varphi_q(x) = \frac{x - x^q}{1-q};$$

thus, from Proposition 8, T_q^π is convex if and only if φ_q is convex and $-1/\varphi_q''$ is $(2-q)$-convex. Since $\varphi_q''(x) = q x^{q-2}$, φ_q is convex for $x \ge 0$ and $q \ge 0$. To show the $(2-q)$-convexity of $-1/\varphi_q''(x) = -(1/q)x^{2-q}$, for $x_t \ge 0$, and $q \in [0,1]$, we use a version of the power mean inequality (Steele, 2006),

$$-\left(\sum_{i=1}^l \lambda_i x_i\right)^{2-q} \le -\sum_{i=1}^l (\lambda_i x_i)^{2-q} = -\sum_{i=1}^l \lambda_i^{2-q} x_i^{2-q},$$

thus concluding that $-1/\varphi_q''$ is in fact $(2-q)$-convex. ■

A consequence of Corollary 9 is that, for finite $\mathcal{X}$ and any $q \in [0,1]$, the JT q-difference is upper bounded, namely $T_q^\pi(p_1,\ldots,p_m) \leq S_q(\pi)$. Indeed, since T_q^π is convex and its domain is the Cartesian product of m simplices (a convex polytope), its maximum must occur on a vertex, that is, when each argument p_t is a degenerate distribution at x_t, denoted δ_{x_t}. In particular, if $|\mathcal{X}| \geq |\mathcal{T}|$, this maximum occurs at a vertex corresponding to disjoint degenerate distributions, that is, such that $x_i \neq x_j$ if $i \neq j$. At this maximum,

$$\begin{aligned} T_q^\pi(\delta_{x_1},\ldots,\delta_{x_m}) &= S_q\left(\sum_{t=1}^m \pi_t \delta_{x_t}\right) - \sum_{t=1}^m \pi_t S_q(\delta_{x_t}) \\ &= S_q\left(\sum_{t=1}^m \pi_t \delta_{x_t}\right) \qquad (28) \\ &= S_q(\pi) \end{aligned}$$

where the equality in (28) results from $S_q(\delta_{x_t}) = 0$. (Notice that this maximum may not be achieved if $|\mathcal{X}| < |\mathcal{T}|$.) The next proposition provides a stronger result: it establishes upper and lower bounds for the JT q-difference to any non-negative q and to countable $\mathcal{X}$ and $\mathcal{T}$.

Proposition 10 *Let $\mathcal{T}$ and $\mathcal{X}$ be countable sets. For $q \geq 0$,*

$$T_q^\pi((p_t)_{t\in\mathcal{T}}) \leq S_q(\pi), \qquad (29)$$

and, if $|\mathcal{X}| \geq |\mathcal{T}|$, the maximum of T_q^π is reached for a set of disjoint degenerate distributions. This maximum may not be attained if $|\mathcal{X}| < |\mathcal{T}|$.

For $q \geq 1$,

$$T_q^\pi((p_t)_{t\in\mathcal{T}}) \geq 0,$$

and the minimum of T_q^π is attained in the purely deterministic case, that is, when all distributions are equal to the same degenerate distribution.

For $q \in [0,1]$ and $\mathcal{X}$ a finite set with $|\mathcal{X}| = n$,

$$T_q^\pi((p_t)_{t\in\mathcal{T}}) \geq S_q(\pi)[1-n^{1-q}]. \qquad (30)$$

This lower bound (which is zero or negative) is attained when all distributions are uniform.

Proof The proof is given in Appendix C. ■

Finally, the next proposition characterizes the convexity/concavity of the JT q-difference on each argument.

Proposition 11 *Let $\mathcal{T}$ and $\mathcal{X}$ be countable sets. The JT q-difference is convex in each argument, for $q \in [0,2]$, and concave in each argument, for $q \geq 2$.*

Proof Notice that the JT q-difference can be written as $T_q^\pi(p_1,\ldots,p_m) = \sum_j \psi(p_{1j},\ldots,p_{mj})$, with

$$\psi(y_1,\ldots,y_m) = \frac{1}{q-1}\left[\sum_i (\pi_i - \pi_i^q)y_i + \sum_i \pi_i^q y_i^q - \left(\sum_i \pi_i y_i\right)^q\right].$$

It suffices to consider the second derivative of ψ with respect to y_1. Introducing $z = \sum_{i=2}^m \pi_i y_i$,

$$\begin{aligned}\frac{\partial^2\psi}{\partial y_1^2} &= q\left[\pi_1^q y_1^{q-2} - \pi_1^2(\pi_1 y_1 + z)^{q-2}\right] \\ &= q\pi_1^2\left[(\pi_1 y_1)^{q-2} - (\pi_1 y_1 + z)^{q-2}\right]. \qquad (31)\end{aligned}$$

Since $\pi_1 y_1 \le (\pi_1 y_1 + z) \le 1$, the quantity in (31) is nonnegative for $q \in [0,2]$ and non-positive for $q \ge 2$. ■

5.3 Joint and Conditional JT q-Differences and a Chain Rule

This subsection introduces joint and conditional JT q-differences, which will later be used as a contrast measure between stochastic processes. A chain rule is derived that relates conditional and joint JT q-differences.

Definition 12 *Let $\mathcal{X}$, $\mathcal{Y}$ and $\mathcal{T}$ be measure spaces. Let $(p_t)_{t\in\mathcal{T}} \in [M_+^1(\mathcal{X}\times\mathcal{Y})]^{\mathcal{T}}$ be a family of measures in $M_+^1(\mathcal{X}\times\mathcal{Y})$ indexed by $\mathcal{T}$, and let p be a random probability distribution taking values in $\{p_t\}_{t\in\mathcal{T}}$ according to a distribution $\pi \in M_+^1(\mathcal{T})$. Consider also:*

- *for each $t\in\mathcal{T}$, the marginals $p_t(Y) \in M_+^1(\mathcal{Y})$,*
- *for each $t\in\mathcal{T}$ and $y\in\mathcal{Y}$, the conditionals $p_t(X|Y=y) \in M_+^1(\mathcal{X})$,*
- *the mixture $r(X,Y) \triangleq \int_{\mathcal{T}} \pi(t)\, p_t(X,Y) \in M_+^1(\mathcal{X}\times\mathcal{Y})$,*
- *the marginal $r(Y) \in M_+^1(\mathcal{Y})$,*
- *for each $y\in\mathcal{Y}$, the conditionals $r(X|Y=y) \in M_+^1(\mathcal{X})$.*

For notational convenience, we also append a subscript to p to emphasize its joint or conditional dependency of the random variables X and Y, that is, $p_{XY} \triangleq p$, and $p_{X|Y}$ denotes a random conditional probability distribution taking values in $\{p_t(.|Y)\}_{t\in\mathcal{T}}$ according to the distribution π.

For $q \ge 0$, we refer to the joint JT q-difference *of p_{XY} by*

$$T_q^\pi(p_{XY}) \triangleq T_q^\pi(p) = S_q(r) - E_{q,T\sim\pi(T)}[S_q(p_t)]$$

and to the conditional JT q-difference *of $p_{X|Y}$ by*

$$T_q^\pi(p_{X|Y}) \triangleq E_{q,Y\sim r(Y)}[S_q(r(.|Y=y))] - E_{q,T\sim\pi(T)}\left[E_{q,Y\sim p_t(Y)}[S_q(p_t(.|Y=y))]\right], \qquad (32)$$

where we appended the random variables being used in each q-expectation, for the sake of clarity.

Note that the joint JT q-difference is just the usual JT q-difference of the joint random variable $X \times Y$, which equals (cf. 25)

$$T_q^\pi(p_{XY}) \;=\; S_q(X,Y) - S_q(X,Y|T) \;=\; I_q(X \times Y; T), \tag{33}$$

and the conditional JT q-difference is simply the usual JT q-difference with all entropies replaced by conditional entropies (conditioned on Y). Indeed, expression (32) can be rewritten as:

$$T_q^\pi(p_{X|Y}) \;=\; S_q(X|Y) - S_q(X|T,Y) \;=\; I_q(X;T \mid Y), \tag{34}$$

that is, the conditional JT q-difference may also interpreted as a Tsallis mutual information, as in (25), but now *conditioned* on the random variable Y.

Note also that, for the extensive case $q = 1$, (32) may also be rewritten in terms of the conditional KL divergences,

$$\begin{aligned} J^\pi(p_{X|Y}) \triangleq T_1^\pi(p_{X|Y}) \;&=\; E_{Y\sim r(Y)}\left[H(r(.|Y=y))\right] - E_{T\sim\pi(T)}\left[E_{Y\sim p_t(Y)}\left[H(p_t(.|Y=y))\right]\right] \\ &=\; E_{T\sim\pi(T)}\left[E_{Y\sim r(Y)}\left[D(p_t(.|Y=y)\|r(.|Y=y))\right]\right]. \end{aligned}$$

Proposition 13 *The following chain rule holds:*

$$T_q^\pi(p_{XY}) = T_q^\pi(p_{X|Y}) + T_q^\pi(p_Y)$$

Proof Writing the joint/conditional JT q-differences as joint/conditional mutual informations (33–34) and invoking the chain rule provided by (4), we have that

$$\begin{aligned} I_q(X;T|Y) + I_q(Y;T) \;&=\; S_q(X|T,Y) - S_q(X|Y) + S_q(Y|T) - S_q(Y) \\ &=\; S_q(X,Y|T) - S_q(X,Y), \end{aligned}$$

which is the joint JT q-difference associated with the random variable $X \times Y$. ■

Let us now turn our attention to the case where $Y = X^k$ for some $k \in \mathbb{N}$. In the following, the notation $(A_n)_{n\in\mathbb{N}}$ denotes a stationary ergodic process with values on some finite alphabet $\mathcal{A}$.

Definition 14 *Let $\mathcal{X}$ and $\mathcal{T}$ be measure spaces, with $\mathcal{X}$ finite, and let $\mathscr{F} = [(X_n)_{n\in\mathbb{N}}]^{\mathcal{T}}$ be a family of stochastic processes (taking values on the alphabet $\mathcal{X}$) indexed by $\mathcal{T}$. The k-*th order JT q-difference *of $\mathscr{F}$ is defined, for $k = 1,\ldots,n$, as*

$$T_{q,k}^{\text{joint},\pi}(\mathscr{F}) \triangleq T_q^\pi(p_{X^k})$$

*and the k-*th order conditional JT q-difference *of $\mathscr{F}$ is defined, for $k = 1,\ldots,n$, as*

$$T_{q,k}^{\text{cond},\pi}(\mathscr{F}) \triangleq T_q^\pi(p_{X|X^k}),$$

and, for $k = 0$, as $T_{q,0}^{\text{cond},\pi}(\mathscr{F}) \triangleq T_{q,1}^{\text{joint},\pi}(\mathscr{F}) = T_q^\pi(p_X)$.

Proposition 15 *The joint and conditional k-th order JT q-differences are related through:*

$$T_{q,k}^{\text{joint},\pi}(\mathscr{F}) = \sum_{i=0}^{k-1} T_{q,i}^{\text{cond},\pi}(\mathscr{F}) \tag{35}$$

Proof Use Proposition 13 and induction. ■

5.4 Asymptotic Analysis in the Extensive Case

We now focus on the extensive case ($q = 1$) for a brief asymptotic analysis of the k-th order joint and conditional JT 1-differences (or *conditional Jensen-Shannon divergences*) when k goes to infinity.

The conditional Jensen-Shannon divergence was introduced by El-Yaniv et al. (1998) to address the *two-sample problem* for strings emitted by Markovian sources. Given two strings s and t, the goal is to decide whether they were emitted by the same source or by different sources. Under some fair assumptions, the most likely k-th order Markovian joint source of s and t is governed by a distribution $\hat{r}$ given by

$$\hat{r} = \arg\min_{r} \lambda D(\hat{p}_s \| r) + (1-\lambda) D(\hat{p}_t \| r). \tag{36}$$

where $D(.\|.)$ are conditional KL divergences, $\hat{p}_s$ and $\hat{p}_t$ are the empirical $(k-1)$-th order conditionals associated with s and t, respectively, and $\lambda = |s|/(|s|+|t|)$ is the length ratio. The solution of the optimization problem is

$$\hat{r}(a|c) = \frac{\lambda \hat{p}_s(c)}{\lambda \hat{p}_s(c) + (1-\lambda)\hat{p}_t(c)}\, \hat{p}_s(a|c) + \frac{(1-\lambda)\hat{p}_t(c)}{\lambda \hat{p}_s(c) + (1-\lambda)\hat{p}_t(c)}\, \hat{p}_t(a|c),$$

where $a \in \mathcal{A}$ is a symbol and $c \in \mathcal{A}^{k-1}$ is a context; this can be rewritten as $\hat{r}(a,c) = \lambda \hat{p}_s(a,c) + (1-\lambda)\hat{p}_t(a,c)$; that is, the optimum in (36) is a mixture of $\hat{p}_s$ and $\hat{p}_t$ weighted by the string lengths. Notice that, at the minimum, we have

$$\lambda D(\hat{p}_s \| \hat{r}) + (1-\lambda) D(\hat{p}_t \| \hat{r}) = JS_k^{\text{cond},(\lambda,1-\lambda)}(\hat{p}_s, \hat{p}_t).$$

It is tempting to investigate the asymptotic behavior of the conditional and joint JS divergences when $k \to \infty$; however, unlike other asymptotic information theoretic quantities, like the entropy or cross entropy rates, this behavior fails to characterize the sources s and t. Intuitively, this is justified by the fact that observing more and more symbols drawn from the mixture of the two sources rapidly decreases the uncertainty about which source generated the sample. Indeed, from the asymptotic equipartition property of stationary ergodic sources (Cover and Thomas, 1991), we have that $\lim_{k\to\infty} \frac{1}{k} H(p_{X_k}) = \lim_{k\to\infty} H(p_{X|X_k})$, which implies

$$\lim_{k\to\infty} JS_k^{\text{cond},\pi} \;=\; \lim_{k\to\infty} \frac{1}{k} JS_k^{\text{joint},\pi} \;\leq\; \lim_{k\to\infty} \frac{1}{k} H(\pi) \;=\; 0,$$

where we used the fact that the JS divergence is upper-bounded by the entropy of the mixture $H(\pi)$ (see Proposition 10). Since the conditional JS divergence must be non-negative, we therefore conclude that $\lim_{k\to\infty} JS_k^{\text{cond},\pi} = 0$, pointwise.

6. Nonextensive Mutual Information Kernels

In this section we consider the application of extensive and nonextensive entropies to define kernels on measures; since kernels involve pairs of measures, throughout this section $|\mathcal{T}| = 2$. Based on the denormalization formulae presented in Section 2.3, we devise novel kernels related to the JS divergence and the JT q-difference; these kernels allow setting a weight for each argument, thus will be called *weighted Jensen-Tsallis kernels*. We also introduce kernels related to the JR divergence (Section 3.3) and the JT divergence (Section 3.4), and establish a connection between the Tsallis kernels and a family of kernels investigated by Hein et al. (2004) and Fuglede (2005), placing those kernels under a new information-theoretic light. After that, we give a brief overview of string kernels, and using the results of Section 5.3, we devise k-th order Jensen-Tsallis kernels between stochastic processes that subsume the well-known p-spectrum kernel of Leslie et al. (2002).

6.1 Positive and Negative Definite Kernels

We start by recalling basic concepts from kernel theory (Schölkopf and Smola, 2002); in the following, $\mathcal{X}$ denotes a nonempty set.

Definition 16 *Let $\varphi : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ be a symmetric function, that is, a function satisfying $\varphi(y,x) = \varphi(x,y)$, for all $x, y \in \mathcal{X}$. φ is called a* positive definite *(pd) kernel if and only if*

$$\sum_{i=1}^{n} \sum_{j=1}^{n} c_i c_j \varphi(x_i, x_j) \geq 0$$

for all $n \in \mathbb{N}$, $x_1, \ldots, x_n \in \mathcal{X}$ and $c_1, \ldots, c_n \in \mathbb{R}$.

Definition 17 *Let $\psi : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ be symmetric. ψ is called a* negative definite *(nd) kernel if and only if*

$$\sum_{i=1}^{n} \sum_{j=1}^{n} c_i c_j \psi(x_i, x_j) \leq 0$$

for all $n \in \mathbb{N}$, $x_1, \ldots, x_n \in \mathcal{X}$ and $c_1, \ldots, c_n \in \mathbb{R}$, satisfying the additional constraint $c_1 + \ldots + c_n = 0$. In this case, $-\psi$ is called conditionally *pd; obviously, positive definiteness implies conditional positive definiteness.*

The sets of pd and nd kernels are both closed under pointwise sums/integrations, the former being also closed under pointwise products; moreover, both sets are closed under pointwise convergence. While pd kernels "correspond" to inner products via embedding in a Hilbert space, nd kernels that vanish on the diagonal and are positive anywhere else, "correspond" to squared Hilbertian distances. These facts, and the following propositions and lemmas, are shown in Berg et al. (1984).

Proposition 18 *Let $\psi : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ be a symmetric function, and $x_0 \in \mathcal{X}$. Let $\varphi : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ be given by*

$$\varphi(x,y) = \psi(x, x_0) + \psi(y, x_0) - \psi(x,y) - \psi(x_0, x_0).$$

Then, φ is pd if and only if ψ is nd.

Proposition 19 *The function $\psi : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is a nd kernel if and only if $\exp(-t\psi)$ is pd for all $t > 0$.*

Proposition 20 *The function $\psi : \mathcal{X} \times \mathcal{X} \to \mathbb{R}_+$ is a nd kernel if and only if $(t + \psi)^{-1}$ is pd for all $t > 0$.*

Lemma 21 *If ψ is nd and nonnegative on the diagonal, that is, $\psi(x,x) \geq 0$ for all $x \in \mathcal{X}$, then ψ^α, for $\alpha \in [0,1]$, and $\ln(1 + \psi)$, are also nd.*

Lemma 22 *If $f : \mathcal{X} \to \mathbb{R}$ satisfies $f \geq 0$, then, for $\alpha \in [1,2]$, the function $\psi_\alpha(x,y) = -(f(x)+f(y))^\alpha$ is a nd kernel.*

The following definition (Berg et al., 1984) has been used in a machine learning context by Cuturi and Vert (2005).

Definition 23 *Let $(\mathcal{X}, \oplus)$ be a semigroup.*[2] *A function $\varphi : \mathcal{X} \to \mathbb{R}$ is called pd (in the semigroup sense) if $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, defined as $k(x,y) = \varphi(x \oplus y)$, is a pd kernel. Likewise, φ is called nd if k is a nd kernel. Accordingly, these are called* semigroup kernels.

6.2 Jensen-Shannon and Tsallis Kernels

The basic result that allows deriving pd kernels based on the JS divergence and, more generally, on the JT q-difference, is the fact that the denormalized Tsallis q-entropies (10) are nd functions on $(M_+^{S_q}(\mathcal{X}), +)$, for $q \in [0,2]$. Of course, this includes the denormalized Shannon-Boltzmann-Gibbs entropy (8) as a particular case, corresponding to $q = 1$. Although part of the proof was given by Berg et al. (1984) (and by Topsøe 2000 and Cuturi and Vert 2005 for the Shannon entropy case), we present a complete proof here.

Proposition 24 *For $q \in [0,2]$, the denormalized Tsallis q-entropy S_q is a nd function on $(M_+^{S_q}(\mathcal{X}), +)$.*

Proof Since nd kernels are closed under pointwise integration, it suffices to prove that φ_q (see Equation 11) is nd on $(\mathbb{R}_+, +)$. For $q \neq 1$, $\varphi_q(y) = (q-1)^{-1}(y - y^q)$. Let us consider two cases separately: if $q \in [0,1)$, $\varphi_q(y)$ equals a positive constant times $-\iota + \iota^q$, where $\iota(y) = y$ is the identity map defined on $\mathbb{R}_+$. Since the set of nd functions is closed under sums, we only need to show that both $-\iota$ and ι^q are nd. Both ι and $-\iota$ are nd, as can easily be seen from the definition; besides, since ι is nd and nonnegative, Lemma 21 guarantees that ι^q is also nd. For the second case, where $q \in (1,2]$, $\varphi_q(y)$ equals a positive constant times $\iota - \iota^q$. It only remains to show that $-\iota^q$ is nd for $q \in (1,2]$: Lemma 22 guarantees that the kernel $k(x,y) = -(x+y)^q$ is nd; therefore $-\iota^q$ is a nd function.

For $q = 1$, we use the fact that,

$$\varphi_1(x) = \varphi_H(x) = -x \ln x = \lim_{q \to 1} \frac{x - x^q}{q-1} = \lim_{q \to 1} \varphi_q(x),$$

where the limit is obtained by L'Hôpital's rule; since the set of nd functions is closed under limits, $\varphi_1(x)$ is nd. ■

The following lemma, proved in Berg et al. (1984), will also be needed below.

Lemma 25 *The function $\zeta_q : \mathbb{R}_{++} \to \mathbb{R}$, defined as $\zeta_q(y) = y^{-q}$ is pd, for $q \in [0,1]$.*

We are now in a position to present the main contribution of this section, which is a family of *weighted Jensen-Tsallis kernels*, generalizing the JS-based (and other) kernels in two ways: (i) they allow using unnormalized measures; equivalently, they allow using different weights for each of the two arguments; (ii) they extend the mutual information feature of the JS kernel to the nonextensive scenario.

2. Recall that $(\mathcal{X}, \oplus)$ is a *semigroup* if $\oplus$ is a binary operation in $\mathcal{X}$ that is associative and has an identity element.

Definition 26 (weighted Jensen-Tsallis kernels) *The kernel* $\widetilde{k}_q : M_+^{S_q}(\mathcal{X}) \times M_+^{S_q}(\mathcal{X}) \to \mathbb{R}$ *is defined as*

$$\begin{aligned} \widetilde{k}_q(\mu_1,\mu_2) &\triangleq \widetilde{k}_q(\omega_1 p_1, \omega_2 p_2) \\ &= \left(S_q(\pi) - T_q^{\pi}(p_1,p_2)\right)(\omega_1+\omega_2)^q, \end{aligned}$$

where $p_1 = \mu_1/\omega_1$ *and* $p_2 = \mu_2/\omega_2$ *are the normalized counterparts of* μ_1 *and* μ_2*, with corresponding masses* $\omega_1, \omega_2 \in \mathbb{R}_+$*, and* $\pi = (\omega_1/(\omega_1+\omega_2), \omega_2/(\omega_1+\omega_2))$.

The kernel $k_q : \left(M_+^{S_q}(\mathcal{X}) \setminus \{0\}\right)^2 \to \mathbb{R}$ *is defined as*

$$k_q(\mu_1,\mu_2) \triangleq k_q(\omega_1 p_1, \omega_2 p_2) = S_q(\pi) - T_q^{\pi}(p_1,p_2).$$

Recalling (25), notice that $S_q(\pi) - T_q^{\pi}(p_1,p_2) = S_q(T) - I_q(X;T) = S_q(T|X)$ can be interpreted as the *Tsallis posterior conditional entropy*. Hence, k_q can be seen (in Bayesian classification terms) as a nonextensive expected measure of uncertainty in correctly identifying the class, given the prior $\pi = (\pi_1, \pi_2)$, and a sample from the mixture $\pi_1 p_1 + \pi_2 p_2$. The more similar the two distributions are, the greater this uncertainty.

Proposition 27 *The kernel* $\widetilde{k}_q$ *is pd, for* $q \in [0,2]$*. The kernel* k_q *is pd, for* $q \in [0,1]$.

Proof With $\mu_1 = \omega_1 p_1$ and $\mu_2 = \omega_2 p_2$ and using the denormalization formula of Remark 1, we obtain $\widetilde{k}_q(\mu_1,\mu_2) = -S_q(\mu_1+\mu_2) + S_q(\mu_1) + S_q(\mu_2)$. Now invoke Proposition 18 with $\psi = S_q$ (which is nd by Proposition 24), $x = \mu_1$, $y = \mu_2$, and $x_0 = 0$ (the null measure). Observe now that $k_q(\mu_1,\mu_2) = \widetilde{k}_q(\mu_1,\mu_2)(\omega_1+\omega_2)^{-q}$. Since the product of two pd kernels is a pd kernel and (Proposition 25) $(\omega_1+\omega_2)^{-q}$ is a pd kernel, for $q \in [0,1]$, we conclude that k_q is pd. ■

As we can see, the weighted Jensen-Tsallis kernels have two inherent properties: they are parameterized by the entropic index q and they allow their arguments to be unbalanced, that is, to have different weights ω_i. We now mention some instances of kernels where each of these degrees of freedom is suppressed. We start with the following subfamily of kernels, obtained by setting $q = 1$.

Definition 28 (weighted Jensen-Shannon kernels) *The kernel* $\widetilde{k}_{WJS} : (M_+^H(\mathcal{X}))^2 \to \mathbb{R}$ *is defined as* $\widetilde{k}_{WJS} \triangleq \widetilde{k}_1$*, that is,*

$$\begin{aligned} \widetilde{k}_{WJS}(\mu_1,\mu_2) &= \widetilde{k}_{WJS}(\omega_1 p_1, \omega_2 p_2) \\ &= \left(H(\pi) - J^{\pi}(p_1,p_2)\right)(\omega_1+\omega_2), \end{aligned}$$

where $p_1 = \mu_1/\omega_1$ *and* $p_2 = \mu_2/\omega_2$ *are the normalized counterpart of* μ_1 *and* μ_2*, and* $\pi = (\omega_1/(\omega_1+\omega_2), \omega_2/(\omega_1+\omega_2))$.

Analogously, the kernel $k_{WJS} : \left(M_+^H(\mathcal{X}) \setminus \{0\}\right)^2 \to \mathbb{R}$ *is simply* $k_{WJS} \triangleq k_1$*, that is,*

$$k_{WJS}(\mu_1,\mu_2) = k_{WJS}(\omega_1 p_1, \omega_2 p_2) = H(\pi) - J^{\pi}(p_1,p_2).$$

Corollary 29 *The weighted Jensen-Shannon kernels* $\widetilde{k}_{WJS}$ *and* k_{WJS} *are pd.*

Proof Invoke Proposition 27 with $q = 1$. ■

The following family of *weighted exponentiated JS kernels*, generalize the so-called *exponentiated JS* kernel, that has been used, and shown to be pd, by Cuturi and Vert (2005).

Definition 30 (Exponentiated JS kernel) *The kernel $k_{EJS} : M_+^1(\mathcal{X}) \times M_+^1(\mathcal{X}) \to \mathbb{R}$ is defined, for $t > 0$, as*

$$k_{EJS}(p_1, p_2) = \exp\left[-t\, JS(p_1, p_2)\right].$$

Definition 31 (Weighted exponentiated JS kernels) *The kernel $k_{WEJS} : M_+^H(\mathcal{X}) \times M_+^H(\mathcal{X}) \to \mathbb{R}$ is defined, for $t > 0$, as*

$$\begin{aligned} k_{WEJS}(\mu_1, \mu_2) &= \exp[t\, k_{WJS}(\mu_1, \mu_2)] \\ &= \exp(t\, H(\pi)) \exp\left[-tJ^\pi(p_1, p_2)\right]. \end{aligned} \tag{37}$$

Corollary 32 *The kernels k_{WEJS} are pd. In particular, k_{EJS} is pd.*

Proof Results from Proposition 19 and Corollary 29. Notice that although k_{WEJS} is pd, none of its two exponential factors in (37) is pd. ■

We now keep $q \in [0, 2]$ but consider the weighted JT kernel family restricted to normalized measures, $k_q|_{(M_+^1(\mathcal{X}))^2}$. This corresponds to setting uniform weights ($\omega_1 = \omega_2 = 1/2$); note that in this case $\widetilde{k}_q$ and k_q collapse into the same kernel,

$$\widetilde{k}_q(p_1, p_2) = k_q(p_1, p_2) = \ln_q(2) - T_q(p_1, p_2).$$

Proposition 27 guarantees that these kernels are pd for $q \in [0, 2]$. Remarkably, we recover three well-known particular cases for $q \in \{0, 1, 2\}$. We start with the Jensen-Shannon kernel, introduced and shown to be pd by Hein et al. (2004); it is a particular case of a weighted Jensen-Shannon kernel in Definition 28.

Definition 33 (Jensen-Shannon kernel) *The kernel $k_{JS} : M_+^1(\mathcal{X}) \times M_+^1(\mathcal{X}) \to \mathbb{R}$ is defined as*

$$k_{JS}(p_1, p_2) = \ln 2 - JS(p_1, p_2).$$

Corollary 34 *The kernel k_{JS} is pd.*

Proof k_{JS} is the restriction of k_{WJS} to $M_+^1(\mathcal{X}) \times M_+^1(\mathcal{X})$. ■

Finally, we study two other particular cases of the family of Tsallis kernels: the Boolean and linear kernels.

Definition 35 (Boolean kernel) *Let the kernel $k_{\text{Bool}} : M_+^{S_0,1}(\mathcal{X}) \times M_+^{S_0,1}(\mathcal{X}) \to \mathbb{R}$ be defined as $k_{\text{Bool}} = k_0$, that is,*

$$k_{\text{Bool}}(p_1, p_2) = \nu\left(\text{supp}(p_1) \cap \text{supp}(p_2)\right),$$

that is, $k_{\text{Bool}}(p_1, p_2)$ equals the measure of the intersection of the supports (cf. Equation 26). In particular, if $\mathcal{X}$ is finite and ν is the counting measure, the above may be written as

$$k_{\text{Bool}}(p_1, p_2) = \|p_1 \odot p_2\|_0.$$

Definition 36 (Linear kernel) *Let the kernel $k_{\text{lin}} : M_+^{S_2,1}(\mathcal{X}) \times M_+^{S_2,1}(\mathcal{X}) \to \mathbb{R}$ be defined as*

$$k_{\text{lin}}(p_1, p_2) = \frac{1}{2} \langle p_1, p_2 \rangle.$$

Corollary 37 *The kernels k_{Bool} and k_{lin} are pd.*

Proof Invoke Proposition 27 with $q = 0$ and $q = 2$. Notice that, for $q = 2$, we just recover the well-known property of the inner product kernel (Schölkopf and Smola, 2002), which is equal to k_{lin} up to a scalar. ■

In conclusion, the Boolean kernel, the Jensen-Shannon kernel, and the linear kernel are simply particular elements of the much wider family of Jensen-Tsallis kernels, continuously parameterized by $q \in [0,2]$. Furthermore, the Jensen-Tsallis kernels are a particular subfamily of the even wider set of weighted Jensen-Tsallis kernels.

One of the key features of our generalization is that the kernels are defined on unnormalized measures, with arbitrary mass. This is relevant, for example, in applications of kernels on empirical measures (e.g., word counts, pixel intensity histograms); instead of the usual step of normalization Hein et al. 2004, we may leave these empirical measures unnormalized, thus allowing objects of different size (e.g., total number of words in a document, total number of image pixels) to be weighted differently. Another possibility opened by our generalization is the explicit inclusion of weights: given two normalized measures, they can be multiplied by arbitrary (positive) weights before being fed to the kernel function.

6.3 Other Kernels Based on Jensen Differences and q-Differences

It is worthwhile to note that the Jensen-Rényi and the Jensen-Tsallis divergences also yield positive definite kernels, albeit there are not any obvious "weighted generalizations" like the ones presented above for the Tsallis kernels.

Proposition 38 (Jensen-Rényi and Jensen-Tsallis kernels) *For any $q \in [0,2]$, the kernel*

$$(p_1, p_2) \mapsto S_q\left(\frac{p_1 + p_2}{2}\right)$$

and the (unweighted) Jensen-Tsallis divergence J_{S_q} (20) are nd kernels on $M_+^1(\mathcal{X}) \times M_+^1(\mathcal{X})$.

Also, for any $q \in [0,1]$, the kernel

$$(p_1, p_2) \mapsto R_q\left(\frac{p_1 + p_2}{2}\right)$$

and the (unweighted) Jensen-Rényi divergence J_{R_q} (19) are nd kernels on $M_+^1(\mathcal{X}) \times M_+^1(\mathcal{X})$.

Proof The fact that $(p_1, p_2) \mapsto S_q(\frac{p_1+p_2}{2})$ is nd results from the embedding $x \mapsto x/2$ and Proposition 24. Since $(p_1, p_2) \mapsto \frac{S_q(p_1)+S_q(p_2)}{2}$ is trivially nd, we have that J_{S_q} is a sum of nd functions, which turns it nd. To prove the negative definiteness of the kernel $(p_1, p_2) \mapsto R_q\left(\frac{p_1+p_2}{2}\right)$, notice first that the kernel $(x,y) \mapsto (x+y)/2$ is clearly nd. From Lemma 21 and integrating, we have that $(p_1,p_2) \mapsto \int \left(\frac{p_1+p_2}{2}\right)^q$ is nd for $q \in [0,1]$. From the same lemma we have that $(p_1,p_2) \mapsto \ln\left(t + \int \left(\frac{p_1+p_2}{2}\right)^q\right)$ is nd for any $t > 0$. Since $\int \left(\frac{p_1+p_2}{2}\right)^q > 0$, the nonnegativity of $(p_1,p_2) \mapsto R_q\left(\frac{p_1+p_2}{2}\right)$ follows by taking the limit $t \to 0$. By the same argument as above, we conclude that J_{R_q} is nd. ■

As a consequence, we have from Lemma 19 that the following kernels are pd for any $q \in [0,1]$ and $t > 0$:

$$\tilde{k}_{\text{EJR}}(p_1,p_2) = \exp\left(-tR_q\left(\frac{p_1+p_2}{2}\right)\right) = \left(\int \left(\frac{p_1+p_2}{2}\right)^q\right)^{-\frac{t}{1-q}},$$

and its "normalized" counterpart,

$$k_{\text{EJR}}(p_1,p_2) = \exp(-tJ_{R_q}(p_1,p_2)) = \left(\frac{\int \left(\frac{p_1+p_2}{2}\right)^q}{\sqrt{\int p_1^q \int p_2^q}}\right)^{-\frac{t}{1-q}}.$$

Although we could have derived its positive definiteness without ever referring to the Rényi entropy, the latter has in fact a suggestive interpretation: it corresponds to an exponentiation of the Jensen-Rényi divergence; it generalizes the case $q = 1$ which corresponds to the exponentiated Jensen-Shannon kernel.

Finally, we point out a relationship between the Jensen-Tsallis divergences (Section 3.4) and a family of difference kernels introduced by Fuglede (2005),

$$\psi_{\alpha,\beta}(x,y) = \left(\frac{x^\alpha + y^\alpha}{2}\right)^{1/\alpha} - \left(\frac{x^\beta + y^\beta}{2}\right)^{1/\beta}.$$

Fuglede (2005) derived the negative definiteness of the above family of kernels provided $1 \le \alpha \le \infty$ and $1/2 \le \beta \le \alpha$; he went further by providing representations for these kernels. Hein et al. (2004) used the fact that the integral $\int \psi_{\alpha,\beta}(x(t),y(t))d\tau(t)$ is also nd to derive a family of pd kernels for probability measures that included the Jensen-Shannon kernel (see Def. 33).

We start by noting the following property of the extended Tsallis entropy, which is very easy to establish:

$$S_q(\mu) = q^{-1}S_{1/q}(\mu^q)$$

As a consequence, by making the substitutions $r \triangleq q^{-1}$, $x_1 \triangleq y_1^q$ and $x_2 \triangleq y_2^q$, we have that

$$\begin{aligned} J_{S_q}(y_1,y_2) &= S_q\left(\frac{y_1+y_2}{2}\right) - \left(\frac{S_q(y_1)+S_q(y_2)}{2}\right) \\ &= r\left[S_r\left(\left(\frac{x_1^r+x_2^r}{2}\right)^{1/r}\right) - \frac{S_r(x_1)+S_r(x_2)}{2}\right] \\ &\triangleq r\tilde{J}_{S_r}(x_1,x_2) \end{aligned}$$

where we introduced

$$\begin{aligned}\tilde{J}_{S_r}(x_1,x_2) &= S_r\left(\left(\frac{x_1^r+x_2^r}{2}\right)^{1/r}\right)-\frac{S_r(x_1)+S_r(x_2)}{2}\\ &= (r-1)^{-1}\int\left[\left(\frac{x_1^r+x_2^r}{2}\right)^{1/r}-\frac{x_1+x_2}{2}\right]. \end{aligned}\tag{38}$$

Since J_{S_q} is nd for $q\in[0,2]$, we have that $\tilde{J}_{S_r}$ is nd for $r\in[1/2,\infty]$.

Notice that while J_{S_q} may be interpreted as "the difference between the Tsallis q-entropy of the mean and the mean of the Tsallis q-entropies," $\tilde{J}_{S_q}$ may be interpreted as "the difference between the Tsallis q-entropy of the q-power mean and the mean of the Tsallis q-entropies."

From (38) we have that

$$\int\psi_{\alpha,\beta}(x,y)=(\alpha-1)\tilde{J}_{S_\alpha}(x,y)-(\beta-1)\tilde{J}_{S_\beta}(x,y),$$

so the family of probabilistic kernels studied in Hein et al. (2004) can be written in terms of Jensen-Tsallis divergences.

6.4 *k*-th Order Jensen-Tsallis String Kernels

This subsection introduces a new class of string kernels inspired by the k-th order JT q-difference introduced in Section 5.3. Although we refer to them as "string kernels," they are more generally kernels between stochastic processes.

Several string kernels (i.e., kernels operating on the space of strings) have been proposed in the literature (Haussler, 1999; Lodhi et al., 2002; Leslie et al., 2002; Vishwanathan and Smola, 2003; Shawe-Taylor and Cristianini, 2004; Cortes et al., 2004). These are kernels defined on $\mathcal{A}^*\times\mathcal{A}^*$, where $\mathcal{A}^*$ is the Kleene closure of a finite alphabet $\mathcal{A}$ (i.e., the set of all finite strings formed by characters in $\mathcal{A}$ together with the empty string ε). The *p-spectrum kernel* (Leslie et al., 2002) is associated with a feature space indexed by $\mathcal{A}^p$ (the set of length-p strings). The feature representation of a string s, $\Phi^p(s)\triangleq(\phi_u^p(s))_{u\in\mathcal{A}^p}$, counts the number of times each $u\in\mathcal{A}^p$ occurs as a substring of s,

$$\phi_u^p(s)=|\{(v_1,v_2):s=v_1uv_2\}|.$$

The p-spectrum kernel is then defined as the standard inner product in $\mathbb{R}^{|\mathcal{A}|^p}$

$$k_{\mathrm{SK}}^p(s,t)=\langle\Phi^p(s),\Phi^p(t)\rangle.\tag{39}$$

A more general kernel is the *weighted all-substrings kernel* (Vishwanathan and Smola, 2003), which takes into account the contribution of all the substrings weighted by their length. This kernel can be viewed as a conic combination of p-spectrum kernels and can be written as

$$k_{\mathrm{WASK}}(s,t)=\sum_{p=1}^{\infty}\alpha_p k_{\mathrm{SK}}^p(s,t),\tag{40}$$

where α_p is often chosen to decay exponentially with p and truncated; for example, $\alpha_p=\lambda^p$, if $p_{\min}\le p\le p_{\max}$, and $\alpha_p=0$, otherwise, where $0<\lambda<1$ is the decaying factor.

Both k_{SK}^{p} and k_{WASK} are trivially positive definite, the former by construction and the latter because it is a conic combination of positive definite kernels. A remarkable fact is that both kernels may be computed in $O(|s|+|t|)$ time (i.e., with cost that is linear in the length of the strings), as shown by Vishwanathan and Smola (2003), by using data structures such as suffix trees or suffix arrays (Gusfield, 1997). Moreover, with s fixed, any kernel $k(s,t)$ may be computed in time $O(|t|)$, which is particularly useful for classification applications.

We will now see how Jensen-Tsallis kernels may be used as string kernels. In Section 5.3, we have introduced the concept of *joint* and *conditional* JT q-differences. We have seen that joint JT q-differences are just JT q-differences in a product space of the form $\mathcal{X} = \mathcal{X}_1 \times \mathcal{X}_2$; for k-th order joint JT q-differences this product space is of the form $\mathcal{A}^k = \mathcal{A} \times \mathcal{A}^{k-1}$. Therefore, they still yield positive definite kernels as those introduced in Definition 26, where $\mathcal{X} = \mathcal{A}^k$. The next definition and proposition summarize these statements.

Definition 39 (k-th order weighted JT kernels) *Let $\mathscr{S}(\mathcal{A})$ be the set of stationary and ergodic stochastic processes that take values on the alphabet $\mathcal{A}$. For $k \in \mathbb{N}$ and $q \in [0,2]$, let the kernel $\widetilde{k}_{q,k} : (\mathbb{R}_+ \times \mathscr{S}(\mathcal{A}))^2 \to \mathbb{R}$ be defined as*

$$\begin{aligned} \widetilde{k}_{q,k}((\omega_1,s_1),(\omega_2,s_2)) &\triangleq \widetilde{k}_q(\omega_1 p_{s_1,k}, \omega_2 p_{s_2,k}) \\ &= \left(S_q(\pi) - T_{q,k}^{\mathrm{joint},\pi}(s_1,s_2)\right)(\omega_1+\omega_2)^q, \end{aligned} \tag{41}$$

where $p_{s_1,k}$ and $p_{s_2,k}$ are the k-th order joint probability functions associated with the stochastic sources s_1 and s_2, and $\pi = (\omega_1/(\omega_1+\omega_2), \omega_2/(\omega_1+\omega_2))$.

Let the kernel $k_{q,k} : (\mathbb{R}_{++} \times \mathscr{S}(\mathcal{A}))^2 \to \mathbb{R}$ be defined as

$$\begin{aligned} k_{q,k}((\omega_1,s_1),(\omega_2,s_2)) &\triangleq k_q(\omega_1 p_{s_1,k}, \omega_2 p_{s_2,k}) \\ &= \left(S_q(\pi) - T_{q,k}^{\mathrm{joint},\pi}(s_1,s_2)\right), \end{aligned} \tag{42}$$

Proposition 40 *The kernel $\widetilde{k}_{q,k}$ is pd, for $q \in [0,2]$. The kernel $k_{q,k}$ is pd, for $q \in [0,1]$.*

Proof Define the map $g : \mathbb{R}_+ \times \mathscr{S}(\mathcal{A}) \to \mathbb{R}_+ \times M_+^{1,S_q}(\mathcal{A}^k)$ as $(\omega, s) \mapsto g(\omega,s) = (\omega, p_{s,k})$. From Proposition 27, the kernel $\widetilde{k}_q(g(\omega_1,s_1), g(\omega_2,s_2))$ is pd and therefore so is $\widetilde{k}_{q,k}((\omega_1,s_1),(\omega_2,s_2))$; proceed analogously for $k_{q,k}$. ■

At this point, one might wonder whether the "k-th order conditional JT kernel" $\widetilde{k}_{q,k}^{\mathrm{cond}}$ that would be obtained by replacing $T_{q,k}^{\mathrm{joint},\pi}$ with $T_{q,k}^{\mathrm{cond},\pi}$ in (41–42) is also pd. Formula (35) shows that such "conditional JT kernel" is a difference between two joint JT kernels, which is inconclusive. The following proposition shows that $\widetilde{k}_{q,k}^{\mathrm{cond}}$ and $k_{q,k}^{\mathrm{cond}}$ are not pd in general. The proof, which is in Appendix D, proceeds by building a counterexample.

Proposition 41 *Let $\widetilde{k}_{q,k}^{\mathrm{cond}}$ be defined as $\widetilde{k}_{q,k}^{\mathrm{cond}}(s_1,s_2) \triangleq \left(S_q(\pi) - T_{q,k}^{\mathrm{cond},\pi}(s_1,s_2)\right)(\omega_1+\omega_2)^q$; and $k_{q,k}^{\mathrm{cond}}$ be defined as $k_{q,k}^{\mathrm{cond}}(s_1,s_2) \triangleq \left(S_q(\pi) - T_{q,k}^{\mathrm{cond},\pi}(s_1,s_2)\right)$. It holds that $\widetilde{k}_{q,k}^{\mathrm{cond}}$ and $k_{q,k}^{\mathrm{cond}}$ are not pd in general.*

Despite the negative result in Proposition 41, the chain rule in Proposition 15 still allows us to define pd kernels by combining conditional JT q-differences.

Proposition 42 *Let $(\beta_k)_{k\in\mathbb{N}}$ be a non-increasing infinitesimal sequence, that is, satisfying*

$$\beta_0 \geq \beta_1 \geq \ldots \geq \beta_n \to 0$$

Any kernel of the form

$$\sum_{k=0}^{\infty} \beta_k \, \widetilde{k}_{q,k}^{\text{cond}} \tag{43}$$

is pd for $q \in [0,2]$; and any kernel of the form

$$\sum_{k=0}^{\infty} \beta_k \, k_{q,k}^{\text{cond}}$$

is pd for $q \in [0,1]$, provided both series above converge pointwise.

Proof From the chain rule, we have that (defining the 0-th order joint JT q-difference as $\widetilde{k}_{q,0} \triangleq 0$)

$$\sum_{k=0}^{\infty} \beta_k \widetilde{k}_{q,k}^{\text{cond}} = \sum_{k=0}^{\infty} \beta_k \,(\widetilde{k}_{q,k+1} - \widetilde{k}_{q,k}) = \lim_{n\to\infty} \sum_{k=1}^{n} \alpha_k \, \widetilde{k}_{q,k} + \beta_n \, \widetilde{k}_{q,n+1} = \sum_{k=1}^{\infty} \alpha_k \, \widetilde{k}_{q,k} \tag{44}$$

with $\alpha_k = \beta_{k-1} - \beta_k$ (the term $\lim \beta_n \widetilde{k}_{q,n+1}$ was dropped because $\beta_n \to 0$ and $\widetilde{k}_{q,n+1}$ is bounded). Since $(\beta_k)_{k\in\mathbb{N}}$ is non-increasing, we have that $(\alpha_k)_{k\in\mathbb{N}\setminus\{0\}}$ is non-negative, which makes (44) the pointwise limit of a conic combination of pd kernels, and therefore a pd kernel. The proof for $\sum_{k=0}^{\infty} \beta_k k_{q,k}^{\text{cond}}$ is analogous. ■

Notice that if we set $\beta_0 = \ldots = \beta_{k-1} = 1$ and $\beta_j = 0$, $\forall j \geq k$, in the above proposition, we recover the k-th order joint JT q-difference.

Finally, notice that, in the same way that the linear kernel is a special case of a JT kernel when $q = 2$ (see Cor. 37), the p-spectrum kernel (39) is a particular case of a p-th order joint JT kernel, and the weighted all substrings kernel (40) is a particular case of a combination of joint JT kernels in the form (43), both obtained when we set $q = 2$ and the weights ω_1 and ω_2 equal to the length of the strings. Therefore, we conclude that the JT string kernels introduced in this section subsume these two well-known string kernels.

7. Experiments

We illustrate the performance of the proposed nonextensive information theoretic kernels, in comparison with common kernels, for SVM-based text classification. We performed experiments with two standard data sets: *Reuters-21578*[3] and *WebKB*.[4] Since our objective was to evaluate the kernels, we considered a simple binary classification task that tries to discriminate among the two largest categories of each data set; this led us to the *earn-vs-acq* classification task for the first data set, and *stud-vs-fac* (students' *vs.* faculty webpages) in the second data set. Two different frameworks were considered: modeling documents as bags of words, and modeling them as strings of characters. Therefore, both bags of words kernels and string kernels were employed for each task.

3. Available at `www.daviddlewis.com/resources/testcollections`.
4. Available at `www.cs.cmu.edu/afs/cs.cmu.edu/project/theo-20/www/data`.

7.1 Documents as Bags of Words

For the bags of words framework, after the usual preprocessing steps of stemming and stop-word removal, we mapped text documents into probability distributions over words using the bag-of-words model and maximum likelihood estimation; this corresponds to normalizing the *term frequencies* (tf) using the ℓ_1-norm, and is referred to as *tf* (Joachims, 2002; Baeza-Yates and Ribeiro-Neto, 1999). We also used the *tf-idf* (term frequency-inverse document frequency) representation, which penalizes terms that occur in many documents (Joachims, 2002; Baeza-Yates and Ribeiro-Neto, 1999). To weight the documents for the Tsallis kernels, we tried four strategies: uniform weighting, word counts, square root of the word counts, and one plus the logarithm of the word counts; however, for both tasks, uniform weighting revealed the best strategy, which may be due to the fact that documents in both collections are usually short and do not differ much in size.

As baselines, we used the linear kernel with ℓ_2 normalization, commonly used for this task (Joachims, 2002), and the heat kernel approximation introduced by Lafferty and Lebanon (2005):

$$k_{\text{heat}}(p_1,p_2) = (4\pi t)^{-\frac{n}{2}} \exp\left(-\frac{1}{4t}\, d_g^2(p_1,p_2)\right),$$

where $t > 0$ and $d_g(p_1,p_2) = 2\arccos\left(\sum_i \sqrt{p_{1i}p_{2i}}\right)$. Although Lafferty and Lebanon (2005) provide empirical evidence that the heat kernel outperforms the linear kernel, it is not guaranteed to be pd for an arbitrary choice of t, as we show in Appendix E. This parameter and the SVM C parameter were tuned by cross-validation over the training set. The SVM-Light package (available at `http://svmlight.joachims.org/`) was used to solve the SVM quadratic optimization problem.

Figures 2–3 summarize the results. We report the performance of the Tsallis kernels as a function of the entropic index q. For comparison, we also plot the performance of an instance of a Tsallis kernel with q tuned by cross-validation. For the first task, this kernel and the two baselines exhibit similar performance for both the *tf* and the *tf-idf* representations; differences are not statistically significant. In the second task, the Tsallis kernel outperformed the ℓ_2-normalized linear kernel for both representations, and the heat kernel for *tf-idf*; the differences are statistically significant (using the unpaired t test at the 0.05 level). Regarding the influence of the entropic index, we observe that in both tasks, the optimal value of q is usually higher for *tf-idf* than for *tf*.

The results on these two problems are representative of the typical relative performance of the kernels considered: in almost all tested cases, both the heat kernel and the Tsallis kernels (for a suitable value of q) outperform the ℓ_2-normalized linear kernel; the Tsallis kernels are competitive with the heat kernel.

7.2 Documents as Strings

In the second set of experiments, each document is mapped into a probability distribution over character p-grams, using maximum likelihood estimation; we did experiments for $p = 3,4,5$. To weight the documents for the p-th order joint Jensen-Tsallis kernels, four strategies were attempted: uniform weighting, document lengths (in characters), square root of the document lengths, and one plus the logarithm of the document lengths. For the *earn-vs-acq* task, all strategies performed similarly, with a slight advantage for the square root and logarithm of the document lengths; for the *stud-vs-fac* task, uniform weighting revealed the best strategy. For simplicity, all experiments reported here use uniform weighting.

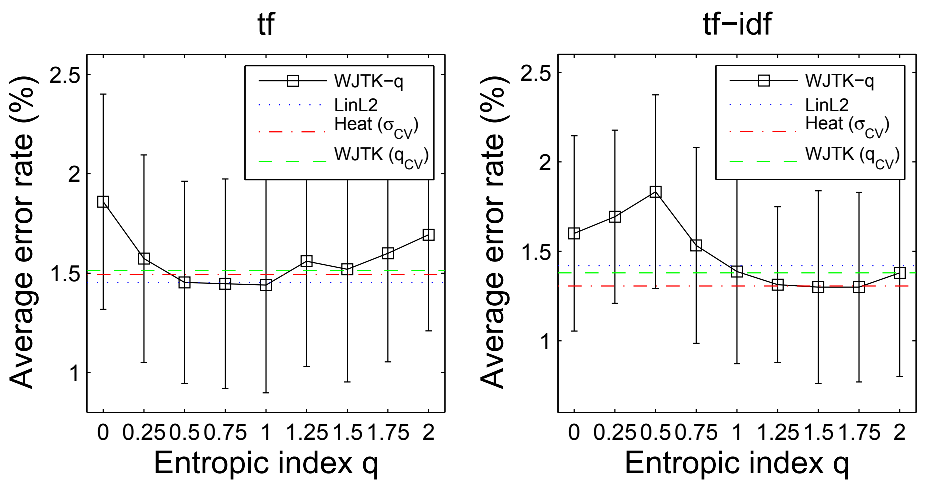


Figure 2: Results for *earn-vs-acq* using *tf* and *tf-idf* representations. The error bars represent ± 1 standard deviation on 30 runs. Training (resp. testing) with 200 (resp. 250) samples per class.

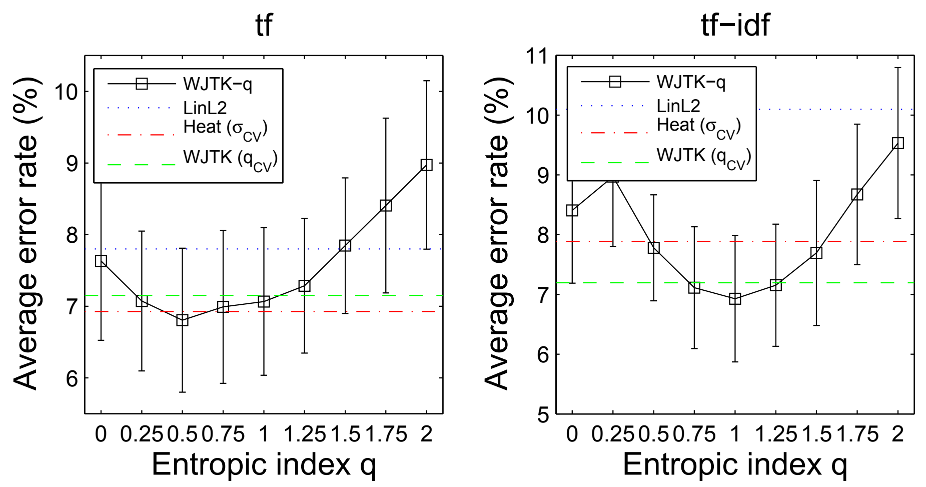


Figure 3: Results for *stud-vs-fac* using *tf* and *tf-idf* representations. The error bars represent ± 1 standard deviation on 30 runs. Training (resp. testing) with 200 (resp. 250) samples per class.

As baselines, we used the p-spectrum kernel (PSK, see 39) for the values of p referred above, and the weighted all substrings kernel (WASK, see 40) with decaying factor tuned to $\lambda = 0.75$ (which yielded the best results), with $p_{\min} = p$ set to the values above, and $p_{\max} = \infty$. The SVM C parameter was tuned by cross-validation over the training set.

Figures 4–5 summarize the results.

For the first task, the JT string kernel and the WASK outperformed the PSK (with statistical significance for $p = 3$), all kernels performed similarly for $p = 4$, and the JT string kernel outperformed the WASK for $p = 5$; all other differences are not statiscally significant. In the second task, the JT string kernel outperformed both the WASK and the PSK (and the WASK outperformed the PSK), with statistical significance for $p = 3,4,5$. Furthermore, by comparing Figures 3 and 5, we

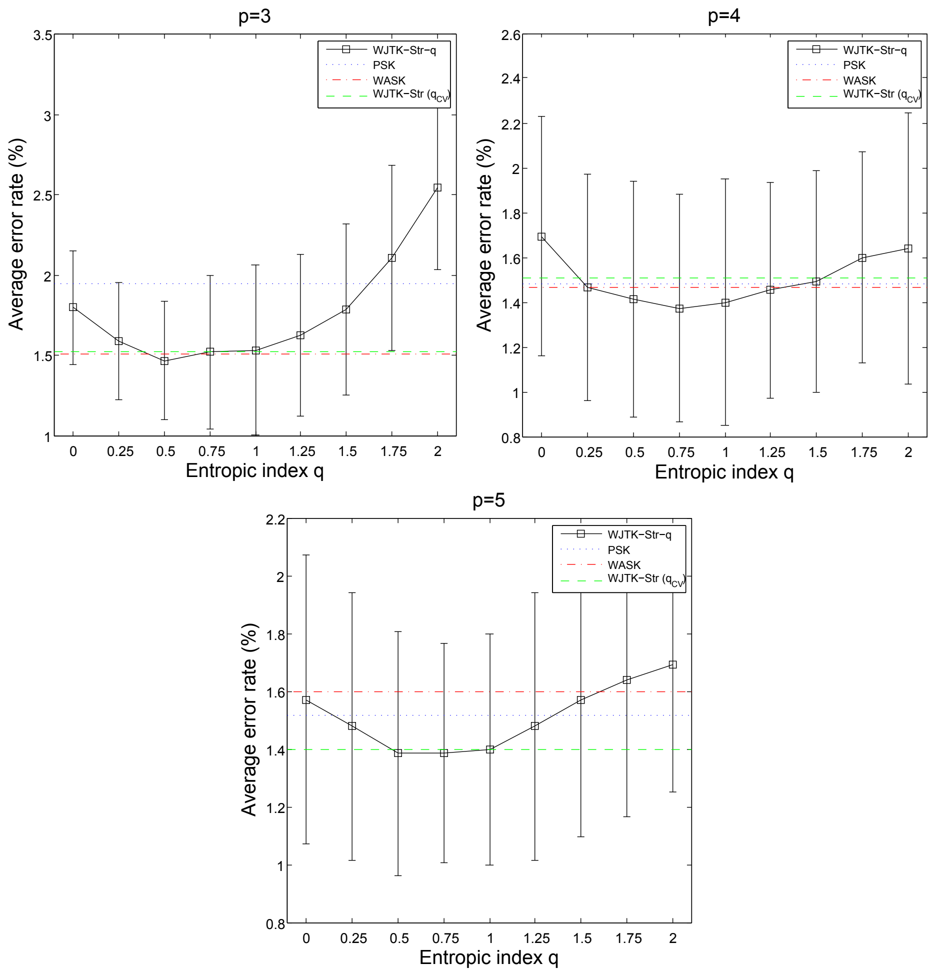


Figure 4: Results for *earn-vs-acq* using string kernels and $p = 3, 4, 5$. The error bars represent ± 1 standard deviation on 15 runs. Training (resp. testing) with 200 (resp. 250) samples per class.

also observe that the 5-th order JT string kernel remarkably outperforms all bags of words kernels for the *stud-vs-fac* task, even though it does not use or build any sort of language model at the word level.

8. Conclusions

In this paper we have introduced a new family of positive definite kernels between measures, which includes previous information-theoretic kernels on probability measures as particular cases. One of the key features of the new kernels is that they are defined on unnormalized measures (not necessarily normalized probabilities). This is relevant, for example, for kernels on empirical measures (such as word counts, pixel intensity histograms); instead of the usual step of normalization (Hein et al., 2004), we may leave these empirical measures unnormalized, thus allowing objects of different sizes (e.g., documents of different lengths, images with different sizes) to be weighted differently. Another possibility is the explicit inclusion of weights: given two normalized measures, they can

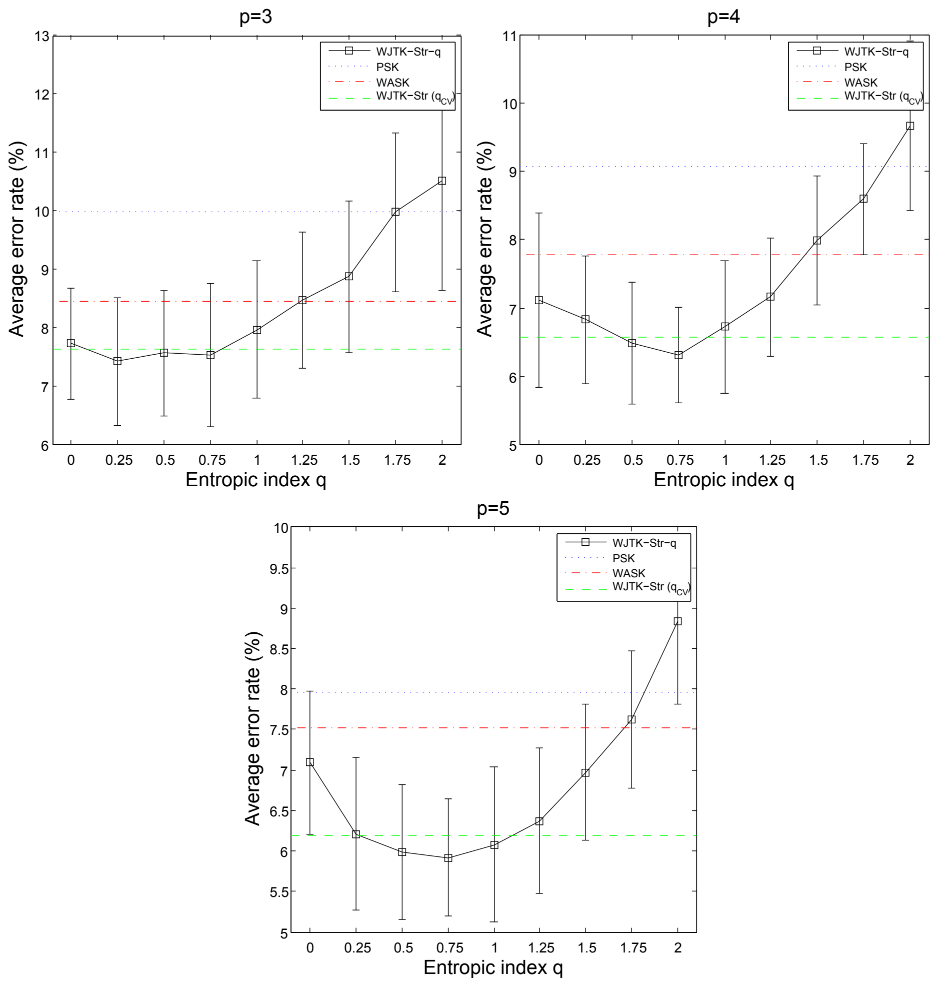


Figure 5: Results for *stud-vs-fac* using string kernels and $p = 3, 4, 5$. The error bars represent ± 1 standard deviation on 15 runs. Training (resp. testing) with 200 (resp. 250) samples per class.

be multiplied by arbitrary (positive) weights before being fed to the kernel function. In addition, we define positive definite kernels between stochastic processes that subsume well-known string kernels.

The new kernels and the proofs of positive definiteness rely on other main contributions of this paper: the new concept of q-convexity, for which we proved a *Jensen q-inequality*; the concept of *Jensen-Tsallis q-difference*, a nonextensive generalization of the Jensen-Shannon divergence; denormalization formulae for several entropies and divergences.

We have reported experiments in which these new kernels were used in support vector machines for text classification tasks. Although the reported experiments do not lead to strong conclusions, they show that the new kernels are competitive with the state-of-the-art, in some cases yielding a significant performance improvement.

Future research will concern applying Jensen-Tsallis q-differences to other learning problems, like clustering, possibly exploiting the fact that they accept more than two arguments.

Acknowledgments

The authors would like to thank the reviewers for their valuable comments, and Guy Lebanon for the fruitful discussions regarding the heat kernel approximation. This work was partially supported by *Fundação para a Ciência e Tecnologia* (FCT), Portugal, grant PTDC/EEA-TEL/72572/2006 and by the European Commission under project SIMBAD. A.M. was supported by a grant from FCT through the CMU-Portugal Program and the Information and Communications Technologies Institute (ICTI) at CMU, and also by Priberam Informática. N.S. was supported by NSF IIS-0713265, DARPA HR00110110013, and an IBM Faculty Award. E.X. was supported by NSF DBI-0546594, DBI-0640543, IIS-0713379, and an Alfred Sloan Foundation Fellowship in Computer Science.

Appendix A. Suyari's Axioms for Nonextensive Entropies

Suyari (2004) proposed the following set of axioms (above referred as Suyari's axioms) that determine nonextensive entropies of the form stated in (1). Below, $q \geq 0$ is any fixed scalar and f_q is a function defined on $\cup_{n=1}^{\infty} \Delta^{n-1}$.

(A1) *Continuity*: $f_q|_{\Delta^{n-1}}$ is continuous, for any $n \in \mathbb{N}$;

(A2) *Maximality*: For any $n \in \mathbb{N}$ and $(p_1, \ldots, p_n) \in \Delta^{n-1}$,

$$f_q(p_1, \ldots, p_n) \leq S_q(1/n, \ldots, 1/n);$$

(A3) *Generalized additivity*: For $i = 1, \ldots, n$, $j = 1, \ldots, m_i$, $p_{ij} \geq 0$, and $p_i = \sum_{j=1}^{m_i} p_{ij}$,

$$\begin{aligned} f_q(p_{11}, \ldots, p_{nm_n}) &= f_q(p_1, \ldots, p_n) + \\ & \sum_{i=1}^{n} p_i^q f_q\left(\frac{p_{i1}}{p_i}, \ldots, \frac{p_{im_i}}{p_i}\right); \end{aligned}$$

(A4) *Expandability*: $f_q(p_1, \ldots, p_n, 0) = f_q(p_1, \ldots, p_n)$.

Appendix B. Proof of Proposition 8

Proof The case $q = 1$ corresponds to the Jensen difference and was proved by Burbea and Rao (1982) (Theorem 1). Our proof extends that to $q \neq 1$. Let $y = (y_1, \ldots, y_m)$, where $y_t = (y_{t1}, \ldots, y_{tn})$. Thus

$$\begin{aligned} T_{q,\Psi}^{\pi}(y) &= \Psi\left(\sum_{t=1}^{m} \pi_t y_t\right) - \sum_{t=1}^{m} \pi_t^q \, \Psi(y_t) \\ &= \sum_{i=1}^{n} \left[\sum_{t=1}^{m} \pi_t^q \varphi(y_{ti}) - \varphi\left(\sum_{t=1}^{m} \pi_t y_{ti}\right)\right], \end{aligned}$$

showing that it suffices to consider $n = 1$, where each $y_t \in [0, 1]$, that is,

$$T_{q,\Psi}^{\pi}(y_1, \ldots, y_m) = \sum_{t=1}^{m} \pi_t^q \varphi(y_t) - \varphi\left(\sum_{t=1}^{m} \pi_t y_t\right);$$

this function is convex on $[0,1]^m$ if and only if, for every fixed $a_1,\dots,a_m \in [0,1]$, and $b_1,\dots,b_m \in \mathbb{R}$, the function

$$f(x) = T^{\pi}_{q,\Psi}(a_1 + b_1 x, \dots, a_m + b_m x)$$

is convex in $\{x \in \mathbb{R} : a_t + b_t x \in [0,1], t = 1,\dots,m\}$. Since f is C^2, it is convex if and only if $f''(t) \geq 0$.

We first show that convexity of f (equivalently of $T^{\pi}_{q,\Psi}$) implies convexity of φ. Letting $c_t = a_t + b_t x$,

$$f''(x) = \sum_{t=1}^{m} \pi_t^q b_t^2 \varphi''(c_t) - \left(\sum_{t=1}^{m} \pi_t b_t\right)^2 \varphi''\left(\sum_{t=1}^{m} \pi_t c_t\right). \tag{45}$$

By choosing $x = 0$, $a_t = a \in [0,1]$, for $t = 1,...,m$, and $b_1,\dots,b_m$ satisfying $\sum_t \pi_t b_t = 0$ in (45), we get

$$f''(0) = \varphi''(a) \sum_{t=1}^{m} \pi_t^q b_t^2,$$

hence, if f is convex, $\varphi''(a) \geq 0$ thus φ is convex.

Next, we show that convexity of f also implies $(2-q)$-convexity of $-1/\varphi''$. By choosing $x = 0$ (thus $c_t = a_t$) and $b_t = \pi_t^{1-q}(\varphi''(a_t))^{-1}$, we get

$$\begin{aligned} f''(0) &= \sum_{t=1}^{m} \frac{\pi_t^{2-q}}{\varphi''(a_t)} - \left(\sum_{t=1}^{m} \frac{\pi_t^{2-q}}{\varphi''(a_t)}\right)^2 \varphi''\left(\sum_{t=1}^{m} \pi_t a_t\right) \\ &= \left[\frac{1}{\varphi''\left(\sum_{t=1}^{m} \pi_t a_t\right)} - \sum_{t=1}^{m} \frac{\pi_t^{2-q}}{\varphi''(a_t)}\right] \left(\sum_{t=1}^{m} \frac{\pi_t^{2-q}}{\varphi''(a_t)}\right) \varphi''\left(\sum_{t=1}^{m} \pi_t a_t\right), \end{aligned}$$

where the expression inside the square brackets is the Jensen $(2-q)$-difference of $1/\varphi''$ (see Definition 7). Since $\varphi''(x) \geq 0$, the factor outside the square brackets is non-negative, thus the Jensen $(2-q)$-difference of $1/\varphi''$ is also nonnegative and $-1/\varphi''$ is $(2-q)$-convex.

Finally, we show that if φ is convex and $-1/\varphi''$ is $(2-q)$-convex, then $f'' \geq 0$, thus $T^{\pi}_{q,\Psi}$ is convex. Let $r_t = (q\pi_t^{2-q}/\varphi''(c_t))^{1/2}$ and $s_t = b_t(\pi_t^q \varphi''(c_t)/q)^{1/2}$; then, non-negativity of f'' results from the following chain of inequalities/equalities:

$$\begin{aligned} 0 &\leq \left(\sum_{t=1}^{m} r_t^2\right)\left(\sum_{t=1}^{m} s_t^2\right) - \left(\sum_{t=1}^{m} r_t s_t\right)^2 && (46) \\ &= \sum_{t=1}^{m} \frac{\pi_t^{2-q}}{\varphi''(c_t)} \sum_{t=1}^{m} b_t^2 \pi_i^q \varphi''(c_t) - \left(\sum_{t=1}^{m} b_t \pi_t\right)^2 && (47) \\ &\leq \frac{1}{\varphi''\left(\sum_{t=1}^{m} \pi_t c_t\right)} \sum_{t=1}^{m} b_t^2 \pi_t^q \varphi''(c_t) - \left(\sum_{t=1}^{m} b_t \pi_t\right)^2 && (48) \\ &= \frac{1}{\varphi''\left(\sum_{t=1}^{m} \pi_t c_t\right)} \cdot f''(t), && (49) \end{aligned}$$

where: (46) is the Cauchy-Schwarz inequality; Equality (47) results from the definitions of r_t and s_t and from the fact that $r_t s_t = b_t \pi_t$; Inequality (48) states the $(2-q)$-convexity of $-1/\varphi''$; equality (49) results from (45). ■

Appendix C. Proof of Proposition 10

Proof The proof of (29), for $q \geq 0$, results from

$$\begin{aligned} T_q^\pi(p_1,\ldots,p_m) &= \frac{1}{q-1}\left[1-\sum_{j=1}^n\left(\sum_{t=1}^m \pi_t p_{tj}\right)^q - \sum_{t=1}^m \pi_t^q\left(1-\sum_{j=1}^n p_{tj}^q\right)\right] \\ &= S_q(\pi)+\frac{1}{q-1}\sum_{j=1}^n\left[\sum_{t=1}^m(\pi_t p_{tj})^q-\left(\sum_{t=1}^m \pi_t p_{tj}\right)^q\right] \\ &\leq S_q(\pi), \end{aligned}$$

where the inequality holds since, for $y_i \geq 0$: if $q \geq 1$, then $\sum_i y_i^q \leq (\sum_i y_i)^q$; if $q \in [0,1]$, then $\sum_i y_i^q \geq (\sum_i y_i)^q$.

The proof that $T_q^\pi \geq 0$ for $q \geq 1$, uses the notion of q-convexity. Since $\mathcal{X}$ is countable, the Tsallis entropy is as in (2), thus $S_q \geq 0$. Since $-S_q$ is 1-convex, then, by Proposition 6, it is also q-convex for $q \geq 1$. Consequently, from the q-Jensen inequality (Proposition 5), for finite $\mathcal{T}$, with $|\mathcal{T}| = m$,

$$T_q^\pi(p_1,\ldots,p_m) = S_q\left(\sum_{t=1}^m \pi_t p_t\right) - \sum_{t=1}^m \pi_t^q S_q(p_t) \geq 0.$$

Since S_q is continuous, so is T_q^π, thus the inequality is valid in the limit as $m \to \infty$, which proves the assertion for $\mathcal{T}$ countable. Finally, $T_q^\pi(\delta_1,\ldots,\delta_1,\ldots) = 0$, where δ_1 is some degenerate distribution.

Finally, to prove (30), for $q \in [0,1]$ and $\mathcal{X}$ finite,

$$\begin{aligned} T_q^\pi(p_1,\ldots,p_m) &= S_q\left(\sum_{t=1}^m \pi_t p_t\right) - \sum_{t=1}^m \pi_t^q S_q(p_t) \\ &\geq \sum_{t=1}^m \pi_t S_q(p_t) - \sum_{t=1}^m \pi_t^q S_q(p_t) \qquad (50) \\ &= \sum_{t=1}^m (\pi_t - \pi_t^q) S_q(p_t) \\ &\geq S_q(U)\sum_{t=1}^m(\pi_t-\pi_t^q) \qquad (51) \\ &= S_q(\pi)[1-n^{1-q}]. \end{aligned}$$

where the Inequality (50) results from S_q being concave, and the Inequality (51) holds since $\pi_t - \pi_t^q \leq 0$, for $q \in [0,1]$, and the uniform distribution U maximizes S_q, with $S_q(U) = (1-n^{1-q})/(q-1)$. ■

Appendix D. Proof of Proposition 41

Proof We show a counterexample with $q = 1$ (the extensive case), $\pi = (1/2, 1/2)$ and $k = 1$, that discards both cases. It suffices to show that $\sqrt{JS_1^{\text{cond}}} \triangleq \sqrt{T_{1,1}^{\text{cond},(1/2,1/2)}}$ violates the triangle inequality for some choice of stochastic processes s_1, s_2, s_3 and therefore is a not a squared distance;

this in turn implies that $\sqrt{JS_1^{\text{cond}}}$ is not nd and, from Proposition 18, that the above two kernels are not pd. We define s_1, s_2, s_3 to be stationary first order Markov processes in a binary alphabet $\mathcal{A} = \{0,1\}$ defined by the following transition matrices, respectively:

$$S_1 = \lim_{\varepsilon \to 0} \begin{bmatrix} 1-\varepsilon & \varepsilon \\ 1/4 & 3/4 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 1/4 & 3/4 \end{bmatrix},$$

$$S_2 = \lim_{\varepsilon \to 0} \begin{bmatrix} 3/4 & 1/4 \\ \varepsilon & 1-\varepsilon \end{bmatrix} = \begin{bmatrix} 3/4 & 1/4 \\ 0 & 1 \end{bmatrix},$$

and

$$S_3 = \lim_{\varepsilon \to 0} \begin{bmatrix} \varepsilon & 1-\varepsilon \\ 1/4 & 3/4 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 1/4 & 3/4 \end{bmatrix},$$

whose stationary distributions are

$$\sigma_1 = \lim_{\varepsilon \to 0} \frac{1}{1+4\varepsilon} \begin{bmatrix} 1 \\ 4\varepsilon \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix},$$

$$\sigma_2 = \lim_{\varepsilon \to 0} \frac{1}{1+4\varepsilon} \begin{bmatrix} 4\varepsilon \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix},$$

and

$$\sigma_3 = \lim_{\varepsilon \to 0} \frac{1}{5-4\varepsilon} \begin{bmatrix} 1 \\ 4-4\varepsilon \end{bmatrix} = \begin{bmatrix} 1/5 \\ 4/5 \end{bmatrix}.$$

The matrix of first order conditional JT 1-differences (or first order conditional Jensen-Shannon divergences) is

$$\begin{bmatrix} 0 & 0 & \frac{3}{5}H(\frac{5}{6}) \\ * & 0 & \frac{9}{10}H(\frac{8}{9}) - \frac{2}{5}H(\frac{1}{4}) \\ * & * & 0 \end{bmatrix} \approx \begin{bmatrix} 0 & 0 & 0.390 \\ * & 0 & 0.128 \\ * & * & 0 \end{bmatrix}, \tag{52}$$

which fails to be negative definite, since

$$\sqrt{JS_1^{\text{cond}}(s_1,s_2)} + \sqrt{JS_1^{\text{cond}}(s_2,s_3)} < \sqrt{JS_1^{\text{cond}}(s_1,s_3)},$$

which violates the triangle inequality required for $\sqrt{JS_1^{\text{cond}}}$ to be a metric.

Interestingly, the 0-th order conditional Jensen-Shannon divergence matrix (this one ensured to be negative definite because it equals a standard Jensen-Shannon divergence matrix) is

$$\begin{bmatrix} 0 & 1 & H(\frac{2}{5}) - \frac{1}{2}H(\frac{1}{5}) \\ * & 0 & H(\frac{1}{10}) - \frac{1}{2}H(\frac{1}{5}) \\ * & * & 0 \end{bmatrix} \approx \begin{bmatrix} 0 & 1 & 0.610 \\ * & 0 & 0.108 \\ * & * & 0 \end{bmatrix}. \tag{53}$$

From the chain rule (35), we have that the sum of the matrices (52) and (53) is the second order joint Jensen-Shannon divergence, and therefore is also guaranteed to be negative definite. ■

Appendix E. The Heat Kernel Approximation

The diffusion kernel for statistical manifolds, recently proposed by Lafferty and Lebanon (2005), is grounded in information geometry (Amari and Nagaoka, 2001). It models the diffusion of "information" over a statistical manifold according to the heat equation. Since in the case of the multinomial manifold (the relative interior of Δ^n), the diffusion kernel has no closed form, the authors adopt the so-called "first-order parametrix expansion," which resembles the Gaussian kernel replacing the Euclidean distance by the geodesic distance that is induced when the manifold is endowed with a Riemannian structure given by the Fisher information (we refer to Lafferty and Lebanon 2005 for further details). The resulting heat kernel approximation is

$$k_{\text{heat}}(p_1, p_2) = (4\pi t)^{-\frac{n}{2}} \exp\left(-\frac{1}{4t}\, d_g^2(p_1, p_2)\right),$$

where $t > 0$ and $d_g(p_1, p_2) = 2 \arccos\left(\sum_i \sqrt{p_{1i} p_{2i}}\right)$. Whether k_{heat} is pd has been an open problem (Hein et al., 2004; Zhang et al., 2005). Let $\mathbb{S}_+^n$ be the positive orthant of the n-dimensional sphere, that is,

$$\mathbb{S}_+^n = \left\{ (x_1, \ldots, x_{n+1}) \in \mathbb{R}^{n+1} \mid \sum_{i=1}^{n+1} x_i^2 = 1,\ \forall i\, x_i \geq 0 \right\}.$$

The problem can be restated as follows: is there an isometric embedding from $\mathbb{S}_+^n$ to some Hilbert space? In this section we answer that question in the negative.

Proposition 43 *Let $n \geq 2$. For sufficiently large t, the kernel k_{heat} is* not *pd.*

Proof From Proposition 19, k_{heat} is pd, for all $t > 0$, if and only if d_g^2 is nd. We provide a counterexample, using the following four points in Δ^2: $p_1 = (1,0,0)$, $p_2 = (0,1,0)$, $p_3 = (0,0,1)$ and $p_4 = (1/2, 1/2, 0)$. The squared distance matrix $[D_{ij}] = [d_g^2(p_i, p_j)]$ is

$$D = \frac{\pi^2}{4} \cdot \begin{bmatrix} 0 & 4 & 4 & 1 \\ 4 & 0 & 4 & 1 \\ 4 & 4 & 0 & 4 \\ 1 & 1 & 4 & 0 \end{bmatrix}.$$

Taking $c = (-4, -4, 1, 7)$ we have $c^T D c = 2\pi^2 > 0$, showing that D is not nd. Although p_1, p_2, p_3, p_4 lie on the boundary of Δ^2, continuity of d_g^2 implies that it is not nd on the relative interior of Δ^2. The case $n > 2$ follows easily, by appending zeros to the four vectors above. ■

References

S. Abe. Foundations of nonextensive statistical mechanics. In *Chaos, Nonlinearity, Complexity*. Springer, 2006.

S. Amari and H. Nagaoka. *Methods of Information Geometry*. Oxford University Press, 2001.

R. Baeza-Yates, and B. Ribeiro-Neto. *Modern information retrieval*. ACM Press New York, 1999.

A. Banerjee, S. Merugu, I. Dhillon, and J. Ghosh. Clustering with Bregman divergences. *J. of Mach. Learn. Res.*, 6:1705–1749, 2005.

A. Ben-Hamza. A nonextensive information-theoretic measure for image edge detection. *J. of Electronic Imaging*, 15-1:13011.1–13011.8, 2006.

A. Ben-Hamza and H. Krim. Image registration and segmentation by maximizing the jensen-rényi divergence. In *Proc. of Int. Workshop on Energy Minimization Methods in Computer Vision and Pattern Recognition*, pages 147–163. Springer, 2003.

C. Berg, J. Christensen, and P. Ressel. *Harmonic Analysis on Semigroups*. Springer, Berlin, 1984.

J. Burbea and C. Rao. On the convexity of some divergence measures based on entropy functions. *IEEE Trans. on Information Theory*, 28(3):489–495, 1982.

C. Cortes, P. Haffner and M. Mohri. Rational Kernels: Theory and Algorithms. *J. of Mach. Learn. Res.*, 5:1035–1062, 2004.

T. Cover and J. Thomas. *Elements of Information Theory*. Wiley, 1991.

I. Csiszar. I-divergence geometry of probability distributions and minimization problems. *The Annals of Probability*, 3(1):146–158, 1975.

M. Cuturi and J.-P. Vert. Semigroup kernels on finite sets. In L. Saul, Y. Weiss, and L. Bottou, editors, *Advances in Neural Information Processing Systems 17*, pages 329–336. MIT Press, Cambridge, MA, 2005.

M. Cuturi, K. Fukumizu, and J.-P. Vert. Semigroup kernels on measures. *J. of Mach. Learn. Res.*, 6:1169–1198, 2005.

Z. Daróczy. Generalized information functions. *Information and Control*, 16(1):36–51, 1970.

F. Desobry, M. Davy, and W. Fitzgerald. Density kernels on unordered sets for kernel-based signal processing. In *Proc. of the IEEE Int. Conf. on Acoustics, Speech, and Signal Proc.*, 2007.

R. El-Yaniv, S. Fine, and N. Tishby. Agnostic classification of markovian sequences. In M. Jordan, M. Kearns, and S. Solla, editors, *Advances in Neural Information Processing Systems 10*, pages 465–471. MIT Press, Cambridge, MA, 1998.

D. Endres and J. Schindelin. A new metric for probability distributions. *IEEE Trans. on Information Theory*, 49(7):1858–1860, 2003.

B. Fuglede. Spirals in Hilbert space, with an application in information theory. *Expositiones Mathematicae*, 25(1):23–46, 2005.

S. Furuichi. Information theoretical properties of Tsallis entropies. *J. of Mathematical Physics*, 47 (2), 2006.

M. Gell-Mann and C. Tsallis. *Nonextensive Entropy: Interdisciplinary Applications*. Oxford University Press, 2004.

I. Grosse, P. Bernaola-Galvan, P. Carpena, R. Roman-Roldan J. Oliver, and H. E. Stanley. Analysis of symbolic sequences using the Jensen-Shannon divergence. *Phys. Review E*, 65, 2002.

D. Gusfield. *Algorithms on Strings, Trees, and Sequences*. Cambridge University Press, 1997.

D. Haussler. Convolution kernels on discrete structures. In Technical Report UCS-CRL-99-10, 1999.

M. Havrda and F. Charvát. Quantification method of classification processes: concept of structural α-entropy. *Kybernetika*, 3:30–35, 1967.

Y. He, A. Ben-Hamza, and H. Krim. A generalized divergence measure for robust image registration. *IEEE Trans. on Signal Proc.*, 51(5):1211–1220, 2003.

M. Hein and O. Bousquet. Hilbertian metrics and positive definite kernels on probability measures. In Z. Ghahramani and R. Cowell, editors, *Proc. of the 10th Int. Workshop on Artificial Intell. and Stat.*, 2005.

M. Hein, T. Lal, and O. Bousquet. Hilbertian metrics on probability measures and their application in SVMs. In *Proc. of the 26th DAGM Symposium*, pages 270–277. Springer, 2004.

T. Jebara, R. Kondor, and A. Howard. Probability product kernels. *J. of Mach. Learn. Res.*, 5: 819–844, 2004.

J. Jensen. Sur les fonctions convexes et les inégalités entre les valeurs moyennes. *Acta Mathematica*, 30:175–193, 1906.

T. Joachims. *Learning to Classify Text Using Support Vector Machines: Methods, Theory and Algorithms*. Kluwer Academic Publishers, 2002.

D. Karakos, S. Khudanpur, J. Eisner, and C. Priebe. Iterative denoising using Jensen-Rényi divergences with an application to unsupervised document categorization. In *Proc. of IEEE Int. Conf. on Acoustics, Speech and Signal Proc.*, Baltimore, MD, 2007.

A. Khinchin. *Mathematical Foundations of Information Theory*. Dover, New York, 1957.

S. Kullback and R.A. Leibler. On information and sufficiency. *The Annals of Mathematical Statistics*, 79–86, 1951.

J. Lafferty and G. Lebanon. Diffusion kernels on statistical manifolds. *J. of Mach. Lear. Res.*, 6: 129–163, 2005.

P. Lamberti and A. Majtey. Non-logarithmic Jensen-Shannon divergence. *Physica A: Statistical Mechanics and its Applications*, 329:81–90, 2003.

C. Leslie, E. Eskin, and W. Noble. The spectrum kernel: A string kernel for svm protein classification. In *Proc. of the Pacific Symposium on Biocomputing*, pages 564–575, 2002.

Y. Li, X. Fan, and G. Li. Image segmentation based on Tsallis-entropy and Renyi-entropy and their comparison. In *IEEE Int. Conf. on Industrial Informatics*, pages 943–948, 2006.

J. Lin. Divergence measures based on the Shannon entropy. *IEEE Trans. on Information Theory*, 37(1):145–151, 1991.

J. Lin and S. Wong. A new directed divergence measure and its characterization. *Int. J. of General Systems*, 17:73–81, 1990.

J. Lindhard. *On the Theory of Measurement and Its Consequences in Statistical Dynamics*. Munksgaard, Copenhagen, 1974.

J. Lindhard and V. Nielsen. *Studies in Statistical Dynamics*. Munksgaard, Copenhagen, 1971.

H. Lodhi, C. Saunders, J. Shawe-Taylor, N. Cristianini, and C. Watkins. Text classification using string kernels. *J. of Mach. Learn. Res.*, 2:419–444, 2002.

A. F. T. Martins, P. M. Q. Aguiar, and M. A. T. Figueiredo. Tsallis kernels on measures. In *Proc. of the IEEE Information Theory Workshop*, Porto, Portugal, 2008a.

A. F. T. Martins, M. A. T. Figueiredo, P. M. Q. Aguiar, N. A. Smith, and E. P. Xing. Nonextensive entropic kernels. In *Proc. of the Int. Conf. on Machine Learning – ICML'08*, Helsinki, Finland, 2008b.

P. Moreno, P. Ho, and N. Vasconcelos. A Kullback-Leibler divergence based kernel for SVM classification in multimedia applications. In S. Thrun, L. Saul, and B. Schölkopf, editors, *Advances in Neural Information Processing Systems 16*. MIT Press, 2004.

A. Rényi. On measures of entropy and information. In *Proc. of the 4th Berkeley Symposium on Mathematics, Statistics, and Probability*, volume 1, pages 547–561, Berkeley, 1961. University of California Press.

B. Schölkopf and A. J. Smola. *Learning with Kernels*. The MIT Press, Cambridge, MA, 2002.

C. Shannon and W. Weaver. *The Mathematical Theory of Communication*. University of Illinois Press, Urbana, Ill., 1949.

C. E. Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27 (3):379–423, 1948.

J. Shawe-Taylor and N. Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, 2004.

J. Steele. *The Cauchy-Schwarz Master Class*. Cambridge University Press, Cambridge, 2006.

H. Suyari. Generalization of Shannon-Khinchin axioms to nonextensive systems and the uniqueness theorem for the nonextensive entropy. *IEEE Trans. on Information Theory*, 50(8):1783–1787, 2004.

F. Topsøe. Some inequalities for information divergence and related measures of discrimination. *IEEE Trans. on Information Theory*, 46(4):1602–1609, 2000.

C. Tsallis. Possible generalization of Boltzmann-Gibbs statistics. *J. of Statistical Physics*, 52: 479–487, 1988.

S. Vishwanathan and A. Smola. Fast kernels for string and tree matching. In K. Tsuda, B. Schölkopf, and J.P. Vert, editors, *Kernels and Bioinformatics*, Cambridge, MA, 2003. MIT Press.

D. Zhang, X. Chen, and W. Lee. Text classification with kernels on the multinomial manifold. In *Proc. of the 28th Annual Int. ACM SIGIR Conf. on Research and Development in Information Retrieval*, pages 266–273, New York, NY, 2005.

Journal of Machine Learning Research 10 (2009) 977-996 Submitted 2/08; Revised 9/08; Published 4/09

On Uniform Deviations of General Empirical Risks with Unboundedness, Dependence, and High Dimensionality

Wenxin Jiang WJIANG@NORTHWESTERN.EDU
Department of Statistics
Northwestern University
Evanston, IL 60208, USA

Editor: Gábor Lugosi

Abstract

The statistical learning theory of risk minimization depends heavily on probability bounds for uniform deviations of the empirical risks. Classical probability bounds using Hoeffding's inequality cannot accommodate more general situations with unbounded loss and dependent data. The current paper introduces an inequality that extends Hoeffding's inequality to handle these more general situations. We will apply this inequality to provide probability bounds for uniform deviations in a very general framework, which can involve discrete decision rules, unbounded loss, and a dependence structure that can be more general than either martingale or strong mixing. We will consider two examples with high dimensional predictors: autoregression (AR) with ℓ_1-loss, and ARX model with variable selection for sign classification, which uses both lagged responses and exogenous predictors.

Keywords: dependence, empirical risk, probability bound, unbounded loss, uniform deviation

1. Introduction

In machine learning, a problem of central importance is to bound a probability of uniform deviation $P[\sup_{b\in B}|n^{-1}\sum_{t=1}^{n}\rho(\omega_t,b)-n^{-1}\sum_{t=1}^{n}E\rho(\omega_t,b)|>\delta]$, where $\delta>0$ is a positive deviation (which can be allowed to depend on n and characterize a convergence rate), b is a parameter in a parameter space B (which is typically a Borel measurable subset of an Euclidean space), $D=(\omega_1,...,\omega_n)$ form the data set of n random observations, $\rho(\cdot,\cdot)$ is a loss function (measurable to a certain product σ-field), $\hat{R}(b)=n^{-1}\sum_{t=1}^{n}\rho(\omega_t,b)$ is an empirical risk, and $R(b)=n^{-1}\sum_{t=1}^{n}E\rho(\omega_t,b)$ is its expectation.[1]

Such a probability is of interest since it is well known to bound the performance $R(\hat{b})$ of an empirical risk minimizer $\hat{b}=\arg\min_{b\in B}\hat{R}(b)$, relative to the optimal performance $\inf_{b\in B}R(b)$ over B:

$$P[R(\hat{b})-\inf_{b\in B}R(b)>2\delta]\le P[\sup_{b\in B}|\hat{R}(b)-R(b)|>\delta],$$

due to, for example, Lemma 8.2 (Devroye, Györfi and Lugosi, 1996). Recently, Jiang and Tanner (2007, 2008) indicate that the probability of uniform deviation is also of central importance in

1. In this paper, we will not be concerned about the measurability problem that may be involved in quantities such as $\sup_{b\in B}|\hat{R}(b)-R(b)|$. Works on 'universal measurability' described in, for example, Yu (1994, Appendix) and Davidson (1994, Section 21.1) imply that this is not a problem for all our examples later where B is a Borel measurable subset of a compact metric space. Alternatively we could consider P as the 'outer probability' as in Newey (1991).

studying the performance of a Bayesian approach of empirical risk minimization considered in, for example, Zhang (2006), when b is generated randomly according to a Gibbs posterior $\pi_{b|D}(db) \propto e^{-n\psi\hat{R}(b)}\pi_b(db)$ where $\pi_b(db)$ is a prior distribution on B and $\psi^{-1} > 0$ is a 'temperature parameter'. This includes the usual Bayesian posterior as a special case when $\psi = 1$ and when $-n\hat{R}$ is the log-likelihood function. A straightforward application of Jiang and Tanner (2008, Proposition 6) renders

$$P[R(b) - \inf_{b\in B} R(b) > 5\delta] \leq P[\sup_{b\in B} |\hat{R}(b) - R(b)| > \delta] + e^{-2n\psi\delta}/\pi_b[R(b) - \inf_{b\in B} R(b) < \delta],$$

when $b|D \sim \pi_{b|D}$ and D is generated from a true distribution. This again shows the dependence of the risk performance on the probability of uniform deviation.

The probability of uniform deviation is treated in the standard machine learning text such as Devroye, Györfi and Lugosi (1996) by the Vapnik-Chervonenkis theory using a Hoeffding's inequality on the probability of pointwise deviation $P[|n^{-1}\sum_{t=1}^n \rho(\omega_t, b) - n^{-1}\sum_{t=1}^n E\rho(\omega_t, b)| > \delta]$, which typically assumes that ω_t's are iid (independent and identically distributed), and that the loss function ρ is bounded. The goal of this paper is to generalize in several directions, so that ρ can be unbounded and ω_t's can be dependent. In addition, we will allow b to have a possibly high dimension that can increase with n in certain ways. When ρ has sufficiently thin tail in the distribution and when ω_t's have certain kind of decaying dependence over t, we derive bounds of the form

$$P[\sup_{b\in B} |n^{-1}\sum_{t=1}^n \rho(\omega_t, b) - n^{-1}\sum_{t=1}^n E\rho(\omega_t, b)| > n^{-0.5+\gamma_1}] = O(e^{-c_1 n^{c_2}}),$$

for any small positive γ_1, where c_1, c_2 are some positive constants depending on γ_1. Such a result indicates uniform convergence of the empirical risk at a near 'parametric' rate (close to $O_P(n^{-0.5})$) despite high dimensionality in b, dependence in ω_t, and unbounded loss function ρ.

Such results are obtained using a very general 'pointwise' inequality that generalizes Hoeffding's inequality, which will be introduced in Section 2. This allows unbounded loss and a framework of dependence that is more general than strong mixing, which is therefore more general than previous works using strong mixing (e.g., Vidyasagar, 2005; Zou and Li, 2007) or β-mixing (e.g., Yu, 1994; Lozano, Kulkarni and Schapire, 2006). The 'uniform aspect' is then treated in a very general framework in Section 3 allowing both continuity and discontinuity of ρ in b. Examples of applications of this general framework are given in Sections 4 and 5.

2. An Inequality

We first introduce an inequality that is more general than Hoeffding's inequality (Hoeffding, 1963). This inequality may be called a 'triplex inequality' since its right hand side has three parts. In addition to a term that is of an exponential form as in the Hoeffding's inequality, it also includes a term to gauge the dependence, and a term to control the unboundedness of the random variables. The result is therefore almost assumption free and generally applicable: it does not assume independence or boundedness of the random variables.

Theorem 1 *(A triplex inequality.) let $\{\mathcal{F}_t\}_{-\infty}^{\infty}$ be an increasing sequence of σ-fields and ρ_t be a random variable that is $\mathcal{F}_t$-measurable for each t. Then for any $\varepsilon, C > 0$ and positive integers n, m,* *we have*

$$P[|\sum_{t=1}^{n}(\rho_t - E\rho_t)| > n\varepsilon] \leq 2me^{-n\varepsilon^2/(288m^2C^2)}$$
$$+(6/\varepsilon)n^{-1}\sum_{t=1}^{n} E|E(\rho_t|\mathcal{F}_{t-m}) - E\rho_t|$$
$$+(15/\varepsilon)n^{-1}\sum_{t=1}^{n} E|\rho_t|I(|\rho_t| > C),$$

as long as the right hand side exists and does not exceed one.

2.1 Remarks

1. The bound is not necessarily very tight; the constants appearing in the theorem may be improved. However, these typically do not affect the convergence rates in the later applications of this inequality.

2. In later applications, the choices of m and C can be made to depend on n so that the combination of all three terms converge to zero as $n \to \infty$.

3. The last term will be called the 'tail term' since it is related to the tail behavior of ρ_t. This often can be bounded by techniques similar to the Markov inequalities. Note that for nonnegative $X = |\rho_t|$, $EXI(X > C) \leq EX^{k+1}C^{-k}$ and $EXI(X > C) \leq \sqrt{EX^2}\sqrt{P(X > C)} \leq \sqrt{EX^2}\sqrt{Ee^{\theta X}}e^{-\theta C/2}$, for $k, \theta > 0$. So existence of moments of X will imply a power law and existence of the moment generating function in a neighborhood of zero will imply an exponential law for the decay of the 'tail term' in C.

4. The second term will be called the 'dependence term' since it is related to the dependence described in the framework of L_1-mixingale (see, e.g., Chapter 16, Davidson, 1994), which is more general than either martingale or strong mixing. When $\{\rho_t\}$ is a sequence of martingale differences, the dependence term vanishes. If $\{\rho_t\}$ is strong mixing with coefficients α_m, and has bounded L_q norms ($q > 1$), then Theorem 14.2 of Davidson (1994) would imply that the dependence term decreases according to order $O(\alpha_m^{1-1/q})$ as m increases.

5. The mixingale formulation of the dependence term can also handle a process ρ_t that is *not* strong mixing. We will provide an example below when ρ_t is not strong mixing but is *approximable* to a strong mixing process, where we can still make the dependence term small for large m. Such an extension from 'strong mixing' to 'approximable by strong mixing', although seemingly a small improvement, is very significant. The problem of strong mixing is that a function of a mixing sequence (even an independent sequence) that depends on an infinite number of lags is not generally mixing. This is regarded as a 'serious drawback from the viewpoint of applications in time-series modelling' (Davidson, 1994, p.261), and has led to the 'approximability' framework summarized in Davidson (1994, Chapter 17), which is popular in modern time series study but has not been paid much attention to by the machine learning society. Our work can incorporate this approximability concept and provide a 'bridge' introducing this framework to our field.

2.2 An Example for the Dependence Term

Suppose that $\{\rho_t\}_{t=-\infty}^{\infty}$ can be approximated in an L_1-sense by a strong mixing sequence $\{\rho_{t,k}\}_{t=-\infty}^{\infty}$ as k increases (where $\rho_{t,k}$ is measurable -$\mathcal{F}_t$ for each t):

$$n^{-1}\sum_{t=1}^{n} E|\rho_t - \rho_{t,k}| \le c_1\nu_k,$$

where $c_1 > 0$ and $\nu_k > 0$ are nonstochastic and ν_k decreases to zero as $k \to \infty$. Suppose the qth moment $||\rho_{t,k}||_q \equiv (E|\rho_{t,k}|^q)^{1/q} \le c_2$ for some constants $q > 1$, $c_2 > 0$. Then Theorem 14.2 of Davidson (1994) implies that

$$n^{-1}\sum_{t=1}^{n} E|E(\rho_{t,k}|\mathcal{F}_{t-m}) - E\rho_{t,k}| \le 6c_2\alpha_m(\{\rho_{t,k}\}_{t=-\infty}^{\infty})^{1-1/q}.$$

Then apply the triangular inequality and note that the dependence term is proportional to

$$\begin{aligned} n^{-1}\sum_{t=1}^{n} E|E(\rho_t|\mathcal{F}_{t-m}) - E\rho_t| &\le n^{-1}\sum_{t=1}^{n} E|E(\rho_{t,k}|\mathcal{F}_{t-m}) - E\rho_{t,k}| + 2n^{-1}\sum_{t=1}^{n} E|\rho_t - \rho_{t,k}| \\ &\le 6c_2\alpha_m(\{\rho_{t,k}\}_{t=-\infty}^{\infty})^{1-1/q} + 2c_1\nu_k. \end{aligned}$$

We may be able to choose $k = k(m)$ to increase with m somehow so that both terms above are small for large m. Such a choice $k(m)$ depends on the mechanism of approximation. When k indicates the number of lags involved as in the following example, one can choose $k(m) \approx m/2$.

For example, consider an $MA(\infty)$ process $\rho_t = \sum_{j=0}^{\infty}\theta_j V_{t-j}$ where $\{V_t\}_{-\infty}^{\infty}$ is a zero-mean, L_q-bounded sequence (i.e., $\sup_t ||V_t||_q < \infty$) for some $q > 1$. (We can take $\mathcal{F}_t$ to be the σ-field generated by $\{V_s\}_{s=-\infty}^{t}$.) Then ρ_t is not necessarily strong mixing even when V_t's are independent innovations, even when $|\theta_j|$ decreases very rapidly, due to the infinitely many lags involved (see, e.g., Section 14.3, Davidson 1994). On the other hand, when $|\theta|_1 \equiv \sum_1^{\infty}|\theta_j| < \infty$, we can define 'finite-lag' approximators $\rho_{t,k} = \sum_{j=0}^{k}\theta_j V_{t-j}$ so that $E|\rho_t - \rho_{t,k}| = E|\sum_{k+1}^{\infty}\theta_j V_{t-j}| \le \sup_t ||V_t||_1 \sum_{k+1}^{\infty}|\theta_j|$ which is of the form $c_1\nu_k$ where $\nu_k = \sum_{k+1}^{\infty}|\theta_j| \to 0$ as $k \to \infty$.

Suppose V_t is strong mixing (e.g., when innovations are independent) with mixing coefficient $\alpha_m(\{V_t\}_{-\infty}^{\infty})$. Then the strong mixing coefficient of $\rho_{t,k}$ satisfies

$$\alpha_m(\{\rho_{t,k}\}_{t=-\infty}^{\infty}) \le \alpha_{m-k}(\{V_t\}_{-\infty}^{\infty}),$$

since the $\rho_{t,k}$ depends on lags $V_t, V_{t-1}, ..., V_{t-k}$. Note that $||\rho_{t,k}||_q \le \sum_{j=0}^{k}|\theta_j|\sup_t ||V_{t-k}||_q \le |\theta|_1 \sup_t ||V_t||_q$ which can be taken as the constant c_2.

Now note that the dependence term of interest is proportional to

$$\begin{aligned} & n^{-1}\sum_{t=1}^{n} E|E(\rho_t|\mathcal{F}_{t-m}) - E\rho_t| \\ & \le 6c_2\alpha_m(\{\rho_{t,k}\}_{t=-\infty}^{\infty})^{1-1/q} + 2c_1\nu_k \\ & \le 6c_2\alpha_{m-k}(\{V_t\}_{-\infty}^{\infty})^{1-1/q} + 2c_1\nu_k. \end{aligned}$$

Then one can take, for example, $k = \lceil m/2 \rceil$ (the integer part of $m/2$) and make the upperbound small for large m.

This shows that the current formulation of the inequality can handle dependence that is more general than strong mixing.

2.3 Proving the Triplex Inequality

The idea behind an upperbound with a decomposition into such three terms has appeared in econometric literature. For example, de Jong and Woutersen (2004) used this idea to treat an unbounded sum appearing in the binary choice models. The idea of our proof is related to a mixingale treatment seen in, for example, Chapter 16 of Davidson (1994). Since the specific form of the current inequality is not seen in these literatures, we will provide a proof below for completeness. The following Lemma will be used in the proof.

Lemma 1 *let $\{\mathcal{F}_t\}_{-\infty}^{\infty}$ be an increasing sequence of σ-fields. Let X_t be a random variable that is $\mathcal{F}_t$-measurable and is bounded so that $|X_t| \leq C$ for some constant C for each t. Then for any $\varepsilon > 0$ and positive integers n, m, we have*

$$P[|\sum_{t=1}^{n} X_t - E\sum_{t=1}^{n} X_t| > n\varepsilon] \leq 2me^{-n\varepsilon^2/(32m^2C^2)} + (2/\varepsilon)n^{-1}\sum_{t=1}^{n} E|E(X_t|\mathcal{F}_{t-m}) - EX_t|, \tag{1}$$

as long as the right hand side exists.

Proof for Lemma 1 Consider $U_n \equiv \sum_{t=1}^{n} X_t - E\sum_{t=1}^{n} X_t$, which can be 'telescoped' into $U_n = U_{1,n} + U_{2,n} + ... + U_{m,n} + V_n$ where $U_{1,n} = \{X_1 - E(X_1|\mathcal{F}_{1-1})\} + ... + \{X_n - E(X_n|\mathcal{F}_{n-1})\}$, $U_{2,n} = \{E(X_1|\mathcal{F}_{1-1}) - E(X_1|\mathcal{F}_{1-2})\} + ... + \{E(X_n|\mathcal{F}_{n-1}) - E(X_n|\mathcal{F}_{n-2})\}$,..., $U_{m,n} = \{E(X_1|\mathcal{F}_{1-(m-1)}) - E(X_1|\mathcal{F}_{1-m})\} + ... + \{E(X_n|\mathcal{F}_{n-(m-1)}) - E(X_n|\mathcal{F}_{n-m})\}$, $V_n = \{E(X_1|\mathcal{F}_{1-m}) - EX_1\} + ... + \{E(X_n|\mathcal{F}_{n-m}) - EX_n\}$. Then a union bound leads to

$$\begin{aligned}
&P[|U_n| > n\varepsilon] \\
&\leq P[|U_{1,n}| > n\varepsilon/(2m)] + P[|U_{2,n}| > n\varepsilon/(2m)] + ... + P[|U_{m,n}| > n\varepsilon/(2m)] \\
&+P[|V_n| > n\varepsilon/2].
\end{aligned} \tag{2}$$

Note that $U_{1,n}$ is a sum of n martingale differences each bounded in magnitude by $2C$. So $P[|U_{1,n}| > n\varepsilon/(2m)] \leq 2e^{-n\varepsilon^2/(32m^2C^2)}$ by applying a generalization of the Hoeffding's inequality to the martigale differences (see, e.g., Theorem 15.20, Davidson, 1994, or Theorem 9.1, Devroye, Györfi and Lugosi, 1996). Similarly is $P[|U_{j,n}| > n\varepsilon/(2m)] \leq 2e^{-n\varepsilon^2/(32m^2C^2)}$ for all $j = 1,...,m$. Now $P[|V_n| > n\varepsilon/2] \leq (2/\varepsilon)n^{-1}E|V_n| \leq (2/\varepsilon)n^{-1}\sum_{t=1}^{n} E|E(X_t|\mathcal{F}_{t-m}) - EX_t|$ using the Markov inequality and the triangular inequalities. Combining these upperbounds for the terms on the right hand side of (2) leads to the proof. Q.E.D.

The inequality appearing in the current lemma holds without assumption of a dependence structure. It still assumes a bounded X_t. Next we remove the boundedness assumption by incorporating a term related to the 'tail behavior' of a random variable ρ_t, which is now possibly unbounded.

Proof for Theorem 1 We will decompose $\rho_t = X_t + Y_t$ where $X_t = \rho_t I[|\rho_t| \leq C]$ and $Y_t = \rho_t I[|\rho_t| > C]$. Then $|\sum(\rho_t - E\rho_t)| \leq |\sum(X_t - EX_t)| + \sum|Y_t| + \sum E|Y_t|$ by using triangular inequalities. Then $P[\sum_1^n(\rho_t - E\rho_t)| > n\varepsilon] \leq P[|\sum_1^n(X_t - EX_t)| > n\varepsilon/3] + P[\sum_1^n |Y_t| > n\varepsilon/3] + P[\sum_1^n E|Y_t| > n\varepsilon/3]$ The first term is bounded by the preceding lemma by

$$2me^{-n\varepsilon^2/(288m^2C^2)} + (6/\varepsilon)n^{-1}\sum_{t=1}^{n} E|E(X_t|\mathcal{F}_{t-m}) - EX_t|.$$

The second term is bounded above by $(3/\varepsilon)n^{-1}\sum_1^n E|Y_t|$. The third term is a probability of a deterministic event, which is zero if the righthand side of the triplex inequality in Theorem 1 does not exceed one. Therefore $P[\sum_1^n(\rho_t - E\rho_t)| > n\varepsilon] \leq 2me^{-n\varepsilon^2/(288m^2C^2)} + (6/\varepsilon)n^{-1}\sum_{t=1}^n E|E(X_t|\mathcal{F}_{t-m}) - EX_t| + (3/\varepsilon)n^{-1}\sum_1^n E|Y_t|$. Now note that $|E|E(X_t|\mathcal{F}_{t-m}) - EX_t| - E|E(\rho_t|\mathcal{F}_{t-m}) - E\rho_t|| \leq E|E(Y_t|\mathcal{F}_{t-m}) - EY_t| \leq 2E|Y_t|$ using the triangular inequalities and the Jensen's inequality. Then $E|E(X_t|\mathcal{F}_{t-m}) - EX_t| \leq E|E(\rho_t|\mathcal{F}_{t-m}) - E\rho_t| + 2E|Y_t|$ and $P[\sum_1^n(\rho_t - E\rho_t)| > n\varepsilon] \leq 2me^{-n\varepsilon^2/(288m^2C^2)}$ $+(6/\varepsilon)n^{-1}\sum_{t=1}^n\{E|E(\rho_t|\mathcal{F}_{t-m}) - E\rho_t| + 2E|Y_t|\} + (3/\varepsilon)n^{-1}\sum_1^n E|Y_t|$ which leads to the proof of the theorem. Q.E.D.

3. Uniform Deviation

The above inequality (in Theorem 1) can be used to bound the probability of a large (pointwise) deviation $T_n(b) = \hat{R}(b) - R(b)$, where b is a parameter, $\hat{R}(b)$ is a sample average $\hat{R}(b) = n^{-1}\sum_{t=1}^n \rho(\omega_t, b)$ (where for each t, ω_t is measurable -$\mathcal{F}_t$ from an increasing sequence of σ-fields), and $R(b)$ is its expectation $R(b) = E\hat{R}(b)$. It is often of interest to bound the probability of a large *uniform* deviation $\sup_{b\in B}|T_n(b)|$ over a parameter space B for b.

The connection between the pointwise and uniform deviations can be obtained by covering B with many (say, Γ) smaller sets B_i's, so that $B \subset \cup_{i=1}^{\Gamma} B_i$. Choose b_i to be some parameter located in B_i for each i. Note that $\sup_{b\in B}|T_n(b)| \leq \max_{i=1}^{\Gamma}|T_n(b_i)| + \max_{i=1}^{\Gamma}\sup_{b\in B_i}|T_n(b) - T_n(b_i)|$. Then a union bound leads to:

Proposition 1 *For any nonstochastic* $\delta > 0$ *and any positive integer n,*

$$P[\sup_{b\in B}|T_n(b)| > 2\delta] \leq \sum_{i=1}^{\Gamma} P[|T_n(b_i)| > \delta] + \sum_{i=1}^{\Gamma} P[\sup_{b\in B_i}|T_n(b) - T_n(b_i)| > \delta]. \tag{3}$$

This is the basis for us to bound the probability of uniform large deviation. The first term involves pointwise deviations and can be bounded by the inequality derived before. The second term can be bounded when $T_n(b)$ 'often changes little' in a small set B_i. This can often achieved by assuming a Lipshitz condition for the summand $\rho(\omega_t, b)$ in argument b (see, e.g., Newey, 1991).

In machine learning, however, we often encounter summand $\rho(\omega_t, b)$ that is discontinuous in b. For example, the classification error can be written as $\rho(\omega_t, b) = |y_t - I[x_t'b > 0]|$, which is discontinuous in b, when a linear boundary (in predictor x_t) is used to classify a $\{0,1\}$ valued label y_t. (Here $\omega_t = (y_t, x_t)$.)

We will use a quite general framework that allows some continuous cases and some discontinuous cases as well as some 'mixed' cases. Let $\rho(\omega_t, b)$ be of the form $\rho(\omega_t, b) = f_t(b, A_t(b))$ where $f_t(\cdot,\cdot)$ is continuous in the first argument, but the second argument $A_t(b) = I[g(\omega_t, b) > 0]$ for some fixed function g that determines a decision boundary. The function f_t depends on t through observation ω_t. This framework can then include the following examples:

- (continuous) L_1-loss: $\rho = |y_t - x_t'b|$ (when $f_t(b,\cdot)$ is constant);
- (discontinuous) classification loss: $\rho = |y_t - I[x_t'b > 0]|$ (when $f_t(\cdot, A_t(b))$ is constant);
- 'mixed' loss such as $\rho = (1 - y_t)\alpha(x_t'b)I[\alpha(x_t'b) > 0]$, which may result from a loan decision of lending out amount $\alpha(x_t'b)$ (according to a continuous parametric model α) when $100y_t\%$ of the loan is paid back.

Under this framework we will bound the deviation $\sup_{b\in B_i}|T_n(b)-T_n(b_i)|$. This is summarized in the following Proposition, the proof of which is included in the Appendix.

Proposition 2 *For each parameter b in a convex set B_i that contains b_i, denote $T_n(b)=\hat{R}(b)-E\hat{R}(b)$, where $\hat{R}(b)=n^{-1}\sum_{t=1}^{n}\rho(\omega_t,b)$ and for each t, ω_t is measurable -$\mathcal{F}_t$ (from an increasing sequence of σ-fields). Assume that ρ has the form $\rho(\omega_t,b)=f_t(b,A_t(b))$ where $A_t(b)=I[g(\omega_t,b)>0]$ for some fixed function g that determines a decision boundary.*

Define S_i as the 'boundary set' $S_i=\cup_{b\in B_i}\{\omega_t : g(\omega_t,b)=0\}$. Assume that:

(A1): $g(\omega_t,b)$ is continuous in b and measurable in ω_t;

(A2): (Lipshitz condition) $\sup_{a=0,1}|f_t(b,a)-f_t(b^*,a))|\le N_{it}|b-b^*|_q$ for some $q>0$, for any $b,b^*\in B_i$;[2]

(A3): (Small boundary condition) The boundary set S_i is measurable and $n^{-1}\sum_{t=1}^{n}EI(\omega_t\in S_i)\le\delta/(12C)$ for some constants $\delta,C>0$.

Denote $\lambda=\sup_{b,b^*\in B_i}|b-b^*|_q$ and $M_{it}=|f_t(b_i,1)-f_t(b_i,0)|$.

For any constants $\delta,C>0$ and positive integers n,m, if (A3) holds, then we have:

$$\begin{aligned}
&P[\sup_{b\in B_i}|T_n(b)-T_n(b_i)|>\delta]\\
&\le(6\lambda/\delta)n^{-1}\sum_{t=1}^{n}E(N_{it}+EN_{it})I[N_{it}+EN_{it}>\delta/(6\lambda)]\\
&+(6/\delta)n^{-1}\sum_{t=1}^{n}EM_{it}I(M_{it}>C)\\
&+2me^{-n\delta^2/(1152m^2C^2)}+(12C/\delta)n^{-1}\sum_{t=1}^{n}E|E(I(\omega_t\in S_i)|\mathcal{F}_{t-m})-EI(\omega_t\in S_i)|,
\end{aligned}$$

as long as the right hand side exists.

3.1 Remarks

6. In the 'continuous case', $f_t(b,a)$ is constant in $a\in\{0,1\}$. We can then drop the last three terms in the above bound. This is because $M_{it}=0$ in this case and we can take $C\to 0$. In the 'discontinuous case', $f_t(b,a)$ is constant in b and we can drop the first term in the above bound, since the Lipshitz constant N_{it} can be taken as 0. The result above holds also for the more general 'mixed case' when $f_t(b,a)$ varies with both b and a.

7. Assumption A2 can often be validated by bounding the partial derivative of $f_t(b,a)$ on the first argument. In a later example with L_1 loss we will use a triangular inequality to validate this assumption.

8. Assumption A3 is related to $P(\omega_t\in S_i)$, the probability of an observation falling in the 'boundary set' S_i corresponding to a parameter set B_i. When B_i is small enough, we expect that the 'boundary set' will have small probability and A3 can be satisfied. The situation is clarified when $\omega_t=(y_t,x_t)$, $g(\omega_t,b)$ depends on ω_t only through predictor x_t, and $x_t=(w_t,v_t')'$ has a

2. For a vector v with component v_j's, define the ℓ_q norm as $|v|_q=(\sum_{j=1}^{\dim(v)}|v_j|^q)^{1/q}$ for $q\in(0,\infty)$, and $|v|_\infty=\sup_{j=1}^{\dim(v)}|v_j|$. We will also formally denote $|v|_0=\sum_{j=1}^{\dim(v)}I[|v_j|>0]$.

scalar component w_t and other components v_t, so that the decision boundary $[g(\omega_t, b) = 0]$ 'can be solved' as $[w_t = w(v_t, b)]$ for some fixed function w. In this case the boundary set $S_i = \{(w_t, v_t) : w_t = w(v_t, b), b \in B_i\} = [\inf_{b \in B_i} w(v_t, b) \le w_t \le \sup_{b \in B_i} w(v_t, b)]$ if B_i is compact and $w(v_t, b)$ is continuous in b. Suppose we use the ℓ_∞ norm and define $\lambda = \sup_{b,b^* \in B_i} |b - b^*|_\infty$. Denote $d_0 = \sup_{b,b^* \in B_i} |b - b^*|_0$. Then in the Appendix we will show that

$$P(\omega_t \in S_i) \le E_{\mathcal{V}_t}\{\sup_{w_t} p(w_t|\mathcal{V}_t) \sup_{b \in B_i} |\partial_b w(v_t, b)|_\infty\}\lambda d_0. \tag{4}$$

Here $\partial_b w(v_t, b)$ denotes a partial derivative of w, $\mathcal{V}_t$ is some σ-field such that v_t is measurable -$\mathcal{V}_t$ for each t, and $p(w_t|\mathcal{V}_t)$ denotes the conditional density. In the 'linear case'[3] assuming $\lambda < 2$, $g(\omega_t, b) = \pm w_t + v_t' b_v$ (so A1 is satisfied), we can take $w(v_t, b) = \mp v_t' b_v$ and $\sup_{b \in B_i} |\partial_b w(v_t, b)|_\infty = |v_t|_\infty$. If the conditional density is bounded above by constant c, then (4) becomes $P(\omega_t \in S_i) \le cE|v_t|_\infty \lambda d_0$. The assumption A3 will be satisfied for choosing $\lambda \le \delta/(12Cc \sup_t E|v_t|_\infty d_0)$, which restricts the size of B_i.

9. It is also noted that in this paper, we will consider 'boundary sets' of the 'solvable' form $S_i = [\inf_{b \in B_i} w(v_t, b) \le w_t \le \sup_{b \in B_i} w(v_t, b)]$ which is assumed to be measurable. In our later examples we will focus on 'linear solvable type' described above, with decision boundary $[w_t = \mp v_t' b_v]$, and B_i being a closed ℓ_∞ ball centered at b_i and with radius $h = \lambda/2 > 0$. Then $S_i = [\mp v_t'(b_i)_v - h|v_t|_1 \le w_t \le \mp v_t'(b_i)_v + h|v_t|_1]$ which is indeed measurable. [More generally, when $\sup_{b \in B_i} w(v_t, b)$ and $\inf_{b \in B_i} w(v_t, b)$ are both continuous in v_t, S_i is measurable.]

We will analyze the bound in Proposition 2 term by term in the later examples. The terms $E(N_{it} + EN_{it})I[N_{it} + EN_{it} > \delta/(6\lambda)]$ and $EM_{it}I(M_{it} > C)$ are tail terms. We can choose sufficiently small λ and sufficiently large C to make them small.

The dependence term $E|E(I(\omega_t \in S_i)|\mathcal{F}_{t-m}) - EI(\omega_t \in S_i)|$ will be small for large m when ω_t can be approximated by strong mixing sequences in some sense.

The exponential term $2me^{-n\delta^2/(1152m^2C^2)}$ can be made small by choosing m and C to depend on n in certain ways. We can allow δ to depend on n also, which will lead to convergence rates.

4. A Continuous Example

Consider $\rho_t = \rho(\omega_t, b) = |Y_t - b_1 Y_{t-1} - \ldots - b_r Y_{t-r}|$ (where $\omega_t = (Y_t, \ldots, Y_{t-r})$ and $b = (b_1, \ldots, b_r)$), which represents predicting Y_t by r of its own lags under an L_1 loss. We will allow r to increase with n later to allow high dimensionality. We will bound the probability of a large uniform deviation $\sup_{b \in B} |n^{-1} \sum_{t=1}^n \rho_t - n^{-1} \sum_{t=1}^n E\rho_t|$ over an ℓ_∞ ball $B = [|b|_\infty \le C_b]$ with a constant radius $C_b > 0$.

Suppose the true model for Y_t follows an $MA(\infty)$ model $Y_t = \sum_{j=0}^\infty \theta_j Z_{t-j}$, where θ_j's are fixed coefficients with a finite ℓ_1 norm $|\theta|_1 = \sum_0^\infty |\theta_j| < \infty$, and Z_j's are 'innovations', which are assumed to be iid (independent and identically distributed) with zero mean and finite variance. Although we have assumed Y_t to be centered to have mean zero and that there is no intercept term used in the L_1 loss, this is only for convenience and similar results can be obtained without this assumption.

We will consider a case of exponentially decaying θ_j's, but each θ_j can be nonzero:

3. In the 'linear case' the decision rule $[g(\omega_t, b) > 0]$ has the form $[w_t b_w + v_t' b_v > 0]$. We can always rescale the coefficients $b = (b_w, b_v')'$ by a scalar multiple. One such standardization used in Horowitz (1992) is such that $|b_w| = 1$ or $b_w \in \{-1, +1\}$. Note that for small enough set B_i with $\lambda = \sup_{b,b' \in B_i} |b - b'|_\infty < 2$, b_w is constant in B_i and takes a common sign. We can pick either sign to proceed.

Condition (B1): $\sum_{j=k}^{\infty}|\theta_j| < \nu^k$ *for all large enough* k*, for some* $\nu \in (0,1)$.

This is a situation when Y_t can have a dependence structure that is not strong mixing, since it involves the infinite past of Z_{t-j}'s (see, e.g., Davidson, 1994, Section 14.3). On the other hand, the dependence term in Theorem 1 can be bounded by L_1 approximation of strong mixing and we have the following result (proved in Appendix, where $\mathcal{F}_t$ is the σ-field generated by $\{Z_s\}_{s=-\infty}^{t}$ for each t):

$$E|E(\rho_t|\mathcal{F}_{t-m}) - E\rho_t| \le 2(r+1)(C_b+1)E|Z_1| \sum_{j=k+1}^{\infty}|\theta_j| \text{ for any positive integer } k < m-r. \quad (5)$$

In order to bound the tail term in Theorem 1, we use, for some finite constants $u, C_u > 0$,

$$E|\rho_t|I(|\rho_t| > C) \le (r+1)^2(C_b+1)||Z_1||_2|\theta|_1 C_u e^{-uC(C_b+1)^{-1}(r+1)^{-1}/2}, \quad (6)$$

which is proved in the Appendix assuming an additional condition:
Condition (B2): For innovation Z_1*, the cumulant generating function* $K(u) = \ln Ee^{Z_1 u}$ *is continuously differentiable at* 0. *(E.g.,* Z_1 *can be a Gaussian innovation.)*

Now we apply Proposition 2, where only the first term of the bound is relevant in this continuous case due to Remark 6. The Lipshitz constant N_{it} can be obtained from the triangular inequality $|\rho(\omega_t,b) - \rho(\omega_t,b^*)| \le |b_1 - b_1^*||Y_{t-1}| + \dots + |b_r - b_r^*||Y_{t-r}| \le (|Y_{t-1}| + \dots + |Y_{t-r}|)|b-b^*|_\infty$. So we can take $N_{it} = |Y_{t-1}| + \dots + |Y_{t-r}|$ (using ℓ_∞ norm). We have

$$E(N_{it} + EN_{it})I(N_{it} + EN_{it} > \delta/(6\lambda)) \le (2r^2|\theta|_1||Z_1||_2)C_u e^{-u\psi/2}, \quad (7)$$

for some finite constants $u, C_u > 0$, where $\psi = \delta/(6r\lambda) - E|Z_1||\theta|_1$, which is proved in the Appendix.

Now we apply (3), Theorem 1 and Proposition 2 and combine all terms together (using (5), (6) and (7)) to obtain:

For any positive integers $k < m-r$, m, n, and positive C, δ, we have

$$\begin{aligned}
&P[\sup_{b\in B}|n^{-1}\sum_{t=1}^{n}(\rho_n - E\rho_t)| > 2\delta] \\
&\le \Gamma 2me^{-n\delta^2/(288m^2C^2)} \\
&+\Gamma(6/\delta)2(r+1)(C_b+1)E|Z_1|\sum_{j=k+1}^{\infty}|\theta_j| \\
&+\Gamma(15/\delta)(r+1)^2(C_b+1)||Z_1||_2|\theta|_1 C_u e^{-uC(C_b+1)^{-1}(r+1)^{-1}/2} \\
&+\Gamma(6\lambda/\delta)(2r^2|\theta|_1||Z_1||_2)C_u e^{-u(\delta/(6r\lambda)-E|Z_1||\theta|_1)/2}. \quad (8)
\end{aligned}$$

Here $B = [|b|_\infty \le C_b] = [-C_b, C_b]^r$ for some constant radius $C_b > 0$. We will consider a high dimensional case where the number of lags can increase with sample size n:
Condition (B3): $r = O((\ln n)^M)$ *for some power* $M > 0$.

Note that we can take $B_1, \dots, B_\Gamma$ to be Γ closed ℓ_∞ balls of radius $\lambda/2$ to cover B, where $\Gamma \le (2C_b/\lambda + 1)^r$.

We will let $\delta = n^{-0.5+\gamma_1}/2$ for some small $\gamma_1 > 0$, $m = C = \lceil n^{\gamma_1/4}\rceil$, $\lambda = n^{-1}$, $k = m - 2r$. Then under conditions (B1) to (B3), $\ln\Gamma = O((\ln n)^{M_1})$ for some $M_1 > 0$ and all four terms in the above inequality (8) are $O(e^{-c_1 n^{c_2}})$ for some $c_1, c_2 > 0$ dependent on γ_1. Therefore we have:

Proposition 3 *Under Conditions (B1) to (B3), for any small* $\gamma_1 > 0$,

$$P[\sup_{b \in B} |n^{-1} \sum_{t=1}^{n} (\rho_t - E\rho_t)| > n^{-0.5+\gamma_1}] = O(e^{-c_1 n^{c_2}}).$$

The rate of uniform convergence remains nearly 'parametric' $O_P(n^{-0.5})$ in this case, despite the high dimensionality of set B.

5. A Discontinuous Example

Let $\omega_t = (y_t, x_t)$, where y_t is a real-valued response at t and $x_t = (1, y_{t-1}, ..., y_{t-r}, z_t')'$ is a vector of predictors that can include r lags of y as well as a vector of exogenous variable z_t. Suppose we are interested in predicting the sign of y_t by using a discontinuous loss $\rho_t = |I(y_t > 0) - I(x_t' b > 0)|$. We will bound the probability of a large uniform deviation $\sup_{b \in B} |n^{-1} \sum_{t=1}^{n} (\rho_t - E\rho_t)|$ over a set of 'variable selection' $B = [|b|_0 \leq v, |b|_\infty \leq C_b, |b_{r+2}| = 1]$, where $|b|_0 \equiv \sum_{j=1}^{\dim(b)} I|b_j| > 0]$ counts the number of selected x-components, $|b|_\infty \leq C_b$ bounds the parameter space, and $|b_{r+2}| = 1$ is due to a standardization for the coefficient of the first component of z_t (see Footnote 2). Later we will allow v (maximal number of selected variables), r (number of lags allowed) and $K \equiv \dim(z_t)$ to increase with n in certain ways for high dimensional variable selection.

The true model of y_t is assumed to be an $MA(\infty)$ transform of a strong mixing process: $y_t = \sum_{j=0}^{\infty} \theta_j f_j(z_{t-j}, \varepsilon_{t-j})$, where f_j is some fixed measurable function for each j so that $\sup_{t,j} ||f_j(z_t, \varepsilon_t)||_1 < \infty$, and ε_t is a stochastic sequence independent of z_t called the 'innovation'. We assume that:

Condition (C1): $\{z_t, \varepsilon_t\}$ *is strong mixing with mixing coefficient* α_m *decreasing exponentially fast in* m.

Condition (C2): $\sum_{j=k+1}^{\infty} |\theta_j|$ *decreases exponentially fast in* k.

Note that y_t itself may no longer be strong mixing due to its dependence on the infinite past. These assumptions are satisfied in many situations. For example, in an ARX model $y_t = \varphi y_{t-1} + z_t' \beta + \varepsilon_t$ ($|\varphi| < 1$), an $MA(\infty)$ representation gives $y_t = \sum_{j=0}^{\infty} \varphi^j (z_{t-j}' \beta + \varepsilon_{t-j})$. Here, ε_t does not have to be iid; it can be an 'exponential' strong (or β-) mixing process such as a GARCH process (see, e.g., Francq and Zakoïan, 2006) when y_t follows an ARX-GARCH model.

In the Appendix, we show that under some additional conditions (C3 and C4) on the underlying process $\{z_t, \varepsilon_t\}$, we have, for any positive integer $k < m - r$,

$$E|E(\rho_t|\mathcal{F}_{t-m}) - E\rho_t| \leq 6\alpha_{m-r-k} + 8(\sqrt{2M_y} + \sqrt{2M_x r C_b}) \sqrt{\sup_{t,l} ||f_l(z_t, \varepsilon_t)||_1 \sum_{j>k} |\theta_j|}. \qquad (9)$$

Here M_x, M_y are constants appearing in these additional conditions:

Condition (C3): y_t *follows a model of the form* $y_t = F_t + \varepsilon_t$ *where* F_t *depends on the history* $(\{z_s\}_{\infty}^{t}, \{\varepsilon_s\}_{-\infty}^{t-1})$ *and* ε_t *is an innovation that has a conditional density* $p(\varepsilon_t | \{z_s\}_{\infty}^{t}, \{\varepsilon_s\}_{-\infty}^{t-1})$ *bounded above by a constant* M_y *(which is satisfied, for example, by* $N(0, \sigma^2)$ *innovations).*

Condition (C4): The conditional density $p(z_{t,1} | z_{t,2}, ..., z_{t,K}, \{z_s, \varepsilon_s\}_{-\infty}^{t-1})$ *is bounded above by a constant* M_x.

We can cover B by Γ sets $B_i \subset B$, $i = 1, ..., \Gamma$, with each B_i being a closed ℓ_∞ ball centered at some b_i, with radius $h = \lambda/2 > 0$, and with dimension at most $v - 1$. This is explained in the Appendix, where we also show that we can take

$$\Gamma \leq 2v(K + r)^{v-1} (2C_b/\lambda + 1)^{v-1} \qquad (10)$$

as an upperbound obtained from a combinatorial argument. Now we try to apply Proposition 2.

Assumption (A1) is obviously satisfied since $g(\omega_t, b) = x_t'b$. We will show that assumption (A3) holds for all large n in the Appendix, with an additional condition (C5) and with suitable choices of δ, λ (the ℓ_∞ diameter of B_i), r (the number of lags) and v (the maximal number of selected variables) to be specified later. This additional condition is:

Condition (C5): The exogenous variables are bounded above by a finite constant: $|z_t|_\infty \leq C_z$.

Assumption (A2) is satisfied with $N_{it} = 0$ due to this 'discrete' situation (see Remark 6). So the first term of the bound in Proposition 2 is zero. We can take $M_{it} = ||I(y_t > 0) - 1| - |I(y_t > 0) - 0|| = 1$ and $C = 1$ so the second term of the bound is zero also.

In the Appendix we evaluate the last term which is determined by $E|E(I(\omega_t \in S_i)|\mathcal{F}_{t-m}) - EI(\omega_t \in S_i)|$. Assuming (C4), we have, for any positive integer $k < m - r$,

$$E|E(I(\omega_t \in S_i)|\mathcal{F}_{t-m}) - EI(\omega_t \in S_i)| \leq 6\alpha_{m-r-k} + 4\sqrt{r\sum_{j>k}|\theta_j|\sup_{t,l}||f_l(z_t,\varepsilon_t)||_1(2M_x 2(1+C_b+h))}. \tag{11}$$

Now combine the applications of (3), Theorem 1 (with $C = 1$, or just use Lemma 1), and Proposition 2, apply Equations (9) and (11) and we obtain:

For any positive integers $k < m - r$, m, n, and positive δ,

$$\begin{aligned}
&P[\sum_{b\in B}|n^{-1}\sum_{t=1}^{n}(\rho_t - E\rho_t)| > 2\delta] \\
&\leq \Gamma 2me^{-n\delta^2/(32m^2)} \\
&+\Gamma(2/\delta)\left\{6\alpha_{m-r-k} + 8(\sqrt{2M_y} + \sqrt{2M_x rC_b})\sqrt{\sup_{t,l}||f_l(z_t,\varepsilon_t)||_1\sum_{j>k}|\theta_j|}\right\} \\
&+\Gamma 2me^{-n\delta^2/(1152m^2)} \\
&+\Gamma(12/\delta)\left\{6\alpha_{m-r-k} + 4\sqrt{r\sum_{j>k}|\theta_j|\sup_{t,l}||f_l(z_t,\varepsilon_t)||_1(2M_x 2(1+C_b+\lambda/2))}\right\}.
\end{aligned} \tag{12}$$

Now choose parameters to make the bound small. Note that we can take $\Gamma \leq 2v(K+r)^{v-1}(2C_b/\lambda + 1)^{v-1}$. We have assumed exponential decay for α_k and $\sum_{j>k}|\theta_j|$ in k.

Let $C_b > 0$ be a constant in n. Assume the following condition on the various dimension parameters:

Condition (C6): The number of lags $r = O((\ln n)^{M_1})$ for some power $M_1 > 0$; the number of exogenous variables $K = O(n^{M_2})$ for some power $M_2 > 0$, which can form a very-high dimensional candidate predictor, with dimension possibly large than sample size n; the number of selected variables $v = O((\ln n)^{M_3})$ for some power $M_3 > 0$.

We will let $\delta = n^{-0.5+\gamma_1}/2$ for some small $\gamma_1 > 0$, $m = \lceil n^{\gamma_1/4}\rceil$, $\lambda = n^{-1}$, $k = \lceil(m-r)/2\rceil$. Then $\ln\Gamma = O((\ln n)^{M_4})$ for some $M_4 > 0$ and all four terms in the above inequality (12) are $O(e^{-c_1 n^{c_2}})$ (for some $c_1, c_2 > 0$ dependent on γ_1), when Conditions (C1) to (C6) are assumed. Therefore we have:

Proposition 4 *Under conditions (C1) to (C6), for any small $\gamma_1 > 0$,*

$$P[\sup_{b\in B}|n^{-1}\sum_{t=1}^{n}(\rho_n - E\rho_t)| > n^{-0.5+\gamma_1}] = O(e^{-c_1 n^{c_2}}).$$

The rate of uniform convergence remains nearly 'parametric' $O_P(n^{-0.5})$ in this case, despite the high dimensionality of set B.

6. Discussion

This paper presents a very general inequality that generalizes Hoeffding's inequality to dependent and unbounded summands. The inequality may not be very tight, but it involves few assumptions and can be very useful in deriving convergence rates of pointwise and uniform deviations in a number of situations that cannot be dealt with before. We gave two examples here, one with L_1 loss and another on sign classification. There are other examples that may be worked out (e.g., with L_2 loss or with log-likelihood) which are not considered here. We hope that the current work can serve as a probablistic foundation to the theory of empirical risk minimization for many situations with dependent data and unbounded loss.

The current results involve a high dimensional parameter b; near-parametric convergence rates are obtained in examples with exponentially small 'unboundedness' (characterized by existence of some moment generating function) and with certain kinds of exponentially decaying temporal dependence. We expect that slower convergence rates may be obtained with more severe 'unboundedness', or with a slower decay of temporal dependence, using the same techniques.

Although the number of selected variables is restricted to $O((\log n)^M)$, we can allow these variables to be selected from a much higher number of candidate regressors of dimension up to n^M for any finite positive M, and still maintain a near-parametric convergence rate. This is demonstrated in Section 5 and is also true if we add in regressors and make an ARX model for Section 4. (In fact it is also possible to allow a higher number of selected variables such as n^a for some $a \in (0,1)$, but this will correspond to a slower convergence rate.)

It is noted that there exists much previous work in addressing the problems considered in this paper, in addition to the related work mentioned in the Introduction (we thank the reviewers for bringing our attention to these additional references). In the direction of unbounded loss, Meir and Zhang (2003) consider uniform deviations for iid data using a bound of the Rademaker complexity. Various ratios of empirical processes can also be used to handle unboundedness (see, e.g., Haussler, 1992; Pollard, 1995; Bartlett and Lugosi, 1999; Bercu, Gassiat and Rio, 2002). Bercu, Gassiat and Rio (2002) present a ratio-type result that can relax the assumption on the 'unboundedness' while proving exponential concentration for iid data. In the direction of dependence, McDiarmid (1998, e.g., Theorem 3.8) uses the method of bounded differences and includes a term related to a 'bad set' of events involving a large variance. Uniform inequalities have also been obtained for martingales (see, e.g., van de Geer, 2000, Theorem 8.13). In the direction of high dimensionality, typical uniform deviation results deal with infinite-dimensional function space (see, e.g, good summaries in Bousquet, Boucheron, and Lugosi, 2003 and van de Geer, 2000).

In comparison, our method addresses the three aspects (unboundedness, dependence, high dimensionality) simultaneously with a relative simple approach. While the additional references can sometimes handle one aspect better, they typically do not address the other aspects in the same time. For example, McDiarmid (1998, e.g., Theorem 3.8) can potentially handle data that are dependent 'in multi-dimensions' (such as random graph or spatial dependence), while our method only addresses 'one-dimensional' dependence (i.e., a time series). On the other hand, McDiarmid's method also requires bounded differences which would require some kind of boundedness (e.g., bounded summands in the case of an iid average) while we do not need this assumption. In addition, we note that the 'bad set' term bounding the probability of a large variance will often require applying a large deviation inequality again, while our triplex inequality is one in closed form.

Many of these additional references do not treat dependence to the same degree of generality as the current paper. For example, traditional treatments with symmetrization and Rademaker average such as Meir and Zhang (2003) are suitable only for iid data. Ratio-type results such as Bercu, Gassiat, Rio (2002) can relax the assumption on the 'unboundedness' while proving exponential concentration for iid data, but it is not clear how this can be extended to general dependent situations. The chaining technique described in, for example, van de Geer (2000, Section 3.2) and Bousquet, Boucheron and Lugosi (2003, Section 5) may be used to improve the convergence rate by a $\log n$ factor in the finite dimensional case. However, most applications of this technique are for independent data or martingales. Our dependence term, on the other hand, is put in the framework of mixingale, which is more general than martingales as discussed in Remark 4.

Typical uniform deviation results deal with infinite-dimensional function spaces. Our formulation is for a somewhat less general situation where the functions are parameterized, and we formulated uniform convergence on a high dimensional parameter space. Although it may be possible to formulate uniform convergence on function spaces directly for dependent data (see, e.g., Yu, 1994 for beta-mixing and Vidyasagar, 2005 for alpha-mixing), we choose the parametric covering framework which is less abstract and easier to understand, and demonstrates the convergence rates more clearly. The result of such a formulation is that a reader with an elementary background on probability and real analysis can follow the development easily, and arrive at such advanced results as the convergence rates with high dimensional variable selection. Although the formulation is deliberately elementary, the results are powerful enough to handle such complicated dependent situations as the sign prediction for an ARX process with GARCH error in Section 5.

Acknowledgments

Supported in part by NSF Grant DMS-0706885. We thank the reviewers and the editors for providing additional references and helpful suggestions.

Appendix A.

Proof of Proposition 2 Note that for any $b \in B_i$, $|T_n(b) - T_n(b_i)| = |n^{-1}\sum_{t=1}^n \{f_t(b, A_t(b)) - f_t(b_i, A_t(b_i))\} - n^{-1}\sum_{t=1}^n E\{f_t(b, A_t(b)) - f_t(b_i, A_t(b_i))\}|$. We therefore investigate the differences of the form $f_t(b, A_t(b)) - f_t(b_i, A_t(b_i)) = \{f_t(b, A_t(b)) - f_t(b_i, A_t(b))\} + \{f_t(b_i, A_t(b)) - f_t(b_i, A_t(b_i))\}$.

Note that $|f_t(b_i, A_t(b)) - f_t(b_i, A_t(b_i))| = M_{it}|A_t(b) - A_t(b_i)|$ where $M_{it} = |f_t(b_i, 1) - f_t(b_i, 0)|$, and $|f_t(b, A_t(b)) - f_t(b_i, A_t(b))| \le \sup_{b,b^* \in B_i}\sup_{a=0,1}|f_t(b,a) - f_t(b^*,a))| \le N_{it}\lambda$, where $\lambda \equiv \sup_{b,b^* \in B_i}|b - b^*|_q$, if we assume a Lipshitz condition $\sup_{a=0,1}|f_t(b,a) - f_t(b^*,a))| \le N_{it}|b - b^*|_q$ under an ℓ_q-norm for some $q > 0$.

We then combine the statements before and apply the triangular inequalities to obtain $|T_n(b) - T_n(b_i)| \le n^{-1}\sum_{t=1}^n N_{it}\lambda + n^{-1}\sum_{t=1}^n M_{it}|A_t(b) - A_t(b_i)| + n^{-1}\sum_{t=1}^n EN_{it}\lambda + n^{-1}\sum_{t=1}^n EM_{it}|A_t(b) - A_t(b_i)|$.

Now note that $|A_t(b) - A_t(b_i)| \le I(\omega_t \in S_i)$ where S_i is the 'boundary set $S_i = \cup_{b \in B_i}\{\omega_t : g(\omega_t, b) = 0\}$. This is true when we assume that $g(\omega_t, b)$ is continuous in b and B_i is convex.

[The difference $|A_t(b)-A_t(b_i)|$ is $\{0,1\}$ valued and takes value 1 only when only one of $g(\omega_t,b)$ and $g(\omega_t,b_i)$ is positive. This would imply $g(\omega_t,b^*)=0$ at some intermediate point b^* on the line segment between b and b_i, which must fall in B_i due to its convexity. A similar technique is used in Jiang and Tanner (2007) for a binary choice model with $g(\omega_t,b)$ linear in b.]

The above statements hold for any $b\in B_i$. Therefore

$$\begin{aligned}
&\sup_{b\in B_i}|T_n(b)-T_n(b_i)|\\
&\le n^{-1}\sum_{t=1}^n N_{it}\lambda+n^{-1}\sum_{t=1}^n M_{it}I(\omega_t\in S_i)+n^{-1}\sum_{t=1}^n EN_{it}\lambda+n^{-1}\sum_{t=1}^n EM_{it}I(\omega_t\in S_i)\\
&\le n^{-1}\sum_{t=1}^n (N_{it}+EN_{it})\lambda+n^{-1}\sum_{t=1}^n CI(\omega_t\in S_i)+n^{-1}\sum_{t=1}^n CEI(\omega_t\in S_i)\\
&+n^{-1}\sum_{t=1}^n\{M_{it}I(M_{it}>C)+EM_{it}I(M_{it}>C)\},
\end{aligned}$$

by noting that $M_{it}I(\omega_t\in S_i)\le CI(\omega_t\in S_i)+M_{it}I(M_{it}>C)$ for any constant $C>0$.

Then

$$\begin{aligned}
&P[\sup_{b\in B_i}|T_n(b)-T_n(b_i)|>\delta]\\
&\le P[n^{-1}\sum_{t=1}^n(N_{it}+EN_{it})\lambda>\delta/3]\\
&+P[n^{-1}\sum_{t=1}^n CI(\omega_t\in S_i)+n^{-1}\sum_{t=1}^n CEI(\omega_t\in S_i)>\delta/3]\\
&+P[n^{-1}\sum_{t=1}^n\{M_{it}I(M_{it}>C)+EM_{it}I(M_{it}>C)\}>\delta/3]\\
&\le P[n^{-1}\sum_{t=1}^n(N_{it}+EN_{it})\lambda>\delta/3]\\
&+(6/\delta)n^{-1}\sum_{t=1}^n EM_{it}I(M_{it}>C)\\
&+P[n^{-1}\sum_{t=1}^n\{I(\omega_t\in S_i)-EI(\omega_t\in S_i)\}>\delta/(3C)-2n^{-1}\sum_{t=1}^n EI(\omega_t\in S_i)]\\
&\le(6\lambda/\delta)n^{-1}\sum_{t=1}^n E(N_{it}+EN_{it})I[N_{it}+EN_{it}>\delta/(6\lambda)]\\
&+(6/\delta)n^{-1}\sum_{t=1}^n EM_{it}I(M_{it}>C)\\
&+P[n^{-1}\sum_{t=1}^n\{CI(\omega_t\in S_i)-CEI(\omega_t\in S_i)\}>\delta/6],
\end{aligned}$$

where in the last step we assume that $n^{-1}\sum_{t=1}^n EI(\omega_t\in S_i)\le\delta/(12C)$ and have used $P[n^{-1}\sum_{t=1}^n X_t>2Q]\le Q^{-1}n^{-1}\sum_{t=1}^n EX_tI[X_t>Q]$ for nonnegative $X=N_{it}+EN_{it}$ and constant $Q=\delta/(6\lambda)$. [Note that $n^{-1}\sum_{t=1}^n X_t=n^{-1}\sum_{t=1}^n X_tI[X_t>Q]+n^{-1}\sum_{t=1}^n X_tI[X_t\le Q]\le Q+n^{-1}\sum_{t=1}^n X_tI[X_t>Q]$ and use Markov inequality.]

Now apply the pointwise inequality (1) on the last term and we obtain Proposition 2. Q.E.D.

Proof of Equation (4) in Remark 8

$$\begin{aligned}
&P(\omega_t \in S_i) = P[w_t \in [\inf_{b\in B_i} w(v_t,b), \sup_{b\in B_i} w(v_t,b)]] \\
&\leq E_{\mathcal{V}_t} \sup_{w_t} p(w_t|\mathcal{V}_t)|\sup_{b\in B_i} w(v_t,b) - \inf_{b\in B_i} w(v_t,b)| \\
&\leq E_{\mathcal{V}_t} \sup_{w_t} p(w_t|\mathcal{V}_t) \sup_{b,b^*\in B_i} |w(v_t,b) - w(v_t,b^*)| \\
&\leq E_{\mathcal{V}_t} \{\sup_{w_t} p(w_t|\mathcal{V}_t) \sup_{b\in B_i} |\partial_b w(v_t,b)|_\infty\} \sup_{b,b^*\in B_i} |b-b^*|_\infty \sup_{b,b^*\in B_i} |b-b^*|_0 \\
&= E_{\mathcal{V}_t} \{\sup_{w_t} p(w_t|\mathcal{V}_t) \sup_{b\in B_i} |\partial_b w(v_t,b)|_\infty\} \lambda d_0.
\end{aligned}$$

Q.E.D.

Proof of Equation (5) Define $Y_{t,k} = \sum_{j=0}^{k} \theta_j Z_{t-j}$ and $\rho_{t,k} = |Y_{t,k} - b_1 Y_{t-1,k} - \dots - b_r Y_{t-r,k}|$, which are strong mixing due to dependence on finite number of lags. We then have the L_1 approximation error $E|\rho_t - \rho_{t,k}| \leq (C_b+1)E(|Y_t - Y_{t,k}| + \dots + |Y_{t-r} - Y_{t-r,k}|) \leq (r+1)(C_b+1)E|Z_1|\sum_{j=k+1}^{\infty}|\theta_j|$. Then the technique in Section 2.2 implies that $E|E(\rho_t|\mathcal{F}_{t-m}) - E\rho_t| \leq 6\sup_{t,k}||\rho_{t,k}||_2 \alpha_{m-k-r}(\{Z_t\})^{1/2}$ $+2(r+1)(C_b+1)E|Z_1|\sum_{j=k+1}^{\infty}|\theta_j|$.

Note that $\alpha_{m-k-r}(\{Z_t\}) = 0$ for $m > k+r$, We have (5). Q.E.D.

Proof of Equation (6) We note that $|\rho_t| \leq (C_b+1)(|Y_t| + \dots + |Y_{t-r}|)$ and therefore $E|\rho_t|I(|\rho_t| > C) \leq (C_b+1)E(|Y_t| + \dots + |Y_{t-r}|)\sum_{s=0}^{r} I(|Y_{t-s}| > C(C_b+1)^{-1}(r+1)^{-1}) \leq (1+r)^2(C_b+1)\sup_{t,s} E|Y_t|I(|Y_s| > C(C_b+1)^{-1}(r+1)^{-1})$. Note that for $\eta = C(C_b+1)^{-1}(r+1)^{-1}$, we have

$$\begin{aligned}
&E|Y_t|I(|Y_s| > \eta) \\
&\leq ||Y_t||_2 ||I(|Y_s| > \eta)||_2 \\
&\leq (\sum_{j=0}^{\infty} |\theta_j| ||Z_{t-j}||_2) \sqrt{(Ee^{u|Y_s|})e^{-u\eta}} \\
&= ||Z_1||_2 |\theta|_1 \sqrt{(Ee^{u|Y_1|})} e^{-u\eta/2} \\
&\leq ||Z_1||_2 |\theta|_1 C_u e^{-u\eta/2}
\end{aligned}$$

for some positive u and C_u such that $Ee^{u|Y_s|} \leq C_u^2 < \infty$. This is achieved for some small enough u since

$$\begin{aligned}
&Ee^{\pm uY_s} = \exp\{\sum_{j=0}^{\infty} \ln Ee^{\pm u\theta_j Z_1}\} = \exp\{\sum_{j=0}^{\infty} K(\pm u\theta_j)\} \\
&\leq \exp\{\sum_{j=0}^{\infty} \sup_{|v|\leq u|\theta|_1} |K'(v)||u\theta_j|\} \leq \exp\{\sup_{|v|\leq u|\theta|_1} |K'(v)|u|\theta|_1\}.
\end{aligned}$$

Then

$$Ee^{u|Y_s|} \leq Ee^{uY_s} + Ee^{-uY_s} \leq 2\exp\{\sup_{|v|\leq u|\theta|_1} |K'(v)|u|\theta|_1\} \equiv C_u^2 < \infty$$

for some small enough u, due to continuous differentiability of $K(\cdot)$ at 0, which is assumed in Condition (B2). These lead to (6). Q.E.D.

Proof of Equation (7) Note that $EN_{it} = rE|Y_1| \leq rE|Z_1||\theta|_1$, and

$$\begin{aligned}
&E(N_{it}+EN_{it})I(N_{it}+EN_{it} > \delta/(6\lambda)) \\
&\leq E(|Y_{t-1}|+...+|Y_{t-r}|+rE|Y_1|)I(|Y_{t-1}|+...+|Y_{t-r}| > \delta/(6\lambda) - rE|Z_1||\theta|_1) \\
&\leq E(|Y_{t-1}|+...+|Y_{t-r}|+rE|Y_1|)\sum_{s=1}^{r} I(|Y_{t-s}| > \delta/(6r\lambda) - E|Z_1||\theta|_1) \\
&\leq |||Y_{t-1}|+...+|Y_{t-r}|+rE|Y_1|||_2||\sum_{s=1}^{r} I(|Y_{t-s}| > \delta/(6r\lambda) - E|Z_1||\theta|_1)||_2 \\
&\leq (2r||Y_1||_2)r\sqrt{Ee^{u|Y_1|}}e^{-u\psi/2} \\
&\leq (2r^2|\theta|_1||Z_1||_2)C_u e^{-u\psi/2}
\end{aligned}$$

for some small enough $u > 0$, where $\psi = \delta/(6r\lambda) - E|Z_1||\theta|_1$, and C_u is defined in Proof of Equation (6). Q.E.D.

Proof of Equation (9) We notice the process (y_t, x_t) can be approximated by strong mixing processes. This is because if we define $y_{t,k} = \sum_{j=0}^{k}\theta_j f_j(z_{t-j}, \varepsilon_{t-j})$, and $x_{t,k} = (1, y_{t-1,k}, ..., y_{t-r,k}, z_t')'$ then $||y_t - y_{t,k}||_1 \leq \sum_{j>k}|\theta_j|\sup_{t,l}||f_l(z_t,\varepsilon_t)||_1$ and $||x_t - x_{t,k}||_1 \equiv E|x_t - x_{t,k}|_1 = \sum_{s=t-1}^{t-r}||y_s - y_{s,k}||_1 \leq r\sum_{j>k}|\theta_j|\sup_{t,l}||f_l(z_t,\varepsilon_t)||_1$, both of which will decrease exponentially fast with k. On the other hand, $(y_{t,k}, x_{t,k})$ is a measurable transform of $(z_t, ..., z_{t-r-k}, \varepsilon_t, ..., \varepsilon_{t-r-k})$ and is therefore strong mixing with mixing coefficient α_{m-r-k} for $m > r+k$.

Now define $\rho_{t,k} = |I(y_{t,k} > 0) - I(x_{t,k}'b > 0)|$. The technique in Section 2.2 implies that $E|E(\rho_t|\mathcal{F}_{t-m}) - E\rho_t| \leq E|E(\rho_{t,k}|\mathcal{F}_{t-m}) - E\rho_{t,k}| + 2E|\rho_t - \rho_{t,k}|$ where $\mathcal{F}_t$ represents the σ-field generated by $(z_s, \varepsilon_s)_{-\infty}^{t}$. The first term is bounded by $6\alpha_{m-r-k}$ by using Theorem 14.2, Davidson (1994).

Applying the triangular inequalities, the second term is at most $2||I(y_t > 0) - I(y_{t,k} > 0)||_1 + 2||I(x_t'b > 0) - I(x_{t,k}'b > 0)||_1$. We will assume that (†) $P[|y_t| \leq \Delta] \leq M_y(2\Delta)$ for any small $\Delta > 0$, for some constant $M_y < \infty$. This is true under Condition (C3).

We will also assume that (‡) $P[|x_t'b| \leq \Delta] \leq M_x(2\Delta)$ for any small $\Delta > 0$, for some constant $M_x < \infty$. Notice that $x_t'b$ is of the form "$\pm z_{t,1} +$ *a linear combination of* $y_{t-1}, ..., y_{t-r}, z_{t,2}, ..., z_{t,K}$" due to the standardization of the coefficient of $z_{t,1}$. Then it is obvious that (‡) holds under Condition (C4).

Now notice that for two random variables W and W^*,

$$||I(W > 0) - I(W^* > 0)||_1 \leq 4\sqrt{2M}\sqrt{||W - W^*||_1}, \tag{13}$$

if $P(|W| \leq \Delta) \leq M(2\Delta)$ for $\Delta = \sqrt{||W - W^*||_1}/\sqrt{2M}$. This is proved by noting

$$EI(W > 0, W^* \leq 0)$$

$$
\begin{aligned}
&\le EI(W>0, W^* \le 0, |W-W^*| \le \Delta) + EI(|W-W^*| > \Delta) \\
&\le P[|W| \le \Delta] + E|W-W^*|/\Delta \\
&\le M(2\Delta) + E|W-W^*|/\Delta \\
&= 2\sqrt{2M}\sqrt{||W-W^*||_1}.
\end{aligned}
$$

Similarly $EI(W^*>0, W \le 0) \le 2\sqrt{2M}\sqrt{||W-W^*||_1}$. Now $||I(W>0)-I(W^*>0)||_1 = EI(W>0, W^* \le 0) + EI(W^*>0, W \le 0)$ leads to (13).

Now apply (13) and we obtain

$$
\begin{aligned}
&2||I(y_t>0)-I(y_{t,k}>0)||_1 + 2||I(x_t'b>0)-I(x_{t,k}'b>0)||_1 \\
&\le 8\sqrt{2M_y}\sqrt{||y_t-y_{t,k}||_1} + 8\sqrt{2M_x}\sqrt{||x_t'b-x_{t,k}'b||_1} \\
&\le 8\sqrt{2M_y}\sqrt{||y_t-y_{t,k}||_1} + 8\sqrt{2M_x|b|_\infty}\sqrt{||x_t-x_{t,k}||_1} \\
&\le 8\sqrt{2M_y}\sqrt{\sum_{j>k}|\theta_j|\sup_{t,l}||f_l(z_t,\varepsilon_t)||_1} + 8\sqrt{2M_xC_b}\sqrt{r\sum_{j>k}|\theta_j|\sup_{t,l}||f_l(z_t,\varepsilon_t)||_1},
\end{aligned}
$$

and therefore we have (9). Q.E.D.

Proof of Equation (10) Note that the $B = B_+ \cup B_-$ where $B_\pm = \{b \in B : b_{r+1} = \pm 1\}$ represents the two halfs of B with $b_{r+1} = \pm 1$, respectively. Note that $B_\pm$ can be written as a union $B_\pm = \cup_\gamma B_\pm(\gamma)$, where γ represents the subset of 'selected indices' in addition to $r+2$, and the union is taken over all subsets γ of $\{1,...,K+r+1\}\backslash\{r+2\}$ with card$(\gamma) \le v-1$. Here $B_\pm(\gamma) = \{b \in \Re^{K+r+1} : b_{r+2} = \pm 1, |b_j| \le C_b, \forall j \in \gamma; |b_j| = 0, \forall j \notin \gamma \cup \{r+2\}\}$. Each $B_\pm(\gamma)$ can be covered by at most $(2C_b/\lambda+1)^{\text{card}(\gamma)}$ sets B_i's of the form $B_i = \{b \in \Re^{K+r+1} : b_{r+2} = \pm 1, |b_j - (b_i)_j| \le \lambda/2, \forall j \in \gamma; |b_j-(b_i)_j| = 0, \forall j \notin \gamma \cup \{r+2\}\}$ for some $b_i \in B_\pm(\gamma)$, where card$(\gamma) \le v-1$. So a combinatorial argument leads to upperbound (10). Q.E.D.

Proof of Assumption (A3) for the example in Section 5 Assume Condition (C5), so that $|z_t|_\infty \le C_z < \infty$.

We can apply the arguments in Remark 8 by identifying $w_t = z_{t,1}$, $v_t = (1, y_{t-1}, ..., y_{t-r}, z_{t,2}, ..., z_{t,K})'$. Note that $d_0 \le v$ and we can choose $\mathcal{V}_t$ as the σ-field generated by $\{z_s, \varepsilon_s\}_{-\infty}^{t-1}$ and $z_{t,2}, ..., z_{t,K}$. Then A3 is satisfied if (C4) holds [the conditional density $p(w_t|\mathcal{V}_t)$ is bounded above by c (=M_x)] and if

$$\lambda \le \delta/(12vCc(1+C_z+r|\theta|_1\sup_{t,l}||f_l(z_t,\varepsilon_t)||_1)), \tag{14}$$

which would then be $\le \delta/(12vCc(1+C_z+r\sup_t||y_t||_1)) \le \delta/(12d_0Cc\sup_t E|v_t|_\infty)$. This inequality (14) is satisfied for all large n, when we take $\delta = n^{-0.5+\gamma_1}/2$ for any $\gamma_1 > 0$, and $\lambda = n^{-1}$ (the ℓ_∞-diameter of B_i), and assume that the number of lags r and the maximal number of selected variables v follow condition (C6). Q.E.D.

Proof of Equation (11) Similar to the approximation argument used before in Proof of Equation (9), we define $\omega_{t,k} = (y_{t,k}, x_{t,k})$ and obtain $E|E(I(\omega_t \in S_i)|\mathcal{F}_{t-m}) - EI(\omega_t \in S_i)| \le E|E(I(\omega_{t,k} \in S_i)|\mathcal{F}_{t-m}) - EI(\omega_{t,k} \in S_i)| + 2E|I(\omega_t \in S_i) - I(\omega_{t,k} \in S_i)|$, where the first term is bounded above by

$6\alpha_{m-r-k}$ for $m > r+k$, due to a treatment similar to the one used in Proof of Equation (9). Now we try to bound $E|I(\omega_t \in S_i) - I(\omega_{t,k} \in S_i)| \equiv ED$ in the second term, where $S_i \equiv \cup_{b \in B_i}[x_t' b = 0]$ will be computed below.

For our vector $x_t = (1, y_{t-1}, ..., y_{t-r}, z_{t,1}, z_{t,2}, ..., z_{t,K})'$, we here identify $w_t = z_{t,1} = (x_t)_{r+2}$, and $v_t = (1, y_{t-1}, ..., y_{t-r}, z_{t,2}, ..., z_{t,K})'$ including all $\{(x_t)_j\}_{j \neq r+2}$, for applying Remarks 8 and 9. Note that according to Remark 9, $S_i = [w^-(v_t) \le w_t \le w^+(v_t)]$ where $w^\pm(v_t) = v_t'(b_i)_v \pm h|v_t|_1$. [We have picked a sign $b_{r+2} = -1$ to proceed. The other sign is similar. To be more precise, here $|v_t|_1 = \sum_{j \in \gamma} |(x_t)_j|$ where γ is a subset of 'selected indices' in addition to $r+2$, when B_i is as described in Proof of Equation (10).]

Now $D \equiv |I(\omega_t \in S_i) - I(\omega_{t,k} \in S_i)| = 1$ implies that only one of the two points $\{x_t, x_{t,k}\} = \{(v_t', w_t)', \ (v_{t,k}', w_{t,k})'\}$ can lie in S_i, so there must be an intermediate point lying on a boundary of S_i, either on the upper boundary and denoted as $x^+ = ((v^+)', w^+(v^+))'$, or on the lower boundary and denoted as $x^- = ((v^-)', w^-(v^-))'$. [Here we have re-ordered the components of the vectors. For examples, for vector x_t, its component $(x_t)_{r+2} = w_t$ is now placed behind other components (denoted as v_t).]

If in addition, we also have a small distance $|x_t - x_{t,k}|_1 \le \eta$, then one of the following two events must happen (with $\pm$ option depending on whether the intermediate point falls on the upper or lower boundary of S_i):

$$\begin{aligned}
&|w_t - w^\pm(v_t)| \\
&\le |w_t - w^\pm(v^\pm)| + |w^\pm(v^\pm) - w^\pm(v_t)| \\
&\le |w_t - w^\pm(v^\pm)| + (C_b + h)|v_t - v^\pm|_1 \\
&\le |(1 + C_b + h)|x_t - x^\pm|_1 \\
&\le (1 + C_b + h)|x_t - x_{t,k}|_1 \le (1 + C_b + h)\eta.
\end{aligned}$$

Now we can bound

$$\begin{aligned}
ED &\le P[D = 1, |x_t - x_{t,k}|_1 \le \eta] + P[|x_t - x_{t,k}|_1 > \eta] \\
&\le P[\cup |w_t - w^\pm(v_t)| \le (1 + C_b + h)\eta] + P[|x_t - x_{t,k}|_1 > \eta] \\
&\overset{(a)}{\le} 2c2(1 + C_b + h)\eta + E|x_t - x_{t,k}|_1/\eta \\
&\le 2c2(1 + C_b + h)\eta + r\sum_{j>k} |\theta_j| \sup_{t,l} ||f_l(z_t, \varepsilon_t)||_1/\eta.
\end{aligned}$$

Now take $\eta = \sqrt{r\sum_{j>k} |\theta_j| \sup_{t,l} ||f_l(z_t, \varepsilon_t)||_1 / (2c2(1 + C_b + h))}$ and obtain

$$ED \le 2\sqrt{r\sum_{j>k} |\theta_j| \sup_{t,l} ||f_l(z_t, \varepsilon_t)||_1 (2c2(1 + C_b + h))}.$$

As in the previous Proof of Assumption (A3), we identify $c = M_x$. We have assumed Condition (C4) for the inequality (a) above. Therefore we have (11). Q.E.D.

References

P. L. Bartlett and G. Lugosi. An inequality for uniform deviations of sample averages from their means. *Statistics and Probability Letters*, 44:55-62, 1999.

B. Bercu, E. Gassiat, and E. Rio. Concentration inequalities, large and moderate deviations for self-normalized empirical processes. *Annals of Probability*, 30:1576-1604, 2002.

O. Bousquet, S. Boucheron, and G. Lugosi. Introduction to statistical learning theory. *Advanced Lectures on Machine Learning 2003* (O. Bousquet, U. von Luxburg, G. Rätsch, eds.) Springer, Berlin, 169-207, 2003.

J. Davidson. *Stochastic Limit Theory*. Oxford University Press, Oxford, 1994.

R. M. de Jong and T. M. Woutersen. Dynamic time series binary choice. *Manuscript, Ohio State University,* 2004. Downloadable at `http://www.econ.ohio-state.edu/dejong/tiemen45.pdf`.

L. Devroye, L. Györfi, and G. Lugosi. *A Probabilistic Theory of Pattern Recognition*. Springer, New York, 1996.

C. Francq and J.-M. Zakoïan. Mixing properties of a general class of GARCH(1,1) models without moment assumptions on the observed process. *Econometric Theory*, 22:815-834, 2006.

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

W. Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58:13-30, 1963.

J. L. Horowitz. A smoothed maximum score estimator for the binary response model. *Econometrica*, 60:505-531, 1992.

W. Jiang and M. A. Tanner. Risk minimization for time series binary choice with variable selection. *Technical Report 07-02, Department of Statistics, Northwestern University*, 2007. Downloadable at `http://newton.stats.northwestern.edu/~jiang/tr/choice1.tr.pdf`.

W. Jiang and M. A. Tanner. Gibbs posterior for variable selection in high dimensional classification and data mining. *Annals of Statistics*, 36:2207-2231. 2008. Downloadable at `http://newton.stats.northwestern.edu/~jiang/tr/gibbsone2.tr.pdf`.

A. C. Lozano, S. R. Kulkarni, and R. E. Schapire. Convergence and consistency of regularized boosting algorithms with stationary beta-mixing observations. In *Advances in Neural Information Processing Systems* 18, 2006. Downloadable at `http://www.cs.princeton.edu/~schapire/boost.html`.

C. McDiarmid. Concentration. *Probabilistic Methods for Algorithmic Discrete Mathematics* (M. Habib, C. McDiarmid, J. Ramirez, B. Reed, eds.) 195–248, Springer, Berlin, 1998.

R. Meir and T. Zhang. Generalization error bounds for Bayesian mixture algorithms. *Journal of Machine Learning Research*, 4:839-860, 2003.

W. K. Newey. Uniform convergence in probability and stochastic equicontinuity. *Econometrica*, 59:1161-1167, 1991.

D. Pollard. Uniform ratio limit theorems for empirical processes. *Scandinavian Journal of Statistics*, 22:271-278, 1995.

S. Van de Geer. *Empirical Processes in M-Estimation*. Cambridge University Press, Cambridge, 2000.

M. Vidyasagar. Convergence of empirical means with alpha-mixing input sequences, and an application to PAC learning. *Proceedings of the 44th IEEE Conference on Decision and Control, and the European Control Conference 2005*, pages 560-565, 2005.

B. Yu. Rates of convergence for empirical processes of stationary mixing sequences. *Annals of Probability*, 22:94-114, 1994.

T. Zhang. Information theoretical upper and lower bounds for statistical estimation. *IEEE Transaction on Information Theory*, 52:1307- 1321, 2006.

B. Zou and L. Li. The performance bounds of learning machines based on exponentially strong mixing sequences, *Computers and Mathematics with Applications*, 53:1050-1058, 2007.

Journal of Machine Learning Research 10 (2009) 997-1070 Submitted 5/08; Revised 3/09; Published 5/09

Fourier Theoretic Probabilistic Inference over Permutations

Jonathan Huang JCH1@CS.CMU.EDU
Robotics Institute
Carnegie Mellon University
Pittsburgh, PA 15213

Carlos Guestrin GUESTRIN@CS.CMU.EDU
Machine Learning Department
Carnegie Mellon University
Pittsburgh, PA 15213

Leonidas Guibas GUIBAS@CS.STANFORD.EDU
Department of Computer Science
Stanford University
Stanford, CA 94305

Editor: Marina Meila

Abstract

Permutations are ubiquitous in many real-world problems, such as voting, ranking, and data association. Representing uncertainty over permutations is challenging, since there are $n!$ possibilities, and typical compact and factorized probability distribution representations, such as graphical models, cannot capture the mutual exclusivity constraints associated with permutations. In this paper, we use the "low-frequency" terms of a Fourier decomposition to represent distributions over permutations compactly. We present *Kronecker conditioning*, a novel approach for maintaining and updating these distributions directly in the Fourier domain, allowing for polynomial time bandlimited approximations. Low order Fourier-based approximations, however, may lead to functions that do not correspond to valid distributions. To address this problem, we present a quadratic program defined directly in the Fourier domain for projecting the approximation onto a relaxation of the polytope of legal marginal distributions. We demonstrate the effectiveness of our approach on a real camera-based multi-person tracking scenario.

Keywords: identity management, permutations, approximate inference, group theoretical methods, sensor networks

1. Introduction

Probability distributions over permutations arise in a diverse variety of real world problems. While they were perhaps first studied in the context of gambling and card games, they have now been found to be applicable to many important problems such as multi-object tracking, information retrieval, webpage ranking, preference elicitation, and voting. Probabilistic reasoning problems over permutations, however, are not amenable to the typical representations afforded by machine learning such as Bayesian networks and Markov random fields. This paper explores an alternative representation and inference algorithms based on Fourier analysis for dealing with permutations.

As an example, consider the problem of tracking n people based on a set of noisy measurements of identity and position. A typical tracking system might attempt to manage a set of n tracks along

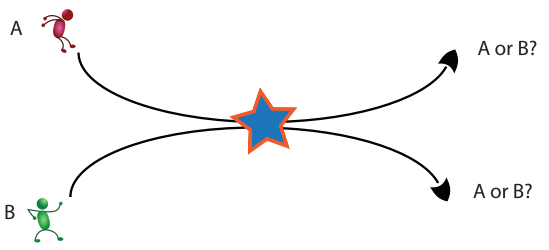


Figure 1: When two persons pass near each other, their identities can get confused.

with an identity corresponding to each track, in spite of ambiguities arising from imperfect identity measurements. When the people are well separated, the problem is easily decomposed and measurements about each individual can be clearly associated with a particular track. When people pass near each other, however, confusion can arise as their signal signatures may mix; see Figure 1. After the individuals separate again, their positions may be clearly distinguishable, but their identities can still be confused, resulting in identity uncertainty which must be propagated forward in time with each person, until additional observations allow for disambiguation. This task of maintaining a belief state for the correct association between object tracks and object identities while accounting for local mixing events and sensor observations, was introduced in Shin et al. (2003) and is called the *identity management problem*.

The identity management problem poses a challenge for probabilistic inference because it needs to address the fundamental combinatorial challenge that there is a factorial number of associations to maintain between tracks and identities. Distributions over the space of all permutations require storing at least $n! - 1$ numbers, an infeasible task for all but very small n. Moreover, typical compact representations, such as graphical models, cannot efficiently capture the mutual exclusivity constraints associated with permutations.

While there have been many approaches for coping with the factorial complexity of maintaining a distribution over permutations, most attack the problem using one of two ideas—storing and updating a small subset of likely permutations, or, as in our case, restricting consideration to a tractable subspace of possible distributions. Willsky (1978) was the first to formulate the probabilistic filtering/smoothing problem for group-valued random variables. He proposed an efficient FFT based approach of transforming between primal and Fourier domains so as to avoid costly convolutions, and provided efficient algorithms for dihedral and metacyclic groups. Kueh et al. (1999) show that probability distributions on the group of permutations are well approximated by a small subset of Fourier coefficients of the actual distribution, allowing for a principled tradeoff between accuracy and complexity. The approach taken in Shin et al. (2005), Schumitsch et al. (2005), and Schumitsch et al. (2006) can be seen as an algorithm for maintaining a particular fixed subset of Fourier coefficients of the log density. Most recently, Kondor et al. (2007) allow for a general set of Fourier coefficients, but assume a restrictive form of the observation model in order to exploit an efficient FFT factorization.

In the following, we outline our main contributions and provide a roadmap of the sections ahead.[1]

1. A much shorter version this work appeared in *NIPS* 2007 (Huang et al., 2007). We provide a more complete discussion of our Fourier based methods in this extended paper.

- In Sections 4 and 5, we provide a gentle introduction to the theory of group representations and noncommutative Fourier analyis. While none of the results of these sections are novel, and have indeed been studied by mathematicians for decades (Diaconis, 1989; Terras, 1999; Willsky, 1978; Chen, 1989), noncommutative Fourier analysis is still fairly new to the machine learning community, which has just begun to discover some of its exciting applications (Huang et al., 2007, 2009; Kondor et al., 2007; Kondor and Borgwardt, 2008). Our tutorial sections are targeted specifically at the machine learning community and describe its connections to probabilistic inference problems that involve permutations.

- In Section 6, we discuss performing probabilistic inference operations in the Fourier domain. In particular, we present Fourier theoretic algorithms for two ubiquitous operations which appear in filtering applications and beyond: prediction/rollup and conditioning with Bayes rule. Our main contribution in this section is a novel and conceptually simple algorithm, called *Kronecker Conditioning*, which performs all conditioning operations completely in the Fourier domain, allowing for a principled tradeoff between computational complexity and approximation accuracy. Our approach generalizes upon previous work in two ways—first, in the sense that it can address any transition model or likelihood function that can be represented in the Fourier domain, and second, in the sense that many of our results hold for arbitrary finite groups.

- In Section 7, we analyze the errors which can be introduced by bandlimiting a probability distribution and show how they propagate with respect to inference operations. We argue that approximate conditioning based on bandlimited distributions can sometimes yield Fourier coefficients which do not correspond to any valid distribution, even returning negative "probabilities" on occasion. We address possible negative and inconsistent probabilities by presenting a method for projecting the result back into the polytope of coefficients which correspond to nonnegative and consistent marginal probabilities using a simple quadratic program.

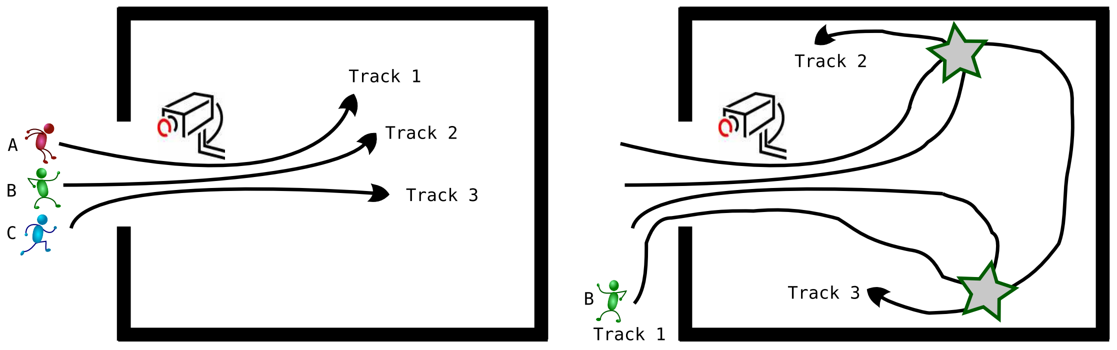
- In Section 8, we present a collection of general techniques for efficiently computing the Fourier coefficients of probabilistic models that might be useful in practical inference problems, and give a variety of examples of such computations for probabilistic models that might arise in identity management or ranking scenarios.

- Finally in Section 10, we empirically evaluate the accuracy of approximate inference on simulated data drawn from our model and further demonstrate the effectiveness of our approach on a real camera-based multi-person tracking scenario.

2. Filtering Over Permutations

As a prelude to the general problem statement, we begin with a simple identity management problem on three tracks (illustrated in Figure 2) which we will use as a running example. In this problem, we observe a stream of localization data from three people walking inside a room. Except for a camera positioned at the entrance, however, there is no way to distinguish between identities once they are inside. In this example, an internal tracker declares that two tracks have 'mixed' whenever they get too close to each other and announces the identity of any track that enters or exits the room.

In our particular example, three people, Alice, Bob and Cathy, enter a room separately, walk around, and we observe Bob as he exits. The events for our particular example in the figure are



(a) *Before* (b) *After*

Figure 2: Identity Management example. Three people, Alice, Bob and Charlie enter a room and we receive a position measurement for each person at each time step. With no way to observe identities inside the room, however, we are confused whenever two tracks get too close. In this example, track 1 crosses with track 2, then with track 3, then leaves the room, at which point it is observed that the identity at Track 1 is in fact Bob.

recorded in Table 1. Since Tracks 2 and 3 never mix, we know that Cathy cannot be in Track 2 in the end, and furthermore, since we observe Bob to be in Track 1 when he exits, we can deduce that Cathy must have been in Track 3, and therefore Alice must have been in Track 2. Our simple example illustrates the combinatorial nature of the problem—in particular, reasoning about the mixing events allows us to exactly decide where Alice and Cathy were even though we only made an observation about Bob at the end.

Event #	*Event Type*
1	*Tracks 1 and 2 mixed*
2	*Tracks 1 and 3 mixed*
3	*Observed Identity Bob at Track 1*

Table 1: Table of Mixing and Observation events logged by the tracker.

In identity management, a permutation σ represents a joint assignment of identities to internal tracks, with $\sigma(i)$ being the track belonging to the ith identity. When people walk too closely together, their identities can be confused, leading to uncertainty over σ. To model this uncertainty, we use a *Hidden Markov Model (HMM)* on permutations, which is a joint distribution over latent permutations $\sigma^{(1)},\dots,\sigma^{(T)}$, and observed variables $z^{(1)},\dots,z^{(T)}$ which factors as:

$$P(\sigma^{(1)},\dots,\sigma^{(T)},z^{(1)},\dots,z^{(T)}) = P(\sigma^{(1)})P(z^{(1)}|\sigma^{(1)})\prod_{t=2}^{T} P(z^t|\sigma^{(t)})\cdot P(\sigma^{(t)}|\sigma^{(t-1)}).$$

The conditional probability distribution $P(\sigma^{(t)}|\sigma^{(t-1)})$ is called the *transition model*, and might reflect, for example, that the identities belonging to two tracks were swapped with some probability by a mixing event. The distribution $P(z^{(t)}|\sigma^{(t)})$ is called the *observation model*, which might, for example, capture a distribution over the color of clothing for each individual.

We focus on *filtering*, in which one queries the HMM for the posterior at some time step, conditioned on all past observations. Given the distribution $P(\sigma^{(t)}|z^{(1)},\dots,z^{(t)})$, we recursively compute $P(\sigma^{(t+1)}|z^{(1)},\dots,z^{(t+1)})$ in two steps: a *prediction/rollup* step and a *conditioning* step. Taken together, these two steps form the well known *Forward Algorithm* (Rabiner, 1989). The prediction/rollup step multiplies the distribution by the transition model and marginalizes out the previous time step:

$$P(\sigma^{(t+1)}|z^{(1)},\dots,z^{(t)}) = \sum_{\sigma^{(t)}} P(\sigma^{(t+1)}|\sigma^{(t)})P(\sigma^{(t)}|z^{(1)},\dots,z^{(t)}).$$

The conditioning step conditions the distribution on an observation $z^{(t+1)}$ using Bayes rule:

$$P(\sigma^{(t+1)}|z^{(1)},\dots,z^{(t+1)}) \propto P(z^{(t+1)}|\sigma^{(t+1)})P(\sigma^{(t+1)}|z^{(1)},\dots,z^{(t)}).$$

Since there are $n!$ permutations, a single iteration of the algorithm requires $O((n!)^2)$ flops and is consequently intractable for all but very small n. The approach that we advocate is to maintain a compact approximation to the true distribution based on the Fourier transform. As we discuss later, the Fourier based approximation is equivalent to maintaining a set of low-order marginals, rather than the full joint, which we regard as being analogous to an *Assumed Density Filter* (Boyen and Koller, 1998).

Although we use hidden Markov models and filtering as a running example, the approach we describe is useful for other probabilistic inference tasks over permutations, such as ranking objects and modeling user preferences. For example, operations such as marginalization and conditioning are fundamental and are widely applicable. In particular, conditioning using Bayes rule, one of the main topics of our paper, is one of the most fundamental probabilistic operations, and we provide a completely general formulation.

3. Probability Distributions over the Symmetric Group

A permutation on n elements is a one-to-one mapping of the set $\{1,\dots,n\}$ into itself and can be written as a tuple,

$$\sigma = [\sigma(1)\ \sigma(2)\ \dots\ \sigma(n)],$$

where $\sigma(i)$ denotes where the ith element is mapped under the permutation (called *one line notation*). For example, $\sigma = [2\ 3\ 1\ 4\ 5]$ means that $\sigma(1) = 2$, $\sigma(2) = 3$, $\sigma(3) = 1$, $\sigma(4) = 4$, and $\sigma(5) = 5$. The set of all permutations on n elements forms a group under the[2] operation of function composition—that is, if σ_1 and σ_2 are permutations, then

$$\sigma_1\sigma_2 = [\sigma_1(\sigma_2(1))\ \sigma_1(\sigma_2(2))\ \dots\ \sigma_1(\sigma_2(n))]$$

is itself a permutation. The set of all $n!$ permutations is called the *symmetric group*, or just S_n.

We will actually notate the elements of S_n using the more standard *cycle notation*, in which a *cycle* $(i,j,k,\dots,\ell)$ refers to the permutation which maps i to j, j to k, $\dots$, and finally ℓ to i. Though not every permutation can be written as a single cycle, any permutation can always be written as a product of disjoint cycles. For example, the permutation $\sigma = [2\ 3\ 1\ 4\ 5]$ written in cycle notation is $\sigma = (1,2,3)(4)(5)$. The number of elements in a cycle is called the *cycle length* and we typically drop the length 1 cycles in cycle notation when it creates no ambiguity—in our

2. See Appendix A for a list of the basic group theoretic definitions used in this paper.

example, $\sigma = (1,2,3)(4)(5) = (1,2,3)$. We refer to the identity permutation (which maps every element to itself) as ε.

A probability distribution over permutations can be thought of as a joint distribution on the n random variables $(\sigma(1),\dots,\sigma(n))$ subject to the *mutual exclusivity constraints* that $P(\sigma : \sigma(i) = \sigma(j)) = 0$ whenever $i \neq j$. For example, in the identity management problem, Alice and Bob cannot both be in Track 1 simultaneously. Due to the fact that all of the $\sigma(i)$ are coupled in the joint distribution, graphical models, which might have otherwise exploited an underlying conditional independence structure, are ineffective. Instead, our Fourier based approximation achieves compactness by exploiting the *algebraic structure* of the problem.

3.1 Compact Summary Statistics

While continuous distributions like Gaussians are typically summarized using moments (like mean and variance), or more generally, expected features, it is not immediately obvious how one might, for example, compute the 'mean' of a distribution over permutations. There is a simple method that might spring to mind, however, which is to think of the permutations as *permutation matrices* and to average the matrices instead.

Example 1 *For example, consider the two permutations* $\varepsilon, (1,2) \in S_3$ *(*ε *is the identity and* $(1,2)$ *swaps 1 and 2). We can associate the identity permutation* ε *with the* 3×3 *identity matrix, and similarly, we can associate the permutation* $(1,2)$ *with the matrix:*

$$(1,2) \mapsto \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

The 'average' of ε *and* $(1,2)$ *is therefore:*

$$\frac{1}{2}\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} + \frac{1}{2}\begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1/2 & 1/2 & 0 \\ 1/2 & 1/2 & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

As we will later show, computing the 'mean' (as described above) of a distribution over permutations, P, compactly summarizes P by storing a marginal distribution over each of $\sigma(1), \sigma(2), \dots, \sigma(n)$, which requires storing only $O(n^2)$ numbers rather than the full $O(n!)$ for the exact distribution. As an example, one possible summary might look like:

$$\widehat{P} = \left[\begin{array}{c|ccc} & \text{Alice} & \text{Bob} & \text{Cathy} \\ \hline \text{Track 1} & 2/3 & 1/6 & 1/6 \\ \text{Track 2} & 1/3 & 1/3 & 1/3 \\ \text{Track 3} & 0 & 1/2 & 1/2 \end{array}\right].$$

Such doubly stochastic "first-order summaries" have been studied in various settings (Shin et al., 2003; Helmbold and Warmuth, 2007). In identity management (Shin et al., 2003),[3] first-order sum-

3. Strictly speaking, a map from identities to tracks is not a permutation since a permutation always maps a set into itself. In fact, the set of all such identity-to-track assignments does not actually form a group since there is no way to compose any two such assignments to obtain a legitimate group operation. We abuse the notation by referring to these assignments as a group, but really the elements of the group here should be thought of as the 'deviation' from the original identity-to-track assignment (where only the tracks are permuted, for example, when they are confused). In the group theoretic language, there is a faithful group action of S_n on the set of all identity-to-track assignments.

maries maintain, for example,

$$P(\text{Alice is at Track 1}) = 2/3,$$
$$P(\text{Bob is at Track 3}) = 1/2.$$

What cannot be captured by first-order summaries however, are the higher order statements like:

$$P(\text{Alice is in Track 1 } \textit{and} \text{ Bob is in Track 2}) = 0.$$

Over the next two sections, we will show that the first-order summary of a distribution $P(\sigma)$ can equivalently be viewed as the lowest frequency coefficients of the Fourier transform of $P(\sigma)$, and that by considering higher frequencies, we can capture higher order marginal probabilities in a principled fashion. Furthermore, the Fourier theoretic perspective, as we will show, provides a natural framework for formulating inference operations with respect to our compact summaries. In a nutshell, we will view the prediction/rollup step as a convolution and the conditioning step as a pointwise product—then we will formulate the two inference operations in the Fourier domain as a pointwise product and convolution, respectively.

4. The Fourier Transform on Finite Groups

Over the last fifty years, the Fourier Transform has been ubiquitously applied to everything digital, particularly with the invention of the Fast Fourier Transform (Cooley and Tukey, 1965; Rockmore, 2000). On the real line, the Fourier Transform is a well-studied method for decomposing a function into a sum of sine and cosine terms over a spectrum of frequencies. Perhaps less familiar to the machine learning community though, is its group theoretic generalization. In this section we review group theoretic generalizations of the Fourier transform with an eye towards approximating functions on S_n. None of the results stated in this section or the next are original. Noncommutative generalizations of the Fourier transform have been studied quite extensively throughout the last century from both the mathematics (Lang, 1965) and physics communities (Chen, 1989). Applications to permutations were first pioneered by Persi Diaconis who studied problems in card shuffling and since then, there have been many papers on related topics in probability and statistics. For further information, see Diaconis (1988) and Terras (1999).

4.1 Group Representation Theory

The generalized definition of the Fourier Transform relies on the theory of group representations, which formalize the concept of associating permutations with matrices and are used to construct a complete basis for the space of functions on a group G, thus also playing a role analogous to that of sinusoids on the real line.

Definition 1 *A* representation *of a group G is a map ρ from G to a set of invertible $d_\rho \times d_\rho$ (complex) matrix operators ($\rho : G \to \mathbb{C}^{d_\rho \times d_\rho}$) which preserves algebraic structure in the sense that for all $\sigma_1, \sigma_2 \in G$, $\rho(\sigma_1\sigma_2) = \rho(\sigma_1) \cdot \rho(\sigma_2)$. The matrices which lie in the image of ρ are called the* representation matrices*, and we will refer to d_ρ as the* degree *of the representation.*

The requirement that $\rho(\sigma_1\sigma_2) = \rho(\sigma_1) \cdot \rho(\sigma_2)$ is analogous to the property that $e^{i(\theta_1+\theta_2)} = e^{i\theta_1} \cdot e^{i\theta_2}$ for the conventional sinusoidal basis. Each matrix entry, $\rho_{ij}(\sigma)$ defines some function over S_n:

$$\rho(\sigma) = \begin{bmatrix} \rho_{11}(\sigma) & \rho_{12}(\sigma) & \cdots & \rho_{1d_\rho}(\sigma) \\ \rho_{21}(\sigma) & \rho_{22}(\sigma) & \cdots & \rho_{2d_\rho}(\sigma) \\ \vdots & \vdots & \ddots & \vdots \\ \rho_{d_\rho 1}(\sigma) & \rho_{d_\rho 2}(\sigma) & \cdots & \rho_{d_\rho d_\rho}(\sigma) \end{bmatrix},$$

and consequently, each representation ρ simultaneously defines a set of d_ρ^2 functions over S_n. We will eventually think of group representations as the set of Fourier basis functions onto which we can project arbitrary functions.

Before moving onto examples, we make several remarks about the generality of this paper. First, while our paper is primarily focused on the symmetric group, many of its results hold for arbitrary finite groups. For example, there are a variety of finite groups that have been studied in applications, like metacyclic groups (Willsky, 1978), wreath product groups (Foote et al., 2004), etc. However, while some of these results will even extend with minimal effort to more general cases, such as locally compact groups, the assumption in all of the following results will be that G is finite, even if it is not explicitly stated. Specifically, most of the results in Sections 4, 6, and Appendix D.2 are intended to hold over any finite group, while the results of the remaining sections are specific to probabilistic inference over the symmetric group. Secondly, given an arbitrary finite group G, some of the algebraic results that we use require that the underlying field be the complex numbers. For the particular case of the symmetric group, however, we can in fact assume that the representations are real-valued matrices. Thus, throughout the paper, we will explicitly assume that the representations are real-valued.[4]

Example 2 *We begin by showing three examples of representations on the symmetric group.*

1. *The simplest example of a representation is called the* trivial representation $\rho_{(n)} : S_n \to \mathbb{R}^{1\times 1}$, *which maps each element of the symmetric group to 1, the multiplicative identity on the real numbers. The trivial representation is actually defined for every group, and while it may seem unworthy of mention, it plays the role of the constant basis function in the Fourier theory.*

2. *The* first-order permutation representation *of S_n, which we alluded to in Example 1, is the degree n representation, $\tau_{(n-1,1)}$ (we explain the terminology in Section 5) , which maps a permutation σ to its corresponding permutation matrix given by $[\tau_{(n-1,1)}(\sigma)]_{ij} = \mathbb{1}\{\sigma(j) = i\}$. For example, the first-order permutation representation on S_3 is given by:*

$$\tau_{(2,1)}(\varepsilon) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad \tau_{(2,1)}(1,2) = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad \tau_{(2,1)}(2,3) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

$$\tau_{(2,1)}(1,3) = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix} \qquad \tau_{(2,1)}(1,2,3) = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \qquad \tau_{(2,1)}(1,3,2) = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix}$$

4. To recover similar results for more complex-valued representations, one would have to replace matrix transposes by adjoints, etc.

3. *The* alternating representation *of S_n, maps a permutation σ to the determinant of $\tau_{(n-1,1)}(\sigma)$, which is $+1$ if σ can be equivalently written as the composition of an even number of pairwise swaps, and -1 otherwise. We write the alternating representation as $\rho_{(1,\ldots,1)}$ with n 1's in the subscript. For example, on S_4, we have:*

$$\rho_{(1,1,1,1)}((1,2,3)) = \rho_{(1,1,1,1)}((13)(12)) = +1.$$

The alternating representation can be interpreted as the 'highest frequency' basis function on the symmetric group, intuitively due to its high sensitivity to swaps. For example, if $\tau_{(1,\ldots,1)}(\sigma) = 1$, then $\tau_{(1,\ldots,1)}((12)\sigma) = -1$. In identity management, it may be reasonable to believe that the joint probability over all n identity labels should only change by a little if just two objects are mislabeled due to swapping—in this case, ignoring the basis function corresponding to the alternating representation should still provide an accurate approximation to the joint distribution.

In general, a representation corresponds to an overcomplete set of functions and therefore does not constitute a valid basis for any subspace of functions. For example, the set of nine functions on S_3 corresponding to $\tau_{(2,1)}$ span only four dimensions, because there are six normalization constraints (three on the row sums and three on the column sums), of which five are independent—and so there are five redundant dimensions. To find a valid complete basis for the space of functions on S_n, we will need to find a family of representations whose basis functions are independent, and span the entire $n!$-dimensional space of functions.

In the following two definitions, we will provide two methods for constructing a new representation from old ones such that the set of functions on S_n corresponding to the new representation is linearly *dependent* on the old representations. Somewhat surprisingly, it can be shown that dependencies which arise amongst the representations can always be recognized in a certain sense, to come from the two possible following sources (Serre, 1977).

Definition 2

1. ***Equivalence.*** *Given a representation ρ_1 and an invertible matrix C, one can define a new representation ρ_2 by "changing the basis" for ρ_1:*

$$\rho_2(\sigma) \triangleq C^{-1} \cdot \rho_1(\sigma) \cdot C. \tag{1}$$

We say, in this case, that ρ_1 and ρ_2 are equivalent *as representations (written $\rho_1 \equiv \rho_2$, as opposed to $\rho_1 = \rho_2$), and the matrix C is known as the* intertwining operator. *Note that $d_{\rho_1} = d_{\rho_2}$.*

It can be checked that the functions corresponding to ρ_2 can be reconstructed from those corresponding to ρ_1. For example, if C is a permutation matrix, the matrix entries of ρ_2 are exactly the same as the matrix entries of ρ_1, only permuted.

2. ***Direct Sum.*** *Given two representations ρ_1 and ρ_2, we can always form a new representation, which we will write as $\rho_1 \oplus \rho_2$, by defining:*

$$\rho_1 \oplus \rho_2(\sigma) \triangleq \left[\begin{array}{c|c} \rho_1(\sigma) & 0 \\ \hline 0 & \rho_2(\sigma) \end{array}\right].$$

$\rho_1 \oplus \rho_2$ *is called the* direct sum representation. *For example, the direct sum of two copies of the trivial representation is:*

$$\rho_{(n)} \oplus \rho_{(n)}(\sigma) = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix},$$

with four corresponding functions on S_n, each of which is clearly dependent upon the trivial representation itself.

Most representations can be seen as being equivalent to a direct sum of strictly smaller representations. Whenever a representation ρ can be decomposed as $\rho \equiv \rho_1 \oplus \rho_2$, we say that ρ is *reducible*. As an example, we now show that the first-order permutation representation is a reducible representation.

Example 3 *Instead of using the standard basis vectors $\{e_1, e_2, e_3\}$, the first-order permutation representation for S_3, $\tau_{(2,1)} : S_3 \to \mathbb{C}^{3\times 3}$, can be equivalently written with respect to a new basis $\{v_1, v_2, v_3\}$, where:*

$$v_1 = \frac{e_1 + e_2 + e_3}{|e_1 + e_2 + e_3|},$$
$$v_2 = \frac{-e_1 + e_2}{|-e_1 + e_2|},$$
$$v_3 = \frac{-e_1 - e_2 + 2e_3}{|-e_1 - e_2 + 2e_3|}.$$

To 'change the basis', we write the new basis vectors as columns in a matrix C:

$$C = \begin{bmatrix} | & | & | \\ v_1 & v_2 & v_3 \\ | & | & | \end{bmatrix} = \begin{bmatrix} \frac{1}{\sqrt{3}} & -\frac{\sqrt{2}}{2} & -\frac{1}{\sqrt{6}} \\ \frac{1}{\sqrt{3}} & \frac{\sqrt{2}}{2} & -\frac{1}{\sqrt{6}} \\ \frac{1}{\sqrt{3}} & 0 & \frac{2}{\sqrt{6}} \end{bmatrix},$$

and conjugate the representation $\tau_{(2,1)}$ by C (as in Equation 1) to obtain the equivalent representation $C^{-1} \cdot \tau_{(2,1)}(\sigma) \cdot C$:

$$C^{-1} \cdot \tau_{(2,1)}(\varepsilon) \cdot C = \left[\begin{array}{c|cc} 1 & 0 & 0 \\ \hline 0 & 1 & 0 \\ 0 & 0 & 1 \end{array}\right] \qquad C^{-1} \cdot \tau_{(2,1)}(1,2) \cdot C = \left[\begin{array}{c|cc} 1 & 0 & 0 \\ \hline 0 & -1 & 0 \\ 0 & 0 & 1 \end{array}\right]$$

$$C^{-1} \cdot \tau_{(2,1)}(2,3) \cdot C = \left[\begin{array}{c|cc} 1 & 0 & 0 \\ \hline 0 & 1/2 & \sqrt{3}/2 \\ 0 & \sqrt{3}/2 & -1/2 \end{array}\right] \qquad C^{-1} \cdot \tau_{(2,1)}(1,3) \cdot C = \left[\begin{array}{c|cc} 1 & 0 & 0 \\ \hline 0 & 1/2 & -\sqrt{3}/2 \\ 0 & -\sqrt{3}/2 & -1/2 \end{array}\right]$$

$$C^{-1} \cdot \tau_{(2,1)}(1,2,3) \cdot C = \left[\begin{array}{c|cc} 1 & 0 & 0 \\ \hline 0 & -1/2 & -\sqrt{3}/2 \\ 0 & \sqrt{3}/2 & -1/2 \end{array}\right] \qquad C^{-1} \cdot \tau_{(2,1)}(1,3,2) \cdot C = \left[\begin{array}{c|cc} 1 & 0 & 0 \\ \hline 0 & -1/2 & \sqrt{3}/2 \\ 0 & -\sqrt{3}/2 & -1/2 \end{array}\right]$$

The interesting property of this particular basis is that the new representation matrices all appear to be the direct sum of two smaller representations, a trivial representation, $\rho_{(3)}$ as the top left block, and a degree 2 representation in the bottom right which we will refer to as $\rho_{(2,1)}$.

Geometrically, the representation $\rho_{(2,1)}$ *can also be thought of as the group of rigid symmetries of the equilateral triangle with vertices:*

$$P_1 = \begin{bmatrix} \sqrt{3}/2 \\ 1/2 \end{bmatrix}, P_2 = \begin{bmatrix} -\sqrt{3}/2 \\ 1/2 \end{bmatrix}, P_3 = \begin{bmatrix} 0 \\ -1 \end{bmatrix}.$$

The matrix $\rho_{(2,1)}(1,2)$ *acts on the triangle by reflecting about the x-axis, and* $\rho_{(2,1)}(1,2,3)$ *by a* $\pi/3$ *counter-clockwise rotation.*

In general, there are infinitely many reducible representations. For example, given any dimension d, there is a representation which maps every element of a group G to the $d \times d$ identity matrix (the direct sum of d copies of the trivial representation). However, for any finite group, there exists a finite collection of atomic representations which can be used to build up any other representation (up to equivalence) using the direct sum operation. These representations are referred to as the *irreducibles* of a group, and they are defined simply to be the collection of representations (up to equivalence) which are not reducible. It can be shown that any (complex) representation of a finite group G is equivalent to a direct sum of irreducibles (Diaconis, 1988), and hence, for any representation τ, there exists a matrix C for which

$$C^{-1} \cdot \tau \cdot C = \bigoplus_{\rho} \bigoplus_{j=1}^{z_\rho} \rho, \tag{2}$$

where ρ ranges over all distinct irreducible representations of the group G, and the inner $\oplus$ refers to some finite number (z_ρ) of copies of each irreducible ρ.

As it happens, there are only three irreducible representations of S_3 (Diaconis, 1988), up to equivalence: the trivial representation $\rho_{(3)}$, the degree 2 representation $\rho_{(2,1)}$, and the alternating representation $\rho_{(1,1,1)}$. The complete set of irreducible representation matrices of S_3 are shown in the Table 2. Unfortunately, the analysis of the irreducible representations for $n > 3$ is far more complicated and we postpone this more general discussion for Section 5.

4.2 The Fourier Transform

The link between group representation theory and Fourier analysis is given by the celebrated *Peter-Weyl theorem* (Diaconis, 1988; Terras, 1999; Sagan, 2001) which says that the matrix entries of the irreducibles of G form a *complete* set of *orthogonal* basis functions on G.[5] The space of functions on S_3, for example, is orthogonally spanned by the 3! functions $\rho_{(3)}(\sigma)$, $[\rho_{(2,1)}(\sigma)]_{1,1}$, $[\rho_{(2,1)}(\sigma)]_{1,2}$, $[\rho_{(2,1)}(\sigma)]_{2,1}$, $[\rho_{(2,1)}(\sigma)]_{2,2}$ and $\rho_{(1,1,1)}(\sigma)$, where $[\rho(\sigma)]_{ij}$ denotes the (i,j) entry of the matrix $\rho(\sigma)$.

As a replacement for projecting a function f onto a complete set of sinusoidal basis functions (as one would do on the real line), the Peter-Weyl theorem suggests instead to project onto the basis provided by the irreducibles of G. As on the real line, this projection can be done by computing the inner product of f with each element of the basis, and we define this operation to be the generalized form of the Fourier Transform.

5. Technically the Peter-Weyl result, as stated here, is only true if all of the representation matrices are unitary. That is, $\rho(\sigma)^*\rho(\sigma) = I$ for all $\sigma \in S_n$, where the matrix A^* is the conjugate transpose of A. For the case of real-valued (as opposed to complex-valued) matrices, however, the definitions of unitary and orthogonal matrices coincide.

While most representations are not unitary, there is a standard result from representation theory which shows that for *any* representation of G, there exists an equivalent unitary representation.

σ	$\rho_{(3)}$	$\rho_{(2,1)}$	$\rho_{(1,1,1)}$
ε	1	$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$	1
$(1,2)$	1	$\begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}$	-1
$(2,3)$	1	$\begin{bmatrix} 1/2 & \sqrt{3}/2 \\ \sqrt{3}/2 & -1/2 \end{bmatrix}$	-1
$(1,3)$	1	$\begin{bmatrix} 1/2 & -\sqrt{3}/2 \\ -\sqrt{3}/2 & -1/2 \end{bmatrix}$	-1
$(1,2,3)$	1	$\begin{bmatrix} -1/2 & -\sqrt{3}/2 \\ \sqrt{3}/2 & -1/2 \end{bmatrix}$	1
$(1,3,2)$	1	$\begin{bmatrix} -1/2 & \sqrt{3}/2 \\ -\sqrt{3}/2 & -1/2 \end{bmatrix}$	1

Table 2: The irreducible representation matrices of S_3.

Definition 3 *Let $f : G \to \mathbb{R}$ be any function on a group G and let ρ be any representation on G. The* Fourier Transform *of f at the representation ρ is defined to be the matrix of coefficients:*

$$\hat{f}_\rho = \sum_\sigma f(\sigma)\rho(\sigma).$$

The collection of Fourier Transforms at all irreducible representations of G form the Fourier Transform of f.

There are two important points which distinguish this Fourier Transform from its familiar formulation on the real line—first, the outputs of the transform are matrix-valued, and second, the inputs to $\hat{f}$ are *representations* of G rather than real numbers. As in the familiar formulation, the Fourier Transform is invertible and the inversion formula is explicitly given by the Fourier Inversion Theorem.

Theorem 4 (Fourier Inversion Theorem)

$$f(\sigma) = \frac{1}{|G|} \sum_\lambda d_{\rho_\lambda} Tr \left[\hat{f}^T_{\rho_\lambda} \cdot \rho_\lambda(\sigma) \right], \tag{3}$$

where λ indexes over the collection of irreducibles of G.

Note that the trace term in the inverse Fourier Transform is just the 'matrix dot product' between $\hat{f}_{\rho_\lambda}$ and $\rho_\lambda(\sigma)$, since $\mathrm{Tr}\left[A^T \cdot B\right] = \langle \mathrm{vec}(A), \mathrm{vec}(B) \rangle$, where by vec we mean mapping a matrix to a vector on the same elements arranged in column-major order.

We now provide several examples for intuition. For functions on the real line, the Fourier Transform at zero frequency gives the DC component of a signal. The same holds true for functions on a group; If $f : G \to \mathbb{R}$ is any function, then since $\rho_{(n)} = 1$, the Fourier Transform of f at the trivial representation is constant, with $\hat{f}_{\rho_{(n)}} = \sum_\sigma f(\sigma)$. Thus, for any probability distribution P, we have $\hat{P}_{\rho_{(n)}} = 1$. If P were the uniform distribution, then $\hat{P}_\rho = 0$ at every irreducible ρ except at the trivial representation.

The Fourier Transform at $\tau_{(n-1,1)}$ also has a simple interpretation:

$$[\hat{f}_{\tau_{(n-1,1)}}]_{ij} = \sum_{\sigma\in S_n} f(\sigma)[\tau_{(n-1,1)}(\sigma)]_{ij} = \sum_{\sigma\in S_n} f(\sigma)\mathbb{1}\{\sigma(j)=i\} = \sum_{\sigma:\sigma(j)=i} f(\sigma).$$

The set $\Delta_{ij} = \{\sigma : \sigma(j) = i\}$ is the set of the $(n-1)!$ possible permutations which map element j to i. In identity management, Δ_{ij} can be thought of as the set of assignments which, for example, have Alice at Track 1. If P is a distribution, then $\hat{P}_{\tau_{(n-1,1)}}$ is a matrix of *first-order* marginal probabilities, where the (i,j)-th element is the marginal probability that a random permutation drawn from P maps element j to i.

Example 4 *Consider the following probability distribution on S_3:*

σ	ε	$(1,2)$	$(2,3)$	$(1,3)$	$(1,2,3)$	$(1,3,2)$
$P(\sigma)$	$1/3$	$1/6$	$1/3$	0	$1/6$	0

The set of all first order marginal probabilities is given by the Fourier transform at $\tau_{(2,1)}$:

$$\widehat{P}_{\tau_{(2,1)}} = \left[\begin{array}{c|ccc} & \mathbf{A} & \mathbf{B} & \mathbf{C} \\ \hline \mathbf{1} & 2/3 & 1/6 & 1/6 \\ \mathbf{2} & 1/3 & 1/3 & 1/3 \\ \mathbf{3} & 0 & 1/2 & 1/2 \end{array}\right].$$

In the above matrix, each column j represents a marginal distribution over the possible tracks that identity j can map to under a random draw from P. We see, for example, that Alice is at Track 1 with probability 2/3, or at Track 2 with probability 1/3. Simultaneously, each row i represents a marginal distribution over the possible identities that could have been mapped to track i under a random draw from P. In our example, Bob and Cathy are equally likely to be in Track 3, but Alice is definitely not in Track 3. Since each row and each column is itself a distribution, the matrix $\widehat{P}_{\tau_{(2,1)}}$ must be doubly stochastic. We will elaborate on the consequences of this observation later.

The Fourier transform of the same distribution at all irreducibles is:

$$\widehat{P}_{\rho_{(3)}} = 1, \quad \widehat{P}_{\rho_{(2,1)}} = \begin{bmatrix} 1/4 & \sqrt{3}/4 \\ \sqrt{3}/4 & 1/4 \end{bmatrix}, \quad \widehat{P}_{\rho_{(1,1,1)}} = 0.$$

The first-order permutation representation, $\tau_{(n-1,1)}$, captures the statistics of how a random permutation acts on a single object irrespective of where all of the other $n-1$ objects are mapped, and in doing so, compactly summarizes the distribution with only $O(n^2)$ numbers. Unfortunately, as mentioned in Section 3, the Fourier transform at the first-order permutation representation cannot capture more complicated statements like:

$$P(\text{Alice and Bob occupy Tracks 1 and 2}) = 0.$$

To avoid collapsing away so much information, we might define richer summary statistics that might capture 'higher-order' effects. We define the *second-order unordered permutation representation* by:

$$[\tau_{(n-2,2)}(\sigma)]_{\{i,j\},\{k,\ell\}} = \mathbb{1}\left\{\sigma(\{k,\ell\}) = \{i,j\}\right\},$$

where we index the matrix rows and columns by unordered pairs $\{i,j\}$. The condition inside the indicator function states that the representation captures whether the pair of objects $\{k,\ell\}$ maps to the pair $\{i,j\}$, but is indifferent with respect to the ordering; that is, either $k \mapsto i$ and $\ell \mapsto j$, *or*, $k \mapsto j$ and $\ell \mapsto i$.

Example 5 *For $n=4$, there are six possible unordered pairs: $\{1,2\}$,$\{1,3\}$,$\{1,4\}$,$\{2,3\}$,$\{2,4\}$, and $\{3,4\}$. The matrix representation of the permutation $(1,2,3)$ is:*

$$\tau_{(2,2)}(1,2,3) = \left[\begin{array}{c|cccccc} & \{1,2\} & \{1,3\} & \{1,4\} & \{2,3\} & \{2,4\} & \{3,4\} \\ \hline \{1,2\} & 0 & 0 & 0 & 1 & 0 & 0 \\ \{1,3\} & 1 & 0 & 0 & 0 & 0 & 0 \\ \{1,4\} & 0 & 0 & 0 & 0 & 1 & 0 \\ \{2,3\} & 0 & 1 & 0 & 0 & 0 & 0 \\ \{2,4\} & 0 & 0 & 0 & 0 & 0 & 1 \\ \{3,4\} & 0 & 0 & 1 & 0 & 0 & 0 \end{array}\right].$$

The *second order ordered permutation representation*, $\tau_{(n-2,1,1)}$, is defined similarly:

$$[\tau_{(n-2,1,1)}(\sigma)]_{(i,j),(k,\ell)} = \mathbb{1}\left\{\sigma((k,\ell)) = (i,j)\right\},$$

where (k,ℓ) denotes an *ordered* pair. Therefore, $[\tau_{(n-2,1,1)}(\sigma)]_{(i,j),(k,\ell)}$ is 1 if and only if σ maps k to i *and* ℓ to j.

As in the first-order case, the Fourier transform of a probability distribution at $\tau_{(n-2,2)}$, returns a matrix of marginal probabilities of the form: $P(\sigma : \sigma(\{k,\ell\}) = \{i,j\})$, which captures statements like, "Alice and Bob occupy Tracks 1 and 2 with probability 1/2". Similarly, the Fourier transform at $\tau_{(n-2,1,1)}$ returns a matrix of marginal probabilities of the form $P(\sigma : \sigma((k,\ell)) = (i,j))$, which captures statements like, "Alice is in Track 1 *and* Bob is in Track 2 with probability 9/10".

We can go further and define third-order representations, fourth-order representations, and so on. In general however, the permutation representations as they have been defined above are reducible, intuitively due to the fact that it is possible to recover lower order marginal probabilities from higher order marginal probabilities. For example, one can recover the normalization constant (corresponding to the trivial representation) from the first order matrix of marginals by summing across either the rows or columns, and the first order marginal probabilities from the second order marginal probabilities by summing across appropriate matrix entries. To truly leverage the machinery of Fourier analysis, it is important to understand the Fourier transform at the irreducibles of the symmetric group, and in the next section, we show how to derive the irreducible representations of the Symmetric group by first defining permutation representations, then "subtracting off the lower-order effects".

5. Representation Theory on the Symmetric Group

In this section, we provide a brief introduction to the representation theory of the Symmetric group. Rather than giving a fully rigorous treatment of the subject, our goal is to give some intuition about the kind of information which can be captured by the irreducible representations of S_n. Roughly speaking, we will show that Fourier transforms on the Symmetric group, instead of being indexed by frequencies, are indexed by *partitions* of n (tuples of numbers which sum to n), and certain partitions correspond to more complex basis functions than others. For proofs, we point the reader to consult: Diaconis (1989), James and Kerber (1981), Sagan (2001) and Vershik and Okounkov (2006).

Instead of the singleton or pairwise marginals which were described in the previous section, we will now focus on using the Fourier coefficients of a distribution to query a much wider class of marginal probabilities. As an example, we will be able to compute the following (more complicated) marginal probability on S_6 using Fourier coefficients:

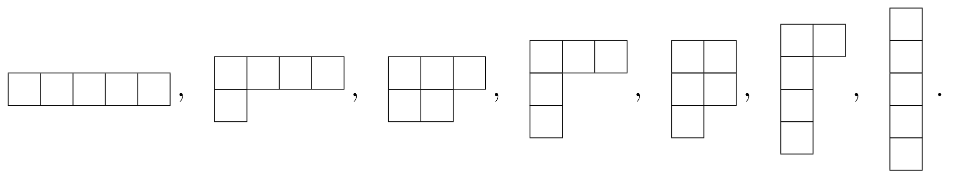
$$P\left(\sigma : \sigma\left(\left\{\begin{ytableau} 1 & 2 & 3 \\ 4 & 5 \\ 6 \end{ytableau}\right\}\right) = \left\{\begin{ytableau} 1 & 2 & 6 \\ 4 & 5 \\ 3 \end{ytableau}\right\}\right), \tag{4}$$

which we interpret as the joint marginal probability that the rows of the diagram on the left map to corresponding rows on the right as unordered sets. In other words, Equation 4 is the joint probability that *unordered* set $\{1,2,3\}$ maps to $\{1,2,6\}$, the unordered pair $\{4,5\}$ maps to $\{4,5\}$, and the singleton $\{6\}$ maps to $\{3\}$.

The diagrams in Equation 4 are known as *Ferrer's diagrams* and are commonly used to visualize *partitions* of n, which are defined to be unordered tuples of positive integers, $\lambda = (\lambda_1, \dots, \lambda_\ell)$, which sum to n. For example, $\lambda = (3,2)$ is a partition of $n = 5$ since $3+2 = 5$. Usually we write partitions as weakly decreasing sequences by convention, so the partitions of $n = 5$ are:

$$(5),\ (4,1),\ (3,2),\ (3,1,1),\ (2,2,1),\ (2,1,1,1),\ (1,1,1,1,1),$$

and their respective Ferrers diagrams are:



A *Young tabloid* is an assignment of the numbers $\{1,\dots,n\}$ to the boxes of a Ferrers diagram for a partition λ, where each row represents an unordered set. There are 6 Young tabloids corresponding to the partition $\lambda = (2,2)$, for example:

$$\left\{\begin{ytableau} 1 & 2 \\ 3 & 4 \end{ytableau}\right\}, \left\{\begin{ytableau} 1 & 3 \\ 2 & 4 \end{ytableau}\right\}, \left\{\begin{ytableau} 1 & 4 \\ 2 & 3 \end{ytableau}\right\}, \left\{\begin{ytableau} 2 & 3 \\ 1 & 4 \end{ytableau}\right\}, \left\{\begin{ytableau} 2 & 4 \\ 1 & 3 \end{ytableau}\right\}, \left\{\begin{ytableau} 3 & 4 \\ 1 & 2 \end{ytableau}\right\}.$$

The Young tabloid, $\begin{ytableau} 1 & 2 \\ 3 & 4 \end{ytableau}$, for example, represents the two underordered sets $\{1,2\}$ and $\{3,4\}$, and if we were interested in computing the joint probability that $\sigma(\{1,2\}) = \{3,4\}$ and $\sigma(\{3,4\}) = \{1,2\}$, then we could write the problem in terms of Young tabloids as:

$$P\left(\sigma : \sigma\left(\left\{\begin{ytableau} 1 & 2 \\ 3 & 4 \end{ytableau}\right\}\right) = \left\{\begin{ytableau} 3 & 4 \\ 1 & 2 \end{ytableau}\right\}\right).$$

In general, we will be able to use the Fourier coefficients at irreducible representations to compute the marginal probabilities of Young tabloids. As we shall see, with the help of the *James Submodule theorem* (James and Kerber, 1981), the marginals corresponding to “simple” partitions will require very few Fourier coefficients to compute, which is one of the main strengths of working in the Fourier domain.

Example 6 *Imagine three separate rooms containing two tracks each, in which Alice and Bob are in room 1 occupying Tracks 1 and 2; Cathy and David are in room 2 occupying Tracks 3 and 4; and Eric and Frank are in room 3 occupying Tracks 5 and 6, but we are not able to distinguish which person is at which track in any of the rooms. Then*

$$P\left(\sigma : \left(\left\{\begin{array}{|c|c|}\hline A & B \\ \hline C & D \\ \hline E & F \\ \hline\end{array}\right\}\right) \to \left\{\begin{array}{|c|c|}\hline 1 & 2 \\ \hline 3 & 4 \\ \hline 5 & 6 \\ \hline\end{array}\right\}\right) = 1.$$

It is in fact, possible to recast the first-order marginals which were described in the previous section in the language of Young tabloids by noticing that, for example, if 1 maps to 1, then the unordered set $\{2,\ldots,n\}$ must map to $\{2,\ldots,n\}$ since permutations are one-to-one mappings. The marginal probability that $\sigma(1) = 1$, then, is equal to the marginal probability that $\sigma(1) = 1$ *and* $\sigma(\{2,\ldots,n\}) = \{2,\ldots,n\}$. If $n = 6$, then the marginal probability written using Young tabloids is:

$$P\left(\sigma : \sigma\left(\left\{\begin{array}{|c|c|c|c|c|}\hline 2 & 3 & 4 & 5 & 6 \\ \hline 1 \\ \cline{1-1}\end{array}\right\}\right) = \left\{\begin{array}{|c|c|c|c|c|}\hline 2 & 3 & 4 & 5 & 6 \\ \hline 1 \\ \cline{1-1}\end{array}\right\}\right).$$

The first-order marginal probabilities correspond, therefore, to the marginal probabilities of Young tabloids of shape $\lambda = (n-1,1)$.

Likewise, the second-order unordered marginals correspond to Young tabloids of shape $\lambda = (n-2,2)$. If $n = 6$ again, then the marginal probability that $\{1,2\}$ maps to $\{2,4\}$ corresponds to the following marginal probability for tabloids:

$$P\left(\sigma : \sigma\left(\left\{\begin{array}{|c|c|c|c|}\hline 3 & 4 & 5 & 6 \\ \hline 1 & 2 \\ \cline{1-2}\end{array}\right\}\right) = \left\{\begin{array}{|c|c|c|c|}\hline 1 & 3 & 5 & 6 \\ \hline 2 & 4 \\ \cline{1-2}\end{array}\right\}\right).$$

The second-order *ordered* marginals are captured at the partition $\lambda = (n-2,1,1)$. For example, the marginal probability that $\{1\}$ maps to $\{2\}$ *and* $\{2\}$ maps to $\{4\}$ is given by:

$$P\left(\sigma : \sigma\left(\left\{\begin{array}{|c|c|c|c|}\hline 3 & 4 & 5 & 6 \\ \hline 1 \\ \cline{1-1} 2 \\ \cline{1-1}\end{array}\right\}\right) = \left\{\begin{array}{|c|c|c|c|}\hline 1 & 3 & 5 & 6 \\ \hline 2 \\ \cline{1-1} 4 \\ \cline{1-1}\end{array}\right\}\right).$$

And finally, we remark that the $(1,\ldots,1)$ partition of n recovers all original probabilities since it asks for a joint distribution over $\sigma(1),\ldots,\sigma(n)$. The corresponding matrix of marginals has $n! \times n!$ entries (though there will only be $n!$ distinct probabilities.

To see how the marginal probabilities of Young tabloids of shape λ can be thought of as Fourier coefficients, we will define a representation (which we call the *permutation representation*) associated with λ and show that the Fourier transform of a distribution at a permutation representation gives marginal probabilities. We begin by fixing an ordering on the set of possible Young tabloids, $\{t_1\}, \{t_2\}, \ldots$, and define the permutation representation $\tau_\lambda(\sigma)$ to be the matrix:

$$[\tau_\lambda(\sigma)]_{ij} = \begin{cases} 1 & \text{if } \sigma(\{t_j\}) = \{t_i\} \\ 0 & \text{otherwise} \end{cases} .$$

It can be checked that the function τ_λ is indeed a valid representation of the Symmetric group, and therefore we can compute Fourier coefficients at τ_λ. If $P(\sigma)$ is a probability distribution, then

$$\begin{aligned} \left[\widehat{P}_{\tau_\lambda}\right]_{ij} &= \sum_{\sigma \in S_n} P(\sigma)\,[\tau_\lambda(\sigma)]_{ij}, \\ &= \sum_{\{\sigma:\sigma(\{t_j\})=\{t_i\}\}} P(\sigma), \\ &= P(\sigma:\ \sigma(\{t_j\}) = \{t_i\}), \end{aligned}$$

and therefore, *the matrix of marginals corresponding to Young tabloids of shape* λ *is given* exactly *by the Fourier transform at the representation* τ_λ.

As we showed earlier, the simplest marginals (the zeroth order normalization constant), correspond to the Fourier transform at $\tau_{(n)}$, while the first-order marginals correspond to $\tau_{(n-1,1)}$, and the second-order unordered marginals correspond to $\tau_{(n-2,2)}$. The list goes on and on, with the marginals getting more complicated. At the other end of the spectrum, we have the Fourier coefficients at the representation $\tau_{(1,1,\ldots,1)}$ which exactly recover the original probabilities $P(\sigma)$.

We use the word 'spectrum' suggestively here, because the different levels of complexity for the marginals are highly reminiscent of the different frequencies for real-valued signals, and a natural question to ask is how the partitions might be ordered with respect to the 'complexity' of the corresponding basis functions. In particular how might one characterize this vague notion of complexity for a given partition?

The 'correct' characterization, as it turns out, is to use the *dominance ordering* of partitions, which, unlike the ordering on frequencies, is not a linear order, but rather, a partial order.

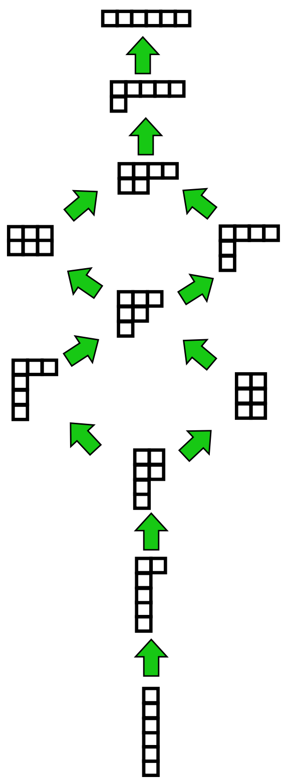
Definition 5 (Dominance Ordering) *Let* λ,μ *be partitions of* n. *Then* $\lambda \trianglerighteq \mu$ *(we say* λ dominates μ*), if for each* i, $\sum_{k=1}^{i} \lambda_k \geq \sum_{k=1}^{i} \mu_k$.

For example, $(4,2) \trianglerighteq (3,2,1)$ since $4 \geq 3$, $4+2 \geq 3+2$, and $4+2+0 \geq 3+2+1$. However, $(3,3)$ and $(4,1,1)$ cannot be compared with respect to the dominance ordering since $3 \leq 4$, but $3+3 \geq 4+1$. The ordering over the partitions of $n=6$ is depicted in Figure 3(a).

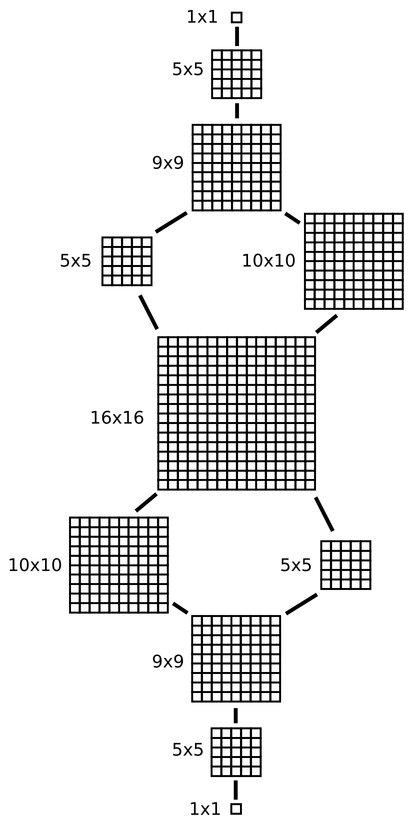
Partitions with fat Ferrers diagrams tend to be greater (with respect to dominance ordering) than those with skinny Ferrers diagrams. Intuitively, representations corresponding to partitions which are *high* in the dominance ordering are 'low frequency', while representations corresponding to partitions which are *low* in the dominance ordering are 'high frequency.'[6]

Having defined a family of intuitive permutation representations over the Symmetric group, we can now ask whether the permutation representations are irreducible or not: the answer in general, is to the negative, due to the fact that it is often possible to reconstruct lower order marginals by summing over the appropriate higher order marginal probabilities. However, it is possible to show

6. The direction of the ordering is slightly counterintuitive given the frequency interpretation, but is standard in the literature.



(a) Dominance ordering for $n = 6$.



(b) Fourier coefficient matrices for S_6.

Figure 3: The dominance order for partitions of $n = 6$ are shown in the left diagram (a). Fat Ferrer's diagrams tend to be higher in the order and long, skinny diagrams tend to be lower. The corresponding Fourier coefficient matrices for each partition (at irreducible representations) are shown in the right diagram (b). Note that since the Fourier basis functions form a complete basis for the space of functions on the Symmetric group, there must be exactly $n!$ coefficients in total.

that, for each permutation representation τ_λ, there exists a corresponding irreducible representation ρ_λ, which, loosely, captures all of the information at the 'frequency' λ which was not already captured at lower frequency irreducibles. Moreover, it can be shown that there exists no irreducible representation besides those indexed by the partitions of n. These remarkable results are formalized in the *James Submodule Theorem*, which we state here without proof (see Diaconis 1988, James and Kerber 1981 and Sagan 2001).

Theorem 6 (James' Submodule Theorem)

1. *(Uniqueness) For each partition, λ, of n, there exists an irreducible representation, ρ_λ, which is unique up to equivalence.*
2. *(Completeness) Every irreducible representation of S_n corresponds to some partition of n.*

3. *There exists a matrix C_λ associated with each partition λ, for which*

$$C_\lambda^T \cdot \tau_\lambda(\sigma) \cdot C_\lambda = \bigoplus_{\mu \trianglerighteq \lambda} \bigoplus_{\ell=1}^{K_{\lambda\mu}} \rho_\mu(\sigma), \quad \textit{for all } \sigma \in S_n. \tag{5}$$

4. $K_{\lambda\lambda} = 1$ *for all partitions* λ.

In plain English, part (3) of the James Submodule theorem says that we can always reconstruct marginal probabilities of λ-tabloids using the Fourier coefficients at irreducibles which lie *at* λ *and above* in the dominance ordering, if we have knowledge of the matrix C_λ (which can be precomputed using methods detailed in Appendix D), and the multiplicities $K_{\lambda\mu}$. In particular, combining Equation 5 with the definition of the Fourier transform, we have that

$$\hat{f}_{\tau_\lambda} = C_\lambda \cdot \left[\bigoplus_{\mu \trianglerighteq \lambda} \bigoplus_{\ell=1}^{K_{\lambda\mu}} \hat{f}_{\rho_\mu} \right] \cdot C_\lambda^T, \tag{6}$$

and so to obtain marginal probabilities of λ-tabloids, we simply construct a block diagonal matrix using the appropriate irreducible Fourier coefficients, and conjugate by C_λ. The multiplicities $K_{\lambda\mu}$ are known as the *Kostka numbers* and can be computed using Young's rule (Sagan, 2001). To illustrate using a few examples, we have the following decompositions:

$$\begin{aligned}
\tau_{(n)} &\equiv \rho_{(n)}, \\
\tau_{(n-1,1)} &\equiv \rho_{(n)} \oplus \rho_{(n-1,1)}, \\
\tau_{(n-2,2)} &\equiv \rho_{(n)} \oplus \rho_{(n-1,1)} \oplus \rho_{(n-2,2)}, \\
\tau_{(n-2,1,1)} &\equiv \rho_{(n)} \oplus \rho_{(n-1,1)} \oplus \rho_{(n-1,1)} \oplus \rho_{(n-2,2)} \oplus \rho_{(n-2,1,1)}, \\
\tau_{(n-3,3)} &\equiv \rho_{(n)} \oplus \rho_{(n-1,1)} \oplus \rho_{(n-2,2)} \oplus \rho_{(n-3,3)}, \\
\tau_{(n-3,2,1)} &\equiv \rho_{(n)} \oplus \rho_{(n-1,1)} \oplus \rho_{(n-1,1)} \oplus \rho_{(n-2,2)} \oplus \rho_{(n-2,2)} \oplus \rho_{(n-2,1,1)} \oplus \rho_{(n-3,3)} \oplus \rho_{(n-3,2,1)}.
\end{aligned}$$

Intuitively, the irreducibles at a partition λ reflect the "pure" λ^{th}-order effects of the underlying distribution. In other words, the irreducibles at λ form a basis for functions that have "interesting" λ^{th}-order marginal probabilities, but uniform marginals at all partitions μ such that $\mu \triangleright \lambda$.

Example 7 *As an example, we demonstrate a "preference" function which is "purely" second-order (unordered) in the sense that its Fourier coefficients are equal to zero at all irreducible representations except* $\rho_{(n-2,2)}$ *(and the trivial representation). Consider the function* $f : S_n \to \mathbb{R}$ *defined by:*

$$f(\sigma) = \begin{cases} 1 & \textit{if } |\sigma(1) - \sigma(2)| \equiv 1 \ (\textit{mod } n) \\ 0 & \textit{otherwise} \end{cases}.$$

Intuitively, imagine seating n people at a round table with n chairs, but with the constraint that the first two people, Alice and Bob, are only happy if they are allowed to sit next to each other. In this case, f can be thought of as the indicator function for the subset of seating arrangements (permutations) which make Alice and Bob happy.

Since f depends only on the destination of the unordered pair $\{1,2\}$*, its Fourier transform is zero at all partitions* μ *such that* $\mu \triangleleft (n-2,2)$ $(\hat{f}_\mu = 0)$*. On the other hand, Alice and Bob*

λ	(n)	$(n-1,1)$	$(n-2,2)$	$(n-2,1,1)$	$(n-3,3)$	$(n-3,2,1)$
$\dim \rho_\lambda$	1	$n-1$	$\frac{n(n-3)}{2}$	$\frac{(n-1)(n-2)}{2}$	$\frac{n(n-1)(n-5)}{6}$	$\frac{n(n-2)(n-4)}{3}$

Table 3: Dimensions of low-order irreducible representation matrices.

have no individual preferences for seating, so the first-order "marginals" of f are uniform, and hence, $\hat{f}_{(n-1,1)} = 0$. The Fourier coefficients at irreducibles can be obtained from the second-order (unordered) "marginals" using Equation 5.

$$C^T_{(n-2,2)} \cdot \hat{P}_{\tau_{(n-2,2)}} \cdot C_{(n-2,2)} = \begin{bmatrix} \boxed{Z} & & \\ & \boxed{0} & \\ & & \boxed{\hat{f}_{\rho_{(n-2,2)}}} \end{bmatrix}.$$

The sizes of the irreducible representation matrices are typically much smaller than their corresponding permutation representation matrices. In the case of $\lambda = (1, \dots, 1)$ for example, $\dim \tau_\lambda = n!$ while $\dim \rho_\lambda = 1$. There is a simple combinatorial algorithm, known as the *Hook Formula* (Sagan, 2001), for computing the dimension of ρ_λ. While we do not discuss it, we provide a few dimensionality computations here (Table 3) to facilitate a dicussion of complexity later. Despite providing polynomial sized function approximations, the Fourier coefficient matrices can grow quite fast, and roughly, one would need $O(n^{2k})$ storage to maintain kth order marginals. For example, we would need to store $O(n^8)$ elements to maintain fourth-order marginals. It is worth noting that since the Fourier transform is invertible, there must be $n!$ Fourier coefficients in total, and so $\sum_\rho d_\rho^2 = |G| = n!$. See Figure 3(b) for an example of what the matrices of a complete Fourier transform on S_6 would look like.

In practice, since the irreducible representation matrices are determined only up to equivalence, it is necessary to choose a basis for the irreducible representations in order to explicitly construct the representation matrices. As in Kondor et al. (2007), we use the *Gel'fand-Tsetlin basis* which has several attractive properties, two advantages being that the matrices are real-valued and orthogonal. See Appendix B for details on constructing irreducible matrix representations with respect to the Gel'fand-Tsetlin basis.

6. Inference in the Fourier Domain

What we have shown thus far is that there is a principled method for compactly summarizing distributions over permutations based on the idea of bandlimiting—saving only the low-frequency terms of the Fourier transform of a function, which, as we discussed, is equivalent to maintaining a set of low-order marginal probabilities. We now turn to the problem of performing probabilistic inference using our compact summaries. One of the main advantages of viewing marginals as Fourier coefficients is that it provides a natural principle for formulating polynomial time approximate inference algorithms, which is to rewrite all inference related operations with respect to the Fourier domain, then to perform the Fourier domain operations ignoring high-order terms.

The idea of bandlimiting a distribution is ultimately moot, however, if it becomes necessary to transform back to the primal domain each time an inference operation is called. Naively, the Fourier Transform on S_n scales as $O((n!)^2)$, and even the fastest Fast Fourier Transforms for functions on S_n are no faster than $O(n^2 \cdot n!)$ (see Maslen 1998 for example). To resolve this issue, we present a formulation of inference which operates solely in the Fourier domain, allowing us to avoid a costly transform. We begin by discussing exact inference in the Fourier domain, which is no more tractable than the original problem because there are $n!$ Fourier coefficients, but it will allow us to discuss the bandlimiting approximation in the next section. There are two operations to consider: prediction/rollup, and conditioning. While we have motivated both of these operations in the familiar context of hidden Markov models, they are fundamental and appear in many other settings. The assumption for the rest of this section is that the Fourier transforms of the transition and observation models are known. We discuss methods for obtaining the models in Section 8. The main results of this section (excluding the discussions about complexity) extend naturally to other finite groups besides S_n.

6.1 Fourier Prediction/Rollup

We will consider one particular class of transition models—that of random walks over a group, which assumes that $\sigma^{(t+1)}$ is generated from $\sigma^{(t)}$ by drawing a random permutation $\pi^{(t)}$ from some distribution $Q^{(t)}$ and setting $\sigma^{(t+1)} = \pi^{(t)}\sigma^{(t)}$.[7] In our identity management example, $\pi^{(t)}$ represents a random identity permutation that might occur among tracks when they get close to each other (what we call a *mixing event*). For example, $Q(1,2) = 1/2$ means that Tracks 1 and 2 swapped identities with probability 1/2. The random walk model also appears in many other applications such as modeling card shuffles (Diaconis, 1988).

The motivation behind the random walk transition model is that it allows us to write the prediction/rollup operation as a *convolution* of distributions on a group. The extension of the familiar notion of convolution to groups simply replaces additions and subtractions by analogous group operations (function composition and inverse, respectively):

Definition 7 *Let Q and P be probability distributions on a group G. Define the* convolution[8] *of Q and P to be the function* $[Q * P](\sigma_1) = \sum_{\sigma_2} Q(\sigma_1\sigma_2^{-1})P(\sigma_2)$.

Using Definition 7, we see that the prediction/rollup step can be written as:

$$\begin{aligned} P(\sigma^{(t+1)}) &= \sum_{\sigma^{(t)}} P(\sigma^{(t+1)}|\sigma^{(t)}) \cdot P(\sigma^{(t)}), \\ &= \sum_{\{(\sigma^{(t)},\pi^{(t)})\,:\,\sigma^{(t+1)}=\pi^{(t)}\cdot\sigma^{(t)}\}} Q^{(t)}(\pi^{(t)}) \cdot P(\sigma^{(t)}), \end{aligned}$$

7. We place π on the left side of the multiplication because we want it to permute tracks and not identities. Had we defined π to map from tracks to identities (instead of identities to tracks), then π would be multiplied from the right. Besides left versus right multiplication, there are no differences between the two conventions.

8. Note that this definition of convolution on groups is *strictly* a generalization of convolution of functions on the real line, and is a non-commutative operation for non-Abelian groups. Thus the distribution $P * Q$ is not necessarily the same as $Q * P$.

$$\begin{aligned}
&\text{(Right-multiplying both sides of } \sigma^{(t+1)} = \pi^{(t)}\sigma^{(t)} \\
&\qquad \text{by } (\sigma^{(t)})^{-1}\text{, we see that } \pi^{(t)} \text{ can be replaced by } \sigma^{(t+1)}(\sigma^{(t)})^{-1}), \\
&= \sum_{\sigma^{(t)}} Q^{(t)}(\sigma^{(t+1)} \cdot (\sigma^{(t)})^{-1}) \cdot P(\sigma^{(t)}), \\
&= \left[Q^{(t)} * P\right](\sigma^{(t+1)}).
\end{aligned}$$

As with Fourier transforms on the real line, the Fourier coefficients of the convolution of distributions P and Q on groups can be obtained from the Fourier coefficients of P and Q individually, using the *convolution theorem* (see also Diaconis 1988):

Proposition 8 (Convolution Theorem) *Let Q and P be probability distributions on a group G. For any representation ρ,*

$$\left[\widehat{Q * P}\right]_\rho = \widehat{Q}_\rho \cdot \widehat{P}_\rho,$$

where the operation on the right side is matrix multiplication.

Therefore, assuming that the Fourier transforms $\widehat{P}^{(t)}_\rho$ and $\widehat{Q}^{(t)}_\rho$ are given, the prediction/rollup update rule is simply:

$$\widehat{P}^{(t+1)}_\rho \leftarrow \widehat{Q}^{(t)}_\rho \cdot \widehat{P}^{(t)}_\rho.$$

Note that the update only requires knowledge of $\hat{P}$ and does not require P. Furthermore, the update is *pointwise* in the Fourier domain in the sense that the coefficients at the representation ρ affect $\widehat{P}^{(t+1)}_\rho$ *only* at ρ. Consequently, prediction/rollup updates in the Fourier domain never increase the representational complexity. For example, if we maintain third-order marginals, then a single step of prediction/rollup called at time t returns the *exact* third-order marginals at time $t+1$, and nothing more.

Example 8 *We run the prediction/rollup routines on the first two time steps of the example in Figure 2, first in the primal domain, then in the Fourier domain. At each mixing event, two tracks, i and j, swap identities with some probability. Using a mixing model given by:*

$$Q(\pi) = \begin{cases} 3/4 & \textit{if } \pi = \varepsilon \\ 1/4 & \textit{if } \pi = (i,j) \\ 0 & \textit{otherwise} \end{cases},$$

we obtain results shown in Tables 4 and 5.

6.1.1 COMPLEXITY OF PREDICTION/ROLLUP

We will discuss complexity in terms of the dimension of the largest maintained irreducible Fourier coefficient matrix, which we will denote by d_{max} (see Table 3 for irreducible dimensions). If we maintain 2^{nd} order marginals, for example, then $d_{max} = O(n^2)$, and if we maintain 3^{rd} order marginals, then $d_{max} = O(n^3)$.

Performing a single prediction/rollup step in the Fourier domain involves performing a single matrix multiplication for each irreducible and thus requires $O(d_{max}^3)$ time using the naive multiplication algorithm.

σ	$P^{(0)}$	$Q^{(1)}$	$P^{(1)}$	$Q^{(2)}$	$P^{(2)}$
ε	1	3/4	3/4	3/4	9/16
$(1,2)$	0	1/4	1/4	0	3/16
$(2,3)$	0	0	0	0	0
$(1,3)$	0	0	0	1/4	3/16
$(1,2,3)$	0	0	0	0	1/16
$(1,3,2)$	0	0	0	0	0

Table 4: Primal domain prediction/rollup example.

	$\widehat{P}^{(0)}$	$\widehat{Q}^{(1)}$	$\widehat{P}^{(1)}$	$\widehat{Q}^{(2)}$	$\widehat{P}^{(2)}$
$\rho_{(3)}$	1	1	1	1	1
$\rho_{(2,1)}$	$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 1/2 & 0 \\ 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 1/2 & 0 \\ 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 7/8 & -\sqrt{3}/8 \\ -\sqrt{3}/8 & 5/8 \end{bmatrix}$	$\begin{bmatrix} 7/16 & -\sqrt{3}/8 \\ -\sqrt{3}/16 & 5/8 \end{bmatrix}$
$\rho_{(1,1,1)}$	1	1/2	1/2	1/2	1/4

Table 5: Fourier domain prediction/rollup example.

In certain situations, faster updates can be achieved. For example, in the pairwise mixing model of Example 8, the Fourier transform of Q distribution takes the form: $\hat{Q}_{\rho_\lambda} = \alpha I_{d_\lambda} + \beta \rho_\lambda(i,j)$, where I_{d_λ} is the $d_\lambda \times d_\lambda$ identity matrix (see also Section 8). As it turns out, the matrix $\rho_\lambda(i,j)$ can be factored into a product of $O(n)$ sparse matrices each with at most $O(d_\lambda)$ nonzero entries. To see why, recall the elementary fact that the transposition (i,j) factors into a sequence of $O(n)$ adjacent transpositions:

$$(i,j) = (i,i+1)(i+1,i+2)\cdots(j-1,j)(j-2,j-1)\cdot(i+1,i+2)(i,i+1).$$

If we use the Gel'fand-Tsetlin basis adapted to the subgroup chain $S_1 \subset \cdots S_n$ (see Appendix B), then we also know that the irreducible representation matrices evaluated at adjacent transpositions are sparse with no more than $O(d_{max}^2)$ nonzero entries. Thus by carefully exploiting sparsity during the prediction/rollup algorithm, one can achieve an $O(nd_{max}^2)$ update, which is faster than $O(d_{max}^3)$ as long as one uses more than first-order terms.

6.1.2 LIMITATIONS OF RANDOM WALK MODELS

While the random walk assumption captures a rather general family of transition models, there do exist certain models which cannot be written as a random walk on a group. In particular, one limitation is that the prediction/rollup update for a random walk model can only increase the entropy of the distribution. As with Kalman filters, localization is thus impossible without making observations.[9] Shin et al. (2005) show that the entropy must increase for a certain kind of random walk on S_n (where π could be either the identity or the transposition (i,j)), but in fact, the result is easily generalized for any random walk mixing model and for any finite group.

Proposition 9

$$H\left[P^{(t+1)}(\sigma^{(t+1)})\right] \geq \max\left\{H\left[Q^{(t)}(\tau^{(t)})\right], H\left[P^{(t)}(\sigma^{(t)})\right]\right\},$$

where $H[P(\sigma)]$ denotes the statistical entropy functional, $H[P(\sigma)] = -\sum_{\sigma\in G} P(\sigma)\log P(\sigma)$.

Proof We have:

$$\begin{aligned} P^{(t+1)}(\sigma^{(t+1)}) &= \left[Q^{(t)} * P^{(t)}\right](\sigma^{(t+1)}) \\ &= \sum_{\sigma^{(t)}} Q(\sigma^{(t+1)}\cdot(\sigma^{(t)})^{-1})P^{(t)}(\sigma^{(t)}) \end{aligned}$$

Applying the Jensen Inequality to the entropy function (which is concave) yields:

$$\begin{aligned} H\left[P^{(t+1)}(\sigma^{(t+1)})\right] &\geq \sum_{\sigma^{(t)}} P^{(t)}(\sigma^{(t)})H\left[Q^{(t)}(\sigma\cdot(\sigma^{(t)})^{-1})\right], && \text{(Jensen's inequality)} \\ &= \sum_{\sigma^{(t)}} P^{(t)}(\sigma^{(t)})H\left[Q^{(t)}(\sigma)\right], && \text{(translation invariance of entropy)} \\ &= H\left[Q^{(t)}(\sigma)\right], && \text{(since } \textstyle\sum_{\sigma^{(t)}} P^{(t)}(\sigma^{(t)}) = 1). \end{aligned}$$

The proof that $H\left[P^{(t+1)}(\sigma^{(t+1)})\right] \geq H\left[P^{(t)}(\sigma^{(t)})\right]$ is similar with the exception that we must rewrite the convolution so that the sum ranges over $\tau^{(t)}$.

$$\begin{aligned} P^{(t+1)}(\sigma^{(t+1)}) &= \left[Q^{(t)} * P^{(t)}\right](\sigma^{(t+1)}), \\ &= \sum_{\tau^{(t)}} Q^{(t)}(\tau^{(t)})P^{(t)}((\tau^{(t)})^{-1}\cdot\sigma^{(t+1)}). \end{aligned}$$

■

Example 9 *This example is based on one from Diaconis (1988). Consider a deck of cards numbered $\{1,\ldots,n\}$. Choose a random permutation of cards by first picking two cards independently, and swapping (a card might be swapped with itself), yielding the following probability distribution over S_n:*

$$Q(\pi) = \begin{cases} \frac{1}{n} & \text{if } \pi = \varepsilon \\ \frac{2}{n^2} & \text{if } \pi \text{ is a transposition} \\ 0 & \text{otherwise} \end{cases}. \tag{7}$$

9. In general, if we are not constrained to using linear Gaussian models, it *is* possible to localize with no observations. Consider a robot walking along the unit interval on the real line (which is not a group). If the position of the robot is unknown, one easy localization strategy might be to simply drive the robot to the right, with the knowledge that given ample time, the robot will slam into the ‘wall’, at which point it will have been localized. With random walk based models on groups however, these strategies are impossible—imagine the same robot walking around the unit circle—since, in some sense, the group structure prevents the existence of ‘walls’.

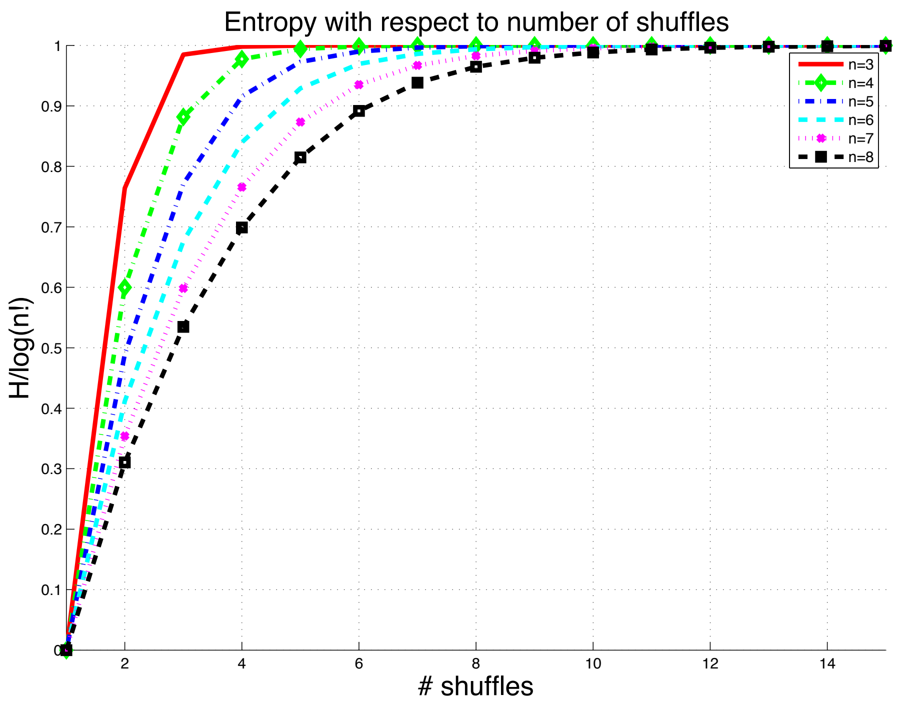


Figure 4: We start with a deck of cards in sorted order, and perform fifteen consecutive shuffles according to the rule given in Equation 7. The plot shows the entropy of the distribution over permutations with respect to the number of shuffles for $n = 3, 4, \ldots, 8$. When $H(P)/\log(n!) = 1$, the distribution has become uniform.

Repeating the above process for generating random permutations π gives a transition model for a hidden Markov model over the symmetric group. We can also see (Figure 4) that the entropy of the deck increases monotonically with each shuffle, and that repeated shuffles with $Q(\pi)$ eventually bring the deck to the uniform distribution.

6.2 Fourier Conditioning

In contrast with the prediction/rollup operation, conditioning can potentially increase the representational complexity. As an example, suppose that we know the following first-order marginal probabilities:

$$P(\text{Alice is at Track 1 or Track 2}) = .9, \text{ and}$$

$$P(\text{Bob is at Track 1 or Track 2}) = .9.$$

If we then make the following first-order observation:

$$P(\text{Cathy is at Track 1 or Track 2}) = 1,$$

then it can be inferred that Alice and Bob cannot *both* occupy Tracks 1 and 2 at the same time, that is,

$$P(\{\text{Alice,Bob}\} \text{ occupy Tracks } \{1{,}2\}) = 0,$$

demonstrating that after conditioning, we are left with knowledge of second-order (unordered) marginals despite the fact that the prior and likelihood functions were only known up to first-order. Intuitively, the example shows that conditioning "smears" information from low-order Fourier coefficients to high-order coefficients, and that one cannot hope for a pointwise operation as was afforded by prediction/rollup. We now show precisely how irreducibles of different complexities "interact" with each other in the Fourier domain during conditioning.

An application of Bayes rule to find a posterior distribution $P(\sigma|z)$ after observing some evidence z requires two steps: a *pointwise product* of likelihood $P(z|\sigma)$ and prior $P(\sigma)$, followed by a normalization step:

$$P(\sigma|z) = \eta \cdot P(z|\sigma) \cdot P(\sigma).$$

For notational convenience, we will refer to the likelihood function as $L(z|\sigma)$ henceforth. We showed earlier that the normalization constant $\eta^{-1} = \sum_\sigma L(z|\sigma) \cdot P(\sigma)$ is given by the Fourier transform of $\widehat{L^{(t)}P^{(t)}}$ at the trivial representation—and therefore the normalization step of conditioning can be implemented by simply dividing each Fourier coefficient by the scalar $\left[\widehat{L^{(t)}P^{(t)}}\right]_{\rho_{(n)}}$.

The pointwise product of two functions f and g, however, is trickier to formulate in the Fourier domain. For functions on the real line, the pointwise product of functions can be implemented by convolving the Fourier coefficients of $\hat{f}$ and $\hat{g}$, and so a natural question is: can we apply a similar operation for functions over general groups? Our answer to this is that there is an analogous (but more complicated) notion of convolution in the Fourier domain of a general finite group. We present a convolution-based conditioning algorithm which we call *Kronecker Conditioning*, which, in contrast to the pointwise nature of the Fourier Domain prediction/rollup step, and much like convolution, smears the information at an irreducible ρ_ν to other irreducibles.

6.2.1 FOURIER TRANSFORM OF THE POINTWISE PRODUCT

Our approach to computing the Fourier transform of the pointwise product in terms of $\hat{f}$ and $\hat{g}$ is to manipulate the function $f(\sigma)g(\sigma)$ so that it can be seen as the result of an inverse Fourier transform (Equation 3). Hence, the goal will be to find matrices R_ν (as a function of $\hat{f}, \hat{g}$) such that for any $\sigma \in G$,

$$f(\sigma) \cdot g(\sigma) = \frac{1}{|G|} \sum_\nu d_{\rho_\nu} \mathrm{Tr}\left(R_\nu^T \cdot \rho_\nu(\sigma)\right), \tag{8}$$

after which we will be able to read off the Fourier transform of the pointwise product as $\left[\widehat{fg}\right]_{\rho_\nu} = R_\nu$.

For any $\sigma \in G$, we can write the pointwise product in terms of $\hat{f}$ and $\hat{g}$ using the inverse Fourier transform:

$$\begin{aligned} f(\sigma) \cdot g(\sigma) &= \left[\frac{1}{|G|} \sum_\lambda d_{\rho_\lambda} \mathrm{Tr}\left(\hat{f}_{\rho_\lambda}^T \cdot \rho_\lambda(\sigma)\right)\right] \cdot \left[\frac{1}{|G|} \sum_\mu d_{\rho_\mu} \mathrm{Tr}\left(\hat{g}_{\rho_\mu}^T \cdot \rho_\mu(\sigma)\right)\right] \\ &= \left(\frac{1}{|G|}\right)^2 \sum_{\lambda,\mu} d_{\rho_\lambda} d_{\rho_\mu} \left[\mathrm{Tr}\left(\hat{f}_{\rho_\lambda}^T \cdot \rho_\lambda(\sigma)\right) \cdot \mathrm{Tr}\left(\hat{g}_{\rho_\mu}^T \cdot \rho_\mu(\sigma)\right)\right]. \end{aligned} \tag{9}$$

1. If A and B are square, $\mathrm{Tr}\,(A \otimes B) = (\mathrm{Tr}A) \cdot (\mathrm{Tr}B)$.
2. $(A \otimes B) \cdot (C \otimes D) = AC \otimes BD$.
3. Let A be an $n \times n$ matrix, and C an invertible $n \times n$ matrix. Then $\mathrm{Tr}A = \mathrm{Tr}\left(C^{-1}AC\right)$.
4. Let A be an $n \times n$ matrix and B_i be matrices of size $m_i \times m_i$ where $\sum_i m_i = n$. Then $\mathrm{Tr}\,(A \cdot (\bigoplus_i B_i)) = \sum_i \mathrm{Tr}\,(A_i \cdot B_i)$, where A_i is the block of A corresponding to block B_i in the matrix $(\bigoplus_i B_i)$.

Table 6: Matrix Identities used in Proposition 10.

Now we want to manipulate this product of traces in the last line to be just one trace (as in Equation 8), by appealing to some properties of the *Kronecker Product*. The Kronecker product of an $n \times n$ matrix $U = (u_{i,j})$ by an $m \times m$ matrix V, is defined to be the $nm \times nm$ matrix

$$U \otimes V = \begin{pmatrix} u_{1,1}V & u_{1,2}V & \dots & u_{1,n}V \\ u_{2,1}V & u_{2,2}V & \dots & u_{2,n}V \\ \vdots & \vdots & \ddots & \vdots \\ u_{n,1}V & u_{n,2}V & \dots & u_{n,n}V \end{pmatrix}.$$

We summarize some important matrix properties in Table 6. The connection to our problem is given by matrix property 1. Applying this to Equation 9, we have:

$$\begin{aligned} \mathrm{Tr}\left(\hat{f}^T_{\rho_\lambda} \cdot \rho_\lambda(\sigma)\right) \cdot \mathrm{Tr}\left(\hat{g}^T_{\rho_\mu} \cdot \rho_\mu(\sigma)\right) &= \mathrm{Tr}\left(\left(\hat{f}^T_{\rho_\lambda} \cdot \rho_\lambda(\sigma)\right) \otimes \left(\hat{g}^T_{\rho_\mu} \cdot \rho_\mu(\sigma)\right)\right) \\ &= \mathrm{Tr}\left(\left(\hat{f}_{\rho_\lambda} \otimes \hat{g}_{\rho_\mu}\right)^T \cdot \left(\rho_\lambda(\sigma) \otimes \rho_\mu(\sigma)\right)\right), \end{aligned}$$

where the last line follows by Property 2. The term on the left, $\hat{f}_{\rho_\lambda} \otimes \hat{g}_{\rho_\mu}$, is a matrix of coefficients. The term on the right, $\rho_\lambda(\sigma) \otimes \rho_\mu(\sigma)$, itself happens to be a representation, called the *Kronecker (or Tensor) Product Representation*. In general, the Kronecker product representation is reducible, and so it can be decomposed into a direct sum of irreducibles. In particular, if ρ_λ and ρ_μ are any two irreducibles of G, there exists a similarity transform $C_{\lambda\mu}$ such that, for any $\sigma \in G$,

$$C^{-1}_{\lambda\mu} \cdot \left[\rho_\lambda \otimes \rho_\mu\right](\sigma) \cdot C_{\lambda\mu} = \bigoplus_{\nu} \bigoplus_{\ell=1}^{z_{\lambda\mu\nu}} \rho_\nu(\sigma). \tag{10}$$

The $\oplus$ symbols here refer to a matrix direct sum as in Equation 2, ν indexes over all irreducible representations of S_n, while ℓ indexes over a number of *copies* of ρ_ν which appear in the decomposition. We index blocks on the right side of this equation by pairs of indices (ν, ℓ). The number of copies of each ρ_ν (for the tensor product pair $\rho_\lambda \otimes \rho_\mu$) is denoted by the integer $z_{\lambda\mu\nu}$, the collection of which, taken over all triples (λ, μ, ν), are commonly referred to as the *Clebsch-Gordan series*. Note that we allow the $z_{\lambda\mu\nu}$ to be zero, in which case ρ_ν does not contribute to the direct sum. The matrices $C_{\lambda\mu}$ are known as the *Clebsch-Gordan coefficients*. The *Kronecker Product Decomposition* problem is that of finding the irreducible components of the Kronecker product representation, and thus to find the Clebsch-Gordan series/coefficients for each pair of irreducible representations (ρ_λ, ρ_μ).

Decomposing the Kronecker product inside Equation 10 using the Clebsch-Gordan series and coefficients yields the desired Fourier transform, which we summarize in the form of a proposition. In the case that f and g are defined over an Abelian group, we will show that the following formulas reduce to the familiar form of convolution.

Proposition 10 *Let $\hat{f}, \hat{g}$ be the Fourier transforms of functions f and g respectively, and for each ordered pair of irreducibles (ρ_λ, ρ_μ), define: $A_{\lambda\mu} \triangleq C_{\lambda\mu}^{-1} \cdot \left(\hat{f}_{\rho_\lambda} \otimes \hat{g}_{\rho_\mu}\right) \cdot C_{\lambda\mu}$. Then the Fourier transform of the pointwise product fg is:*

$$\left[\widehat{fg}\right]_{\rho_\nu} = \frac{1}{d_{\rho_\nu}|G|} \sum_{\lambda\mu} d_{\rho_\lambda} d_{\rho_\mu} \sum_{\ell=1}^{z_{\lambda\mu\nu}} A_{\lambda\mu}^{(\nu,\ell)}, \tag{11}$$

where $A_{\lambda\mu}^{(\nu,\ell)}$ is the block of $A_{\lambda\mu}$ corresponding to the (ν,ℓ) block in $\bigoplus_\nu \bigoplus_{\ell=1}^{z_{\lambda\mu\nu}} \rho_\nu$ from Equation 10.

Proof We use the fact that $C_{\lambda\mu}$ is an orthogonal matrix for all pairs (ρ_λ, ρ_μ), that is, $C_{\lambda\mu}^T \cdot C_{\lambda\mu} = I$.

$$\begin{aligned}
f(\sigma) \cdot g(\sigma) &= \left[\frac{1}{|G|} \sum_\lambda d_{\rho_\lambda} \mathrm{Tr}\left(\hat{f}_{\rho_\lambda}^T \cdot \rho_\lambda(\sigma)\right)\right] \cdot \left[\frac{1}{|G|} \sum_\mu d_{\rho_\mu} \mathrm{Tr}\left(\hat{g}_{\rho_\mu}^T \cdot \rho_\mu(\sigma)\right)\right] \\
&= \left(\frac{1}{|G|}\right)^2 \sum_{\lambda,\mu} d_{\rho_\lambda} d_{\rho_\mu} \left[\mathrm{Tr}\left(\hat{f}_{\rho_\lambda}^T \cdot \rho_\mu(\sigma)\right) \cdot \mathrm{Tr}\left(\hat{g}_{\rho_\mu}^T \cdot \rho_\mu(\sigma)\right)\right] \\
\text{(by Property 1)} &= \left(\frac{1}{|G|}\right)^2 \sum_{\lambda,\mu} d_{\rho_\lambda} d_{\rho_\mu} \left[\mathrm{Tr}\left(\left(\hat{f}_{\rho_\lambda}^T \cdot \rho_\lambda(\sigma)\right) \otimes \left(\hat{g}_{\rho_\mu}^T \cdot \rho_\mu(\sigma)\right)\right)\right] \\
\text{(by Property 2)} &= \left(\frac{1}{|G|}\right)^2 \sum_{\lambda,\mu} d_{\rho_\lambda} d_{\rho_\mu} \mathrm{Tr}\left(\left(\hat{f}_{\rho_\lambda} \otimes \hat{g}_{\rho_\mu}\right)^T \cdot \left(\rho_\lambda(\sigma) \otimes \rho_\mu(\sigma)\right)\right) \\
\text{(by Property 3)} &= \left(\frac{1}{|G|}\right)^2 \sum_{\lambda,\mu} d_{\rho_\lambda} d_{\rho_\mu} \mathrm{Tr}\Big(C_{\lambda\mu}^T \cdot \left(\hat{f}_{\rho_\lambda} \otimes \hat{g}_{\rho_\mu}\right)^T \cdot C_{\lambda\mu} \\
&\qquad \cdot C_{\lambda\mu}^T \cdot \left(\rho_\lambda(\sigma) \otimes \rho_\mu(\sigma)\right) \cdot C_{\lambda\mu}\Big) \\
\text{(by definition of } C_{\lambda\mu} \text{ and } A_{\lambda\mu}) &= \left(\frac{1}{|G|}\right)^2 \sum_{\lambda,\mu} d_{\rho_\lambda} d_{\rho_\mu} \mathrm{Tr}\left(A_{\lambda\mu}^T \cdot \left(\bigoplus_\nu \bigoplus_{\ell=1}^{z_{\lambda\mu\nu}} \rho_\nu(\sigma)\right)\right) \\
\text{(by Property 4)} &= \frac{1}{|G|^2} \sum_{\lambda\mu} d_{\rho_\lambda} d_{\rho_\mu} \sum_\nu d_{\rho_\nu} \sum_{\ell=1}^{z_{\lambda\mu\nu}} \mathrm{Tr}\left(\left(d_{\rho_\nu}^{-1} A_{\lambda\mu}^{(\nu,\ell)}\right)^T \rho_\nu(\sigma)\right) \\
\text{(rearranging terms)} &= \frac{1}{|G|} \sum_\nu d_{\rho_\nu} \mathrm{Tr}\left[\left(\sum_{\lambda\mu} \sum_{\ell=1}^{z_{\lambda\mu\nu}} \frac{d_{\rho_\lambda} d_{\rho_\mu}}{d_{\rho_\nu}|G|} A_{\lambda\mu}^{(\nu,\ell)}\right)^T \rho_\nu(\sigma)\right].
\end{aligned}$$

Recognizing the last expression as an inverse Fourier transform completes the proof. ■

The Clebsch-Gordan series, $z_{\lambda\mu\nu}$, plays an important role in Equation 11, which says that the (ρ_λ, ρ_μ) cross-term contributes to the pointwise product at ρ_ν *only* when $z_{\lambda\mu\nu} > 0$. In the simplest case, we have that

$$z_{(n),\mu,\nu} = \begin{cases} 1 & \text{if } \mu = \nu \\ 0 & \text{otherwise} \end{cases},$$

which is true since $\rho_{(n)}(\sigma) = 1$ for all $\sigma \in S_n$. As another example, it is known that:

$$\rho_{(n-1,1)} \otimes \rho_{(n-1,1)} \equiv \rho_{(n)} \oplus \rho_{(n-1,1)} \oplus \rho_{(n-2,2)} \oplus \rho_{(n-2,1,1)},$$

or equivalently,

$$z_{(n-1,1),(n-1,1),\nu} = \begin{cases} 1 & \text{if } \nu \text{ is one of } (n),(n-1,1),(n-2,2), \text{ or } (n-2,1,1) \\ 0 & \text{otherwise} \end{cases}.$$

So if the Fourier transforms of the likelihood and prior are zero past the first two irreducibles ((n) and $(n-1,1)$), then a single conditioning step results in a Fourier transform which, in general, carries second-order information at $(n-2,2)$ and $(n-2,1,1)$, but is guaranteed to be zero past the first four irreducibles (n), $(n-1,1)$, $(n-2,2)$ and $(n-2,1,1)$.

As far as we know, there are no analytical formulas for finding the entire Clebsch-Gordan series or coefficients, and in practice, acquiring the coefficients requires considerable precomputation. We emphasize however, that as fundamental constants related to the irreducibles of the Symmetric group, they need only be computed *once and for all* (like the digits of π, for example) and can be stored in a table for all future reference. For a detailed discussion of techniques for computing the Clebsch-Gordan series/coefficients, see Appendix D. We have made a set of precomputed coefficients available on our lab website,[10] but we will assume throughout the rest of the paper that both the series and coefficients have been made available as a lookup table.

As a final remark, note that Proposition 10 can be rewritten somewhat more intuitively by absorbing the scalars and submatrices of the Clebsch-Gordan coefficients into projection matrices $P^{(\nu,\ell)}_{\lambda\mu}$.

Proposition 11 *Let $\hat{f}, \hat{g}$ be the Fourier transforms of functions f and g respectively. For each triple of partitions (λ,μ,ν) there exists a positive integer $z_{\lambda,\mu,\nu}$ and projection operators $P^{(\nu,\ell)}_{\lambda\mu}$ for each $\ell \in \{1,2,\ldots,z_{\lambda\mu\nu}\}$ such that the Fourier transform of the pointwise product fg is:*

$$\left[\widehat{fg}\right]_{\rho_\nu} = \sum_{\lambda,\mu}\sum_{\ell=1}^{z_{\lambda\mu\nu}} (P^{(\nu,\ell)}_{\lambda\mu})^T \cdot \left(\hat{f}_{\rho_\lambda} \otimes \hat{g}_{\rho_\mu}\right) \cdot P^{(\nu,\ell)}_{\lambda\mu}. \quad (12)$$

When f and g are functions on an Abelian group G, then it is a well known fact that all irreducible representations are one-dimensional, and so Equation 12 reduces to $\left[\widehat{fg}\right]_{\rho_\nu} = \sum_{\lambda,\mu}\left(\hat{f}_{\rho_\lambda} \cdot \hat{g}_{\rho_\mu}\right)$, where all the tensor products have simply become scalar multiplications and the familiar definition of convolution is recovered.

6.2.2 COMPLEXITY OF CONDITIONING

The complexity of (bandlimited) conditioning (assuming precomputed Clebsch-Gordan series/coefficients) depends on the order of the coefficients maintained for both the prior and the

10. See `http://www.select.cs.cmu.edu/data/index.html`.

observation model. However, it is difficult to state a general complexity bound for arbitrary finite groups due to our limited understanding of the Clebsch-Gordan series. Here we consider conditioning only on the symmetric group of order n with the assumption that the number of irreducibles maintained is very small (and in particular, not allowed to grow with respect to n). Our assumption is realistic in practice since for moderately large n, it is impractical to consider maintaining higher than, say, third-order terms. If we denote the dimension of the largest maintained irreducibles of the prior and likelihood by d_{max}^{prior} and d_{max}^{obs}, respectively, then the complexity of conditioning is dominated by the step that forms a matrix $C^T \cdot (A \otimes B) \cdot C$, where the matrices $A \otimes B$ and C are each $(d_{max}^{prior} \cdot d_{max}^{obs})$-dimensional. Note, however, that since we are only interested in certain blocks of $C^T \cdot (A \otimes B) \cdot C$, the full matrix need not be computed. In particular, the largest extracted block has size d_{max}^{prior}, and so the complexity of conditioning is $O\left((d_{max}^{obs})^2 (d_{max}^{prior})^3\right)$ using the naive matrix multiplication algorithm.

In some situations (see Section 8), the observation model is fully specified by first-order Fourier terms. In such cases, $d_{max}^{obs} = O(n)$ and we can perform conditioning in the Fourier domain in $O(n^2 \cdot (d_{max}^{prior})^3)$ time. If a model is fully specified by second-order terms, for example, then the update requires $O(n^4 \cdot (d_{max}^{prior})^3)$ time.

To speed up conditioning, one can often exploit matrix sparsity in two ways. First, we observe that the Clebsch-Gordan coefficient matrices are often sparse (we cannot yet prove this, see Figure 10.1) and so we can save a conjectured factor of $(d_{max}^{prior} \cdot d_{max}^{obs})$ in practice. Secondly, for certain coset-based observation models (see Section 8), we can show that (under an appropriate relabeling of identities and tracks), the Fourier coefficient matrices of the observation model are sparse (with $O(d_{max}^{obs})$ or sometimes even $O(1)$ nonzero entries for $\hat{L}_\lambda$). For the simplest observations which take the form ("Identity j is at track j"), for example, we can obtain $O((d_{max}^{prior})^3)$ running time (without accounting for the conjectured sparsity of the Clebsch-Gordan coefficients), which matches the time required for the prediction/rollup update. See Appendix B for details.

We now conclude our section on inference with a fully worked example of Kronecker conditioning.

Example 10 *For this example, refer to Table 2 for the representations of S_3. Given functions $f, g : S_3 \to \mathbb{R}$, we will compute the Fourier transform of the pointwise product $f \cdot g$.*

Since there are three irreducibles, there are nine tensor products $\rho_\lambda \otimes \rho_\mu$ to decompose, six of which are trivial either because they are one-dimensional, or involve tensoring against the trivial representation. The nontrivial tensor products to consider are $\rho_{(2,1)} \otimes \rho_{(1,1,1)}$, $\rho_{(1,1,1)} \otimes \rho_{(2,1)}$ and $\rho_{(2,1)} \otimes \rho_{(2,1)}$. The Clebsch-Gordan series for the nontrivial tensor products are:

	$z_{(2,1),(1,1,1),\nu}$	$z_{(1,1,1),(2,1),\nu}$	$z_{(2,1),(2,1),\nu}$
$\nu = (3)$	*0*	*0*	*1*
$\nu = (2,1)$	*1*	*1*	*1*
$\nu = (1,1,1)$	*0*	*0*	*1*

The Clebsch-Gordan coefficients for the nontrivial tensor products are given by the following orthogonal matrices:

$$C_{(2,1)\otimes(1,1,1)} = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}, \quad C_{(1,1,1)\otimes(2,1)} = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}, \quad C_{(2,1)\otimes(2,1)} = \frac{\sqrt{2}}{2} \begin{bmatrix} 1 & 0 & -1 & 0 \\ 0 & -1 & 0 & 1 \\ 0 & -1 & 0 & -1 \\ 1 & 0 & 1 & 0 \end{bmatrix}.$$

As in Proposition 10, define:

$$A_{(2,1)\otimes(1,1,1)} = C^T_{(2,1)\otimes(1,1,1)}\left(\hat{f}_{(2,1)}\otimes\hat{g}_{(1,1,1)}\right)C_{(2,1)\otimes(1,1,1)}, \tag{13}$$

$$A_{(1,1,1)\otimes(2,1)} = C^T_{(1,1,1)\otimes(2,1)}\left(\hat{f}_{(1,1,1)}\otimes\hat{g}_{(2,1)}\right)C_{(1,1,1)\otimes(2,1)}, \tag{14}$$

$$A_{(2,1)\otimes(2,1)} = C^T_{(2,1)\otimes(2,1)}\left(\hat{f}_{(2,1)}\otimes\hat{g}_{(2,1)}\right)C_{(2,1)\otimes(2,1)}, \tag{15}$$

Then Proposition 10 gives the following formulas:

$$\widehat{f\cdot g}_{\rho_{(3)}} = \frac{1}{3!}\cdot\left[\hat{f}_{\rho_{(3)}}\cdot\hat{g}_{\rho_{(3)}} + \hat{f}_{\rho_{(1,1,1)}}\cdot\hat{g}_{\rho_{(1,1,1)}} + 4\cdot\left[A_{(2,1)\otimes(2,1)}\right]_{1,1}\right], \tag{16}$$

$$\widehat{f\cdot g}_{\rho_{(2,1)}} = \frac{1}{3!}\cdot\Big[\hat{f}_{\rho_{(2,1)}}\cdot\hat{g}_{\rho_{(3)}} + \hat{f}_{\rho_{(3)}}\cdot\hat{g}_{\rho_{(2,1)}} + A_{(1,1,1)\otimes(2,1)} + A_{(2,1)\otimes(1,1,1)} + 2\cdot\left[A_{(2,1)\otimes(2,1)}\right]_{2:3,2:3}\Big], \tag{17}$$

$$\widehat{f\cdot g}_{\rho_{(1,1,1)}} = \frac{1}{3!}\cdot\left[\hat{f}_{\rho_{(3)}}\cdot\hat{g}_{\rho_{(1,1,1)}} + \hat{f}_{\rho_{(1,1,1)}}\cdot\hat{g}_{\rho_{(3)}} + 4\cdot\left[A_{(2,1)\otimes(2,1)}\right]_{4,4}\right], \tag{18}$$

where the notation $[A]_{a:b,c:d}$ denotes the block of entries in A between rows a and b, and between columns c and d (inclusive).

Using the above formulas, we can continue on Example 8 and compute the last update step in our identity management problem (Figure 2). At the final time step, we observe that Bob is at track 1 with 100% certainty. Our likelihood function is therefore nonzero only for the permutations which map Bob (the second identity) to the first track:

$$L(\sigma) \propto \begin{cases} 1 & \textit{if } \sigma = (1,2) \textit{ or } (1,3,2) \\ 0 & \textit{otherwise} \end{cases}.$$

The Fourier transform of the likelihood function is:

$$\widehat{L}_{\rho_{(3)}} = 2, \quad \widehat{L}_{\rho_{(2,1)}} = \begin{bmatrix} -3/2 & \sqrt{3}/2 \\ -\sqrt{3}/2 & 1/2 \end{bmatrix}, \quad \widehat{L}_{\rho_{(1,1,1)}} = 0. \tag{19}$$

Plugging the Fourier transforms of the prior distribution ($\widehat{P}^{(2)}$ from Table 5) and likelihood (Equation 19) into Equations 13, 14, 15, we have:

$$A_{(2,1)\otimes(1,1,1)} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}, \; A_{(1,1,1)\otimes(2,1)} = \frac{1}{8}\begin{bmatrix} 1 & \sqrt{3} \\ -\sqrt{3} & -3 \end{bmatrix}, \; A_{(2,1)\otimes(2,1)} = \frac{1}{32}\begin{bmatrix} -7 & -\sqrt{3} & 11 & 5\sqrt{3} \\ -2\sqrt{3} & -10 & -6\sqrt{3} & -14 \\ 20 & 22\sqrt{3} & -4 & 4\sqrt{3} \\ -11\sqrt{3} & -23 & -\sqrt{3} & -13 \end{bmatrix}$$

To invoke Bayes rule in the Fourier domain, we perform a pointwise product using Equations 16, 17, 18, and normalize by dividing by the trivial coefficient, which yields the Fourier transform of the posterior distribution as:

$$\left[\widehat{P(\sigma|z)}\right]_{\rho_{(3)}} = 1, \quad \left[\widehat{P(\sigma|z)}\right]_{\rho_{(2,1)}} = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}, \quad \left[\widehat{P(\sigma|z)}\right]_{\rho_{(1,1,1)}} = -1. \tag{20}$$

Finally, we can see that the result is correct by recognizing that the Fourier transform of the posterior (Equation 20) corresponds exactly to the distribution which is 1 at $\sigma = (1,2)$ and 0 everywhere else. Bob is therefore at Track 1, Alice at Track 2 and Cathy at Track 3.

σ	ε	$(1,2)$	$(2,3)$	$(1,3)$	$(1,2,3)$	$(1,3,2)$
$P(\sigma)$	0	1	0	0	0	0

Algorithm 1: Pseudocode for the Fourier Prediction/Rollup Algorithm.

```
PREDICTIONROLLUP
input : $\hat{Q}^{(t)}_{\rho_\lambda}$ and $\hat{P}^{(t)}_{\rho_\lambda}$, $\rho_\lambda \in \Lambda$
output: $\hat{P}^{(t+1)}_{\rho_\lambda}$, $\rho_\lambda \in \Lambda$
1 foreach $\rho_\lambda \in \Lambda$ do $\hat{P}^{(t+1)}_{\rho_\lambda} \leftarrow \hat{Q}^{(t)}_{\rho_\lambda} \cdot \hat{P}^{(t)}_{\rho_\lambda}$ ;
```

Algorithm 2: Pseudocode for the Kronecker Conditioning Algorithm.

```
KRONECKERCONDITIONING
input : Fourier coefficients of the likelihood function, $\hat{L}_{\rho_\lambda}$, $\rho_\lambda \in \Lambda_L$, and Fourier coefficients
        of the prior distribution, $\hat{P}_{\rho_\mu}$, $\rho_\mu \in \Lambda_P$
output: Fourier coefficients of the posterior distribution, $\widehat{LP}_{\rho_\nu}$, $\rho_\nu \in \Lambda_P$
1  foreach $\rho_\nu \in \Lambda_P$ do $\widehat{LP}_{\rho_\nu} \leftarrow \mathbf{0}$ //Initialize Posterior
   //Pointwise Product
2  foreach $\rho_\lambda \in \Lambda_L$ do
3      foreach $\rho_\mu \in \Lambda_P$ do
4          $z \leftarrow CGseries(\rho_\lambda, \rho_\mu)$ ;
5          $C_{\lambda\mu} \leftarrow CGcoefficients(\rho_\lambda, \rho_\mu)$ ; $A_{\lambda\mu} \leftarrow C^T_{\lambda\mu} \cdot \left(\hat{L}_{\rho_\lambda} \otimes \hat{P}_{\rho_\mu}\right) \cdot C_{\lambda\mu}$ ;
6          for $\rho_\nu \in \Lambda_P$ such that $z_{\lambda\mu\nu} \neq 0$ do
7              for $\ell = 1$ to $z_{\lambda\mu\nu}$ do
8                  $\left[\widehat{L^{(t)}P^{(t)}}\right]_{\rho_\nu} \leftarrow \left[\widehat{L^{(t)}P^{(t)}}\right]_{\rho_\nu} + \frac{d_{\rho_\lambda} d_{\rho_\mu}}{d_{\rho_\nu} n!} A^{(\nu,\ell)}_{\lambda\mu}$; //$A^{(\nu,\ell)}_{\lambda\mu}$ is the $(\nu,\ell)$ block of $A_{\lambda\mu}$
9  $\eta \leftarrow \left[\widehat{L^{(t)}P^{(t)}}\right]^{-1}_{\rho_{(n)}}$ ;
10 foreach $\rho_\nu \in \Lambda$ do $\left[\widehat{L^{(t)}P^{(t)}}\right]_{\rho_\nu} \leftarrow \eta \left[\widehat{L^{(t)}P^{(t)}}\right]_{\rho_\nu}$ //Normalization
```

7. Approximate Inference by Bandlimiting

We now consider the consequences of performing inference using the Fourier transform at a reduced set of coefficients. Important issues include understanding how error can be introduced into the system, and when our algorithms are expected to perform well as an approximation. Specifically, we fix a bandlimit λ^{MIN} and maintain the Fourier transform of P only at irreducibles which are at λ^{MIN} or above in the dominance ordering:

$$\Lambda = \{\rho_\lambda \, : \, \lambda \trianglerighteq \lambda^{MIN}\}.$$

For example, when $\lambda^{MIN} = (n-2,1,1)$, Λ is the set $\left\{\rho_{(n)}, \rho_{(n-1,1)}, \rho_{(n-2,2)}, \text{ and } \rho_{(n-2,1,1)}\right\}$, which corresponds to maintaining second-order (ordered) marginal probabilities of the form $P(\sigma((i,j)) = (k,\ell))$. During inference, we follow the procedure outlined in the previous section but discard the higher order terms which can be introduced during the conditioning step. Pseudocode for bandlimited prediction/rollup and Kronecker conditioning is given in Algorithms 1 and 2. We note that it is not necessary to maintain the same number of irreducibles for both prior and likelihood during the conditioning step. The first question to ask is: when should one expect a bandlimited approximation to be close to $P(\sigma)$ as a function? Qualitatively, if a distribution is relatively smooth, then most of its

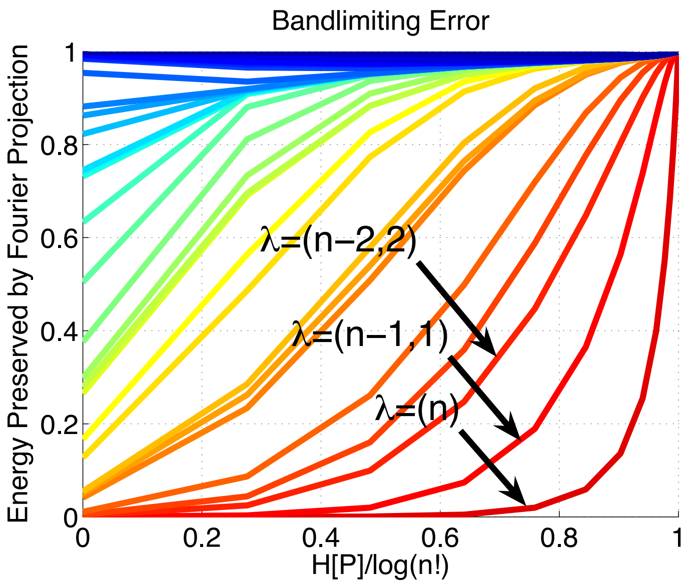


Figure 5: In general, smoother distributions are well approximated by low-order Fourier projections. In this graph, we show the approximation quality of the Fourier projections on distributions with different entropies, starting from sharply peaked delta distributions on the left side of the graph, which get iteratively smoothed until they becomes the maximum entropy uniform distribution on the right side. On the y-axis, we measure how much *energy* is preserved in the bandlimited approximation, which we define to be $\frac{|P'|^2}{|P|^2}$, where P' is the bandlimited approximation to P. Each line represents the approximation quality using a fixed number of Fourier coefficients. At one extreme, we achieve perfect signal reconstruction by using all Fourier coefficients, and at the other, we perform poorly on "spiky" distributions, but well on high-entropy distributions, by storing a single Fourier coefficient.

energy is stored in the low-order Fourier coefficients. However, in a phenomenon quite reminiscent of the Heisenberg uncertainty principle from quantum mechanics, it is exactly when the distribution is sharply concentrated at a small subset of permutations, that the Fourier projection is unable to faithfully approximate the distribution. We illustrate this uncertainty effect in Figure 5 by plotting the accuracy of a bandlimited distribution against the entropy of a distribution.

Even though the bandlimited distribution is sometimes a poor approximation to the true distribution, the marginals maintained by our algorithm are often sufficiently accurate. And so instead of considering the approximation accuracy of the bandlimited Fourier transform to the true joint distribution, we consider the accuracy only at the marginals which are maintained by our method.

7.1 Sources of Error During Inference

We now analyze the errors incurred during our inference procedures with respect to the accuracy at maintained marginals. It is immediate that the Fourier domain prediction/rollup operation is *exact* due to its pointwise nature in the Fourier domain. For example, if we have the second order

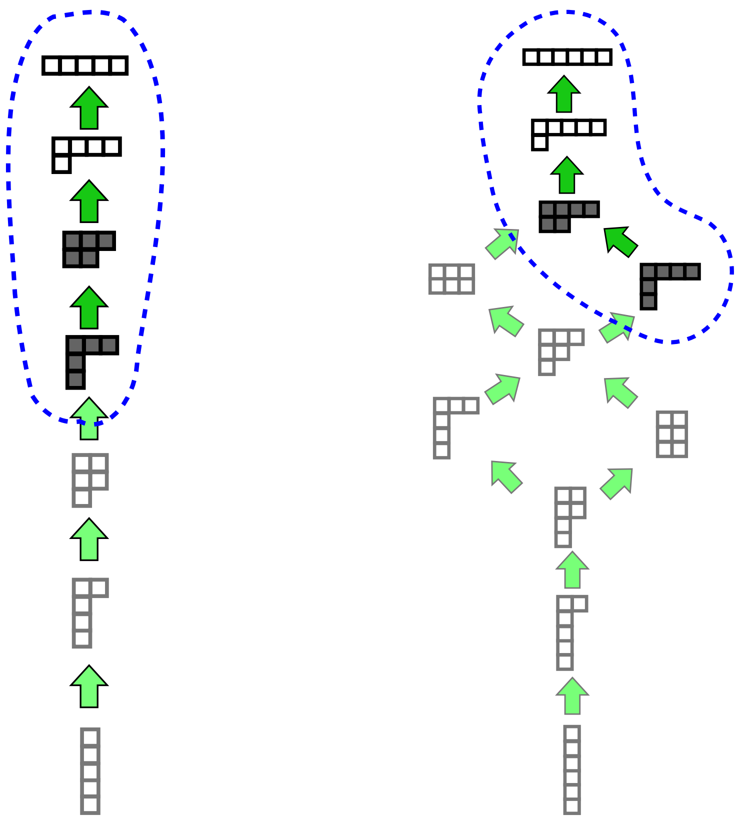

(a) $n = 5$ (b) $n = 6$

Figure 6: We show the dominance ordering for partitions of $n = 5$ and $n = 6$ again. By setting $\lambda^{MIN} = (3,1,1)$ and $(4,1,1)$ respectively, we keep the irreducibles corresponding to the partitions in the dotted regions. If we call Kronecker Conditioning with a first-order observation model, then according to Theorem 12, we can expect to incur some error at the Fourier coefficients corresponding to $(3,1,1)$ and $(3,2)$ for $n = 5$, and $(4,1,1)$ and $(4,2)$ for $n = 6$ (shown as shaded tableaux), but to be exact at first-order coefficients.

marginals at time $t = 0$, then we can find the exact second order marginals at all $t > 0$ if we only perform prediction/rollup operations. Instead, the errors in inference are only committed by Kronecker conditioning, where they are implicitly introduced at coefficients outside of Λ (by effectively setting the coefficients of the prior and likelihood at irreducibles outside of Λ to be zero), then propagated inside to the irreducibles of Λ.

In practice, we observe that the errors introduced at the low-order irreducibles during inference are small if the prior and likelihood are sufficiently diffuse, which makes sense since the high-frequency Fourier coefficients are small in such cases. We can sometimes show that the update is *exact* at low order irreducibles if we maintain *enough* coefficients.

Theorem 12 *If* $\lambda^{MIN} = (n-p, \lambda_2, \ldots)$*, and the Kronecker conditioning algorithm is called with a likelihood function whose Fourier coefficients are nonzero only at* ρ_μ *when* $\mu \trianglerighteq (n-q, \mu_2, \ldots)$*, then* *the approximate Fourier coefficients of the posterior distribution are exact at the set of irreducibles:*

$$\Lambda_{EXACT} = \{\rho_\lambda : \lambda \trianglerighteq (n - |p-q|, \dots)\}.$$

Proof See Appendix D. ■

For example, if we call Kronecker conditioning by passing in third-order terms of the prior and first-order terms of the likelihood, then all first and second-order (unordered and ordered) marginal probabilities of the posterior distribution can be reconstructed without error.

7.2 Projecting to the Marginal Polytope

Despite the encouraging result of Theorem 12, the fact remains that consecutive conditioning steps can propagate errors to all levels of the bandlimited Fourier transform, and in many circumstances, result in a Fourier transform whose "marginal probabilities" correspond to no consistent joint distribution over permutations, and are sometimes negative. To combat this problem, we present a method for projecting to the space of coefficients corresponding to consistent joint distributions (which we will refer to as the *marginal polytope*) during inference.

We begin by discussing the first-order version of the marginal polytope projection problem. Given an $n \times n$ matrix, M, of real numbers, how can we decide whether there exists some probability distribution which has M as its matrix of first-order marginal probabilities? A necessary and sufficient condition, as it turns out, is for M to be *doubly stochastic*. That is, all entries of M must be nonnegative and all rows and columns of M must sum to one (the probability that Alice is at *some track* is 1, and the probability that *some identity* is at Track 3 is 1). The double stochasticity condition comes from the *Birkhoff-von Neumann* theorem (van Lint and Wilson, 2001) which states that a matrix is doubly stochastic *if and only if* it can be written as a convex combination of permutation matrices.

To "renormalize" first-order marginals to be doubly stochastic, some authors (Shin et al., 2003, 2005; Balakrishnan et al., 2004; Helmbold and Warmuth, 2007) have used the *Sinkhorn iteration*, which alternates between normalizing rows and columns independently until convergence is obtained. Convergence is guaranteed under mild conditions and it can be shown that the limit is a nonnegative doubly stochastic matrix which is closest to the original matrix in the sense that the Kullback-Leibler divergence is minimized (Balakrishnan et al., 2004).

There are several problems which cause the Sinkhorn iteration to be an unnatural solution in our setting. First, since the Sinkhorn iteration only works for nonnegative matrices, we would have to first cap entries to lie in the appropriate range, $[0,1]$. More seriously, even though the Sinkhorn iteration would guarantee a doubly stochastic higher order matrix of marginals, there are several natural constraints which are violated when running the Sinkhorn iteration on higher-order marginals. For example, with second-order (ordered) marginals, it seems that we should at least enforce the following symmetry constraint:

$$P(\sigma : \sigma(k,\ell) = (i,j)) = P(\sigma : \sigma(\ell,k) = (j,i)),$$

which says, for example, that the marginal probability that Alice is in Track 1 and Bob is in Track 2 is the same as the marginal probability that Bob is in Track 2 and Alice is in Track 1. Another natural constraint that can be broken is what we refer to as *low-order marginal consistency*. For example, it should always be the case that:

$$P(j) = \sum_i P(i,j) = \sum_k P(j,k).$$

It should be noted that the doubly stochastic requirement is a special case of lower-order marginal consistency—we require that higher-order marginals be consistent on the 0^{th} order marginal.

While compactly describing the constraints of the marginal polytope exactly remains an open problem, we propose a method for projecting onto a *relaxed* form of the marginal polytope which addresses both symmetry and low-order consistency problems by operating directly on irreducible Fourier coefficients instead of on the matrix of marginal probabilities. After each conditioning step, we apply a 'correction' to the approximate posterior $P^{(t)}$ by finding the bandlimited function in the relaxed marginal polytope which is closest to $P^{(t)}$ in an L_2 sense. To perform the projection, we employ the Plancherel Theorem (Diaconis, 1988) which relates the L_2 distance between functions on S_n to a distance metric in the Fourier domain.

Proposition 13 (Plancherel Theorem)

$$\sum_\sigma (f(\sigma)-g(\sigma))^2 \quad = \quad \frac{1}{|G|}\sum_\nu d_{\rho_\nu} Tr\left(\left(\hat{f}_{\rho_\nu}-\hat{g}_{\rho_\nu}\right)^T \cdot \left(\hat{f}_{\rho_\nu}-\hat{g}_{\rho_\nu}\right)\right). \tag{21}$$

To find the closest bandlimited function in the relaxed marginal polytope, we formulate a quadratic program whose objective is to minimize the right side of Equation 21, and whose sum is taken only over the set of maintained irreducibles, Λ, subject to the set of constraints which require all marginal probabilities to be nonnegative. We thus refer to our correction step as *Plancherel Projection*. Our quadratic program can be written as:

$$\begin{aligned}
&\text{minimize}_{\hat{f}^{proj}} && \sum_{\lambda\in\Lambda} d_\lambda \mathrm{Tr}\left[\left(\hat{f}-\hat{f}^{proj}\right)^T_{\rho_\lambda}\left(\hat{f}-\hat{f}^{proj}\right)_{\rho_\lambda}\right] \\
&\text{subject to:} && \left[\hat{f}^{proj}\right]_{(n)} = 1, \\
& && \left[C_{\lambda^{MIN}}\cdot\left(\bigoplus_{\mu\trianglerighteq\lambda^{MIN}}\bigoplus_{\ell=1}^{K_{\lambda^{MIN},\mu}}\hat{f}^{proj}_{\rho_\mu}\right)\cdot C^T_{\lambda^{MIN}}\right]_{ij} \geq 0, \qquad \text{for all } (i,j),
\end{aligned}$$

where $K_{\lambda^{MIN}}$ and $C_{\lambda^{MIN}}$ are the precomputed constants from Equation 6. We remark that even though the projection will produce a Fourier transform corresponding to nonnegative marginals which are consistent with each other, there might not necessarily exist a joint probability distribution on S_n consistent with those marginals except in the special case of first-order marginals.

Example 11 *In Example 10, we ran the Kronecker conditioning algorithm using all of the Fourier coefficients. If only the first-order coefficients are available, however, then the expressions for zeroth and first order terms of the posterior (Equations 16,17) become:*

$$\begin{aligned}
\widehat{f\cdot g}_{\rho_{(3)}} &= \frac{1}{3!}\cdot\left[\hat{f}_{\rho_{(3)}}\cdot\hat{g}_{\rho_{(3)}} + 4\cdot\left[A_{(2,1)\otimes(2,1)}\right]_{1,1}\right], \\
\widehat{f\cdot g}_{\rho_{(2,1)}} &= \frac{1}{3!}\cdot\left[\hat{f}_{\rho_{(2,1)}}\cdot\hat{g}_{\rho_{(3)}} + \hat{f}_{\rho_{(3)}}\cdot\hat{g}_{\rho_{(2,1)}} + 2\cdot\left[A_{(2,1)\otimes(2,1)}\right]_{2:3,2:3}\right],
\end{aligned}$$

Plugging in the same numerical values from Example 10 and normalizing appropriately yields the approximate Fourier coefficients of the posterior:

$$\left[\widehat{P(\sigma|z)}\right]_{\rho_{(3)}} = 1 \qquad \left[\widehat{P(\sigma|z)}\right]_{\rho_{(2,1)}} = \begin{bmatrix} -10/9 & -77/400 \\ 77/400 & 4/3 \end{bmatrix},$$

which correspond to the following first-order marginal probabilities:

$$\hat{P}_{\tau_{(2,1)}} \left[\begin{array}{c|ccc} & A & B & C \\ \hline \textit{Track 1} & 0 & 11/9 & -2/9 \\ \textit{Track 2} & 1 & 0 & 0 \\ \textit{Track 3} & 0 & -2/9 & 11/9 \end{array}\right].$$

In particular, we see that the approximate matrix of 'marginals' contains negative numbers. Applying the Plancherel projection step, we obtain the following marginals:

$$\hat{P}_{\tau_{(2,1)}} \left[\begin{array}{c|ccc} & A & B & C \\ \hline \textit{Track 1} & 0 & 1 & 0 \\ \textit{Track 2} & 1 & 0 & 0 \\ \textit{Track 3} & 0 & 0 & 1 \end{array}\right],$$

which happen to be exactly the true posterior marginals. It should be noted however, that rounding the 'marginals' to be in the appropriate range would have worked in this particular example as well.

8. Probabilistic Models of Mixing and Observations

While the algorithms presented in the previous sections are general in the sense that they work on all mixing and observation models, it is not always obvious how to compute the Fourier transform of a given model. In this section, we discuss a collection of useful models for which we *can* efficiently compute low-order Fourier coefficients or even provide a closed-form expression. See Table 7 for a summary of the various models covered in this section.

We consider both *mixing* and *observation* models. In multiobject tracking, a mixing model might account for the fact that two tracks may have swapped identities with some probability. Or in card shuffling, a mixing model might reflect that a card has been inserted somewhere into the deck. In multiobject tracking, an observation model might tell us that Alice is at some track with probability one. Or it might reflect the fact that some subset of identities occupies some subset of tracks with no order information, as in the case of the *bluetooth model*. In ranking applications, an observation model might, for example, reflect that some object is ranked higher than, or preferred over some other object.

This section is divided into three parts, each describing a different approach to computing the Fourier coefficients of a model, with some being simpler or more efficient to implement in certain situations than others. In *direct constructions*, we naively apply the definition of the Fourier transform to obtain the Fourier coefficients of some model. In *marginal based constructions*, we first compute the low-order 'marginals' of some probabilistic model, then project the result onto the irreducible Fourier basis. Finally, in *coset-based constructions*, we introduce a family of 'atomic' indicator functions of subgroups of the form $S_k \subset S_n$ which are then combined using

Mixing Model	Example Semantics	Relevant Subgroup
Pairwise mixing	Identity confusion at tracks 1 and 2	S_2
k-subset mixing	Identity confusion at tracks in $\{1,2,4,6\}$	S_k
Insertion mixing	Insert top card somewhere in the deck	n/a

Observation Model	Example Semantics	Relevant Subgroup
Single track observation	Alice is at Track 1	S_{n-1}
Multitrack observation	Alice is at Track 1, Bob is at Track 2, etc.	S_{n-k}
Bluetooth observation	The girls occupy tracks $\{1,2,6,8\}$	$S_k \times S_{n-k}$
Pairwise ranking observation	Apples are better than oranges	S_{n-2}

Table 7: Several useful types of mixing and observation models are summarized in the above table. In many of these cases, computing the appropriate Fourier transform reduces to computing the Fourier transform of the indicator function of some related subgroup of S_n, and so we also mention the relevant subgroup in the second column. In the third column we provide an example illustrating the semantics of each model.

scale/shift/convolution operations to form more complex models. As we discuss in Section 11, there also remains the open possibility of learning models *directly* in the Fourier domain. For the sake of succinctness, many of the results in this section will be stated without proof.

8.1 Direct Construction

In some applications we are fortunate enough to have a model that can be "directly" transformed efficiently using the definition of the Fourier transform (Definition 3). We provide two examples.

8.1.1 Pairwise Mixing

The simplest mixing model for identity management assumes that with probability p, nothing happens, and that with probability $(1-p)$, the identities for tracks i and j are swapped. The probability distribution for the *pairwise mixing model* is therefore:

$$Q_{ij}(\pi) = \begin{cases} p & \text{if } \pi = \varepsilon \\ 1-p & \text{if } \pi = (i,j) \\ 0 & \text{otherwise} \end{cases} . \tag{22}$$

Since Q_{ij} is such a sparse distribution (in the sense that $Q_{ij}(\pi) = 0$ for most π), it is possible to directly compute $\widehat{Q}_{ij}$ using Definition 3:

$$\left[\widehat{Q}_{ij}\right]_{\rho_\lambda} = pI + (1-p)\rho_\lambda((i,j)),$$

where I refers to the $d_\lambda \times d_\lambda$ identity matrix (since any representation must map the identity element ε to an identity matrix), and $\rho_\lambda((i,j))$ is the irreducible representation matrix ρ_λ evaluated at the transposition (i,j) (which can be computed using the algorithms from Appendix C).

8.1.2 INSERTION MIXING

As another example, we can consider the *insertion mixing model* (also called the *top-in shuffle* Diaconis 1988) in which we take the top card in some deck of n cards, and with uniform probability, insert it *somewhere* in the deck, preserving all other original relative orderings. Insertions can be useful in ranking applications where we might wish to add a new item into consideration without disturbing the marginal probabilities over relative rankings of existing items. The distribution for the insertion mixing model is given by:

$$Q^{insertion}(\pi) = \begin{cases} \frac{1}{n} & \text{if } \pi \text{ is a cycle of the form } (j, j-1, \ldots, 1) \text{ for some } j \in \{1, \ldots, n\} \\ 0 & \text{otherwise} \end{cases}.$$

Since the insertion mixing model is supported on n permutations, it is again simple to directly construct the Fourier transform from the definition. We have:

$$\widehat{Q}^{insertion}_{\rho_\lambda} = \frac{1}{n} \sum_{j=1}^{n} \rho_\lambda(j, j-1, \ldots, 1).$$

8.2 Marginal Based Construction

In *marginal based constructions*, we first compute the low-order 'marginals'[11] of some probabilistic model, then project the result onto the irreducible Fourier basis. Thus given a function $f : S_n \to \mathbb{R}$, we compute, for example, the first-order marginals $\hat{f}_{\tau_{(n-1,1)}}$, and conjugate by an intertwining operator (Equation 6) to obtain the Fourier coefficients at (n) and $(n-1,1)$. Sometimes when the Fourier transform of f is provably non-zero *only* at low-order terms, a marginal based construction might be the easiest method to obtain Fourier coefficients.

8.2.1 COLOR HISTOGRAM OBSERVATION

The simplest model assumes that we can get observations of the form: 'track ℓ is color k' (which is essentially the model considered by Kondor et al. 2007). The probability of seeing color k at track ℓ given data association σ is

$$L(\sigma) = P(z_\ell = k|\sigma) = \alpha_{\sigma^{-1}(\ell),k},$$

where $\sum_k \alpha_{\sigma^{-1}(\ell),k} = 1$. For each identity, the likelihood $L(\sigma) = P(z_\ell = k|\sigma)$ depends, for example, on a histogram over all possible colors. If the number of possible colors is K, then the likelihood model can be specified by an $n \times K$ matrix of probabilities. For example,

$$\alpha_{\sigma(\ell),k} = \left[\begin{array}{c|cccc} & k = \text{Red} & k = \text{Orange} & k = \text{Yellow} & k = \text{Green} \\ \hline \sigma(\text{Alice}) = \ell & 1/2 & 1/4 & 1/4 & 0 \\ \sigma(\text{Bob}) = \ell & 1/4 & 0 & 0 & 3/4 \\ \sigma(\text{Cathy}) = \ell & 0 & 1/2 & 1/2 & 0 \end{array} \right]. \quad (23)$$

Since the observation model only depends on a single identity, the first-order terms of the Fourier transform suffice to describe the likelihood exactly. To compute the first-order Fourier coefficients

11. The word 'marginals' is technically appropriate only when the function in question is a legal probability distribution (as opposed to likelihood functions, for example), however we use it to refer to similar summary statistics for general functions.

at irreducibles, we proceed by computing the first-order Fourier coefficients at the first-order permutation representation (the first-order “marginals”), then transforming to irreducible coefficients. The Fourier transform of the likelihood at the first-order permutation representation is given by:

$$\left[\widehat{L}_{\tau_{(n-1,1)}}\right]_{ij} = \sum_{\{\sigma:\sigma(j)=i\}} P(z_\ell = k|\sigma) = \sum_{\{\sigma:\sigma(j)=i\}} \alpha_{\sigma^{-1}(\ell),k}.$$

To compute the ij-term, there are two cases to consider.

1. If $i = \ell$ (that is, if Track i is the same as the track that was observed), then the coefficient $\widehat{L}_{ij}$ is proportional to the probability that Identity j is color k.

$$\widehat{L}_{ij} = \sum_{\{\sigma:\sigma(j)=i\}} \alpha_{j,k} = (n-1)! \cdot \alpha_{j,k}. \tag{24}$$

2. If, on the other hand, $i \neq \ell$ (Track i is not the observed track)), then the coefficient $\hat{L}_{ij}$ is proportional to the sum over

$$\widehat{L}_{ij} = \sum_{\{\sigma:\sigma(j)=i\}} \alpha_{\sigma^{-1}(\ell),k} = \sum_{m\neq j} \sum_{\{\sigma:\sigma(j)=i \text{ and } \sigma(m)=\ell\}} \alpha_{\sigma^{-1}(\ell),k} = \sum_{m\neq j} (n-2)! \cdot \alpha_{m,k}. \tag{25}$$

Example 12 *We will compute the first-order marginals of the likelihood function on S_3 which arises from observing a "Red blob at Track 1". Plugging the values from the “Red” column of the α matrix (Equation 23) into Equation 24 and 25 yields the following matrix of first-order coefficients (at the $\tau_{(n-1,1)}$ permutation representation):*

$$\left[\hat{L}_{(n-1,1)}\right]_{ij} = \left[\begin{array}{c|ccc} & \textit{Track 1} & \textit{Track 2} & \textit{Track 3} \\ \hline \textit{Alice} & 1/4 & 1/2 & 3/4 \\ \textit{Bob} & 1/4 & 1/2 & 3/4 \\ \textit{Cathy} & 1 & 1/2 & 0 \end{array}\right].$$

The corresponding coefficients at the irreducible representations are:

$$\hat{L}_{(3)} = 1.5, \qquad \hat{L}_{(2,1)} = \begin{bmatrix} 0 & 0 \\ -\sqrt{3}/4 & -3/4 \end{bmatrix}, \qquad \hat{L}_{(1,1,1)} = 0.$$

8.2.2 UNORDERED SUBSET (BLUETOOTH) OBSERVATION

We sometimes receive measurements in the form of unordered lists. For example, the *bluetooth model* is the likelihood function that arises if tracks $\{1,\ldots,k\}$ are within range of a bluetooth detector and we receive a measurement that identities $\{1,\ldots,k\}$ are in range. In sports, we might observe that the first k tracks belong to the red team and that the last $n-k$ tracks belong to the blue team. And finally, in *approval voting*, one specifies a subset of approved candidates rather than, for example, picking a single favorite.

We consider two options for bluetooth-type situations. In the first option, we allow for some error-tolerance by setting the likelihood to be proportional to the number of tracks that are correctly returned in the measurement:

$$P^{bluetooth}(z_{\{t_1,\ldots,t_k\}} = \{i_1,\ldots,i_k\}|\sigma) \propto |\{t_1,\ldots,t_k\} \cap \sigma(\{i_1,\ldots,i_k\})| + U(\sigma), \tag{26}$$

where $U(\sigma)$ is a constant function on S_n allowing for noisy observations. Our first bluetooth model can be expressed using only first order terms (intuitively because each track makes a linear contribution) and thus $\hat{P}_\lambda^{bluetooth}$ is nonzero only at the first two partitions $\lambda = (n), (n-1,1)$. For simplicity, we consider the Fourier transform of the function: $f(\sigma) = |\sigma(\{1,\dots,k\}) \cap \{1,\dots,k\}|$. The first-order 'marginals' of f are covered in the following four cases:

- *($j \leq k$ and $i \leq k$)*: $L_{ij} = \sum_{\sigma:\sigma(j)=i} f(\sigma) = (k-1)^2(n-2)! + (n-1)!$
- *($j \leq k$ and $i > k$)*: $L_{ij} = \sum_{\sigma:\sigma(j)=i} f(\sigma) = k(k-1)(n-2)!$
- *($j > k$ and $i \leq k$)*: $L_{ij} = \sum_{\sigma:\sigma(j)=i} f(\sigma) = k(k-1)(n-2)!$
- *($j > k$ and $i > k$)*: $L_{ij} = \sum_{\sigma:\sigma(j)=i} f(\sigma) = k^2(n-2)!$

We discuss the second bluetooth-type model after discussing coset based constructions.

8.3 Coset-Based Construction

Most of the time, realistic models are not supported on only a handful of permutations. The approach we take now is to use a collection of 'primitive' functions to form more interesting models via scale/shift/convolution operations. In particular, we will make use of indicator functions of subsets of the form $S_{X,Y} \subset S_n$, where $X = (x_1,\dots,x_k)$ and $Y = (y_1,\dots,y_k)$ are ordered k-tuples with $\{x_1,\dots,x_k\} \subset \{1,\dots,n\}$, $\{y_1,\dots,y_k\} \subset \{1,\dots,n\}$ and no repetitions are allowed. $S_{X,Y}$ denotes the set of elements in S_n which are constrained to map each x_i to y_i:

$$S_{X,Y} \equiv \{\sigma \in S_n \ : \ \sigma(x_i) = y_i, \text{for each i=1,\dots,k}\}. \qquad (27)$$

The $S_{X,Y}$ can also be thought of as two-sided cosets associated with subgroups of the form $S_{n-k} \subset S_n$. For example, if $X = (1,2)$ and $Y = (3,4)$ with $n = 4$, then $S_{X,Y}$ is simply the set of all permutations that map $1 \mapsto 3$ and $2 \mapsto 4$. Thus, $S_{X,Y} = \{(1,3)(2,4), (1,3,2,4)\}$. Since $|X| = |Y| = k$, then $|S_{X,Y}| = (n-k)!$, and in the special case that $X = Y$, we have that $S_{X,Y}$ is in fact a subgroup isomorphic to S_{n-k}.

As we show in Appendix C, the Fourier transform of the indicator $\delta_{S_{X,Y}}$ takes a particularly simple (and low rank) form and can be efficiently computed. The method described in Appendix C is based on the FFT and exploits the same structure of the symmetric group that is used by Kondor et al. (2007). It is thus possible to understand why some observation models afford faster conditioning updates based on sparsity in Fourier domain.

The functions $\delta_{S_{X,Y}}$ can be viewed as a set of function primitives for constructing more complicated models via shift/scale/convolution operations in the Fourier domain. We now discuss the remaining models in Table 7 with the assumption that there exists some blackbox function which constructs the Fourier coefficients of the indicator function of (two-sided) cosets of the form $S_{X,Y} \subset S_n$ (see Algorithm 5 in Appendix D).

8.3.1 k-SUBSET MIXING

It is not always appropriate to mix only two people at once (as in Equation 22) and so we would like to formulate a mixing model which occurs over a subset of tracks, $X = \{t_1,\dots,t_k\} \subset \{1,\dots,n\}$. One way to 'mimic' the desired effect is to repeatedly draw pairs (i,j) from $\{t_1,\dots,t_k\}$ and to convolve against the pairwise mixing models Q_{ij}. A better alternative is to directly construct the Fourier coefficient matrices for the *k-subset mixing model*, in which we allow the tracks in X to be randomly permuted with uniform probability. In the following, $\bar{X}$ denotes some fixed ordering of the complement of X. For example, if $n = 5$, with $X = \{1,2,4\}$, then $\bar{X}$ is either $(3,5)$ or $(5,3)$. The k-subset mixing model is defined as:

$$Q_X(\pi) = \begin{cases} \frac{1}{k!} & \text{if } \pi \in S_{\bar{X},\bar{X}} \subset S_n \\ 0 & \text{otherwise} \end{cases}. \tag{28}$$

Note that $S_{\bar{X},\bar{X}}$ is isomorphic to S_k and that the pairwise mixing model is the special case where $k = 2$. Intuitively, Equation 28 fixes all of the tracks outside of X and says that with uniform probability, the set of tracks in X experience some permutation of their respective identities. Equation 28 can also be written as $Q_X(\pi) = \frac{1}{k!}\delta_{S_{\bar{X},\bar{X}}}(\pi)$, and thus the mixing model is simply a multiple of the indicator function of $S_{\bar{X},\bar{X}}$.

8.3.2 SINGLE/MULTI-TRACK OBSERVATION

In the *single track observation model* (used in Shin et al. 2005, Schumitsch et al. 2005 and Kondor et al. 2007, for example), we acquire an identity measurement z_j at track j. In the simplest version of the model, we write the likelihood function as:

$$P(z_i = j|\sigma) = \begin{cases} \pi & \text{if } \sigma(j) = i \\ \frac{1-\pi}{n-1} & \text{otherwise} \end{cases}, \tag{29}$$

where j ranges over all n possible identities. $P(z_i|\sigma)$ can also be written as a weighted sum of a uniform distribution U, and an indicator function:

$$P(z_i = j|\sigma) = \left(\frac{\pi n - 1}{n-1}\right)\delta_{S_{j,i}}(\sigma) + \left(\frac{1-\pi}{n-1}\right)U(\sigma).$$

Equation 29 is useful when we receive measurements directly as single identities ("Alice is at Track 1 with such and such probability"). It is, however, far more common to receive lower level measurements that *depend* only upon a single identity, which we formalize with the following conditional independence assumption:

$$P(z_i|\sigma) = P(z_i|\sigma(j)).$$

For example, as in Equation 23, we might have a color histogram over each individual ("Alice loves to wear green") and observe a single color per timestep. Or we might acquire observations in the form of color histograms and choose to model a distribution over all possible color histograms. If for each identity j, $P(z_i|\sigma(j) = i) = \alpha_j$, then we can write the likelihood function as a weighted linear combination of n indicators,

$$L(\sigma) = P(z_i|\sigma) = \sum_j \alpha_j \delta_{S_{j,i}}(\sigma),$$

and by the linearity of the Fourier transform, we can obtain the Fourier coefficients of L:

$$\hat{L}_\lambda = \sum_j \alpha_j \left[\widehat{\delta_{S_{j,i}}}\right]_\lambda.$$

Finally, the single-track observations can be generalized to handle joint observations of multiple tracks at once with a higher-order model:

$$P(z_{(t_1,\dots,t_k)} = (i_1,\dots,i_k)|\sigma) = \begin{cases} \pi & \text{if } \sigma(i_\ell) = t_\ell \text{ for each } \ell \in \{1,\dots,k\} \\ \frac{1-\pi}{\frac{n!}{(n-k)!}-1} & \text{otherwise} \end{cases} . \quad (30)$$

Unsurprisingly, while the Fourier coefficients of Equation 29 can be expressed exactly using first-order terms, the Fourier coefficients of the multi-track observation model, Equation 30, requires kth-order terms. It is important to note that joint multi-track observations are distinct from making k independent identity observations at the same timestep—we can handle the latter case by calling the Kronecker conditioning algorithm with a single-track observation model k times. Depending upon the specific sensor setup, one model may be more natural than the other.

8.3.3 BLUETOOTH OBSERVATION

In contrast with the first bluetooth model (Equation 26), our second bluetooth-type model handles a higher-order form of measurement. Like the single/multi-track observation models, it says that with some probability we receive the correct unordered list, and with some probability, we receive some other list drawn uniformly at random:

$$P^{bluetooth2}(z_{\{t_1,\dots,t_k\}} = \{i_1,\dots,i_k\}|\sigma) = \begin{cases} \pi & \text{if } \sigma(\{i_1,\dots,i_k\}) = \{t_1,\dots,t_k\} \\ \frac{1-\pi}{\binom{n}{k}-1} & \text{otherwise} \end{cases} .$$

As with the single/multi-track observation models, the bluetooth model can be written as a weighted linear combination of a uniform distribution and the indicator function of an $S_k \times S_{n-k}$-coset, where:

$$S_k \times S_{n-k} = \{\sigma \in S_n \,:\, \sigma(\{1,\dots,k\}) = \{1,\dots,k\}\}.$$

To compute the Fourier transform of $P^{bluetooth2}$, it is enough to note that the indicator function of $S_k \times S_{n-k}$ can be thought of as a convolution of indicator functions of S_k and S_{n-k} in a certain sense. More precisely.

Proposition 14 *Let $X = (1,\dots,k)$ and $Y = (k+1,\dots,n)$. Then:* $\delta_{S_k \times S_{n-k}} = \delta_{S_{X,X}} * \delta_{S_{Y,Y}}$.

Invoking the convolution theorem (Proposition 8) shows that the Fourier coefficient matrices of $\delta_{S_k\times S_{n-k}}$ can be constructed by first computing the Fourier coefficients of $S_{X,X}$ and $S_{Y,Y}$, and pointwise multiplying corresponding coefficient matrices. We have:

$$\left[\widehat{\delta}_{S_k\times S_{n-k}}\right]_\lambda = \left[\widehat{\delta}_{S_{X,X}}\right]_\lambda \cdot \left[\widehat{\delta}_{S_{Y,Y}}\right]_\lambda, \text{ for all partitions } \lambda.$$

An interesting fact about the bluetooth model is that its Fourier terms are zero at all partitions with more than two rows.

Proposition 15 *Without loss of generality, assume that $k \le \frac{n}{2}$. The Fourier transform of the bluetooth model, $\hat{P}^{bluetooth2}_\lambda$ is nonzero only at partitions of the form $(n-s,s)$ where $s \le k$.*

8.3.4 PAIRWISE RANKING OBSERVATION

Finally in the *pairwise ranking model*, we consider observations of the form "object j is ranked higher than object i" which can appear in various forms of voting and preference elicitation ("I like candidate x better than candidate y") or webpage/advertisement ranking. Here we think of σ as a mapping from objects to ranks. Our pairwise ranking model simply assigns higher probability to observations which agree with the ordering of i and j in σ.

$$P^{rank}(z_{k\ell}|\sigma) = \begin{cases} \pi & \text{if } \sigma(k) < \sigma(\ell) \\ 1-\pi & \text{otherwise} \end{cases} .$$

When $k = n-1$, $\ell = n$ and $\pi = 1$, we have:

$$\begin{aligned} P^{rank}(z_{k\ell}|\sigma) &= \begin{cases} 1 & \text{if } \sigma(n-1) < \sigma(n) \\ 0 & \text{otherwise} \end{cases} \\ &= \sum_{i<j} \delta_{S_{(n-1,n),(i,j)}}(\sigma). \end{aligned}$$

Perhaps unsurprisingly, pairwise ranking models can be sufficiently captured by first-order and second-order (ordered) Fourier coefficients[12].

Proposition 16 *The Fourier coefficients of the pairwise ranking model, $\hat{P}^{rank}_{\lambda}$, are nonzero only at three partitions: $\lambda = (n)$, $(n-1,1)$, and $(n-2,1,1)$.*

9. Related Work

Rankings and permutations have recently become an active area of research in machine learning due to their importance in information retrieval and preference elicitation. Rather than considering full distributions over permutations, many approaches, like RankSVM (Joachims, 2002) and RankBoost (Freund et al., 2003), have instead focused on learning a single 'optimal' ranking with respect to some objective function.

There are also several authors (from both the statistics and machine learning communities) who have studied distributions over permutations/rankings (Mallows, 1957; Critchlow, 1985; Fligner and Verducci, 1986; Meila et al., 2007; Taylor et al., 2008; Lebanon and Mao, 2008). Taylor et al. (2008) consider distributions over S_n which are induced by the rankings of n independent draws from n individually centered Gaussian distributions with equal variance. They compactly summarize their distributions using an $O(n^2)$ matrix which is conceptually similar to our first-order summaries and apply their techniques to ranking web documents. Most other previous approaches at directly modeling distributions on S_n, however, have relied on distance based exponential family models. For example, the Mallows model (Mallows, 1957) defines a Gaussian-like distribution over permutations as:

$$P(\sigma; c, \sigma_0) \propto \exp\left(-cd(\sigma, \sigma_0)\right),$$

where the function $d(\sigma, \sigma_0)$ is the *Kendall's tau distance* which counts the number of adjacent swaps that are required to bring σ^{-1} to σ_0^{-1}.

12. Additionally, $\hat{P}^{rank}_{(n-1,1)}$ and $\hat{P}^{rank}_{(n-2,1,1)}$ are known to be rank 1 matrices, a fact which can potentially be exploited for faster conditioning updates in practice.

Distance based exponential family models have the advantage that they can compactly represent distributions for very large n, and admit conjugate prior distributions (Meila et al., 2007). Estimating parameters has been a popular problem for statisticians—recovering the optimal σ_0 from data is known as the *consensus ranking* or *rank aggregation* problem and is known to be *NP*-hard (Bartholdi et al., 1989). Many authors have focused on approximation algorithms instead.

Like Gaussian distributions, distance based models also tend to lack flexibility, and so Lebanon and Mao (2008) propose a nonparametric model of ranked (and partially ranked) data based on placing weighted Mallows kernels on top of training examples, which, as they show, can realize a far richer class of distributions, and can be learned efficiently. However, they do not address the inference problem, and it is not clear if one can efficiently perform inference operations like marginalization and conditioning in such models.

As we have shown in this paper, Fourier based methods (Diaconis, 1988; Kondor et al., 2007; Huang et al., 2007) offer a principled alternative method for compactly representing distributions over permutations and performing efficient probabilistic inference operations. Our work draws from two strands of research—one from the data association/identity management literature, and one from a more theoretical area on Fourier analysis in statistics. In the following, we review several of the works which have led up to our current Fourier based approach.

9.1 Previous Work in Identity Management

The identity management problem has been addressed in a number of previous works, and is closely related to, but not identical with, the classical data association problem of maintaining correspondences between tracks and observations. Both problems need to address the fundamental combinatorial challenge that there is a factorial or exponential number of associations to maintain between tracks and identities, or between tracks and observations respectively. A vast literature already exists on the the data association problem, beginning with the *multiple hypothesis testing* approach (MHT) of Reid (1979). The MHT is a 'deferred logic' method in which past observations are exploited in forming new hypotheses when a new set of observations arises. Since the number of hypotheses can grow exponentially over time, various heuristics have been proposed to help cope with the complexity blowup. For example, one can choose to maintain only the *k best* hypotheses for some parameter k (Cox and Hingorani, 1994), using Murty's algorithm (Murty, 1968). But for such an approximation to be effective, k may still need to scale exponentially in the number of objects. A slightly more recent filtering approach is the *joint probabilistic data association filter* (JPDA) (Bar-Shalom and Fortmann, 1988), which is a suboptimal single-stage approximation of the optimal Bayesian filter. JPDA makes associations sequentially and is unable to correct erroneous associations made in the past (Poore, 1995). Even though the JPDA is more efficient than the MHT, the calculation of the JPDA association probabilities is still a #P-complete problem (Collins and Uhlmann, 1992), since it effectively must compute matrix permanents. Polynomial approximation algorithms to the JPDA association probabilities have recently been studied using Markov chain Monte Carlo (MCMC) methods (Oh et al., 2004; Oh and Sastry, 2005).

The identity management problem was first explicitly introduced in Shin et al. (2003). Identity management differs from the classical data association problem in that its observation model is not concerned with the low-level tracking details but instead with high level information about object identities. Shin et al. (2003) introduced the notion of the *belief matrix* approximation of the association probabilities, which collapses a distribution over all possible associations to just its first-order marginals. In the case of n tracks and n identities, the belief matrix B is an $n \times n$ doubly-stochastic matrix of non-negative entries b_{ij}, where b_{ij} is the probability that identity i is associated with track j. As we already saw in Section 4, the belief matrix approximation is equivalent to maintaining the zeroth- and first-order Fourier coefficients. Thus our current work is a strict generalization and extension of those previous results.

An alternative representation that has also been considered is an information theoretic approach (Shin et al., 2005; Schumitsch et al., 2005, 2006) in which the density is parameterized as:

$$P(\sigma;\Omega) \propto \exp \mathrm{Tr}\left(\Omega^T \cdot \tau_{(n-1,1)}(\sigma)\right).$$

In our framework, the information form approach can be viewed as a method for maintaining the Fourier transform of the *log* probability distribution at only the first two irreducibles. The information matrix approach is especially attractive in a distributed sensor network setting, since, if the columns of the information matrix are distributed to leader nodes tracking the respective targets, then the observation events become entirely local operations, avoiding the more expensive Kronecker conditioning algorithm in our setting. On the other hand, the information matrix coefficients do not have the same intuitive marginals interpretation afforded in our setting, and moreover, prediction/rollup steps cannot be performed analytically in the information matrix form. As in many classical data structures problems there are representation trade-off issues: some operations are less expensive in one representation and some operations in the the other. The best choice in any particular scenario will depend on the ratio between observation and mixing events.

9.2 Previous Work on Fourier-Based Approximations

The concept of using Fourier transforms to study probability distributions on groups is not new, with the earliest papers in this area having been published in the 1960s (Grenander, 1963). Willsky (1978) was the first to formulate the exact filtering problem in the Fourier domain for finite and locally compact Lie groups and contributed the first noncommutative Fast Fourier Transform algorithm (for Metacyclic groups). However, he does not address approximate inference, suggesting instead to always transform to the appropriate domain for which either the prediction/rollup or conditioning operations can be accomplished using a pointwise product. While providing significant improvements in complexity for smaller groups, his approach is still infeasible for our problem given the factorial order of the Symmetric group.

Diaconis (1988) used the Fourier transform to analyze probability distributions on the Symmetric group in order to study card shuffling and ranking problems. His work laid the ground for much of the progress made over the last two decades on probabilistic group theory and noncommutative FFT algorithms (Clausen and Baum, 1993; Rockmore, 2000).

Kondor et al. (2007) was the first to show that the data association problem could be efficiently approximated using FFT factorizations. In contrast to our framework where every model is assumed to be have been specified in the Fourier domain, they work with an observation model which can be written as the indicator function of cosets of subgroups of the form $S_k \subset S_n$.

Conceptually, one might imagine formulating a conditioning algorithm which applies the Inverse Fast Fourier Transform (IFFT) to the prior distribution, conditions in the primal domain using pointwise multiplication, then transforms back up to the Fourier domain using the FFT to obtain posterior Fourier coefficients. While such a procedure would ordinarily be intractable because of the factorial number of permutations, Kondor et al. (2007) elegantly shows that for certain coset-based observation models, it is not necessary to perform the full FFT recursion to do a pointwise product. They exploit this observation to formulate an efficient conditioning algorithm whose running time depends on the complexity of the observation model (which can roughly be measured by the number of irreducibles required to fully specify it).

Our work generalizes the conditioning formulation from Kondor et al. (2007) in the sense that it can work for *any* observation model and extends easily to similar filtering problems over any finite group. In the case that the observation model is specified at sufficiently many irreducibles, our conditioning algorithm (prior to the projection step) returns the same approximate probabilities as the FFT-based algorithm. For example, we can show that the observation model given in Equation 29 is fully specified by two Fourier components, and that both algorithms have identical output. Additionally, Kondor et al. (2007) do not address the issue of projecting onto legal distributions, which, as we show in our experimental results is fundamental in practice.

10. Experimental Results

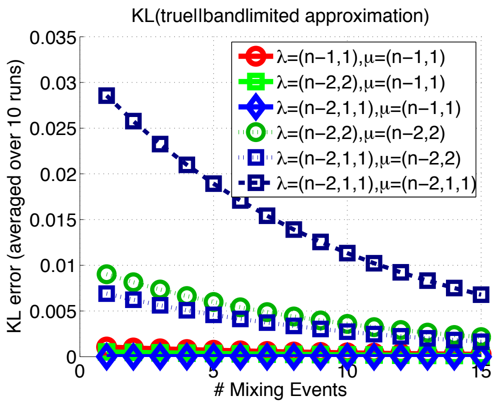
In this section we present the results of several experiments to validate our algorithm. We evaluate performance first by measuring the quality of our approximation for problems where the true distribution is known. Instead of measuring a distance between the true distribution and the inverse Fourier transform of our approximation, it makes more sense in our setting to measure error only at the marginals which are maintained by our approximation. In the results reported below, we measure the L_1 error between the true matrix of marginals and the approximation. If nonnegative marginal probabilities are guaranteed, it also makes sense to measure KL-divergence.

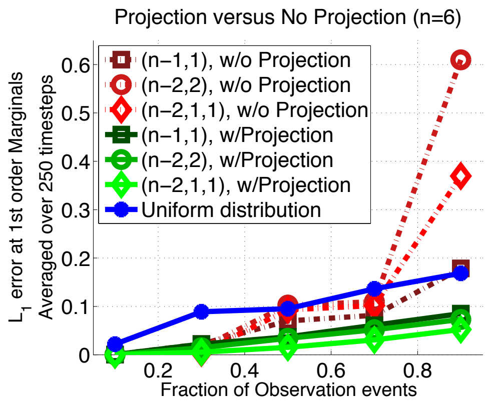
10.1 Simulated Data

We first tested the accuracy of a single Kronecker conditioning step by calling some number of pairwise mixing events (which can be thought roughly as a measure of entropy), followed by a single first-order observation. In the y-axis of Figure 7(a), we plot the Kullback-Leibler divergence between the true first-order marginals and approximate first-order marginals returned by Kronecker conditioning. We compared the results of maintaining first-order, and second-order (unordered and ordered) marginals. As shown in Figure 7(a), Kronecker conditioning is more accurate when the prior is smooth and unsurprisingly, when we allow for higher order Fourier terms. As guaranteed by Theorem 12, we also see that the first-order terms of the posterior are exact when we maintain second-order (ordered) marginals.

To understand how our algorithms perform over many timesteps (where errors can propagate to all Fourier terms), we compared to exact inference on synthetic data sets in which tracks are drawn at random to be observed or swapped. As a baseline, we show the accuracy of a uniform distribution. We observe that the Fourier approximation is better when there are either more mixing events (the fraction of conditioning events is smaller), or when more Fourier coefficients are maintained, as shown in Figure 7(b). We also see that the Plancherel Projection step is fundamental, especially when mixing events are rare.

Figures 10(a) and 10(b) show the per-timeslice accuracy of two typical runs of the algorithm. The fraction of conditioning events is 50% in Figure 10(a), and 70% in Figure 10(b). What we typically observe is that while the projected and nonprojected accuracies are often quite similar, the nonprojected marginals can perform significantly worse during certain segments.





(a) Kronecker Conditioning Accuracy—we measure the accuracy of a single Kronecker conditioning operation after some number of mixing events.

(b) HMM Accuracy—we measure the average accuracy of posterior marginals over 250 timesteps, varying the proportion of mixing and observation events

Figure 7: Simulation results.

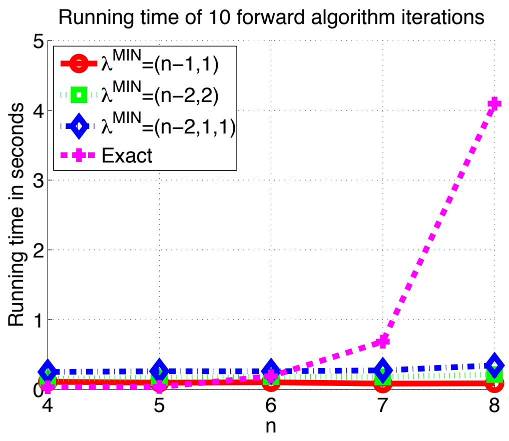


Figure 8: Running times: We compared running times of our polynomial time bandlimited inference algorithms against an exact algorithm with $O(n^3 n!)$ time complexity

Finally, we compared running times against an exact inference algorithm which performs prediction/rollup in the Fourier domain and conditioning in the primal domain. While the prediction/rollup step for pairwise mixing models can be implemented in $O(n!)$ time (linear in the size of the symmetric group), we show running times for the more general mixing models. Instead of the naive $O((n!)^2)$ complexity, its running time is a more efficient $O(n^3 n!)$ due to the Fast Fourier Transform (Clausen and Baum, 1993). It is clear that our algorithm scales gracefully compared to the exact solution (Figure 10.1), and in fact, we could not run exact inference for $n > 8$ due to memory constraints. In Figure 10.1, we show empirically that the Clebsch-Gordan coefficients are indeed sparse, supporting a faster conjectured runtime.

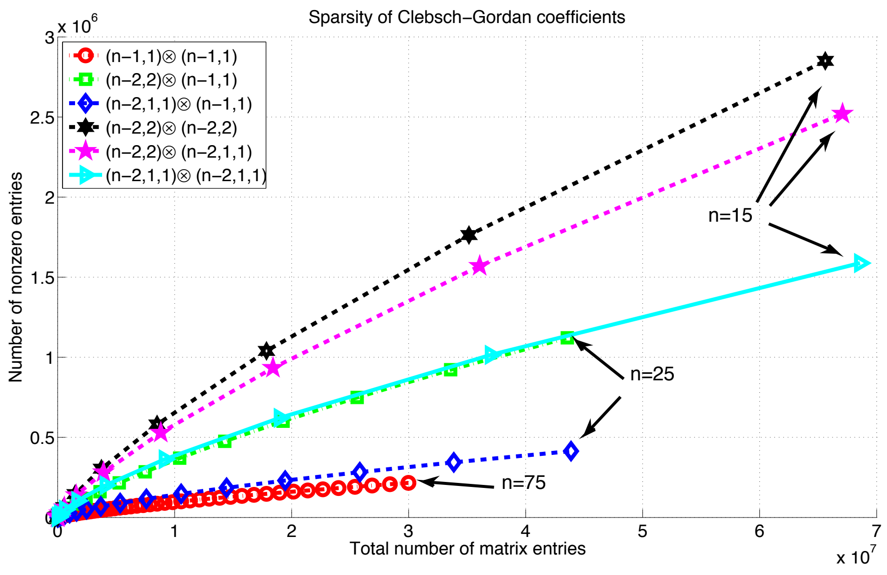
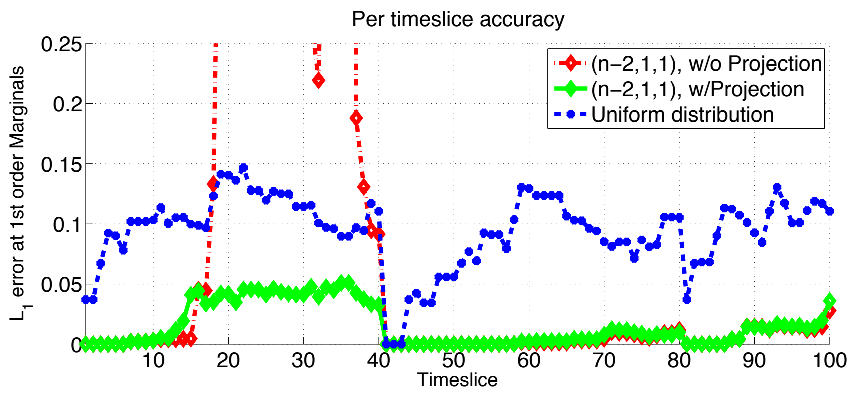


Figure 9: Clebsch-Gordan Sparsity: We measured the sparsity of the Clebsch-Gordan coefficients matrices by plotting the number of nonzero coefficients in a Clebsch-Gordan coefficient matrix against the number of total entries in the matrix for various n and pairs of irreducibles. For each fixed tensor product pair, we see that the number of nonzero entries scales sublinearly with respect to the total number of matrix elements.

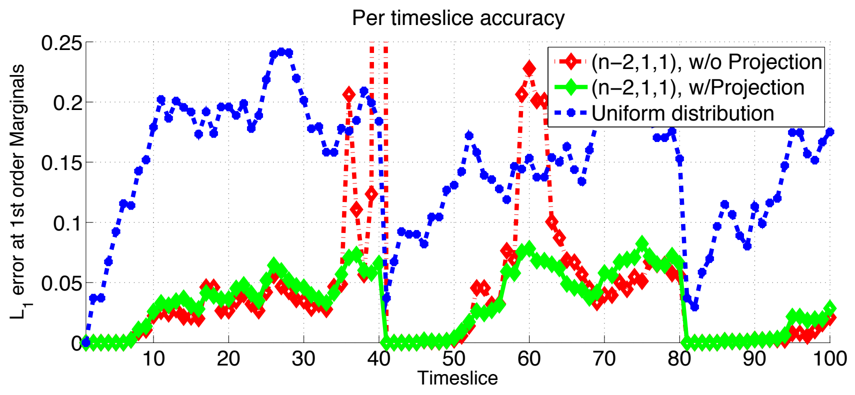
10.2 Real Camera Network

We also evaluated our algorithm on data taken from a real network of eight cameras (Fig. 11(a)). In the data, there are $n = 11$ people walking around a room in fairly close proximity. To handle the fact that people can freely leave and enter the room, we maintain a list of the tracks which are external to the room. Each time a new track leaves the room, it is added to the list and a mixing event is called to allow for m^2 pairwise swaps amongst the m external tracks.

The number of mixing events is approximately the same as the number of observations. For each observation, the network returns a color histogram of the blob associated with one track. The task after conditioning on each observation is to predict identities for all tracks which are inside the room, and the evaluation metric is the fraction of accurate predictions. We compared against a baseline approach of predicting the identity of a track based on the most recently observed histogram at that track. This approach is expected to be accurate when there are many observations and discriminative appearance models, neither of which our problem afforded. As Figure 11(b) shows, both the baseline and first order model(without projection) fared poorly, while the projection step dramatically boosted the prediction accuracy for this problem. To illustrate the difficulty of predicting based on appearance alone, the rightmost bar reflects the performance of an *omniscient* tracker who knows the result of each mixing event and is therefore left only with the task of distinguishing between appearances. We conjecture that the performance of our algorithm (with projection) is near optimal.



(a) $n = 6$ with 50% mixing events and 50% observations



(b) $n = 6$ with 30% mixing events and 70% observations

Figure 10: Accuracy as a function of time on two typical runs.

11. Future Research

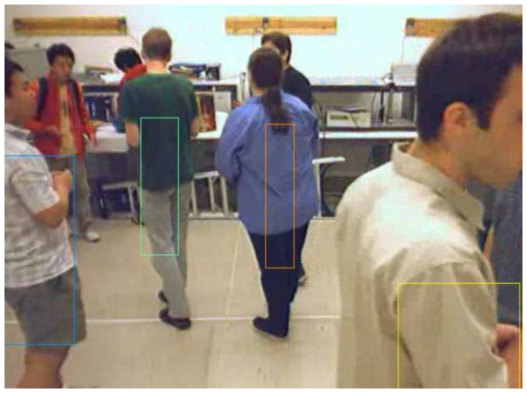
There remain several possible extensions to the current work stemming from both practical and theoretical considerations. We list a few open questions and extensions in the following.

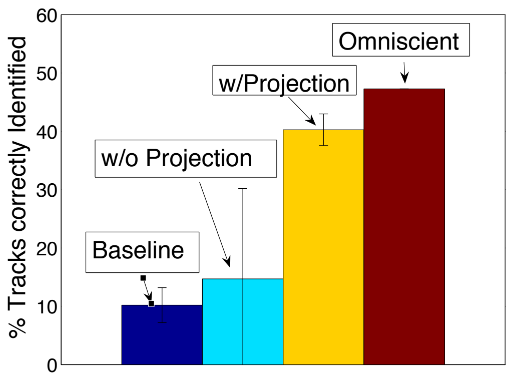
11.1 Adaptive Filtering

While our current algorithms easily beat exact inference in terms of running time, they are still limited by a relatively high (though polynomial) time complexity. In practice however, it seems reasonable to believe that the "difficult" identity management problems typically involve only a small subset of people at a time. A useful extension of our work would be to devise an *adaptive* version of the algorithm which allocates more Fourier coefficients towards the identities which require higher order reasoning. We believe that this kind of extension would be the appropriate way to scale our algorithm to handling massive numbers of objects at a time.

11.2 Characterizing the Marginal Polytope

In our paper, we presented a projection of the bandlimited distribution to a certain polytope, which is exactly the marginal polytope for first-order bandlimited distributions, but strictly an outer bound


(a) Sample Image



(b) Accuracy for Camera Data

Figure 11: Evaluation on data set from a real camera network. In this experiment, there are $n = 11$ people walking in a room begin tracked by 8 cameras.

for higher orders. An interesting project would be to generalize the Birkhoff-von Neumann theorem by exactly characterizing the marginal polytope at higher order marginals. We conjecture that the marginal polytope for low order marginals can be described with polynomially many constraints.

11.3 Learning in the Fourier Domain

Another interesting problem is whether we can learn bandlimited mixing and observation models *directly in the Fourier domain*. Given fully observed permutations $\sigma_1, \dots, \sigma_m$, drawn from a distribution $P(\sigma)$, a naive method for estimating $\hat{P}_\rho$ at low-order ρ is to simply observe that:

$$\hat{P}_\rho = \mathbb{E}_{\sigma \sim P}[\rho(\sigma)],$$

and so one can estimate the Fourier transform by simply averaging $\rho(\sigma_i)$ over all σ_i. However, since we typically do not observe full permutations in real applications like ranking or identity management, it would be interesting to estimate Fourier transforms using partially observed data. In the case of Bayesian learning, it may be possible to apply some of the techniques discussed in this paper.

11.4 Probabilistic Inference on Other Groups

The Fourier theoretic framework presented in this paper is not specific to the symmetric group - in fact, the prediction/rollup and conditioning formulations, as well as most of the results from Appendix D hold over any finite or compact Lie group. As an example, the noncommutative group of rotation operators in three dimensions, $SO(3)$, appears in settings which model the pose of a three dimensional object. Elements in $SO(3)$ might be used to represent the pose of a robot arm in robotics, or the orientation of a mesh in computer graphics; In many settings, it would be useful to have a compact representation of uncertainty over poses. We believe that there are many other application domains with algebraic structure where similar probabilistic inference algorithms might apply, and in particular, that noncommutative settings offer a particularly challenging but exciting opportunity for machine learning research.

12. Conclusions

In this paper, we have presented a Fourier theoretic framework for compactly summarizing distributions over permutations. We showed that common probabilistic inference operations can be performed completely in the Fourier domain and that, using the low-order terms of the Fourier expansion of a distribution, one can obtain polynomial time inference algorithms. Fourier theoretic summaries are attractive because they have tuneable approximation quality, have intuitive interpretations in terms of low-order marginals, and have allowed us to leverage results and insights from noncommutative Fourier analysis to formulate our algorithms.

The main contributions of our paper include methods for performing general probabilistic inference operations completely in the Fourier domain. In particular, we developed the Kronecker conditioning algorithm, which conditions a distribution on evidence using Bayes rule while operating only on Fourier coefficients. While prediction/rollup operations can be written as pointwise products in the Fourier domain, we showed that conditioning operations can be written, in dual fashion, as generalized convolutions in the Fourier domain. Our conditioning algorithm is general in two senses: first, one can use Kronecker conditioning to handle any observation model which can be written in the Fourier domain, and second, the same algorithm can be applied to condition distributions over arbitrary finite groups. Due to this generality, we are able to efficiently compute the Fourier transforms of a wide variety of probabilistic models which may potentially be useful in different applications.

We presented an analysis of the errors which can accumulate in bandlimited inference and argued that Fourier based approaches work well when the underlying distributions are diffuse and are thus well approximated by low-frequency basis functions. During inference, errors in high-order terms due to bandlimiting can be propagated to lower-order terms and bandlimited conditioning can, on occasion, result in Fourier coefficients which correspond to no valid distribution. We showed, however, that the problem can be remedied by projecting to a relaxation of the marginal polytope.

Finally, our evaluation on data from a camera network shows that our methods perform well when compared to the optimal solution in small problems, or to an omniscient tracker in large problems. Furthermore, we demonstrated that our projection step is fundamental in obtaining these high-quality results.

Algebraic methods have recently enjoyed a surge of interest in the machine learning community. We believe that our unified approach for performing probabilistic inference over permutations, as well as our gentle exposition of group representation theory and noncommutative Fourier analysis will significantly lower the barrier of entry for machine learning researchers who are interested in using or further developing algebraically inspired algorithms which are useful for real-world problems.

Acknowledgments

This work is supported in part by the Office of Naval Research under MURI N000140710747, the Army Research Office under grant W911NF-06-1-0275, the National Science Foundation under grants DGE-0333420, EEEC-540865, NeTS-NOSS 0626151 and TF 0634803, and by the Pennsylvania Infrastructure Technology Alliance (PITA). Carlos Guestrin was also supported in part by an Alfred P. Sloan Fellowship. We are grateful to Kyle Heath for collecting the camera data and to

Robert Hough, Emre Oto and Risi Kondor for helpful and insightful discussions. We would like to thank the reviewers for providing thoughtful and detailed feedback.

Appendix A. Groups

This section is intended as a quick glossary for the group theoretic definitions used in the paper. Groups are a generalization of many of the spaces that we typically work with, such as the real numbers, integers, vector spaces, and matrices. The definition of a group unifies all of these spaces under a handful of axioms.

Definition 17 (Group) *A* group *is a set G together with a binary operation $\cdot : G \times G \to G$ (called the* group operation*) such that the following* group axioms *hold:*

1. *(Associativity) The group operation is* associative*. That is, for any group elements $g_1, g_2, g_3 \in G$, we have:*
$$(g_1 \cdot g_2) \cdot g_3 = g_1 \cdot (g_2 \cdot g_3), \text{ for all } g_1, g_2, g_3 \in G.$$
2. *(Identity) There exists an* identity *element (denoted by ε) such that $g \cdot \varepsilon = \varepsilon \cdot g = g$ for any $g \in G$.*
3. *(Inverses) For every $g \in G$, there exists an* inverse element *g^{-1} such that $g \cdot g^{-1} = g^{-1} \cdot g = \varepsilon$.*

Definition 18 (Abelian Group) *If, for any group elements $g_1, g_2 \in G$, we have $g_1 \cdot g_2 = g_2 \cdot g_1$, then G is called an* Abelian *or* commutative *group.*

Perhaps the most familiar group is the set of integers, $\mathbb{Z}$, with respect to the addition operation. It is well known that for any integers $a, b, c \in \mathbb{Z}$, $a + (b + c) = (a + b) + c$. The identity element in the integers is zero, and every element has an additive inverse ($a + (-a) = (-a) + a = 0$). Additionally, the integers are an Abelian group since $a + b = b + a$ for any $a, b \in \mathbb{Z}$. Note that the natural numbers $\mathbb{N} = \{0, 1, 2, 3, \ldots\}$ do not form a group with respect to addition because inverses do not exist.

The main example of a group in this paper, of course, is the symmetric group, the set of permutations of $\{1, \ldots, n\}$. The group operation on permutations is function composition, which is associative, and we discussed inverses and the identity element in Section 3.

Example 13 *There are many groups besides the integers and the symmetric group. The following are several examples.*

- *The positive real numbers $\mathbb{R}^+$ form a group with respect to multiplication. The identity element of $\mathbb{R}^+$ is the multiplicative identity, 1, and given a real number x, there exists an inverse element $\frac{1}{x}$.*
- *As an example of a finite group, the* integers modulo *n, $\mathbb{Z}/n\mathbb{Z}$, form a group with respect to addition modulo n.*
- *The invertible $n \times n$ matrices over the reals, $GL_n(\mathbb{R})$, form a group with respect to matrix multiplication. The $n \times n$ identity matrix serves as the identity element in $GL_n(\mathbb{R})$, and by assumption, every matrix in $GL_n(\mathbb{R})$ is invertible.*

The group axioms impose strong structural constraints on G, and one of the ways that structure is manifested in groups is in the existence of *subgroups*.

Definition 19 (Subgroup) *If G is a group (with group operation $\cdot$), a subset $H \subset G$ is called a* subgroup *if it is itself a group with respect to the same group operation. H is called a* trivial subgroup *if it is either all of G or consists only of a single element.*

Example 14 *We have the following examples of subgroups.*

- *The even integers, $2\mathbb{Z}$, form a subgroup of the integers since the sum of any two even integers is an even integer, and the inverse (negative) of an even integer is again even. However, the odd integers do* not *form a subgroup since the sum of two odd integers is not odd.*
- *The special orthogonal matrices (orthogonal matrices with determinant $+1$) form a subgroup of the group of $n \times n$ matrices, $GL_n(\mathbb{R})$. This can be seen by using the facts (1), that $(\det A)(\det B) = \det(AB)$ and (2), that the inverse of any orthogonal matrix is also orthogonal.*

Appendix B. Constructing Irreducible Representation Matrices

In this section, we present (without proof) some standard algorithms for constructing the irreducible representation matrices with respect to the *Gel'fand-Tsetlin (GZ) basis* (constructed with respect to the subgroup chain $S_1 \subset S_2 \subset \cdots \subset S_n$).[13] None of the techniques in Appendix B are novel. For a more elaborate discussion, see, for example, Kondor (2006), Chen (1989) and Vershik and Okounkov (2006). There are several properties which make the irreducible representation matrices, written with respect to the GZ basis, fairly useful in practice. They are guaranteed to be, for example, real-valued and orthogonal. And as we will show, the matrices have certain useful sparsity properties that can be exploited in implementation.

We begin by introducing a few concepts relating to *Young tableaux* which are like Young tabloids with the distinction that the rows are considered as *ordered tuples* rather than *unordered sets*. For example, the following two diagrams are distinct as Young *tableaux*, but not as Young *tabloids*:

$$\begin{array}{|c|c|c|}\hline 1&2&3\\\hline 4&5\\\cline{1-2}\end{array} \neq \begin{array}{|c|c|c|}\hline 1&3&2\\\hline 5&4\\\cline{1-2}\end{array} \qquad \text{(as Young tableaux).}$$

A Young Tableau t is said to be *standard* if its entries are increasing to the right along rows and down columns. For example, the set of all standard Young Tableaux of shape $\lambda = (3,2)$ is:

$$\left\{ \begin{array}{|c|c|c|}\hline 1&3&5\\\hline 2&4\\\cline{1-2}\end{array},\ \begin{array}{|c|c|c|}\hline 1&2&5\\\hline 3&4\\\cline{1-2}\end{array},\ \begin{array}{|c|c|c|}\hline 1&3&4\\\hline 2&5\\\cline{1-2}\end{array},\ \begin{array}{|c|c|c|}\hline 1&2&4\\\hline 3&5\\\cline{1-2}\end{array},\ \begin{array}{|c|c|c|}\hline 1&2&3\\\hline 4&5\\\cline{1-2}\end{array} \right\}. \tag{31}$$

Given a permutation $\sigma \in S_n$, one can always apply σ to a Young tableau t to get a new Young tableau, which we denote by $\sigma \circ t$, by permuting the labels within the tableau. For example,

$$(1,2) \circ \begin{array}{|c|c|c|}\hline \mathbf{1}&\mathbf{2}&3\\\hline 4&5\\\cline{1-2}\end{array} = \begin{array}{|c|c|c|}\hline \mathbf{2}&\mathbf{1}&3\\\hline 4&5\\\cline{1-2}\end{array}.$$

Note, however, that even if t is a standard tableau, $\sigma \circ t$ is not guaranteed to be standard.

The significance of the standard tableaux is that the set of all standard tableaux of shape λ can be used to index the set of GZ basis vectors for the irreducible representation ρ_λ. Since there are

13. The irreducible representation matrices in this Appendix are also sometimes referred to as *Young's Orthogonal Representation (YOR)*.

five total standard tableaux of shape $(3,2)$, we see, for example, that the irreducible corresponding to the partition $(3,2)$ is 5-dimensional. There is a simple recursive procedure for enumerating the set of all standard tableaux of shape λ, which we illustrate for $\lambda = (3,2)$.

Example 15 *If $\lambda = (3,2)$, there are only two possible boxes that the label 5 can occupy so that both rows and columns are increasing. They are:*

$$\begin{array}{|c|c|c|}\hline \; & \; & 5 \\ \hline \; & \; \\ \cline{1-2}\end{array}, \textit{ and } \begin{array}{|c|c|c|}\hline \; & \; & \; \\ \hline \; & 5 \\ \cline{1-2}\end{array}.$$

To enumerate the set of all standard tableaux of shape $(3,2)$, we need to fill the empty boxes in the above partially filled tableaux with the labels $\{1,2,3,4\}$ so that both rows and columns are increasing. Enumerating the standard tableaux of shape $(3,2)$ thus reduces to enumerating the set of standard tableaux of shapes $(2,2)$ and $(3,1)$, respectively. For $(2,2)$, the set of standard tableaux (which, in implementation would be computed recursively) is:

$$\left\{ \begin{array}{|c|c|}\hline 1 & 3 \\ \hline 2 & 4 \\ \hline\end{array}, \begin{array}{|c|c|}\hline 1 & 2 \\ \hline 3 & 4 \\ \hline\end{array} \right\},$$

and for $(3,1)$, the set of standard tableaux is:

$$\left\{ \begin{array}{|c|c|c|}\hline 1 & 3 & 4 \\ \hline 2 \\ \cline{1-1}\end{array}, \begin{array}{|c|c|c|}\hline 1 & 2 & 4 \\ \hline 3 \\ \cline{1-1}\end{array}, \begin{array}{|c|c|c|}\hline 1 & 2 & 3 \\ \hline 4 \\ \cline{1-1}\end{array} \right\}.$$

The entire set of standard tableaux of shape $(3,2)$ is therefore:

$$\left\{ \begin{array}{|c|c|c|}\hline 1 & 3 & \mathbf{5} \\ \hline 2 & 4 \\ \cline{1-2}\end{array}, \begin{array}{|c|c|c|}\hline 1 & 2 & \mathbf{5} \\ \hline 3 & 4 \\ \cline{1-2}\end{array} \right\} \bigcup \left\{ \begin{array}{|c|c|c|}\hline 1 & 3 & 4 \\ \hline 2 & \mathbf{5} \\ \cline{1-2}\end{array}, \begin{array}{|c|c|c|}\hline 1 & 2 & 4 \\ \hline 3 & \mathbf{5} \\ \cline{1-2}\end{array}, \begin{array}{|c|c|c|}\hline 1 & 2 & 3 \\ \hline 4 & \mathbf{5} \\ \cline{1-2}\end{array} \right\}.$$

Before explicitly constructing the representation matrices, we must define a signed distance on Young Tableaux called the *axial distance*.

Definition 20 *The* axial distance, *$d_t(i,j)$, between entries i and j in tableau t, is defined to be:*

$$d_t(i,j) \equiv (col(t,j) - col(t,i)) - (row(t,j) - row(t,i)),$$

where $row(t,i)$ denotes the row of label i in tableau t, and $col(t,i)$ denotes the column of label i in tableau t.

Intuitively, the axial distance between $i-1$ and i in a standard tableau t is equal to the (signed) number of steps that are required to travel from $i-1$ to i, if at each step, one is allowed to traverse a single box in the tableau in one of the four cardinal directions. For example, the axial distance from 3 to 4 with respect to tableau: $t = \begin{array}{|c|c|c|}\hline 1 & 2 & 3 \\ \hline 4 & 5 \\ \cline{1-2}\end{array}$ is:

$$\begin{aligned} d_t(3,4) &= \left(col\left(\begin{array}{|c|c|c|}\hline 1 & 2 & 3 \\ \hline 4 & 5 \\ \cline{1-2}\end{array}, 4 \right) - col\left(\begin{array}{|c|c|c|}\hline 1 & 2 & 3 \\ \hline 4 & 5 \\ \cline{1-2}\end{array}, 3 \right) \right) - \left(row\left(\begin{array}{|c|c|c|}\hline 1 & 2 & 3 \\ \hline 4 & 5 \\ \cline{1-2}\end{array}, 4 \right) - row\left(\begin{array}{|c|c|c|}\hline 1 & 2 & 3 \\ \hline 4 & 5 \\ \cline{1-2}\end{array}, 3 \right) \right) \\ &= (1-3) - (2-1) = -3 \end{aligned}$$

B.1 Constructing Representation Matrices for Adjacent Transpositions

In the following discussion, we will consider a fixed ordering, $t_1, \ldots, t_{d_\lambda}$, on the set of standard tableaux of shape λ and refer to both standard tableaux and columns of $\rho_\lambda(\sigma)$ interchangeably. Thus t_1 refers to first column, t_2 refers to the second column and so on. And we will index elements in $\rho_\lambda(\sigma)$ using pairs of standard tableau, (t_j, t_k).

To explicitly define the representation matrices with respect to the GZ basis, we will first construct the matrices for adjacent transpositions (i.e., permutations of the form $(i-1,i)$), and then we will construct arbitrary representation matrices by combining the matrices for the adjacent transpositions. The rule for constructing the matrix coefficient $[\rho_\lambda(i-1,i)]_{t_j,t_k}$ is as follows.

1. Define the (t_j, t_k) coefficient of $\rho_\lambda(i-1,i)$ to be zero if it is (1), off-diagonal ($j \neq k$) and (2), not of the form $(t_j, (i-1,i) \circ t_k)$.
2. If (t_j, t_k) is a diagonal element, (i.e., of the form (t_j, t_j)), define:
$$[\rho_\lambda(i-1,i)]_{t_j,t_j} = 1/d_{t_j}(i-1,i),$$
where $d_{t_j}(i-1,i)$ is the axial distance which we defined earlier in the section.
3. If (t_j, t_k) can be written as $(t_j, (i-1,i) \circ t_j)$ define:
$$[\rho_\lambda(i-1,i)]_{t_j, \sigma \circ t_j} = \sqrt{1 - 1/d_{t_j}^2(i-1,i)}.$$

Note that the only time that off-diagonal elements can be nonzero under the above rules is when $(i-i,i) \circ t_j$ happens to also be a standard tableau. If we apply an adjacent transposition, $\sigma = (i-1,i)$ to a standard tableau t, then $\sigma \circ t$ is guaranteed to be standard if and only if $i-1$ and i were neither in the same row nor column of t. This can be seen by examining each case separately.

1. **$i-1$ and i are in the same row or same column of t.** If i and $i-1$ are in the same row of t, then $i-1$ lies to the left of i. Applying $\sigma \circ t$ swaps their positions so that i lies to the left of $i-1$, and so we see that $\sigma \circ t$ cannot be standard. For example,
$$(3,4) \circ \begin{array}{|c|c|c|}\hline 1 & 2 & 5 \\ \hline \mathbf{3} & \mathbf{4} \\ \cline{1-2} \end{array} = \begin{array}{|c|c|c|}\hline 1 & 2 & 5 \\ \hline \mathbf{4} & \mathbf{3} \\ \cline{1-2} \end{array}.$$
Similarly, we see that if i and $i-1$ are in the same column of t, $\sigma \circ t$ cannot be standard. For example,
$$(3,4) \circ \begin{array}{|c|c|c|}\hline 1 & \mathbf{3} & 5 \\ \hline 2 & \mathbf{4} \\ \cline{1-2} \end{array} = \begin{array}{|c|c|c|}\hline 1 & \mathbf{4} & 5 \\ \hline 2 & \mathbf{3} \\ \cline{1-2} \end{array}.$$
2. **$i-1$ and i are neither in the same row nor column of t.** In the second case, $\sigma \circ t$ can be seen to be a standard tableau due to the fact that $i-1$ and i are adjacent indices. For example,
$$(3,4) \circ \begin{array}{|c|c|c|}\hline 1 & 2 & \mathbf{3} \\ \hline \mathbf{4} & 5 \\ \cline{1-2} \end{array} = \begin{array}{|c|c|c|}\hline 1 & 2 & \mathbf{4} \\ \hline \mathbf{3} & 5 \\ \cline{1-2} \end{array}.$$

Therefore, to see if $(i-1,i) \circ t$ is standard, we need only check to see that $i-1$ and i are in different rows and columns of the tableau t. The pseudocode for constructing the irreducible representation matrices for adjacent swaps is summarized in Algorithm 3. Note that the matrices constructed in the algorithm are sparse, with no more than two nonzero elements in any given column.

Algorithm 3: Pseudocode for computing irreducible representations matrices with respect to the Gel'fand-Tsetlin basis at adjacent transpositions.

```
ADJACENTRHO
input : $i \in \{2,\ldots,n\}, \lambda$
output: $\rho_\lambda(i-1,i)$
1  $\rho \leftarrow \mathbf{0}_{d_\lambda \times d_\lambda}$;
2  foreach standard tableaux $t$ of shape $\lambda$ do
3      $d \leftarrow (col(t,i) - col(t,i-1)) - (row(t,i) - row(t,i-1))$;
4      $\rho(t,t) \leftarrow 1/d$;
5      if $i-1$ and $i$ are in different rows and columns of $t$ then
6          $\rho((i-1,i) \circ (t), t) \leftarrow \sqrt{1-1/d^2}$;
7  return $\rho$ ;
```

Example 16 *We compute the representation matrix of* $\rho_{(3,2)}$ *evaluated at the adjacent transposition* $\sigma = (i-1,i) = (3,4)$. *For this example, we will use the enumeration of the standard tableaux of shape* $(3,2)$ *given in Equation 31.*

For each $(3,2)$*-tableau* t_j, *we identify whether* $\sigma \circ t_j$ *is standard and compute the axial distance from* 3 *to* 4 *on the tableau* t_j.

j	*1*	*2*	*3*	*4*	*5*
t_j	$\begin{array}{\|c\|c\|c\|}\hline 1&\mathbf{3}&5\\\hline 2&\mathbf{4}\\\cline{1-2}\end{array}$	$\begin{array}{\|c\|c\|c\|}\hline 1&2&5\\\hline \mathbf{3}&\mathbf{4}\\\cline{1-2}\end{array}$	$\begin{array}{\|c\|c\|c\|}\hline 1&\mathbf{3}&\mathbf{4}\\\hline 2&5\\\cline{1-2}\end{array}$	$\begin{array}{\|c\|c\|c\|}\hline 1&2&\mathbf{4}\\\hline \mathbf{3}&5\\\cline{1-2}\end{array}$	$\begin{array}{\|c\|c\|c\|}\hline 1&2&\mathbf{3}\\\hline \mathbf{4}&5\\\cline{1-2}\end{array}$
$(3,4)\circ t_j$	$\begin{array}{\|c\|c\|c\|}\hline 1&\mathbf{4}&5\\\hline 2&\mathbf{3}\\\cline{1-2}\end{array}$	$\begin{array}{\|c\|c\|c\|}\hline 1&2&5\\\hline \mathbf{4}&\mathbf{3}\\\cline{1-2}\end{array}$	$\begin{array}{\|c\|c\|c\|}\hline 1&\mathbf{4}&\mathbf{3}\\\hline 2&5\\\cline{1-2}\end{array}$	$\begin{array}{\|c\|c\|c\|}\hline 1&2&\mathbf{3}\\\hline \mathbf{4}&5\\\cline{1-2}\end{array}$	$\begin{array}{\|c\|c\|c\|}\hline 1&2&\mathbf{4}\\\hline \mathbf{3}&5\\\cline{1-2}\end{array}$
$(3,4)\circ t_j$ *Standard?*	*No*	*No*	*No*	*Yes*	*Yes*
axial distance ($d_{t_j}(3,4)$)	*-1*	*1*	*1*	*3*	*-3*

Putting the results together in a matrix yields:,

$$\rho_{(3,2)}(3,4) = \left[\begin{array}{c|ccccc} & t_1 & t_2 & t_3 & t_4 & t_5 \\ \hline t_1 & -1 & & & & \\ t_2 & & 1 & & & \\ t_3 & & & 1 & & \\ t_4 & & & & \frac{1}{3} & \sqrt{\frac{8}{9}} \\ t_5 & & & & \sqrt{\frac{8}{9}} & -\frac{1}{3} \end{array}\right],$$

where all of the empty entries are zero.

B.2 Constructing Representation Matrices for General Permutations

To construct representation matrices for general permutations, it is enough to observe that all permutations can be factored into a sequence of adjacent swaps. For example, the permutation $(1,2,5)$ can be factored into:

$$(1,2,5) = (4,5)(3,4)(1,2)(2,3)(3,4)(4,5),$$

Algorithm 4: Pseudocode for computing irreducible representation matrices for arbitrary permutations.

```
GETRHO
input  : $\sigma \in S_n, \lambda$
output: $\rho_\lambda(\sigma)$ (a $d_\lambda \times d_\lambda$ matrix)
1  //Use Bubblesort to factor $\sigma$ into a product of transpositions
2  $k \leftarrow 0$ ;
3  $factors \leftarrow \emptyset$;
4  for $i = 1, 2, \ldots, n$ do
5      for $j = n, n-1, \ldots, i+1$ do
6          if $\sigma(j) < \sigma(j-1)$ then
7              Swap($\sigma(j-1), \sigma(j)$) ;
8              $k \leftarrow k+1$ ;
9              $factors(k) \leftarrow j$ ;
10 //Construct representation matrix using adjacent transpositions
11 $\rho_\lambda(\sigma) \leftarrow I_{d_\lambda \times d_\lambda}$ ;
12 $m \leftarrow length(factors)$;
13 for $j = 1, \ldots, m$ do
14     $\rho_\lambda(\sigma) \leftarrow$ GETADJACENTRHO $(factors(j), \lambda) \cdot \rho_\lambda(\sigma)$ ;
```

and hence, for any partition λ,

$$\rho_\lambda(1,2,5) = \rho_\lambda(4,5) \cdot \rho_\lambda(3,4) \cdot \rho_\lambda(1,2) \cdot \rho_\lambda(2,3) \cdot \rho_\lambda(3,4) \cdot \rho_\lambda(4,5),$$

since ρ_λ is a group representation. Algorithmically, factoring a permutation into adjacent swaps looks very similar to the Bubblesort algorithm, and we show the pseudocode in Algorithm 4.

Appendix C. Fourier Transforming the Indicator Function $\delta_{S_{X,Y}}$

In this section, we derive the Fourier transform of the indicator function of the two-sided coset $S_{X,Y} \subset S_n$ (see Equation 27). To do so, we will need to understand the Gel'fand-Tsetlin basis at a slightly deeper level. For example, the fact that the basis elements are indexed by standard tableaux has not been motivated and may seem unintuitive. We begin this section by motivating the standard tableaux from the perspective of the *branching rule*, a standard fact taken from the representation theory of the symmetric group. We then show how the branching rule leads to a method for computing the desired indicator functions.

C.1 Standard Tableaux and the Gel'fand-Tsetlin Basis

It is straightforward to see that any irreducible representation ρ_λ of S_n, can also be seen as a representation of S_{n-1} when restricted to permutations in S_{n-1} (the set of elements which fix n). We will denote the restricted representation by $\rho_\lambda \downarrow^{S_n}_{S_{n-1}}$. However, the irreducibility property may not be preserved by restriction, which is to say that, as a representation of S_{n-1}, ρ_λ is not necessarily irreducible and might decompose as Equation 2 would dictate. Thus, for all $\sigma \in S_{n-1}$ (or more precisely, $\sigma \in S_n$ such that $\sigma(n) = n$), there exists C_λ and multiplicities $z_{\lambda\mu}$ such that:

$$C_\lambda^{-1} \cdot \rho_\lambda(\sigma) \cdot C_\lambda = \bigoplus_{\mu} \bigoplus_{j=1}^{z_{\lambda\mu}} \rho_\mu(\sigma),$$

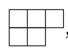
where μ ranges over the partitions of $n-1$.

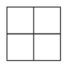
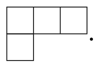
The *branching rule* allows us to state the decomposition even more precisely. Given any partition λ of n, let λ^- index over the set of partitions of $n-1$ whose Ferrers diagrams differ from λ in a single box.

Theorem 21 (Branching Rule, see Vershik and Okounkov (2006) for a proof) *For each irreducible representation ρ_λ of S_n, there exists a matrix C_λ such that:*

$$C_\lambda^{-1} \cdot \rho_\lambda(\sigma) \cdot C_\lambda = \bigoplus_{\lambda^-} \rho_{\lambda^-}(\sigma)$$

holds for any $\sigma \in S_{n-1}$.

Example 17 *If $\lambda = (3,2)$, then its corresponding Ferrers diagram is:* *, and the Ferrers diagrams corresponding to partitions of 4 which differ from λ in a single box are:*

, .

Thus, λ^- indexes over the set $\{(2,2),(3,1)\}$. The branching rule states that given an irreducible matrix representation $\rho_{(3,2)}$ of S_5, then there is a matrix $C_{(3,2)}$ such that, for any permutation $\sigma \in S_5$ such that $\sigma(5) = 5$,

$$C_\lambda^{-1} \cdot \rho_{(3,2)}(\sigma) \cdot C_\lambda = \left[\begin{array}{c|c} \rho_{(2,2)}(\sigma) & 0 \\ \hline 0 & \rho_{(3,1)}(\sigma) \end{array}\right].$$

The Gel'fand-Tsetlin basis is constructed such that the branching rule holds with all $C_\lambda = I$. Thus the irreducible representation matrices constructed with respect to the GZ basis have the property that the equation:

$$\rho_\lambda(\sigma) = \bigoplus_{\lambda^-} \rho_{\lambda^-}(\sigma)$$

holds identically for all $\sigma \in S_{n-1}$. We now can show how the branching rule naturally leads to indexing the basis elements by standard tableaux. First observe that the branching rule allows us to associate each column of the irreducible ρ_λ with some partition of $n-1$.

If we recursively apply the branching rule again (thus restricting to S_{n-2}), we see that the following decomposition holds:

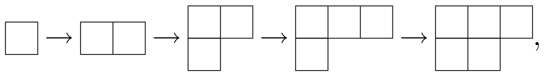
$$\rho_\lambda(\sigma) = \bigoplus_{\lambda^-} \left[\bigoplus_{\lambda^{--}} \rho_{\lambda^{--}}(\sigma) \right],$$

where λ^{--} indexes over partitions which differ from λ^- by a single box. Thus each column can be associated with a partition of $n-1$ *and* a partition of $n-2$. Taking this logic even further, we can restrict to S_{n-3}, S_{n-4}, and so on until we can restrict no further, associating each column with a sequence of partitions[14] $\mu_1 \vdash 1, \mu_2 \vdash 2 \ldots, \mu_n \vdash n$, where each partition μ_i can be obtained by adding

14. Here we use the $\lambda \vdash n$ notation to denote the relation that λ is a partition of n. For example, $(3,2,1) \vdash 6$.

a single box to the Ferrers diagram of μ_{i-1}, and $\mu_n = \lambda$. We will refer to such a sequence as a *branching sequence*. Since the branching rule guarantees multiplicity-free decompositions (that is, $z_{\lambda\mu} = 1$ for all pairs (λ, μ)), it turns out that each column of ρ_λ is *uniquely* specified by a branching sequence.

Example 18 *A possible branching sequence is:*



or written as partitions, $[(1) \to (2) \to (2,1) \to (3,1) \to (3,2)]$.

The set of all possible branching sequences ending in λ can be visualized using a *branching tree* (shown for $\lambda = (3,2)$ in Figure 12(a)), where each branching sequence is a path between the root and some leaf node. We will denote the branching tree corresponding to the partition λ by $\mathcal{T}^\lambda$ and the set of nodes at the r^{th} level of $\mathcal{T}^\lambda$ by $\mathcal{T}^\lambda_r$ (where the root node forms the zeroth level by convention). We can rephrase the branching rule in terms of the branching tree.

Proposition 22 *Let ρ_λ be an irreducible matrix representation of S_n (constructed with respect to the Gel'fand-Tsetlin basis). For any $\sigma \in S_k \subset S_n$, $\rho_\lambda(\sigma)$ decomposes as:*

$$\rho_\lambda(\sigma) = \bigoplus_{\mu \in \mathcal{T}^\lambda_{n-k}} \rho_\mu(\sigma).$$

Example 19 *As an example, consider applying Proposition 22 to $\rho_{(3,2)}$ with $k = 3$. The $(n-k)$th (second) level of the branching tree for $\lambda = (3,2)$, $\mathcal{T}_2^{(3,2)}$ consists of two copies of the partition $(2,1)$ and a single copy of the partition (3). Thus for any element $\sigma \in S_5$ which fixes 4 and 5 ($\sigma(4) = 4$, $\sigma(5) = 5$), we have:*

$$\rho_{(3,2)}(\sigma) = \left[\begin{array}{c|c|c} \rho_{(2,1)}(\sigma) & & \\ \hline & \rho_{(2,1)}(\sigma) & \\ \hline & & \rho_{(3)}(\sigma) \end{array}\right].$$

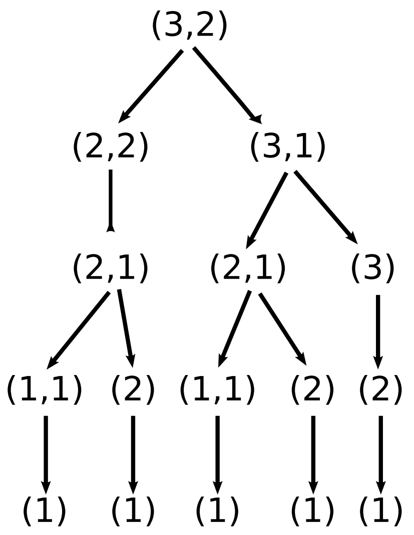
As a final remark, observe that branching sequences can be compactly represented as *standard tableaux*, where the number in each box indicates the point in the sequence at which the box was added. For example, the following standard tableau and sequence of partitions are equivalent:

$$\begin{array}{|c|c|c|} \hline 1 & 2 & 4 \\ \hline 3 & 5 \\ \cline{1-2} \end{array} \qquad \longleftrightarrow \qquad [(1) \to (2) \to (2,1) \to (3,1) \to (3,2)].$$

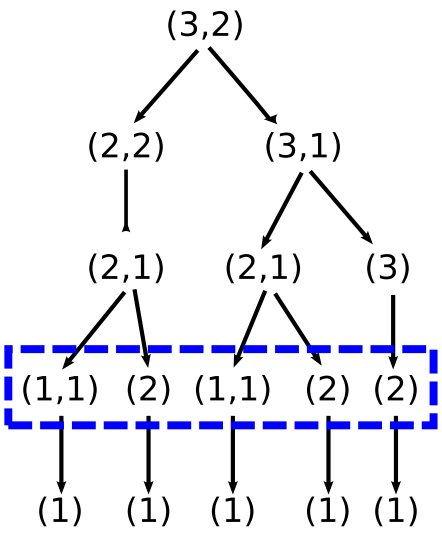
To summarize, the GZ basis (adapted to the subgroup chain $S_1 \subset \cdots \subset S_n$) is defined so that the branching rule holds as a matrix identity (with no need of a change of basis matrix), and furthermore, each basis vector of the representation space for ρ_λ can be associated with a branching sequence, or equivalently, a standard tableau.

C.2 Fourier Transforming $\delta_{S_{X,Y}}$

We are now in a position to compute the Fourier transform of indicators of the form $\delta_{S_{X,Y}}$. First, as a corollary of the branching rule, we see that we can decompose the Fourier transform of functions that are supported on $S_{n-1} \subset S_n$.



(a) $\mathcal{T}^{(3,2)}$



(b) $\mathcal{T}_3^{(3,2)}$

Figure 12: (a) The branching tree for $\lambda = (3,2)$. (b) The 3rd level of $\mathcal{T}^{(3,2)}$ (outlined) is denoted by $\mathcal{T}_3^{(3,2)}$ and consists of two copies of the partition $(1,1)$ and three copies of the partition (2).

Corollary 23 *If $f : S_n \to \mathbb{R}$ is supported on the subgroup S_{n-1}, then for each partition λ, the Fourier transform of f (with respect to the Gel'fand-Tsetlin basis adapted $S_1 \subset S_2 \subset \cdots \subset S_n$) decomposes into a direct sum of Fourier transforms on S_{n-1}. Specifically, we have:*

$$\hat{f}_\lambda = \bigoplus_{\lambda^-} \left[\hat{f} \downarrow^n_{n-1}\right]_{\lambda^-},$$

where $f \downarrow^n_{n-1}$ is defined to be the restriction of f to S_{n-1}.

Consider the Fourier transform of the indicator function of $S_k \subset S_n$:

$$\delta_{S_k}(\sigma) = \begin{cases} 1 & \text{if } \sigma(j) = j \text{ for } j \in \{k+1, \dots, n\} \\ 0 & \text{otherwise} \end{cases}.$$

We now apply the branching rule $n-k$ times to the indicator function δ_{S_k}. Since δ_{S_k} is supported on $S_k \subset S_n$, the Fourier transform of δ_{S_k} at the irreducible ρ_λ can be written as a direct sum of Fourier coefficient matrices at the irreducibles which appear in the $n-k$th level of the branching tree corresponding to λ.

$$\left[\hat{\delta}_{S_k}\right]_\lambda = \bigoplus_{\mu \in \mathcal{T}^\lambda_{n-k}} \left[\hat{\delta}_{S_k} \downarrow^n_k\right]_\mu$$

Furthermore, since the restriction of δ_{S_k} to the subgroup S_k is a constant function, we see that all of the nontrivial irreducible summands are zero (since the Fourier transform of a constant function is zero at all nontrivial terms) and that the trivial terms are exactly $k!$. Because the trivial representation is one-dimensional, only a subset of the diagonal elements of $\left[\hat{\delta}_{S_k}\right]_\lambda$ can be nonzero.

Algorithm 5: Pseudocode for computing the Fourier transform of the indicator function of $S_k \subset S_n$ at the partition λ.

```
$S_k$-INDICATOR
input  : $k, n, \lambda$ (a partition of $n$)
output: $\left[\widehat{\delta_{S_k}}\right]_\lambda$
1  $\left[\widehat{\delta_{S_k}}\right]_\lambda \leftarrow \mathbf{0}_{d_\lambda \times d_\lambda}$ ;
2  foreach standard tableaux $t$ of shape $\lambda$ do
3      if $t \downarrow^n_k = \boxed{1}\boxed{2}\boxed{3}\boxed{\cdots}\boxed{k}$ then
4          $\left[\widehat{\delta_{S_k}}\right]_\lambda (t,t) \leftarrow k!$;
```

Algorithmically we can construct the Fourier transform of δ_{S_k} at λ by enumerating all of the branching sequences for λ and setting the (j, j) diagonal element of $\left[\hat{\delta}_{S_k}\right]_\lambda$ to be $k!$ if the corresponding jth branching sequence contains the partition (k). Alternatively, we can state the procedure in terms of standard tableaux. First, we define a restriction operation on a standard tableau t.

Definition 24 *Given a standard tableau t with n boxes and a positive integer $k < n$, we define the* restriction *of t to S_k (denoted by $t \downarrow^n_k$) to be the standard tableau t after removing boxes containing labels $k+1, \ldots, n$.*

To construct the Fourier transform of δ_{S_k} at λ, we iterate through the standard tableaux of shape λ, and set the (j, j) diagonal element of $\left[\hat{\delta}_{S_k}\right]_\lambda$ to be $k!$ if the restriction of the jth tableau to S_k, $t_j \downarrow^n_k$, takes the form $\boxed{1}\boxed{2}\boxed{3}\boxed{\cdots}\boxed{k}$. See Algorithm 5.

Example 20 *We compute $\left[\hat{\delta}_{S_2}\right]_{(3,2)}$ as an example. The branching sequences for $\lambda = (3,2)$ are:*

$$\begin{array}{|c|c|c|}\hline 1&3&5\\\hline 2&4\\\cline{1-2}\end{array} \longleftrightarrow [(1) \to (1,1) \to (2,1) \to (2,2) \to (3,2)],$$

$$\begin{array}{|c|c|c|}\hline 1&2&5\\\hline 3&4\\\cline{1-2}\end{array} \longleftrightarrow [(1) \to (\mathbf{2}) \to (2,1) \to (2,2) \to (3,2)],$$

$$\begin{array}{|c|c|c|}\hline 1&3&4\\\hline 2&5\\\cline{1-2}\end{array} \longleftrightarrow [(1) \to (1,1) \to (2,1) \to (3,1) \to (3,2)],$$

$$\begin{array}{|c|c|c|}\hline 1&2&4\\\hline 3&5\\\cline{1-2}\end{array} \longleftrightarrow [(1) \to (\mathbf{2}) \to (2,1) \to (3,1) \to (3,2)],$$

$$\begin{array}{|c|c|c|}\hline 1&2&3\\\hline 4&5\\\cline{1-2}\end{array} \longleftrightarrow [(1) \to (\mathbf{2}) \to (3) \to (3,1) \to (3,2)].$$

Since there are only three sequences which contain the partition (2), only those three basis elements have nonzero entries. And finally, noting that the appropriate normalization constant here *is simply* $|S_2| = 2! = 2$*, we see that:*

$$\left[\hat{\delta}_{S_2}\right]_{(3,2)} = \left[\begin{array}{c|ccccc} & \begin{array}{|c|c|c|}\hline 1&3&5\\ \hline 2&4\\ \cline{1-2}\end{array} & \begin{array}{|c|c|c|}\hline \mathbf{1}&\mathbf{2}&5\\ \hline 3&4\\ \cline{1-2}\end{array} & \begin{array}{|c|c|c|}\hline 1&3&4\\ \hline 2&5\\ \cline{1-2}\end{array} & \begin{array}{|c|c|c|}\hline \mathbf{1}&\mathbf{2}&4\\ \hline 3&5\\ \cline{1-2}\end{array} & \begin{array}{|c|c|c|}\hline \mathbf{1}&\mathbf{2}&3\\ \hline 4&5\\ \cline{1-2}\end{array} \\ \hline \begin{array}{|c|c|c|}\hline 1&3&5\\ \hline 2&4\\ \cline{1-2}\end{array} & 0 & 0 & 0 & 0 & 0 \\ \begin{array}{|c|c|c|}\hline \mathbf{1}&\mathbf{2}&5\\ \hline 3&4\\ \cline{1-2}\end{array} & 0 & 2 & 0 & 0 & 0 \\ \begin{array}{|c|c|c|}\hline 1&3&4\\ \hline 2&5\\ \cline{1-2}\end{array} & 0 & 0 & 0 & 0 & 0 \\ \begin{array}{|c|c|c|}\hline \mathbf{1}&\mathbf{2}&4\\ \hline 3&5\\ \cline{1-2}\end{array} & 0 & 0 & 0 & 2 & 0 \\ \begin{array}{|c|c|c|}\hline \mathbf{1}&\mathbf{2}&3\\ \hline 4&5\\ \cline{1-2}\end{array} & 0 & 0 & 0 & 0 & 2 \end{array}\right].$$

Our discussion has been focused on the indicator function δ_{S_k}, but computing $\delta_{S_{X,Y}}$ with $|X| = |Y| = n-k$ can be accomplished by first constructing the Fourier coefficient matrices for δ_{S_k}, then relabeling the tracks and identities using a change of basis. More precisely, suppose that, to achieve this relabeling, we must permute the X (identities) using a permutation π_1 and the Y (tracks) using π_2. The *Shift Theorem* (see Diaconis 1988) can be applied to reorder the Fourier coefficients according to these new labels.

Proposition 25 (Shift Theorem) *Given* $f : S_n \to \mathbb{R}$*, define* $f' : S_n \to \mathbb{R}$ *by* $f'(\sigma) = f(\pi_1\sigma\pi_2)$ *for some fixed* $\pi_1, \pi_2 \in S_n$*. The Fourier transforms of* f *and* f' *are related as:* $\widehat{f'}_\lambda = \rho_\lambda(\pi_1)\cdot\widehat{f}_\lambda\cdot\rho_\lambda(\pi_2)$.

We conclude with a comment on sparsity. It is clear from Algorithm 5 that the coefficient matrices of $\left[\hat{\delta}_{S_{X,Y}}\right]_\lambda$ are all, up to an appropriate relabeling of identities and tracks, diagonal matrices with at most $O(d_\lambda)$ nonzero entries. In fact, we can sometimes show that a given model has $O(1)$ nonzero entries.

Consider, for example, the indicator function $\delta_{S_{n-1}}$, corresponding to observations of the form ("Identity j is at track i"), which is nonzero only at the first two partitions, (n), and $(n-1,1)$. The zeroth-order term is, $\left[\hat{\delta}_{S_{n-1}}\right]_{(n)} = (n-1)!$. The first-order Fourier coefficient matrix, $\left[\hat{\delta}_{S_{n-1}}\right]_{(n-1,1)}$, is a matrix of all zeroes except for a single element on the diagonal, $\left[\hat{\delta}_{S_{n-1}}\right]_{(n-1,1)}(t,t)$, where $t = \begin{array}{|c|c|c|c|}\hline 1&2&3&\cdots\\ \hline n\\ \cline{1-1}\end{array}$, which takes on the value $(n-1)!$.

Appendix D. Decomposing the Tensor Product Representation

We now turn to the *Tensor Product Decomposition* problem, which is that of finding the irreducible components of the typically reducible tensor product representation. If ρ_λ and ρ_μ are irreducible representations of S_n, then there exists an intertwining operator $C_{\lambda\mu}$ such that:

$$C_{\lambda\mu}{}^{-1}\cdot(\rho_\lambda\otimes\rho_\mu(\sigma))\cdot C_{\lambda\mu} = \bigoplus_\nu\bigoplus_{\ell=1}^{z_{\lambda\mu\nu}}\rho_\nu(\sigma). \qquad (32)$$

In this section, we will present a set of numerical methods for computing the Clebsch-Gordan series ($z_{\lambda\mu\nu}$) and Clebsch-Gordan coefficients ($C_{\lambda\mu}$) for a pair of irreducible representations $\rho_\lambda \otimes \rho_\mu$. We begin by discussing two methods for computing the Clebsch-Gordan series. In the second section, we provide a general algorithm for computing the intertwining operators which relate two equivalent representations and discuss how it can be applied to computing the Clebsch-Gordan coefficients (Equation 32) and the matrices which relate marginal probabilities to irreducible Fourier coefficients (Equation 6). The results of Appendix D.1 are specific to the symmetric group, while the results of Appendix D.2 can be applied to arbitrary finite groups.

D.1 Computing the Clebsch-Gordan Series

We begin with a simple, well-known algorithm based on *group characters* for computing the Clebsch-Gordan series that turns out to be computationally intractable, but yields several illuminating theoretical results. See Serre (1977) for proofs of the theoretical results cited in this section.

One of the main results of representation theory was the discovery that there exists a relatively compact way of encoding any representation up to equivalence with a vector which we call the *character* of the representation. If ρ is a representation of a group G, then the character of the representation ρ, is defined simply to be the trace of the representation at each element $\sigma \in G$:

$$\chi_\rho(\sigma) = \mathrm{Tr}\left(\rho(\sigma)\right).$$

The reason characters have been so extensively studied is that they uniquely characterize a representation up to equivalence in the sense that two characters χ_{ρ_1} and χ_{ρ_2} are equal if and only if ρ_1 and ρ_2 are equivalent as representations. Even more surprising is that the space of possible group characters is orthogonally spanned by the characters of the irreducible representations. To make this precise, we first define an inner product on functions from G.

Definition 26 *Let ϕ, ψ be two real-valued functions on G. The* inner product *of ϕ and ψ is defined to be:*

$$\langle \phi, \psi \rangle \equiv \frac{1}{|G|} \sum_{\sigma \in G} \phi(\sigma)\psi(\sigma)$$

With respect to the above inner product, we have the following important result which allows us to test a given representation for irreducibility, and to test two irreducibles for equivalence.

Proposition 27 *Let χ_{ρ_1} and χ_{ρ_2} be characters corresponding to irreducible representations. Then*

$$\langle \chi_{\rho_1}, \chi_{\rho_2} \rangle = \begin{cases} 1 & \textit{if } \rho_1 \equiv \rho_2 \\ 0 & \textit{otherwise} \end{cases}.$$

Proposition 27 shows that the irreducible characters form an orthonormal set of functions. The next proposition says that the irreducible characters *span* the space of all possible characters.

Proposition 28 *Suppose ρ is any representation of G and which decomposes into irreducibles as:*

$$\rho \equiv \bigoplus_{\lambda} \bigoplus_{\ell=1}^{z_\lambda} \rho_\lambda,$$

where λ indexes over all irreducibles of G. Then:

1. *The character of* ρ *is a linear combination of irreducible characters* ($\chi_\rho = \sum_\lambda z_\lambda \chi_{\rho_\lambda}$),
2. *and the multiplicity of each irreducible,* z_λ, *can be recovered using* $\langle \chi_\rho, \chi_{\rho_\lambda} \rangle = z_\lambda$.

A simple way to decompose any group representation ρ, is given by Proposition 28, which says that we can take inner products of χ_ρ against the basis of irreducible characters to obtain the irreducible multiplicities z_λ. To treat the special case of finding the Clebsch-Gordan series, one observes that the character of the tensor product is simply the pointwise product of the characters of each tensor product factor.

Theorem 29 *Let* ρ_λ *and* ρ_μ *be irreducible representations with characters* χ_λ, χ_μ *respectively. Let* $z_{\lambda\mu\nu}$ *be the number of copies of* ρ_ν *in* $\rho_\lambda \otimes \rho_\mu$ *(hence, one term of the Clebsch-Gordan series). Then:*

1. *The character of the tensor product representation is given by:*
$$\chi_{\rho_\lambda \otimes \rho_\mu} = \chi_\lambda \cdot \chi_\mu = \sum_\nu z_{\lambda\mu\nu} \chi_\nu.$$
2. *The terms of the Clebsch-Gordan series can be computed using:*
$$z_{\lambda\mu\nu} = \frac{1}{|G|} \sum_{g \in G} \chi_\lambda(g) \cdot \chi_\mu(g) \cdot \chi_\nu(g),$$
and satisfy the following symmetry:
$$z_{\lambda\mu\nu} = z_{\lambda\nu\mu} = z_{\mu\lambda\nu} = z_{\mu\nu\lambda} = z_{\nu\lambda\mu} = z_{\nu\mu\lambda}. \tag{33}$$

Dot products for characters on the symmetric group can be done in $O(\#(n))$ time where $\#(n)$ is the number of partitions of the number n, instead of the naive $O(n!)$ time. In practice however, $\#(n)$ also grows too quickly for the character method to be tractable.

D.1.1 MURNAGHAN'S FORMULAS

A theorem by Murnaghan (1938) gives us a 'bound' on which representations can appear in the tensor product decomposition on S_n.

Theorem 30 *Let* ρ_1, ρ_2 *be the irreducibles corresponding to the partition* $(n-p, \lambda_2, \dots)$ *and* $(n-q, \mu_2, \dots)$ *respectively. Then the product* $\rho_1 \otimes \rho_2$ *does not contain any irreducibles corresponding to a partition whose first term is less than* $n-p-q$.

In view of the connection between the Clebsch-Gordan series and convolution of Fourier coefficients, Theorem 30 is analogous to the fact that for functions over the reals, the convolution of two compactly supported functions is also compactly supported.

We can use Theorem 30 to show that Kronecker conditioning is exact at certain irreducibles.
Proof [of Theorem 12] Let Λ denote the set of irreducibles at which our algorithm maintains Fourier coefficients. Since the errors in the prior come from setting coefficients outside of Λ to be zero, we see that Kronecker conditioning returns an approximate posterior which is exact at the irreducibles in
$$\Lambda_{EXACT} = \{\rho_\nu \,:\, z_{\lambda\mu\nu} = 0, \text{ where } \lambda \notin \Lambda \text{ and } \mu \trianglerighteq (n-q, \mu_2, \dots)\}.$$

Combining Theorem 30 with Equation 33: if $z_{\lambda\mu\nu} > 0$, with $\lambda = (n-p,\lambda_2,\lambda_3,\dots),\mu = (n-q,\mu_2,\mu_3,\dots)$ and $\nu = (n-r,\nu_2,\nu_3,\dots)$, then we have that: $r \leq p+q, p \leq q+r$, and $q \leq p+r$. In particular, it implies that $r \geq p-q$ and $r \geq q-p$, or more succinctly, $r \geq |p-q|$. Hence, if $\nu = (n-r,\nu_2,\dots)$, then $\rho_\nu \in \Lambda_{EXACT}$ whenever $r \leq |p-q|$, which proves the desired result. ■

The same paper (Murnaghan, 1938) derives several general Clebsch-Gordan series formulas for pairs of low-order irreducibles in terms of n, and in particular, derives the Clebsch-Gordan series for many of the Kronecker product pairs that one would likely encounter in practice. For example,

- $\rho_{(n-1,1)} \otimes \rho_{(n-1,1)} \equiv \rho_{(n)} \oplus \rho_{(n-1,1)} \oplus \rho_{(n-2,2)} \oplus \rho_{(n-2,1,1)}$
- $\rho_{(n-1,1)} \otimes \rho_{(n-2,2)} \equiv \rho_{(n-1,1)} \oplus \rho_{(n-2,2)} \oplus \rho_{(n-2,1,1)} \oplus \rho_{(n-3,3)} \oplus \rho_{(n-3,2,1)}$
- $\rho_{(n-1,1)} \otimes \rho_{(n-2,1,1)} \equiv \rho_{(n-1,1)} \oplus \rho_{(n-2,2)} \oplus \rho_{(n-2,1,1)} \oplus \rho_{(n-3,2,1)} \oplus \rho_{(n-3,1,1,1)}$
- $\rho_{(n-1,1)} \otimes \rho_{(n-3,3)} \equiv \rho_{(n-2,2)} \oplus \rho_{(n-3,3)} \oplus \rho_{(n-3,2,1)} \oplus \rho_{(n-4,4)} \oplus \rho_{(n-4,3,1)}$

D.2 Computing the Clebsch-Gordan Coefficients

In this section, we consider the general problem of finding an orthogonal operator which decomposes an arbitrary complex representation, $X(\sigma)$, of a finite group G.[15] Unlike the Clebsch-Gordan series which are basis-independent, intertwining operators must be recomputed if we change the underlying basis by which the irreducible representation matrices are constructed. However, for a fixed basis, we remind the reader that these intertwining operators need only be computed once and for all and can be stored in a table for future reference. Let X be any degree d group representation of G, and let Y be an equivalent direct sum of irreducibles, for example,

$$Y(\sigma) = \bigoplus_{\nu} \bigoplus_{\ell=1}^{z_\nu} \rho_\nu(\sigma), \tag{34}$$

where each irreducible ρ_ν has degree d_ν. We would like to compute an invertible (and orthogonal) operator C, such that $C \cdot X(\sigma) = Y(\sigma) \cdot C$, for all $\sigma \in G$. Throughout this section, we will assume that the multiplicities z_ν are known. To compute Clebsch-Gordan coefficients, for example, we would set $X = \rho_\lambda \otimes \rho_\mu$, and the multiplicities would be given by the Clebsch-Gordan series (Equation 32). To find the matrix which relates marginal probabilities to irreducible coefficients, we would set $X = \tau_\lambda$, and the multiplicities would be given by the Kostka numbers (Equation 6).

We will begin by describing an algorithm for computing a basis for the space of all possible intertwining operators which we denote by:

$$\mathrm{Int}_{[X;Y]} = \{C \in \mathbb{R}^{d\times d} \ : \ C \cdot X(\sigma) = Y(\sigma) \cdot C, \ \ \forall \sigma \in G\}.$$

We will then discuss some of the theoretical properties of $\mathrm{Int}_{[X;Y]}$ and show how to efficiently select an *orthogonal* element of $\mathrm{Int}_{[X;Y]}$.

15. Though the fundamental ideas in this section hold for a general finite group, we will continue to index irreducible by partitions and think of representations as being real-valued. To generalize the results, one can simply replace all transposes in this section by adjoints and think of η as indexing over the irreducibles of G rather than partitions.

Our approach is to naively[16] view the task of finding elements of $\mathrm{Int}_{[X;Y]}$ as a similarity matrix recovery problem, with the twist that the similarity matrix must be consistent over all group elements. To the best of our knowledge, the technique presented in this section is original. We first cast the problem of recovering a similarity matrix as a nullspace computation.

Proposition 31 *Let A,B,C be matrices and let $K_{AB} = I \otimes A - B^T \otimes I$. Then $AC = CB$ if and only if $vec(C) \in Nullspace(K_{AB})$.*

Proof A well known matrix identity (van Loan, 2000) states that if A,B,C are matrices, then $\mathrm{vec}(ABC) = \left(C^T \otimes A\right)\mathrm{vec}(B)$. Applying the identity to $AC = CB$, we have:

$$\mathrm{vec}(ACI) = \mathrm{vec}(ICB),$$

and after some manipulation:

$$\left(I \otimes A - B^T \otimes I\right)\mathrm{vec}(C) = 0,$$

showing that $\mathrm{vec}(C) \in \mathrm{Nullspace}(K_{AB})$. ■

For each $\sigma \in G$, the nullspace of the matrix $K(\sigma)$ constructed using the above proposition as:

$$K(\sigma) = I \otimes Y(\sigma) - X(\sigma) \otimes I, \tag{35}$$

where I is a $d \times d$ identity matrix, corresponds to the space of matrices C_σ such that

$$C_\sigma \cdot X(\sigma) = Y(\sigma) \cdot C, \qquad \text{for all } \sigma \in G.$$

To find the space of intertwining operators which are consistent across all group elements, we need to find the intersection:

$$\bigcap_{\sigma \in G} Nullspace(K(\sigma)).$$

At first glance, it may seem that computing the intersection might require examining $n!$ nullspaces if $G = S_n$, but as luck would have it, most of the nullspaces in the intersection are extraneous, as we now show.

Definition 32 *We say that a finite group G is* generated *by a set of* generators $S = \{g_1, \ldots, g_m\}$ *if every element of G can be written as a finite product of elements in S.*

For example, the following three sets are all generators for S_n:

- $\{(1,2),(1,3),\ldots,(1,n)\}$,
- $\{(1,2),(2,3),(3,4),\ldots,(n-1,n)\}$, and
- $\{(1,2),(1,2,3,\ldots,n)\}$.

To ensure a consistent similarity matrix for all group elements, we use the following proposition which says that it suffices to be consistent on any set of generators of the group.

16. In implementation, we use a more efficient algorithm for computing intertwining operators known as the *Eigenfunction Method* (EFM) (Chen, 1989). Unfortunately, the EFM is too complicated for us to describe in this paper. The method which we describe in this appendix is conceptually simpler than the EFM and generalizes easily to groups besides S_n.

Proposition 33 *Let X and Y be representations of finite group G and suppose that G is generated by the elements $\sigma_1, \ldots, \sigma_m$. If there exists an invertible linear operator C such that $C \cdot X(\sigma_i) = Y(\sigma_i) \cdot C$ for each $i \in \{1, \ldots, m\}$, then X and Y are equivalent as representations with C as the intertwining operator.*

Proof We just need to show that C is a similarity transform for any other element of G as well. Let π be any element of G and suppose π can be written as the following product of generators: $\pi = \prod_{i=1}^{n} \sigma_i$. It follows that:

$$\begin{aligned} C^{-1} \cdot Y(\pi) \cdot C &= C^{-1} \cdot Y\left(\prod_i \sigma_i\right) \cdot C = C^{-1} \cdot \left(\prod_i Y(\sigma_i)\right) \cdot C \\ &= (C^{-1} \cdot Y(\sigma_1) \cdot C)(C^{-1} \cdot Y(\sigma_2) \cdot C) \cdots (C^{-1} \cdot Y(\sigma_m) \cdot C) \\ &= \prod_i \left(C^{-1} \cdot Y(\sigma_i) \cdot C\right) = \prod_i X(\sigma_i) = X\left(\prod_i \sigma_i\right) = X(\pi). \end{aligned}$$

Since this holds for every $\pi \in G$, we have shown C to be an intertwining operator between the representations X and Y. ■

The good news is that despite having $n!$ elements, S_n can be generated by just two elements, namely, $(1,2)$ and $(1,2,\ldots,n)$, and so the problem reduces to solving for the intersection of two nullspaces, $(K(1,2) \cap K(1,2,\ldots,n))$, which can be done using standard numerical methods. Typically, the nullspace is multidimensional, showing that, for example, the Clebsch-Gordan coefficients for $\rho_\lambda \otimes \rho_\mu$ are not unique even up to scale.

Because $\mathrm{Int}_{[X;Y]}$ contains singular operators (the zero matrix is a member of $\mathrm{Int}_{[X;Y]}$, for example), not every element of $\mathrm{Int}_{[X;Y]}$ is actually a legitimate intertwining operator as we require invertibility. In practice, however, since the singular elements correspond to a measure zero subset of $\mathrm{Int}_{[X;Y]}$, one method for reliably selecting an operator from $\mathrm{Int}_{[X;Y]}$ that "works" is to simply select a random element from the nullspace to be C. It may, however, be desirable to have an *orthogonal* matrix C which works as an intertwining operator. In the following, we discuss an object called the *Commutant Algebra* which will lead to several insights about the space $\mathrm{Int}_{[X;Y]}$, and in particular, will lead to an algorithm for 'modifying' any invertible intertwining operator C to be an *orthogonal* matrix.

Definition 34 *The* Commutant Algebra *of a representation Y is defined to be the space of operators which commute with Y:*[17]

$$Com_Y = \{S \in \mathbb{R}^{d \times d} \ : \ S \cdot Y(\sigma) = Y(\sigma) \cdot S, \ \forall \sigma \in G\}.$$

The elements of the Commutant Algebra of Y can be shown to always take on a particular constrained form (shown using Schur's Lemma in Sagan 2001). In particular, every element of Com_Y takes the form

$$S = \bigoplus_\nu \left(M_{z_\nu} \otimes I_{d_\nu}\right), \tag{36}$$

where M_{z_ν} is some $z_\nu \times z_\nu$ matrix of coefficients and I_{d_ν} is the $d_\nu \times d_\nu$ identity (recall that the z_ν are the multiplicities from Equation 34). Moreover, it can be shown that every matrix of this form must necessarily be an element of the Commutant Algebra.

17. Notice that the definition of the Commutant Algebra does not involve the representation X.

The link between Com_Y and our problem is that the space of intertwining operators can be thought of as a 'translate' of the Commutant Algebra.

Lemma 35 *There exists a vector space isomorphism between $Int_{[X;Y]}$ and Com_Y.*

Proof Let R be any invertible element of $\text{Int}_{[X;Y]}$ and define the linear map $f : \text{Com}_Y \to \mathbb{R}^{d\times d}$ by: $f : S \mapsto (S\cdot R)$. We will show that the image of f is exactly the space of intertwining operators. Consider any element $\sigma \in G$:

$$\begin{aligned}(S\cdot R)\cdot X(\sigma)\cdot(S\cdot R)^{-1} &= S\cdot R\cdot X(\sigma)\cdot R^{-1}\cdot S^{-1},\\ &= S\cdot Y(\sigma)\cdot S^{-1} \quad (\text{since } R\in \text{Int}_{[X;Y]}),\\ &= Y(\sigma) \quad (\text{since } S\in\text{Com}_Y).\end{aligned}$$

We have shown that $S\cdot R\in \text{Int}_{[X;Y]}$, and since f is linear and invertible, we have that $\text{Int}_{[X;Y]}$ and Com_Y are isomorphic as vector spaces. ■

Using the lemma, we can see that the dimension of $\text{Int}_{[X;Y]}$ must be the same as the dimension of Com_Y, and therefore we have the following expression for the dimension of $\text{Int}_{[X;Y]}$.

Proposition 36

$$\dim Int_{[X;Y]} = \sum_\nu z_\nu^2.$$

Proof To compute the dimension of $\text{Int}_{[X;Y]}$, we need to compute the dimension of Com_Y, which can be accomplished simply by computing the number of free parameters in Equation 36. Each matrix M_{z_ν} is free and yields z_ν^2 parameters, and summing across all irreducibles ν yields the desired dimension. ■

To select an orthogonal intertwining operator, we will assume that we are given some invertible $R\in\text{Int}_{[X;Y]}$ which is not necessarily orthogonal (such as a random element of the nullspace of K , Equation 35). To find an orthogonal element, we will 'modify' R to be an orthogonal matrix by applying an appropriate rotation, such that $R\cdot R^T = I$. We begin with a simple observation about $R\cdot R^T$.

Lemma 37 *If both X and Y are orthogonal representations and R is an invertible member of $Int_{[X;Y]}$, then the matrix $R\cdot R^T$ is an element of Com_Y.*

Proof Consider a fixed $\sigma\in G$. Since $R\in\text{Int}_{[X;Y]}$, we have that:

$$X(\sigma) = R^{-1}\cdot Y(\sigma)\cdot R.$$

It is also true that:

$$X(\sigma^{-1}) = R^{-1}\cdot Y(\sigma^{-1})\cdot R. \tag{37}$$

Since $X(\sigma)$ and $Y(\sigma)$ are orthogonal matrices by assumption, Equation 37 becomes:

$$X^T(\sigma) = R^{-1}\cdot Y^T(\sigma)\cdot R.$$

Algorithm 6: Pseudocode for computing an orthogonal intertwining operators

```
INTXY
input : A degree $d$ orthogonal matrix representation $X$ evaluated at permutations $(1,2)$ and
        $(1,\dots,n)$, and the multiplicity $z_\nu$, of the irreducible $\rho_\nu$ in $X$
output: A matrix $C_\nu$ with orthogonal rows such that $C_\nu^T \cdot \oplus^{z_\nu}\rho_\nu \cdot C_\nu = X$
1 $K_1 \leftarrow I_{d\times d} \otimes (\oplus^{z_\nu}\rho_\nu(1,2)) - X(1,2) \otimes I_{d\times d}$;
2 $K_2 \leftarrow I_{d\times d} \otimes (\oplus^{z_\nu}\rho_\nu(1,\dots,n)) - X(1,\dots,n) \otimes I_{d\times d}$;
3 $K \leftarrow [K_1; K_2]$;   //Stack $K_1$ and $K_2$
4 $v \leftarrow \text{SparseNullspace}\big(K, z_\nu^2\big)$;   //Find the $d_\nu^2$-dimensional nullspace
5 $R \leftarrow \text{Reshape}(v; z_\nu d_\nu, d)$;   //Reshape $v$ into a $(z_\nu d_\nu)\times d$ matrix
6 $M \leftarrow \text{KroneckerFactors}(R\cdot R^T)$;   //Find $M$ such that $R\cdot R^T = M\otimes I_{d_\nu}$
7 $S_\nu \leftarrow \text{Eigenvectors}(M)$ ;
8 $C_\nu \leftarrow S_\nu^T \cdot R$ ;
9 NormalizeRows$(C_\nu)$;
```

Taking transposes,

$$X(\sigma) = R^T \cdot Y(\sigma)\cdot (R^{-1})^T.$$

We now multiply both sides on the left by R, and on the right by R^T,

$$\begin{aligned} R\cdot X(\sigma)\cdot R^T &= R\cdot R^T \cdot Y(\sigma)\cdot (R^{-1})^T\cdot R^T \\ &= R\cdot R^T\cdot Y(\sigma). \end{aligned}$$

Since $R \in \text{Int}_{[X;Y]}$,

$$Y(\sigma)\cdot R\cdot R^T = R\cdot R^T\cdot Y(\sigma),$$

which shows that $R\cdot R^T \in \text{Com}_Y$. ■

We can now state and prove our orthogonalization procedure, which works by diagonalizing the matrix $R\cdot R^T$. Due to its highly constrained form, the procedure is quite efficient.

Theorem 38 *Let X be any orthogonal group representation of G and Y an equivalent orthogonal irreducible decomposition (As in Equation 34). Then for any invertible element $R \in Int_{[X;Y]}$, there exists an (efficiently computable) orthogonal matrix T such that the matrix $T\cdot R$ is an element of $Int_{[X;Y]}$ and is* orthogonal.

Proof Lemma 37 and Equation 36 together imply that the matrix $R\cdot R^T$ can always be written in the form

$$R\cdot R^T = \oplus_\nu \left(M_{z_\nu}\otimes I_{d_\nu}\right)$$

Since $R\cdot R^T$ is symmetric, each of the matrices M_{z_ν} is also symmetric and must therefore possess an orthogonal basis of eigenvectors. Define the matrix S_{z_ν} to be the matrix whose columns are the eigenvectors of M_{z_ν}.

The matrix $S = \oplus_\nu(S_{z_\nu}\otimes I_{d_\nu})$ has the following two properties:

1. $(S^T \cdot R)(S^T \cdot R)^T$ is a diagonal matrix:

   Each column of S is an eigenvector of $R \cdot R^T$ by standard properties of the direct sum and Kronecker product. Since each of the matrices, S_{z_ν}, is orthogonal, the matrix S is also orthogonal. We have:

$$\begin{aligned}(S^T \cdot R)(S^T \cdot R)^T &= S^T \cdot R \cdot R^T \cdot S,\\ &= S^{-1} \cdot R \cdot R^T \cdot S,\\ &= D,\end{aligned}$$

   where D is a diagonal matrix of eigenvalues of $R \cdot R^T$.

2. $S^T \cdot R \in \mathrm{Int}_{[X;Y]}$:

   By Equation 36, a matrix is an element of Com_Y if and only if it takes the form $\oplus_\nu (S_{z_\nu} \otimes I_{d_\nu})$. Since S can be written in the required form, so can S^T. We see that $S^T \in \mathrm{Com}_Y$, and by the proof of Lemma 35, we see that $S^T \cdot R \in \mathrm{Int}_{[X;Y]}$.

Finally, setting $T = D^{1/2} \cdot S^T$ makes the matrix $T \cdot R$ orthogonal (and does not change the fact that $T \cdot R \in \mathrm{Int}_{[X;Y]}$). ■

We see that the complexity of computing T is dominated by the eigenspace decomposition of M_{z_ν}, which is $O\left(z_\nu^3\right)$. Pseudocode for computing orthogonal intertwining operators is given Algorithm 6.

References

Hamsa Balakrishnan, Inseok Hwang, and Claire Tomlin. Polynomial approximation algorithms for belief matrix maintenance in identity management. In *Proceedings of the 43rd IEEE Conference on Decision and Control, Bahamas*, 2004.

Yaakov Bar-Shalom and Thomas E. Fortmann. *Tracking and Data Association*. Academic Press, 1988.

John Bartholdi, Craig Tovey, and Michael Trick. Voting schemes for which it can be difficult to tell who won. *Social Choice and Welfare*, 6(2), 1989.

Xavier Boyen and Daphne Koller. Tractable inference for complex stochastic processes. In *UAI '98: Uncertainty in Artificial Intelligence*, 1998.

Jin-Quan Chen. *Group Representation Theory for Physicists*. World Scientific, 1989.

Michael Clausen and Ulrich Baum. Fast Fourier transforms for symmetric groups: Theory and implementation. *Mathematics of Computations*, 61(204):833–847, 1993.

Joseph Collins and Jeffrey Uhlmann. Efficient gating in data association with multivariate distributed states. *IEEE Transactions Aerospace and Electronic Systems*, 28, 1992.

James Cooley and John Tukey. An algorithm for the machine calculation of complex fourier series. *Mathematical Computation*, 19:297–301, 1965.

Ingemar Cox and Sunita Hingorani. An efficient implementation of Reid's multiple hypothesis tracking algorithm and its evaluation for the purpose of visual tracking. In *International Conference on Pattern Recognition*, pages 437–443, 1994.

Douglas E. Critchlow. *Metric Methods for Analyzing Partially Ranked Data*. Springer-Verlag, 1985.

Persi Diaconis. *Group Representations in Probability and Statistics*. Institute of Mathematical Statistics, 1988.

Persi Diaconis. A generalization of spectral analysis with application to ranked data. *The Annals of Statistics*, 17(3):949–979, 1989.

Michael Fligner and Joseph Verducci. Distance based ranking models. *Journal of the Royal Statistical Society*, 48, 1986.

Richard Foote, Gagan Mirchandani, and Dan Rockmore. Two-dimensional wreath product transforms. *Journal of Symbolic Computation*, 37(2):187–207, 2004.

Yoav Freund, Raj Iyer, Robert E. Schapire, and Yoram Singer. An efficient boosting algorithm for combining preferences. *Journal of Machine Learning Research (JMLR)*, 4:933–969, 2003. ISSN 1533-7928.

Ulf Grenander. *Probabilities on Algebraic Structures*. Wiley, 1963.

David P. Helmbold and Manfred K. Warmuth. Learning permutations with exponential weights. In *COLT '07: The Twentieth Annual Conference on Learning Theory*, 2007.

Jonathan Huang, Carlos Guestrin, and Leonidas Guibas. Efficient inference for distributions on permutations. In *NIPS '07: Advances in Neural Information Processing Systems*, Vancouver, Canada, December 2007.

Jonathan Huang, Carlos Guestrin, Xiaoye Jiang, and Leonidas Guibas. Exploiting probabilistic independence for permutations. In *AISTATS '09: Artificial Intelligence and Statistics*, Clearwater Beach, Florida, April 2009.

Gordon James and Adelbert Kerber. *The Representation Theory of the Symmetric Group*. Addison-Wesley, 1981.

Thorsten Joachims. Optimizing search engines using clickthrough data. In *KDD '02: Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 133–142, New York, NY, USA, 2002. ACM.

Risi Kondor. $\mathbb{S}_n$`ob`: a C++ library for fast Fourier transforms on the symmetric group, 2006. Available at `http://www.cs.columbia.edu/~risi/Snob/`.

Risi Kondor and Karsten M. Borgwardt. The skew spectrum of graphs. In *ICML '08: Proceedings of the 25th International Conference on Machine Learning*, pages 496–503, 2008.

Risi Kondor, Andrew Howard, and Tony Jebara. Multi-object tracking with representations of the symmetric group. In *AISTATS '07: Artificial Intelligence and Statistics*, 2007.

Ka-Lam Kueh, Timothy Olson, Dan Rockmore, and Ki-Seng Tan. Nonlinear approximation theory on finite groups. Technical Report PMA-TR99-191, Department of Mathematics, Dartmouth College, 1999.

Serge Lang. *Algebra*. Addison-Wesley, 1965.

Guy Lebanon and Yi Mao. Non-parametric modeling of partially ranked data. In John C. Platt, Daphne Koller, Yoram Singer, and Sam Roweis, editors, *NIPS '07: Advances in Neural Information Processing Systems*, pages 857–864, Cambridge, MA, 2008. MIT Press.

Colin Mallows. Non-null ranking models. *Biometrika*, 44, 1957.

David Maslen. The efficient computation of Fourier transforms on the symmetric group. *Mathematics of Computation*, 67:1121–1147, 1998.

Marina Meila, Kapil Phadnis, Arthur Patterson, and Jeff Bilmes. Consensus ranking under the exponential model. Technical Report 515, University of Washington, Statistics Department, April 2007.

Francis Murnaghan. The analysis of the kronecker product of irreducible representations of the symmetric group. *American Journal of Mathematics*, 60(3):761–784, 1938.

Katta G. Murty. An algorithm for ranking all the assignments in order of increasing cost. *Operations Research*, 16:682–687, 1968.

Songhwai Oh and Shankar Sastry. A polynomial-time approximation algorithm for joint probabilistic data association. In *Proceedings of the American Control Conference, Portland, OR*, 2005.

Songhwai Oh, Stuart Russell, and Shankar Sastry. Markov chain Monte Carlo data association for general multiple-target tracking problems. In *Proceedings of the IEEE International Conference on Decision and Control, Paradise Island, Bahamas*, 2004.

Aubrey B. Poore. Multidimensional assignment and multitarget tracking. In *Partitioning Data Sets*, volume 19, pages 169–196. DIMACS Series in Discrete Mathematics and Theoretical Computer Science, 1995.

Lawrence Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

Donald Reid. An algorithm for tracking multiple targets. *IEEE Transactions on Automatic Control*, 6:843–854, 1979.

Daniel N. Rockmore. The FFT: An algorithm the whole family can use. *Computing in Science and Engineering*, 02(1):60–64, 2000.

Bruce E. Sagan. *The Symmetric Group*. Springer, April 2001. ISBN 0387950672.

Brad Schumitsch, Sebastian Thrun, Gary Bradski, and Kunle Olukotun. The information-form data association filter. In *NIPS '05: Advances in Neural Information Processing Systems*, Cambridge, MA, 2005. MIT Press.

Brad Schumitsch, Sebastian Thrun, Leonidas Guibas, and Kunle Olukotun. The identity management Kalman filter (imkf). In *RSS '06: Proceedings of Robotics: Science and Systems*, Philadelphia, PA, USA, August 2006.

Jean-Pierre Serre. *Linear Representations of Finite Groups*. Springer-Verlag, 1977.

Jaewon Shin, Leonidas Guibas, and Feng Zhao. A distributed algorithm for managing multi-target identities in wireless ad-hoc sensor networks. In *IPSN '03: Information Processing in Sensor Networks*, 2003.

Jaewon Shin, Nelson Lee, Sebastian Thrun, and Leonidas Guibas. Lazy inference on object identities in wireless sensor networks. In *IPSN '05: Information Processing in Sensor Networks*, 2005.

Michael Taylor, John Guiver, Stephen Robertson, and Tom Minka. Softrank: optimizing nonsmooth rank metrics. In *WSDM '08: Proceedings of the international conference on Web search and web data mining*, pages 77–86, New York, NY, USA, 2008. ACM.

Audrey Terras. *Fourier Analysis on Finite Groups and Applications*. London Mathematical Society, 1999.

Jack van Lint and Richard M. Wilson. *A Course in Combinatorics*. Cambridge University Press, 2001.

Charles F. van Loan. The ubiquitous kronecker product. *Journal of Computational and Applied Mathematics*, 123(1-2):85–100, 2000. ISSN 0377-0427.

Anatoly Vershik and Andrei Okounkov. A new approach to the representation theory of symmetric groups. ii. *Journal of Mathematical Sciences*, 131(2):5471–5494, 2006.

Alan Willsky. On the algebraic structure of certain partially observable finite-state markov processes. *Information and Control*, 38:179–212, 1978.

Journal of Machine Learning Research 10 (2009) 1071-1094 Submitted 12/06; Revised 5/08; Published 5/09

An Algorithm for Reading Dependencies from the Minimal Undirected Independence Map of a Graphoid that Satisfies Weak Transitivity

Jose M. Peña JOSPE@IDA.LIU.SE
Department of Computer and Information Science
Linköping University
SE-58183 Linköping, Sweden

Roland Nilsson NILSSON@CHGR.MGH.HARVARD.EDU
Department of Systems Biology
Harvard Medical School
Boston, MA 02115, USA

Johan Björkegren JOHAN.BJORKEGREN@KI.SE
Department of Medicine
Karolinska Institutet
SE-17176 Stockholm, Sweden

Jesper Tegnér[*] JESPERT@IFM.LIU.SE
Department of Physics, Chemistry and Biology
Linköping University
SE-58183 Linköping, Sweden

Editor: Marina Meila

Abstract

We present a sound and complete graphical criterion for reading dependencies from the minimal undirected independence map G of a graphoid M that satisfies weak transitivity. Here, complete means that it is able to read all the dependencies in M that can be derived by applying the graphoid properties and weak transitivity to the dependencies used in the construction of G and the independencies obtained from G by vertex separation. We argue that assuming weak transitivity is not too restrictive. As an intermediate step in the derivation of the graphical criterion, we prove that for any undirected graph G there exists a strictly positive discrete probability distribution with the prescribed sample spaces that is faithful to G. We also report an algorithm that implements the graphical criterion and whose running time is considered to be at most $O(n^2(e+n))$ for n nodes and e edges. Finally, we illustrate how the graphical criterion can be used within bioinformatics to identify biologically meaningful gene dependencies.

Keywords: graphical models, vertex separation, graphoids, weak transitivity, bioinformatics

1. Introduction

A minimal undirected independence map G of an independence model M is used to read independencies that hold in M. Sometimes, however, G can also be used to read dependencies holding in M. For instance, if M is a graphoid that is faithful to G then, by definition, lack of vertex separation is a sound and complete graphical criterion for reading dependencies from G, where complete means

*. Also at the Department of Medicine, Karolinska Institutet, SE-17176 Stockholm, Sweden.

that it is able to read all the dependencies in M. If M is simply a graphoid, then Bouckaert (1995) proposes a sound and complete graphical criterion for reading dependencies from G. In this case, complete means that it is able to read all the dependencies in M that can be derived by applying the graphoid properties to the dependencies used in the construction of G and the independencies obtained from G by vertex separation.

In this paper, we introduce a sound and complete graphical criterion for reading dependencies from G under the assumption that M is a graphoid that satisfies weak transitivity. Here, complete means that it is able to read all the dependencies in M that can be derived by applying the graphoid properties and weak transitivity to the dependencies used in the construction of G and the independencies obtained from G by vertex separation. Our criterion allows reading more dependencies than the criterion in Bouckaert (1995) when the graphoid at hand satisfies weak transitivity. We show that there exist important families of probability distributions that are graphoids and satisfy weak transitivity. These include, for instance, the regular Gaussian probability distributions.

We think that the work presented in this paper can be of great interest for the machine learning community. Graphs are one of the most commonly used metaphors for representing knowledge because they appeal to human intuition (Pearl, 1988). Furthermore, graphs are parsimonious models because they trade off accuracy for simplicity. Consider, for instance, representing the independence model induced by a probability distribution as a graph. Though this graph is typically less accurate than the probability distribution (the graph may not represent all the (in)dependencies and those that are represented are not quantified), it also requires less space to be stored and less time to be communicated than the probability distribution, which may be desirable features in some applications. Thus, it seems sensible developing tools for reasoning with graphs. Our criterion is one such a tool: As vertex separation makes the discovery of independencies amenable to human reasoning by enabling to read independencies off G without numerical calculation (Pearl, 1988), so does our criterion with respect to the discovery of dependencies. There are fields where discovering dependencies is more important than discovering independencies. It is in these fields where we believe that our criterion has greater potential. In bioinformatics, for instance, the nodes of G represent (the expression levels of) some genes under study. Bioinformaticians are typically more interested in discovering gene dependencies than independencies, because the former provide contexts in which the expression level of some genes is informative about that of some other genes, which may lead to hypothesize dependencies, functional relations, causal relations, the effects of manipulation experiments, etc. As we will illustrate at the end of the paper, our criterion can be very helpful in such a scenario. Our criterion also clarifies a misconception that may exist among some bioinformaticians, namely that two genes are dependent if there exists a path in G between them. We will see that there must exist exactly one path to draw such a conclusion. Hence, the importance of developing a formal criterion like ours to prevent drawing erroneous conclusions. Of course, the conclusions drawn by our criterion may be misleading if G is learnt from a sample, which is most likely the case in bioinformatics. However, this has nothing to do with the correctness of our criterion, which we prove in subsequent sections, but with the fact that G is an estimate.

The rest of the paper is organized as follows. We start by reviewing some concepts in Section 2. We show in Section 3 that assuming weak transitivity is not too restrictive. We prove in Section 4 that for any undirected graph G there exists a strictly positive discrete probability distribution with the prescribed sample spaces that is faithful to G. This is an important result in itself as well as for proving the completeness of the graphical criterion that we present in Section 5. An algorithmic implementation of the graphical criterion is described in Section 6. We illustrate in Section 7 how the graphical criterion works in practice with a real world example taken from bioinformatics. Finally, we close with some discussion in Section 8.

2. Preliminaries

The definitions and results in this section are taken from Lauritzen (1996), Pearl (1988) and Studený (2005). We use the juxtaposition $\mathbf{XY}$ to denote $\mathbf{X} \cup \mathbf{Y}$, and X to denote the singleton $\{X\}$. We use upper-case letters to denote random variables and the same letters in lower-case to denote their states. We use $Val(\mathbf{X})$ to denote the set of possible states of a random variable $\mathbf{X}$. Let $\mathbf{U}$ denote a set of random variables. Unless otherwise stated, all the probability distributions, independence models and graphs in this paper are defined over $\mathbf{U}$. Let $\mathbf{X}, \mathbf{Y}, \mathbf{Z}$ and $\mathbf{W}$ denote four mutually disjoint subsets of $\mathbf{U}$. An independence model M is a set of independencies of the form $\mathbf{X}$ is independent of $\mathbf{Y}$ given $\mathbf{Z}$. We represent that an independence is in M by $\mathbf{X} \perp\!\!\!\perp \mathbf{Y} | \mathbf{Z}$ and that an independence is not in M by $\mathbf{X} \not\perp\!\!\!\perp \mathbf{Y} | \mathbf{Z}$. In the latter case, we may equivalently say that the dependence $\mathbf{X} \not\perp\!\!\!\perp \mathbf{Y} | \mathbf{Z}$ is in M. An independence model is a graphoid when it satisfies the following five properties:

- Symmetry $\mathbf{X} \perp\!\!\!\perp \mathbf{Y} | \mathbf{Z} \Rightarrow \mathbf{Y} \perp\!\!\!\perp \mathbf{X} | \mathbf{Z}$.
- Decomposition $\mathbf{X} \perp\!\!\!\perp \mathbf{YW} | \mathbf{Z} \Rightarrow \mathbf{X} \perp\!\!\!\perp \mathbf{Y} | \mathbf{Z}$.
- Weak union $\mathbf{X} \perp\!\!\!\perp \mathbf{YW} | \mathbf{Z} \Rightarrow \mathbf{X} \perp\!\!\!\perp \mathbf{Y} | \mathbf{ZW}$.
- Contraction $\mathbf{X} \perp\!\!\!\perp \mathbf{Y} | \mathbf{ZW} \wedge \mathbf{X} \perp\!\!\!\perp \mathbf{W} | \mathbf{Z} \Rightarrow \mathbf{X} \perp\!\!\!\perp \mathbf{YW} | \mathbf{Z}$.
- Intersection $\mathbf{X} \perp\!\!\!\perp \mathbf{Y} | \mathbf{ZW} \wedge \mathbf{X} \perp\!\!\!\perp \mathbf{W} | \mathbf{ZY} \Rightarrow \mathbf{X} \perp\!\!\!\perp \mathbf{YW} | \mathbf{Z}$.

The independence model induced by any strictly positive probability distribution is a graphoid. Hereinafter, for the sake of simplicity, we do not make any distinction between a probability distribution and the independence model induced by it and, thus, we always refer to the former. For instance, instead of saying that the independence model of a probability distribution p is a graphoid, we simply say that p is a graphoid. In this paper, we pay particular attention to strictly positive discrete probability distributions and regular Gaussian probability distributions, that is, those whose covariance matrices are positive definite. For the strictly positive discrete probability distributions, we assume that each random variable in $\mathbf{U}$ has a finite sample space with at least two possible states. Note that if p is a strictly positive discrete probability distribution, then $p(\mathbf{X})$ and $p(\mathbf{X}|\mathbf{Y}=\mathbf{y})$ are uniquely defined, strictly positive and discrete. Likewise, if p is a regular Gaussian probability distribution, then $p(\mathbf{X})$ and $p(\mathbf{X}|\mathbf{Y}=\mathbf{y})$ are uniquely determined (by a convention) and regular Gaussian. Moreover, any regular Gaussian probability distribution satisfies composition $\mathbf{X} \perp\!\!\!\perp \mathbf{Y} | \mathbf{Z} \wedge \mathbf{X} \perp\!\!\!\perp \mathbf{W} | \mathbf{Z} \Rightarrow \mathbf{X} \perp\!\!\!\perp \mathbf{YW} | \mathbf{Z}$.

A path between X_1 and X_n in a graph G is a sequence of distinct nodes $X_1, \ldots, X_n$ $(1 \leq n)$ such that there exists an edge in G between every two consecutive nodes in the sequence. Given a path $X_1, \ldots, X_n$ in a directed and acyclic graph (DAG) G, a node X_i $(1 < i < n)$ is a collider in the path if $X_{i-1} \rightarrow X_i \leftarrow X_{i+1}$ in G. Let $sep(\mathbf{X}, \mathbf{Y}|\mathbf{Z})$ denote that $\mathbf{X}$ is separated from $\mathbf{Y}$ given $\mathbf{Z}$ in a graph G. Specifically, $sep(\mathbf{X}, \mathbf{Y}|\mathbf{Z})$ holds when every path in G between $\mathbf{X}$ and $\mathbf{Y}$ is blocked by $\mathbf{Z}$. If G is an undirected graph (UG), then a path in G between $\mathbf{X}$ and $\mathbf{Y}$ is blocked by $\mathbf{Z}$ when there exists some $Z \in \mathbf{Z}$ in the path. If G is a DAG, then a path in G between $\mathbf{X}$ and $\mathbf{Y}$ is blocked by $\mathbf{Z}$ when there exists a node Z in the path such that

- either Z is not a collider in the path and $Z \in \mathbf{Z}$, or
- Z is a collider in the path and neither Z nor any of its descendants in G is in $\mathbf{Z}$.

An independence model M is faithful to an UG or DAG G when $\mathbf{X} \perp\!\!\!\perp \mathbf{Y} | \mathbf{Z}$ iff $sep(\mathbf{X}, \mathbf{Y} | \mathbf{Z})$. Any independence model that is faithful to some UG or DAG is a graphoid. An UG (resp. DAG) G is an undirected (resp. directed) independence map of an independence model M when $\mathbf{X} \perp\!\!\!\perp \mathbf{Y} | \mathbf{Z}$ if $sep(\mathbf{X}, \mathbf{Y} | \mathbf{Z})$. Moreover, an UG G is a minimal undirected independence (MUI) map of M when

(i) G is an undirected independence map of M, and

(ii) no proper subgraph of G satisfies (i).

A Markov boundary of $X \in \mathbf{U}$ in an independence model M is any subset $MB(X)$ of $\mathbf{U} \setminus X$ such that

(i) $X \perp\!\!\!\perp \mathbf{U} \setminus X \setminus MB(X) | MB(X)$, and

(ii) no proper subset of $MB(X)$ satisfies (i).

If M is a graphoid, then

(i) $MB(X)$ is unique for each $X \in \mathbf{U}$,

(ii) the MUI map G of M is unique, and

(iii) two nodes X and Y are adjacent in G iff $X \in MB(Y)$ iff $Y \in MB(X)$ iff $X \not\perp\!\!\!\perp Y | \mathbf{U} \setminus (XY)$.

A Bayesian network (BN) is a pair (G, p) where G is a DAG and p is a discrete probability distribution that factorizes as $p = \prod_{X \in \mathbf{U}} q(X | Pa(X))$, where $q(X | Pa(X))$ denotes a conditional discrete probability distribution of X given the parents of X in G, $Pa(X)$. Recall that a node Y is called a parent of X if $Y \to X$ is in G. We denote by $\mathcal{D}(G)^+$ all the strictly positive discrete probability distributions that can be represented by a BN with DAG G, that is, those that factorize according to G as indicated.

3. Weak Transitivity Graphoids

Let $\mathbf{X}$, $\mathbf{Y}$ and $\mathbf{Z}$ denote three mutually disjoint subsets of $\mathbf{U}$. We define a weak transitivity (WT) graphoid as a graphoid that satisfies weak transitivity $\mathbf{X} \perp\!\!\!\perp \mathbf{Y} | \mathbf{Z} \wedge \mathbf{X} \perp\!\!\!\perp \mathbf{Y} | \mathbf{Z}V \Rightarrow \mathbf{X} \perp\!\!\!\perp V | \mathbf{Z} \vee V \perp\!\!\!\perp \mathbf{Y} | \mathbf{Z}$ with $V \in \mathbf{U} \setminus (\mathbf{XYZ})$. There exist important families of probability distributions that are WT graphoids and, thus, we believe that WT graphoids are worth studying: For instance, any probability distribution that is regular Gaussian or faithful to some UG or DAG is a WT graphoid (Pearl, 1988; Studený, 2005). Other interesting families of probability distributions that are WT graphoids are presented in this section.

We say that a strictly positive discrete probability distribution p has context-specific dependencies if there exists some $\mathbf{W} \subseteq \mathbf{U}$ such that $p(\mathbf{U} \setminus \mathbf{W} | \mathbf{W} = \mathbf{w})$ does not have the same (in)dependencies for all $\mathbf{w}$. The theorem below proves that for any DAG G, in a measure-theoretic sense (Halmos, 1966), almost all the probability distributions in $\mathcal{D}(G)^+$ have no context-specific dependencies. This result is not only relative to the DAG G, but also to the measure considered, the

Lebesgue measure in our case, as well as to the dimension of $\mathcal{D}(G)^+$. For this purpose, we discuss first how we parameterize the probability distributions in $\mathcal{D}(G)^+$. Since each probability distribution p in $\mathcal{D}(G)^+$ factorizes as $p = \prod_{X \in \mathbf{U}} q(X|Pa(X))$, p can be parameterized by parameterizing each probability table $q(X|Pa(X))$. Let $Val(X) = \{x_1, \ldots, x_{n(X)}\}$ denote the $n(X)$ possible states of the random variable X. Let $\theta_{x_i,\mathbf{pa}}$ $(1 \leq i \leq n(X))$ denote the parameter corresponding to $q(X = x_i|Pa(X) = \mathbf{pa})$ with $\mathbf{pa} \in Val(Pa(X))$. Note that the parameters are linearly dependent because $\sum_{i=1}^{n(X)} q(X = x_i|Pa(X) = \mathbf{pa}) = 1$. In order to introduce properly the parameter space for $\mathcal{D}(G)^+$ and the Lebesgue measure on it, we make the convention that $\theta_{x_{n(X)},\mathbf{pa}}$ is linearly dependent on the remaining parameters $\theta_{x_i,\mathbf{pa}}$ with $1 \leq i < n(X)$. Therefore, the number of linearly independent parameters for p is $n = \sum_{X \in \mathbf{U}}(n(X) - 1)(\prod_{Y \in Pa(X)} n(Y))$. Let Δ_d denote the simplex $\{(s_1, \ldots, s_d) \in \mathbb{R}^d : s_i \geq 0 \; (1 \leq i \leq d), \sum_{i=1}^d s_i \leq 1\}$. The Lebesgue measure of Δ_d wrt $\mathbb{R}^d$ is $1/d!$ (Stein, 1966). Let Δ_d^+ denote the set $\{(s_1, \ldots, s_d) \in \mathbb{R}^d : s_i > 0 \; (1 \leq i \leq d), \sum_{i=1}^d s_i < 1\}$. The Lebesgue measure of Δ_d^+ wrt $\mathbb{R}^d$ is also $1/d!$, because the difference between Δ_d and Δ_d^+ has Lebesgue measure zero. Then, the parameter space for $\mathcal{D}(G)^+$ is $\times_{X \in \mathbf{U}} \times_{\mathbf{pa} \in Val(Pa(X))} \Delta^+_{n(X)-1}$, whose Lebesgue measure wrt $\mathbb{R}^n$ is $\prod_{X \in \mathbf{U}} \prod_{\mathbf{pa} \in Val(Pa(X))} 1/(n(X) - 1)!$.

Theorem 1 *Let G be a DAG. $\mathcal{D}(G)^+$ has non-zero Lebesgue measure wrt $\mathbb{R}^n$, where n is the number of linearly independent parameters in the parametrization of the probability distributions in $\mathcal{D}(G)^+$ described above. The probability distributions in $\mathcal{D}(G)^+$ that are not faithful to G or have context-specific dependencies have zero Lebesgue measure wrt $\mathbb{R}^n$.*

Proof First, we prove that there is a one-to-one correspondence between the elements of the parameter space for $\mathcal{D}(G)^+$ and the probability distributions in $\mathcal{D}(G)^+$. This will allow us later to compute the Lebesgue measure wrt $\mathbb{R}^n$ of a subset of $\mathcal{D}(G)^+$ as the Lebesgue measure wrt $\mathbb{R}^n$ of the corresponding subset of the parameter space for $\mathcal{D}(G)^+$. Obviously, different probability distributions in $\mathcal{D}(G)^+$ must correspond to different elements of the parameter space for $\mathcal{D}(G)^+$. On the other hand, different elements of the parameter space for $\mathcal{D}(G)^+$ correspond to different probability distributions in $\mathcal{D}(G)^+$, because the values of the parameters $\theta_{x_i,\mathbf{pa}}$ defining a probability distribution p coincide with the values of the conditional probabilities $p(X = x_i|Pa(X) = \mathbf{pa})$ that are computed from p (Pearl, 1988).

To prove the first statement in the theorem it suffices to note that, as discussed right before the theorem, the parameter space for $\mathcal{D}(G)^+$ has non-zero Lebesgue measure wrt $\mathbb{R}^n$. This implies that $\mathcal{D}(G)^+$ also has non-zero Lebesgue measure wrt $\mathbb{R}^n$ because, as proven above, there is a one-to-one correspondence between the elements of the parameter space for $\mathcal{D}(G)^+$ and the probability distributions in $\mathcal{D}(G)^+$.

To prove the second statement in the theorem, we first prove that the probability distributions in $\mathcal{D}(G)^+$ that have context-specific dependencies have zero Lebesgue measure wrt $\mathbb{R}^n$. The proof basically proceeds in the same way as that of Theorem 7 in Meek (1995). We start by showing that for a probability distribution in $\mathcal{D}(G)^+$ to have context-specific dependencies, some polynomials in the parameters in the parametrization of the probability distributions in $\mathcal{D}(G)^+$ must be satisfied. Specifically, these polynomials are real polynomials in real variables that we interpret as real functions on a real Euclidean space that includes the parameter space for $\mathcal{D}(G)^+$. Let $\mathbf{W} \subseteq \mathbf{U}$ and let $\mathbf{X}$, $\mathbf{Y}$ and $\mathbf{Z}$ denote three disjoint subsets of $\mathbf{U} \setminus \mathbf{W}$. Since G is a directed independence map of any probability distribution $p \in \mathcal{D}(G)^+$ (Neapolitan, 2003), for a constraint such as $\mathbf{X} \perp\!\!\!\perp \mathbf{Y}|\mathbf{Z}$ to be true in $p(\mathbf{U} \setminus \mathbf{W}|\mathbf{W} = \mathbf{w})$ but false in $p(\mathbf{U} \setminus \mathbf{W}|\mathbf{W} = \mathbf{w}')$, two conditions must be met. First, $sep(\mathbf{X}, \mathbf{Y}|\mathbf{Z}\mathbf{W})$

must not hold in G and, second, the following equations must be satisfied:

$$p(\mathbf{X}=\mathbf{x},\mathbf{Y}=\mathbf{y},\mathbf{Z}=\mathbf{z},\mathbf{W}=\mathbf{w})p(\mathbf{Z}=\mathbf{z},\mathbf{W}=\mathbf{w})-$$

$$p(\mathbf{X}=\mathbf{x},\mathbf{Z}=\mathbf{z},\mathbf{W}=\mathbf{w})p(\mathbf{Y}=\mathbf{y},\mathbf{Z}=\mathbf{z},\mathbf{W}=\mathbf{w})=0 \tag{1}$$

for all $\mathbf{x}$, $\mathbf{y}$ and $\mathbf{z}$. Each equation is a polynomial in the parameters in the parametrization of the probability distributions in $\mathcal{D}(G)^+$, because each term $p(\mathbf{V}=\mathbf{v})$ in the equations is a polynomial in the parameters: $p(\mathbf{V}=\mathbf{v})=\sum_{\mathbf{v}'}p(\mathbf{V}=\mathbf{v},\mathbf{U}\setminus\mathbf{V}=\mathbf{v}')$ where each term $p(\mathbf{V}=\mathbf{v},\mathbf{U}\setminus\mathbf{V}=\mathbf{v}')$ is a polynomial in the parameters, since $p=\prod_{X\in\mathbf{U}}q(X|Pa(X))$. Let each variable in the polynomials take values in $\mathbb{R}$. Then, each polynomial in Equation (1) is non-trivial, that is, not all the values of the variables are solutions to the polynomial. To prove this, it suffices to prove that there exists a probability distribution $p'\in\mathcal{D}(G)^+$ for which the polynomial does not hold. Consider the polynomial for $\mathbf{x}, \mathbf{y}$ and $\mathbf{z}$. Note that there exists a probability distribution $p''\in\mathcal{D}(G)^+$ that is faithful to G (Meek, 1995) and, thus, $\mathbf{X}\not\perp\mathbf{Y}|\mathbf{ZW}$ is in p'' because $sep(\mathbf{X},\mathbf{Y}|\mathbf{ZW})$ does not hold in G. Then, there is some instantiation $\mathbf{x}''\mathbf{y}''\mathbf{z}''\mathbf{w}''$ of $\mathbf{XYZW}$ such that

$$p''(\mathbf{X}=\mathbf{x}'',\mathbf{Y}=\mathbf{y}'',\mathbf{Z}=\mathbf{z}'',\mathbf{W}=\mathbf{w}'')p''(\mathbf{Z}=\mathbf{z}'',\mathbf{W}=\mathbf{w}'')-$$

$$p''(\mathbf{X}=\mathbf{x}'',\mathbf{Z}=\mathbf{z}'',\mathbf{W}=\mathbf{w}'')p''(\mathbf{Y}=\mathbf{y}'',\mathbf{Z}=\mathbf{z}'',\mathbf{W}=\mathbf{w}'')\neq 0.$$

Then, by permuting the states of the random variables in $\mathbf{X}$, $\mathbf{Y}$, $\mathbf{Z}$ and $\mathbf{W}$, we can transform p'' into the desired p'. Let $\mathbf{V}\equiv\mathbf{XYZW}$, $\mathbf{v}\equiv\mathbf{xyzw}$, and $\mathbf{v}''\equiv\mathbf{x}''\mathbf{y}''\mathbf{z}''\mathbf{w}''$. One can introduce a permutation π_X on the set of possible states of X for each $X\in\mathbf{U}$: For $X\in\mathbf{V}$, it is the transposition of the states of X in $\mathbf{v}$ and $\mathbf{v}''$, and the identical mapping for $X\in\mathbf{U}\setminus\mathbf{V}$. These random variable permutations together define a permutation π of the joint sample space of $\mathbf{U}$. Then, p'' can be transformed by π to $p'\equiv p''\circ\pi$. Note that $p'\in\mathcal{D}(G)^+$ because $p''\in\mathcal{D}(G)^+$, and that the parameter values of p' are obtained from the parameter values of p'' by local permutations. Finally, note that $p'(\mathbf{V}=\mathbf{v})=p''(\mathbf{V}=\mathbf{v}'')$ and, thus, the polynomial in Equation (1) for $\mathbf{x}$, $\mathbf{y}$ and $\mathbf{z}$ does not hold for p'.

Let $sol(\mathbf{x},\mathbf{y},\mathbf{z},\mathbf{w})$ denote the set of solutions to the polynomial in Equation (1) for $\mathbf{x}$, $\mathbf{y}$ and $\mathbf{z}$. Then, $sol(\mathbf{x},\mathbf{y},\mathbf{z},\mathbf{w})$ has zero Lebesgue measure wrt $\mathbb{R}^n$ because it consists of the solutions to a non-trivial polynomial in real variables (Okamoto, 1973). Let $sol=\bigcup_{\mathbf{X},\mathbf{Y},\mathbf{Z},\mathbf{W}}\bigcup_{\mathbf{w}}\bigcap_{\mathbf{x},\mathbf{y},\mathbf{z}}sol(\mathbf{x},\mathbf{y},\mathbf{z},\mathbf{w})$ and recall from above that the outer-most union only involves those cases for which $sep(\mathbf{X},\mathbf{Y}|\mathbf{ZW})$ does not hold in G. Then, sol has zero Lebesgue measure wrt $\mathbb{R}^n$, because the finite union and intersection of sets of zero Lebesgue measure has zero Lebesgue measure too. Consequently, the probability distributions in $\mathcal{D}(G)^+$ that have context-specific dependencies correspond to a set of elements of the parameter space for $\mathcal{D}(G)^+$ that has zero Lebesgue measure wrt $\mathbb{R}^n$ because it is contained in sol. Since, as proven above, this correspondence is one-to-one, the probability distributions in $\mathcal{D}(G)^+$ that have context-specific dependencies also have zero Lebesgue measure wrt $\mathbb{R}^n$.

To finish the proof of the second statement, it suffices to note that (i) the probability distributions in $\mathcal{D}(G)^+$ that are not faithful to G have zero Lebesgue measure wrt $\mathbb{R}^n$, because they are a subset of the probability distributions that factorize according to G but are not faithful to G, and these have zero Lebesgue measure wrt $\mathbb{R}^n$ (Meek, 1995), (ii) the probability distributions in $\mathcal{D}(G)^+$ that have context-specific dependencies have zero Lebesgue measure wrt $\mathbb{R}^n$ as proven above, and (iii) the union of the probability distributions in (i) and (ii) has zero Lebesgue measure wrt $\mathbb{R}^n$, because the finite union of sets of zero Lebesgue measure has zero Lebesgue measure too. ■

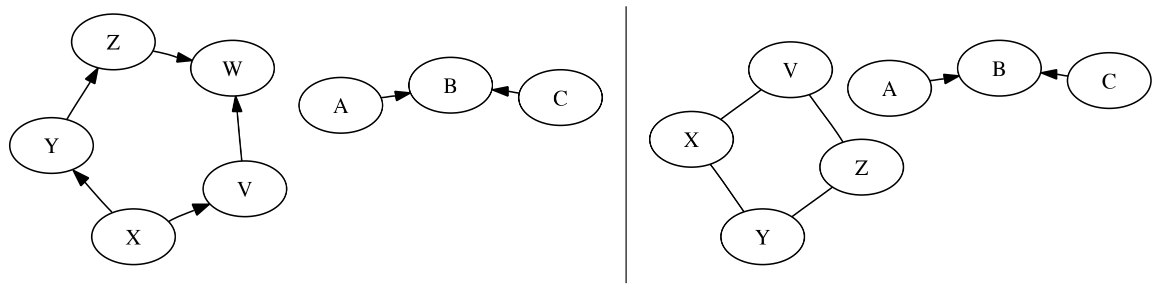


Figure 1: Chain graphs in Example 1.

The theorem below proves that the marginals and conditionals of a strictly positive discrete probability distribution that is a WT graphoid and has no context-specific dependencies are WT graphoids too.

Theorem 2 *Let p be a strictly positive discrete probability distribution that is a WT graphoid. Then, $p(\mathbf{U} \setminus \mathbf{W})$ for any $\mathbf{W} \subseteq \mathbf{U}$ is a WT graphoid. If p has no context-specific dependencies, then $p(\mathbf{U} \setminus \mathbf{W}|\mathbf{W} = \mathbf{w})$ for any $\mathbf{w}$ and $\mathbf{W} \subseteq \mathbf{U}$ is a WT graphoid.*

Proof Let $\mathbf{X}$, $\mathbf{Y}$ and $\mathbf{Z}$ denote three mutually disjoint subsets of $\mathbf{U} \setminus \mathbf{W}$. Then, $\mathbf{X} \perp\!\!\!\perp \mathbf{Y}|\mathbf{Z}$ is in $p(\mathbf{U} \setminus \mathbf{W})$ iff $\mathbf{X} \perp\!\!\!\perp \mathbf{Y}|\mathbf{Z}$ is in p and, thus, $p(\mathbf{U} \setminus \mathbf{W})$ satisfies the WT graphoid properties because p satisfies them. Now, note that $p(\mathbf{U} \setminus \mathbf{W}|\mathbf{W} = \mathbf{w})$ is uniquely defined because p is strictly positive. Furthermore, if p has no context-specific dependencies, then $\mathbf{X} \perp\!\!\!\perp \mathbf{Y}|\mathbf{Z}$ is in $p(\mathbf{U} \setminus \mathbf{W}|\mathbf{W} = \mathbf{w})$ iff $\mathbf{X} \perp\!\!\!\perp \mathbf{Y}|\mathbf{Z}\mathbf{W}$ is in p. Then, $p(\mathbf{U} \setminus \mathbf{W}|\mathbf{W} = \mathbf{w})$ satisfies the WT graphoid properties because p satisfies them. ■

In a nutshell, Theorem 1 proves that for any DAG G, in the measure-theoretic sense explained above, almost all the probability distributions in $\mathcal{D}(G)^+$ are faithful to G and, thus, are WT graphoids (Pearl, 1988). On the other hand, the combination of Theorems 1 and 2 proves that, in the measure-theoretic sense explained above, all the marginals and conditionals of almost all the probability distributions in $\mathcal{D}(G)^+$ are WT graphoids. Finally, we give an example that shows that not all the probability distributions that are WT graphoids are either regular Gaussian or faithful to some UG or DAG.

Example 1 *Let p be a strictly positive discrete probability distribution that is faithful to the DAG in the left-hand side of Figure 1 and that has no context-specific dependencies. Such a probability distribution exists due to Theorem 1 and, moreover, it is a WT graphoid (Pearl, 1988). Then, $p(X,Y,Z,V,A,B,C|W = w)$ for any w, which is uniquely defined because p is strictly positive, is a WT graphoid by Theorem 2. However, this conditional probability distribution is neither regular Gaussian nor faithful to any UG or DAG, because it is discrete and faithful to the chain graph in the right-hand side of Figure 1 (Chickering and Meek, 2002; Peña et al., 2006).*

4. Reading Independencies

By definition, *sep* is sound for reading independencies from the MUI map G of a WT graphoid M, that is, it only identifies independencies in M. In the regular Gaussian case, *sep* in G is also complete in the sense that it identifies all the independencies in M that are shared by all the WT graphoids for which G is the MUI map, because (i) it is proven in Lněnička and Matúš (2007) that there exist regular Gaussian probability distributions that are faithful to G, and (ii) such probability distributions are WT graphoids (Studený, 2005), G is their MUI map, and they only have the independencies that *sep* identifies from G. In this section, we extend this result to the case where $\mathbf{U}$ is discrete. Specifically, we prove that there exist strictly positive discrete probability distributions that are faithful to G for any sample spaces (with at least two possible states) of the random variables in $\mathbf{U}$. Again, such probability distributions are WT graphoids (Pearl, 1988), G is their MUI map, and they only have the independencies that *sep* identifies from G. Therefore, when $\mathbf{U}$ is discrete *sep* in G is also complete in the sense that it identifies all the independencies in M that are shared by all the WT graphoids for which G is the MUI map. These completeness results, in addition to being important in themselves, are crucial for reading as many dependencies as possible from G, as we will see in the next section.

Theorem 3 *Let G be an UG. Let us assume that the sample space of each random variable in* $\mathbf{U}$ *is prescribed and has at least two possible states. Then, there exists a strictly positive discrete probability distribution with the prescribed sample spaces for the random variables in* $\mathbf{U}$ *that is faithful to G.*

Proof Create an extended DAG H of G as follows. For each edge $X - Y$ in G, create an auxiliary discrete random variable W_{XY}. Let $\mathbf{W}$ denote all the auxiliary random variables created. Let H be a DAG over $\mathbf{UW}$ with no edges. For each edge $X - Y$ in G add $X \rightarrow W_{XY} \leftarrow Y$ to H. No more edges are added to H. It is easy to see that for any three mutually disjoint subsets $\mathbf{X}$, $\mathbf{Y}$ and $\mathbf{Z}$ of $\mathbf{U}$, $sep(\mathbf{X},\mathbf{Y}|\mathbf{ZW})$ in H iff $sep(\mathbf{X},\mathbf{Y}|\mathbf{Z})$ in G.

Let $p(\mathbf{U},\mathbf{W})$ denote any strictly positive discrete probability distribution in $\mathcal{D}(H)^+$ that is faithful to H and has no context-specific dependencies. Such a probability distribution exists by Theorem 1. Now, fix any $\mathbf{w}$ and note that $p(\mathbf{U}|\mathbf{W}=\mathbf{w})$ is uniquely defined because $p(\mathbf{U},\mathbf{W})$ is strictly positive. Let $\mathbf{X}$, $\mathbf{Y}$ and $\mathbf{Z}$ denote three mutually disjoint subsets of $\mathbf{U}$. Then, $\mathbf{X}\perp\!\!\!\perp\mathbf{Y}|\mathbf{Z}$ is in $p(\mathbf{U}|\mathbf{W}=\mathbf{w})$ iff $\mathbf{X}\perp\!\!\!\perp\mathbf{Y}|\mathbf{ZW}$ is in $p(\mathbf{U},\mathbf{W})$ iff $sep(\mathbf{X},\mathbf{Y}|\mathbf{ZW})$ in H iff $sep(\mathbf{X},\mathbf{Y}|\mathbf{Z})$ in G. Then, $p(\mathbf{U}|\mathbf{W}=\mathbf{w})$ is faithful to G. Obviously, $p(\mathbf{U}|\mathbf{W}=\mathbf{w})$ is strictly positive and discrete. ■

Note that the theorem above proves that there exists a strictly positive discrete probability distribution that is faithful to G for any sample spaces (with at least two possible states) of the random variables in $\mathbf{U}$. This result is therefore stronger than Theorem 11 in Geiger and Pearl (1993), which proves that there exists a strictly positive discrete probability distribution that is faithful to G for some sample spaces (with at least two possible states) of the random variables in $\mathbf{U}$.

The theorem above proves that, when $\mathbf{U}$ is discrete, *sep* in the MUI map G of a WT graphoid M is complete for any sample spaces of the random variables in $\mathbf{U}$, where complete means that it is able to identify all the independencies in M that are shared by all the WT graphoids for which G is the MUI map. However, *sep* in G is not complete if this is understood as being able to identify all the independencies in M. Actually, no sound criterion for reading independencies from G is complete in the latter sense. An example follows.

Example 2 *Let p be a discrete probability distribution that is faithful to the DAG $\{X \rightarrow Z, Y \rightarrow Z\}$. Such a probability distribution exists (Meek, 1995). Let G denote the MUI map of p, namely the* *complete UG. Note that p is not faithful to G. However, by Theorem 3, there exists a discrete probability distribution p' that is faithful to G. As proven in Pearl (1988), p and p' are WT graphoids. Let us assume that we are dealing with p. Then, no sound criterion can conclude $X \perp\!\!\!\perp Y|\emptyset$ by just studying G because this independence does not hold in p', and it is impossible to know whether we are dealing with p or p' on the sole basis of G.*

5. Reading Dependencies

In this section, we propose a sound and complete criterion for reading dependencies from the MUI map G of a WT graphoid. Here, complete means that it is able to read all the dependencies that can be derived by applying the WT graphoid properties to the dependencies used in the construction of G and the independencies obtained from G by *sep*.

We define the dependence base of an independence model M, denoted $bas(M)$, as the set of dependencies $X \not\perp\!\!\!\perp Y|\mathbf{U} \setminus (XY)$ with $X, Y \in \mathbf{U}$. Recall from Section 2 that X and Y are adjacent in the MUI map of M iff $X \not\perp\!\!\!\perp Y|\mathbf{U} \setminus (XY)$. If M is a WT graphoid, then additional dependencies in M can be derived from $bas(M)$ via the WT graphoid properties. For this purpose, we rephrase the WT graphoid properties as follows. Let $\mathbf{X}$, $\mathbf{Y}$, $\mathbf{Z}$ and $\mathbf{W}$ denote four mutually disjoint subsets of $\mathbf{U}$. Symmetry $\mathbf{Y} \not\perp\!\!\!\perp \mathbf{X}|\mathbf{Z} \Rightarrow \mathbf{X} \not\perp\!\!\!\perp \mathbf{Y}|\mathbf{Z}$. Decomposition $\mathbf{X} \not\perp\!\!\!\perp \mathbf{Y}|\mathbf{Z} \Rightarrow \mathbf{X} \not\perp\!\!\!\perp \mathbf{YW}|\mathbf{Z}$. Weak union $\mathbf{X} \not\perp\!\!\!\perp \mathbf{Y}|\mathbf{ZW} \Rightarrow \mathbf{X} \not\perp\!\!\!\perp \mathbf{YW}|\mathbf{Z}$. Contraction $\mathbf{X} \not\perp\!\!\!\perp \mathbf{YW}|\mathbf{Z} \Rightarrow \mathbf{X} \not\perp\!\!\!\perp \mathbf{Y}|\mathbf{ZW} \vee \mathbf{X} \not\perp\!\!\!\perp \mathbf{W}|\mathbf{Z}$ is problematic for deriving new dependencies because it contains a disjunction in the right-hand side and, thus, it should be split into two properties: Contraction1 $\mathbf{X} \not\perp\!\!\!\perp \mathbf{YW}|\mathbf{Z} \wedge \mathbf{X} \perp\!\!\!\perp \mathbf{Y}|\mathbf{ZW} \Rightarrow \mathbf{X} \not\perp\!\!\!\perp \mathbf{W}|\mathbf{Z}$, and contraction2 $\mathbf{X} \not\perp\!\!\!\perp \mathbf{YW}|\mathbf{Z} \wedge \mathbf{X} \perp\!\!\!\perp \mathbf{W}|\mathbf{Z} \Rightarrow \mathbf{X} \not\perp\!\!\!\perp \mathbf{Y}|\mathbf{ZW}$. Likewise, intersection $\mathbf{X} \not\perp\!\!\!\perp \mathbf{YW}|\mathbf{Z} \Rightarrow \mathbf{X} \not\perp\!\!\!\perp \mathbf{Y}|\mathbf{ZW} \vee \mathbf{X} \not\perp\!\!\!\perp \mathbf{W}|\mathbf{ZY}$ gives rise to intersection1 $\mathbf{X} \not\perp\!\!\!\perp \mathbf{YW}|\mathbf{Z} \wedge \mathbf{X} \perp\!\!\!\perp \mathbf{Y}|\mathbf{ZW} \Rightarrow \mathbf{X} \not\perp\!\!\!\perp \mathbf{W}|\mathbf{ZY}$, and intersection2 $\mathbf{X} \not\perp\!\!\!\perp \mathbf{YW}|\mathbf{Z} \wedge \mathbf{X} \perp\!\!\!\perp \mathbf{W}|\mathbf{ZY} \Rightarrow \mathbf{X} \not\perp\!\!\!\perp \mathbf{Y}|\mathbf{ZW}$. Note that intersection1 and intersection2 are equivalent and, thus, we refer to them simply as intersection. Finally, weak transitivity $\mathbf{X} \not\perp\!\!\!\perp V|\mathbf{Z} \wedge V \not\perp\!\!\!\perp \mathbf{Y}|\mathbf{Z} \Rightarrow \mathbf{X} \not\perp\!\!\!\perp \mathbf{Y}|\mathbf{Z} \vee \mathbf{X} \not\perp\!\!\!\perp \mathbf{Y}|\mathbf{Z}V$ with $V \in \mathbf{U} \setminus (\mathbf{XYZ})$ gives rise to weak transitivity1 $\mathbf{X} \not\perp\!\!\!\perp V|\mathbf{Z} \wedge V \not\perp\!\!\!\perp \mathbf{Y}|\mathbf{Z} \wedge \mathbf{X} \perp\!\!\!\perp \mathbf{Y}|\mathbf{Z} \Rightarrow \mathbf{X} \not\perp\!\!\!\perp \mathbf{Y}|\mathbf{Z}V$, and weak transitivity2 $\mathbf{X} \not\perp\!\!\!\perp V|\mathbf{Z} \wedge V \not\perp\!\!\!\perp \mathbf{Y}|\mathbf{Z} \wedge \mathbf{X} \perp\!\!\!\perp \mathbf{Y}|\mathbf{Z}V \Rightarrow \mathbf{X} \not\perp\!\!\!\perp \mathbf{Y}|\mathbf{Z}$. The independence in the left-hand side of any of the properties above holds if the corresponding *sep* statement holds in the MUI map G of M. This is the best solution we can hope for because, as discussed in Section 4, *sep* in G is sound and complete in the sense that it identifies all and only the independencies in M that are shared by all the WT graphoids for which G is the MUI map. Moreover, this solution does not require more information about M than what it is available, because G can be constructed from $bas(M)$. We denote by $sep(G)$ all the *sep* statements holding in the MUI map G. We define the WT graphoid closure of $bas(M)$ wrt $sep(G)$, denoted $WT_{bas(M)}^{sep(G)}$, as the set of dependencies in $bas(M)$ plus those that can be derived from it and $sep(G)$ by applying the WT graphoid properties. We now introduce our criterion for reading dependencies from the MUI map of a WT graphoid.

Definition 4 *Let $\mathbf{X}$, $\mathbf{Y}$ and $\mathbf{Z}$ denote three mutually disjoint subsets of $\mathbf{U}$. Let $con(\mathbf{X}, \mathbf{Y}|\mathbf{Z})$ denote that $\mathbf{X}$ is connected to $\mathbf{Y}$ given $\mathbf{Z}$ in an UG G. Specifically, $con(\mathbf{X}, \mathbf{Y}|\mathbf{Z})$ holds when there exist some $X_1 \in \mathbf{X}$ and $X_n \in \mathbf{Y}$ such that there exists **exactly one** path in G between X_1 and X_n that is not blocked by $(\mathbf{X} \setminus X_1)(\mathbf{Y} \setminus X_n)\mathbf{Z}$.*

As an illustrative example of *con*, consider the UG in Figure 2. Some *con* statements holding in that graph are $con(A, B|CD)$, $con(AC, BD|\emptyset)$, $con(A, BCD|\emptyset)$ and $con(A, BD|\emptyset)$ because in each of these statement there exists a single path between A and B or D that is not blocked by the rest of

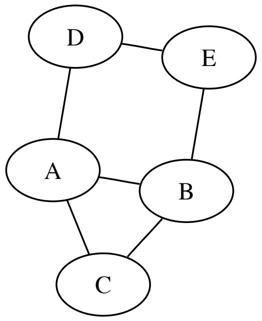


Figure 2: UG to illustrate the definition of *con*: $con(A,B|CD)$, $con(AC,BD|\emptyset)$, $con(A,BCD|\emptyset)$ and $con(A,BD|\emptyset)$ hold in the graph, whereas $con(A,B|\emptyset)$ and $con(A,B|D)$ do not hold.

the nodes in the statement. On the other hand, some *con* statements that do not hold in the graph under consideration are $con(A,B|\emptyset)$ and $con(A,B|D)$, because in both cases there exist several paths between A and B that are not blocked by the rest of the nodes in the statement. Note that the place the nodes take in the statement matters: For instance, $con(A,BD|\emptyset)$ holds but $con(A,B|D)$ does not.

We now prove that *con* is sound for reading dependencies from the MUI map G of a WT graphoid M, that is, it only identifies dependencies in M. Actually, it only identifies dependencies in $WT_{bas(M)}^{sep(G)}$. Hereinafter, we abbreviate a path $X_1,\dots,X_n$ in an UG as $X_{1:n}$.

Theorem 5 *Let M be a WT graphoid and G its MUI map. Then, con in G only identifies dependencies in $WT_{bas(M)}^{sep(G)}$.*

Proof We first prove that if $X_{1:n}$ is the only path in G between X_1 and X_n that is not blocked by $\mathbf{Y} \subseteq \mathbf{U} \setminus X_{1:n}$, then $X_1 \not\perp X_n|\mathbf{Y}$ is in $WT_{bas(M)}^{sep(G)}$. In other words, we prove that $X_1 \not\perp X_n|\mathbf{Y}$ can be derived from $bas(M)$ and $sep(G)$ using the WT graphoid properties. We prove it by induction over n. We first prove it for $n = 2$. Let $\mathbf{W}$ denote all the nodes in $\mathbf{U} \setminus X_{1:2} \setminus \mathbf{Y}$ that are not separated from X_1 given $X_2\mathbf{Y}$ in G. Since X_1 and X_2 are adjacent in G, $X_1 \not\perp X_2|\mathbf{U} \setminus X_{1:2}$ and, thus, $X_1\mathbf{W} \not\perp X_2(\mathbf{U} \setminus X_{1:2} \setminus \mathbf{Y} \setminus \mathbf{W})|\mathbf{Y}$ due to weak union and symmetry. This together with $sep(X_1\mathbf{W}, \mathbf{U} \setminus X_{1:2} \setminus \mathbf{Y} \setminus \mathbf{W}|X_2\mathbf{Y})$, which follows from the definition of $\mathbf{W}$, implies $X_1\mathbf{W} \not\perp X_2|\mathbf{Y}$ due to contraction1. Note that if $\mathbf{U} \setminus X_{1:2} \setminus \mathbf{Y} \setminus \mathbf{W} = \emptyset$, then $X_1\mathbf{W} \not\perp X_2(\mathbf{U} \setminus X_{1:2} \setminus \mathbf{Y} \setminus \mathbf{W})|\mathbf{Y}$ directly implies $X_1\mathbf{W} \not\perp X_2|\mathbf{Y}$. In any case, this dependence together with $sep(\mathbf{W}, X_2|X_1\mathbf{Y})$, because otherwise there exist several unblocked paths in G between X_1 and X_2 which contradicts the definition of $\mathbf{Y}$, implies $X_1 \not\perp X_2|\mathbf{Y}$ due to contraction1 and symmetry. Note that if $\mathbf{W} = \emptyset$, then $X_1\mathbf{W} \not\perp X_2|\mathbf{Y}$ directly implies $X_1 \not\perp X_2|\mathbf{Y}$.

Let us assume as induction hypothesis that the statement that we are proving holds for all $n < m$. We now prove it for $n = m$. Since the paths $X_{1:2}$ and $X_{2:m}$ contain less than m nodes and $\mathbf{Y}$ blocks all the other paths in G between X_1 and X_2 and between X_2 and X_m, because otherwise there exist several unblocked paths in G between X_1 and X_m which contradicts the definition of $\mathbf{Y}$, then $X_1 \not\perp X_2|\mathbf{Y}$ and $X_2 \not\perp X_m|\mathbf{Y}$ due to the induction hypothesis. This together with $sep(X_1, X_m|\mathbf{Y}X_2)$, which follows from the definition of $X_{1:m}$ and $\mathbf{Y}$, implies $X_1 \not\perp X_m|\mathbf{Y}$ due to weak transitivity2.

Let $\mathbf{X}$, $\mathbf{Y}$ and $\mathbf{Z}$ denote three mutually disjoint subsets of $\mathbf{U}$. If $con(\mathbf{X}, \mathbf{Y}|\mathbf{Z})$ holds in G, then there exist some $X_1 \in \mathbf{X}$ and $X_n \in \mathbf{Y}$ such that $X_1 \not\perp X_n|(\mathbf{X} \setminus X_1)(\mathbf{Y} \setminus X_n)\mathbf{Z}$ is in $WT_{bas(M)}^{sep(G)}$ due to the paragraph above and, thus, $\mathbf{X} \not\perp \mathbf{Y}|\mathbf{Z}$ is also in $WT_{bas(M)}^{sep(G)}$ due to weak union and symmetry. ■

We now prove that *con* is complete for reading dependencies from the MUI map G of a WT graphoid M, in the sense that it identifies all the dependencies in M that follow from the information about M that is available, namely the dependencies in $bas(M)$, the independencies in $sep(G)$, and the fact that M is a WT graphoid.

Theorem 6 *Let M be a WT graphoid and G its MUI map. Then, con in G identifies all the dependencies in $WT_{bas(M)}^{sep(G)}$.*

Proof It suffices to prove (i) that all the dependencies in $bas(M)$ are identified by *con* in G, and (ii) that *con* satisfies the WT graphoid properties. Since the first point is trivial, we only prove the second point. Let $\mathbf{X}, \mathbf{Y}, \mathbf{Z}$ and $\mathbf{W}$ denote four mutually disjoint subsets of $\mathbf{U}$.

- Symmetry $con(\mathbf{Y}, \mathbf{X}|\mathbf{Z}) \Rightarrow con(\mathbf{X}, \mathbf{Y}|\mathbf{Z})$. Trivial.

- Decomposition $con(\mathbf{X}, \mathbf{Y}|\mathbf{Z}) \Rightarrow con(\mathbf{X}, \mathbf{YW}|\mathbf{Z})$. Trivial if $\mathbf{W}$ contains no node in the path $X_{1:n}$ in $con(\mathbf{X}, \mathbf{Y}|\mathbf{Z})$. If $\mathbf{W}$ contains some node in $X_{1:n}$, then let X_m denote the closest node to X_1 that is in $X_{1:n}$ and $\mathbf{W}$. Then, the path $X_{1:m}$ ensures $con(\mathbf{X}, \mathbf{YW}|\mathbf{Z})$ because $(\mathbf{X} \setminus X_1)(\mathbf{YW} \setminus X_m)\mathbf{Z}$ blocks all the other paths in G between X_1 and X_m, since $(\mathbf{X} \setminus X_1)(\mathbf{Y} \setminus X_n)\mathbf{Z}$ blocks all the paths in G between X_1 and X_m except $X_{1:m}$ because otherwise there exist several unblocked paths in G between X_1 and X_n, which contradicts $con(\mathbf{X}, \mathbf{Y}|\mathbf{Z})$.

- Weak union $con(\mathbf{X}, \mathbf{Y}|\mathbf{ZW}) \Rightarrow con(\mathbf{X}, \mathbf{YW}|\mathbf{Z})$. Trivial because $\mathbf{W}$ contains no node in the path $X_{1:n}$ in $con(\mathbf{X}, \mathbf{Y}|\mathbf{ZW})$.

- Contraction1 $con(\mathbf{X}, \mathbf{YW}|\mathbf{Z}) \wedge sep(\mathbf{X}, \mathbf{Y}|\mathbf{ZW}) \Rightarrow con(\mathbf{X}, \mathbf{W}|\mathbf{Z})$. Since $\mathbf{ZW}$ blocks all the paths in G between $\mathbf{X}$ and $\mathbf{Y}$, then the path $X_{1:n}$ in $con(\mathbf{X}, \mathbf{YW}|\mathbf{Z})$ must be between $\mathbf{X}$ and $\mathbf{W}$. To prove that $X_{1:n}$ ensures $con(\mathbf{X}, \mathbf{W}|\mathbf{Z})$, we have to prove that this is the only path in G between X_1 and X_n that is not blocked by $(\mathbf{X} \setminus X_1)(\mathbf{W} \setminus X_n)\mathbf{Z}$. Assume to the contrary that there is a second such path in G. This second path cannot contain any node in $\mathbf{Y}$ for $sep(\mathbf{X}, \mathbf{Y}|\mathbf{ZW})$ to hold. Then, this second path is not blocked by $(\mathbf{X} \setminus X_1)\mathbf{Y}(\mathbf{W} \setminus X_n)\mathbf{Z}$ either. However, this contradicts $con(\mathbf{X}, \mathbf{YW}|\mathbf{Z})$, because we have found two paths in G between X_1 and X_n that are not blocked by $(\mathbf{X} \setminus X_1)\mathbf{Y}(\mathbf{W} \setminus X_n)\mathbf{Z}$.

- Contraction2 $con(\mathbf{X}, \mathbf{YW}|\mathbf{Z}) \wedge sep(\mathbf{X}, \mathbf{W}|\mathbf{Z}) \Rightarrow con(\mathbf{X}, \mathbf{Y}|\mathbf{ZW})$. Since $\mathbf{Z}$ blocks all the paths in G between $\mathbf{X}$ and $\mathbf{W}$, the path $X_{1:n}$ in $con(\mathbf{X}, \mathbf{YW}|\mathbf{Z})$ must be between $\mathbf{X}$ and $\mathbf{Y}$ and, thus, it ensures $con(\mathbf{X}, \mathbf{Y}|\mathbf{ZW})$.

- Intersection $con(\mathbf{X}, \mathbf{YW}|\mathbf{Z}) \wedge sep(\mathbf{X}, \mathbf{Y}|\mathbf{ZW}) \Rightarrow con(\mathbf{X}, \mathbf{W}|\mathbf{ZY})$. Since $\mathbf{ZW}$ blocks all the paths in G between $\mathbf{X}$ and $\mathbf{Y}$, the path $X_{1:n}$ in $con(\mathbf{X}, \mathbf{YW}|\mathbf{Z})$ must be between $\mathbf{X}$ and $\mathbf{W}$ and, thus, it ensures $con(\mathbf{X}, \mathbf{W}|\mathbf{ZY})$.

- Weak transitivity2 $con(\mathbf{X}, X_m|\mathbf{Z}) \wedge con(X_m, \mathbf{Y}|\mathbf{Z}) \wedge sep(\mathbf{X}, \mathbf{Y}|\mathbf{Z}X_m) \Rightarrow con(\mathbf{X}, \mathbf{Y}|\mathbf{Z})$ with $X_m \in \mathbf{U} \setminus (\mathbf{XYZ})$. Let $X_{1:m}$ and $X_{m:n}$ denote the paths in $con(\mathbf{X}, X_m|\mathbf{Z})$ and $con(X_m, \mathbf{Y}|\mathbf{Z})$, respectively. For $con(\mathbf{X}, X_m|\mathbf{Z})$ to hold, $X_{1:m}$ cannot contain any node in $(\mathbf{X} \setminus X_1)\mathbf{Z}$. For $con(X_m, \mathbf{Y}|\mathbf{Z})$ to hold, $X_{m:n}$ cannot contain any node in $(\mathbf{Y} \setminus X_n)\mathbf{Z}$. For $sep(\mathbf{X}, \mathbf{Y}|\mathbf{Z}X_m)$ to hold, neither $X_{1:m}$

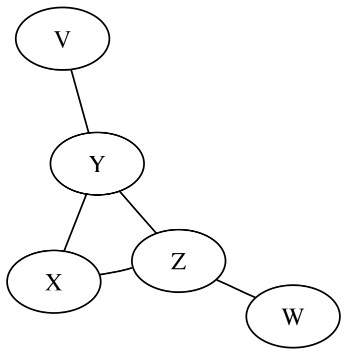


Figure 3: UG in Example 4.

can contain a node in $\mathbf{Y}$, nor $X_{m:n}$ can contain a node in $\mathbf{X}$. Consequently, neither $X_{1:m}$ nor $X_{m:n}$ is blocked by $(\mathbf{X} \setminus X_1)(\mathbf{Y} \setminus X_n)\mathbf{Z}$.

Furthermore, $X_{1:m}$ and $X_{m:n}$ only intersect on X_m. To see this, assume the contrary. Let X_l ($X_l \neq X_m$) denote the closest node to X_1 that is in $X_{1:m}$ and $X_{m:n}$. Then, it follows from the paragraph above that neither $X_{1:l}$ and $X_{l:n}$ contain a node in $\mathbf{Z}X_m$. However, this contradicts $sep(\mathbf{X}, \mathbf{Y}|\mathbf{Z}X_m)$. Consequently, $X_{1:m}$ followed by $X_{m:n}$ is a path in G between X_1 and X_n that, as shown above, is not blocked by $(\mathbf{X} \setminus X_1)(\mathbf{Y} \setminus X_n)\mathbf{Z}$. It remains to prove that this path is unique. Assume to the contrary that there exists a second such path in G. For $sep(\mathbf{X}, \mathbf{Y}|\mathbf{Z}X_m)$ to hold, this second path must pass through X_m. However, this implies that either there exists a second path in G between X_1 and X_m that is not blocked by $(\mathbf{X} \setminus X_1)\mathbf{Z}$, or there exists a second path in G between X_m and X_n that is not blocked by $(\mathbf{Y} \setminus X_n)\mathbf{Z}$. This contradicts $con(\mathbf{X}, X_m|\mathbf{Z})$ or $con(X_m, \mathbf{Y}|\mathbf{Z})$.

- Weak transitivity1 $con(\mathbf{X}, X_m|\mathbf{Z}) \wedge con(X_m, \mathbf{Y}|\mathbf{Z}) \wedge sep(\mathbf{X}, \mathbf{Y}|\mathbf{Z}) \Rightarrow con(\mathbf{X}, \mathbf{Y}|\mathbf{Z}X_m)$ with $X_m \in \mathbf{U} \setminus (\mathbf{XYZ})$. Trivial because the antecedent involves a contradiction: It follows from the proof of weak transitivity2 that $con(\mathbf{X}, X_m|\mathbf{Z})$ and $con(X_m, \mathbf{Y}|\mathbf{Z})$ imply the existence of a path in G between some $X_1 \in \mathbf{X}$ and $X_n \in \mathbf{Y}$ that is not blocked by $(\mathbf{X} \setminus X_1)(\mathbf{Y} \setminus X_n)\mathbf{Z}$, which contradicts $sep(\mathbf{X}, \mathbf{Y}|\mathbf{Z})$.

■

We devote the rest of this section to some remarks on the two theorems above. Note that *con* in G is not complete if this is understood as being able to identify all the dependencies in M. Actually, no sound criterion for reading dependencies from G alone is complete in this sense. Example 2 illustrates this point. Let us now assume that we are dealing with p' instead of with p. Then, no sound criterion can conclude $X \not\perp\!\!\!\perp Y|\emptyset$ by just studying G because this dependence does not hold in p, and it is impossible to know whether we are dealing with p or p' on the sole basis of G.

It seems natural to expect that assuming further independence properties will result in more dependencies being readable from the MUI map of an independence model and, thus, in the necessity of developing a new graphical criterion that identifies them. However, this is not always the case as the following example shows. First, let us define a weak transitivity and composition (WTC) graphoid as a WT graphoid that satisfies composition.

Example 3 *The graphical criterion con is still sound and complete for reading dependencies from the MUI map G of a WTC graphoid M. Here, complete means that it is able to read all the dependencies in M that can be derived from bas(M) and sep(G) by applying the WTC graphoid properties. The reason is that every dependence that follows from composition also follows from contraction1. In other words, when rephrased to derive dependencies, composition looks like* $\mathbf{X}\not\perp\!\!\!\perp\mathbf{YW}|\mathbf{Z} \wedge \mathbf{X}\perp\!\!\!\perp\mathbf{Y}|\mathbf{Z} \Rightarrow \mathbf{X}\not\perp\!\!\!\perp\mathbf{W}|\mathbf{Z}$*. As before, the independence in the left-hand side holds if the corresponding sep statement holds in G. However, if* $sep(\mathbf{X},\mathbf{Y}|\mathbf{Z})$ *then* $sep(\mathbf{X},\mathbf{Y}|\mathbf{ZW})$*. Thus, when composition applies so does contraction1, and both imply the same consequent. Any probability distribution that is regular Gaussian or faithful to some UG or DAG is a WTC graphoid (Pearl, 1988; Studený, 2005). Moreover, similarly to Theorem 2, the marginals and conditionals of a strictly positive discrete probability distribution that is a WTC graphoid and has no context-specific dependencies are WTC graphoids (Chickering and Meek, 2002; Peña et al., 2006).*

A sensible question to ask is whether the definition of complete in Theorem 6 coincides with the definition of complete as able to identify all the dependencies shared by all the WT graphoids for which G is the MUI map. Currently, we do not have an answer to this question, though we incline to think that the definition in the theorem above is weaker than the alternative one. For instance, if we limit ourselves to WTC graphoids, then *con* in G may not identify every dependency shared by all the WTC graphoids for which G is the MUI map, as the following example illustrates.

Example 4 *Consider any regular Gaussian probability distribution whose MUI map is the UG in Figure 3. Such probability distributions exist (Lněnička and Matúš, 2007). Recall that any regular Gaussian probability distribution is a WTC graphoid. Then,* $X\not\perp\!\!\!\perp Y|\emptyset$ *or* $X\not\perp\!\!\!\perp Z|\emptyset$ *because otherwise* $X\perp\!\!\!\perp YZ|\emptyset$ *by composition, which is a contradiction as* $con(X,YZ|\emptyset)$ *holds in G and thus* $X\not\perp\!\!\!\perp YZ|\emptyset$ *by Theorem 5.*

Assume $X\not\perp\!\!\!\perp Y|\emptyset$*. Furthermore,* $con(Y,V|\emptyset)$ *holds in G and thus* $Y\not\perp\!\!\!\perp V|\emptyset$ *by Theorem 5. Furthermore,* $sep(X,V|Y)$ *holds in G and thus* $X\perp\!\!\!\perp V|Y$*. Consequently,* $X\not\perp\!\!\!\perp V|\emptyset$ *by weak transitivity2. Likewise,* $X\not\perp\!\!\!\perp W|\emptyset$ *when assuming* $X\not\perp\!\!\!\perp Z|\emptyset$*. Then,* $X\not\perp\!\!\!\perp V|\emptyset$ *or* $X\not\perp\!\!\!\perp W|\emptyset$ *and, thus,* $X\not\perp\!\!\!\perp VW|\emptyset$ *by decomposition. However,* $con(X,VW|\emptyset)$ *does not hold in G.*

At the beginning of this section, we have defined *bas*(M) as the set of dependencies $X\not\perp\!\!\!\perp Y|\mathbf{U}\setminus(XY)$ with $X,Y\in\mathbf{U}$. However, Theorems 5 and 6 remain valid if we redefine *bas*(M) as the set of dependencies $X\not\perp\!\!\!\perp Y|MB(X)\setminus Y$ with $X\in\mathbf{U}$ and $Y\in MB(X)$. A proof follows. Moreover, recall from Section 2 that X and Y are adjacent in the MUI map of M iff $Y\in MB(X)$.

Proof It suffices to prove that, when *bas*(M) consists of the dependencies in the first definition, $WT^{sep(G)}_{bas(M)}$ includes the dependencies in the second definition, and vice versa. If $X\not\perp\!\!\!\perp Y|\mathbf{U}\setminus(XY)$, then $X\not\perp\!\!\!\perp Y(\mathbf{U}\setminus(XY)\setminus(MB(X)\setminus Y))|MB(X)\setminus Y$ due to weak union. This together with $sep(X,\mathbf{U}\setminus(XY)\setminus(MB(X)\setminus Y)|Y(MB(X)\setminus Y))$ implies $X\not\perp\!\!\!\perp Y|MB(X)\setminus Y$ due to contraction1. On the other hand, if $X\not\perp\!\!\!\perp Y|MB(X)\setminus Y$, then $X\not\perp\!\!\!\perp Y(\mathbf{U}\setminus(XY)\setminus(MB(X)\setminus Y))|MB(X)\setminus Y$ due to decomposition. This together with $sep(X,\mathbf{U}\setminus(XY)\setminus(MB(X)\setminus Y)|Y(MB(X)\setminus Y))$ implies $X\not\perp\!\!\!\perp Y|\mathbf{U}\setminus(XY)$ due to intersection. ■

It is proven in Becker et al. (2000) that if M is a WT graphoid whose MUI map is a forest G, then M is faithful to G. The soundness of *con* allows us to give an alternative proof of this result.

Proof Assume to the contrary that M is not faithful to G. Since G is the MUI map of M, the previous assumption is equivalent to assume that there exists three mutually disjoint subsets of $\mathbf{U}$, say $\mathbf{X}$,

```
FirstPath(X, Y, Z, L)

1   CreatePointers(L)
2   for each node W in L do
3     if $W \in \mathbf{Z}$ then
4       $C[W] = 0$
5     else
6       $C[W] = 1$
7   $P = \emptyset$
8   Push(X, P)
9   $C[X] = 0$
10  while $P \neq \emptyset$ and $Top(P) \neq Y$ do
11    $W = NextAdj(Top(P), L)$
12    if $W = \emptyset$ then
13      Pop(P)
14    else
15      if $C[W] = 1$ then
16        Push(W, P)
17        $C[W] = 0$
18  return P
```

Table 1: $FirstPath(X,Y,\mathbf{Z},L)$.

$\mathbf{Y}$ and $\mathbf{Z}$, such that $\mathbf{X} \perp\!\!\!\perp \mathbf{Y}|\mathbf{Z}$ is in M but $sep(\mathbf{X},\mathbf{Y}|\mathbf{Z})$ does not hold in G. However, if $sep(\mathbf{X},\mathbf{Y}|\mathbf{Z})$ does not hold in G, then there must exist a path in G between some $X_1 \in \mathbf{X}$ and $X_n \in \mathbf{Y}$ that is not blocked by $(\mathbf{X}\setminus X_1)(\mathbf{Y}\setminus X_n)\mathbf{Z}$. Furthermore, since G is a forest, that must be the only path in G between X_1 and X_n that is not blocked by $(\mathbf{X}\setminus X_1)(\mathbf{Y}\setminus X_n)\mathbf{Z}$. Consequently, $con(\mathbf{X},\mathbf{Y}|\mathbf{Z})$ holds in G and, thus, $\mathbf{X} \not\perp\!\!\!\perp \mathbf{Y}|\mathbf{Z}$ is in M due to Theorem 5. This is a contradiction and, thus, M is faithful to G. ■

Finally, we note that the following graphical criterion, denoted *bou* here, for reading dependencies from the MUI map G of a graphoid M is introduced in Bouckaert (1995): Let $\mathbf{X}$, $\mathbf{Y}$ and $\mathbf{Z}$ denote three mutually disjoint subsets of $\mathbf{U}$, then $bou(\mathbf{X},\mathbf{Y}|\mathbf{Z})$ holds when there exist some $X_1 \in \mathbf{X}$ and $X_n \in \mathbf{Y}$ such that $X_1 \in MB(X_n)$ and either $MB(X_1)\setminus X_n \subseteq (\mathbf{X}\setminus X_1)(\mathbf{Y}\setminus X_n)\mathbf{Z}$ or $MB(X_n)\setminus X_1 \subseteq (\mathbf{X}\setminus X_1)(\mathbf{Y}\setminus X_n)\mathbf{Z}$. The criterion is proven to be sound and complete, where complete means that it is able to identify all the dependencies in M that can be derived from $bas(M)$ and $sep(G)$ by applying the graphoid properties. It is clear that *con* identifies a superset of the dependencies identified by *bou* when M satisfies weak transitivity. In such a case, then, *con* represents an advantage over *bou*. As discussed in Section 3, there are important families of probability distributions that are graphoids and satisfy weak transitivity.

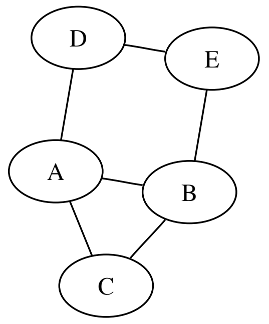
6. An Algorithm for Reading (In)Dependencies

In this section, we present an algorithm that jointly implements *sep* and *con*. We first describe in Table 1 the algorithm $FirstPath(X,Y,\mathbf{Z},L)$ which returns a path in an UG G between X and Y that is not blocked by $\mathbf{Z}$, if such a path exists. We assume that G is represented as a set of adjacency lists L, that is, each node in G has an associated ordered list containing the nodes that are adjacent to it in G. In the algorithm, the function $CreatePointers(L)$ creates a pointer for each adjacency list in L. Each pointer points to the first element of the corresponding list. If the list is empty, the pointer is NULL. The function $NextAdj(W,L)$ returns the element that is pointed by the pointer created by $CreatePointers(L)$ for the adjacency list associated to the node W and, then, moves the pointer to the next element in the list. If there is no next element, the pointer takes value NULL. $NextAdj(W,L)$ returns $\emptyset$ if the pointer is NULL. In the algorithm, P is a stack storing the path currently being explored. The function $Top(P)$ returns the element on the top of P. The function $Pop(P)$ removes the element on the top of P from P. The function $Push(W,P)$ adds the node W to the top of P. Finally, C is an array containing a binary entry for each node in G indicating whether the node should (1) or should not (0) be considered to extend P. The algorithm sets $C[W]$ to 0 iff W is in $\mathbf{Z}$ or W was considered before and the algorithm already found a path from X to W that is not blocked by $\mathbf{Z}$ (see lines 4 and 17). A node considered before should not be considered again because either it is still in P or if it is not in P then it does not lead to Y. To see the latter point, note that the algorithm works in a depth-first fashion: It extends P with a node W that is adjacent to the top element of P if W is not in the blocking set $\mathbf{Z}$ and has not been considered before (see lines 11, 15 and 16). When P cannot be extended further, the algorithm backtracks by removing the top element of P from P and exploring an alternative extension (see lines 12 and 13). The algorithm ends when P is empty, meaning that all the paths in G starting with X were explored and none of them reached Y without visiting $\mathbf{Z}$, or when the top element of P is Y, meaning that a path in G between X and Y that is not blocked by $\mathbf{Z}$ was found and stored in P.

$FirstPath(X,Y,\mathbf{Z},L)$ is considered to run in at most $O(e+n)$ time where e and n are the number of edges and nodes in G, respectively. To see it, note that thanks to the use of C each node is pushed into P at most once. For each node V pushed into P, the algorithm performs as many iterations of lines 10-17 as adjacent nodes V has plus one last iteration when $W=\emptyset$. These iterations need not be consecutive. So, the number of iterations of lines 10-17 that the algorithm performs is bounded from above by $2e+n$. Additionally, the algorithm performs n iterations of lines 2-6 to initialize C. As described in the paragraph above, $CreatePointers(L)$ is considered to run in $O(n)$ time whereas any other operation or function in the algorithm is considered to run in $O(1)$ time. Consequently, $FirstPath(X,Y,\mathbf{Z},L)$ is considered to require at most $O(e+n)$ time.

Recall from above that we assume that G is represented as a set of adjacency lists L and that each of these lists is ordered. Let us now reverse the order of the elements in each of these lists and let L' denote the resulting set of adjacency lists. Our premise is that producing L' from L takes $O(e)$ time. Obviously, L' is also an adjacency list representation of G. However, $FirstPath(X,Y,\mathbf{Z},L)$ and $FirstPath(X,Y,\mathbf{Z},L')$ return the same path iff that is the only path in G between X and Y that is not blocked by $\mathbf{Z}$. We formally prove this assertion below, but first we give an example. Let G, L and L' be as shown in Table 2. Then, $FirstPath(A,B,\emptyset,L)$ returns the path A,B whereas $FirstPath(A,B,\emptyset,L')$ returns the path A,D,E,B.

Proof First, we introduce a total order for the paths between X and Y in G that are not blocked by $\mathbf{Z}$. For this purpose, we associate to each such path a sequence of length $n-1$ of natural numbers



Adjacency lists L

Node	Adjacency list
A	B, C, D
B	A, C, E
C	A, B
D	A, E
E	B, D

Adjacency lists L'

Node	Adjacency list
A	D, C, B
B	E, C, A
C	B, A
D	E, A
E	D, B

Table 2: Example where $FirstPath(A,B,\emptyset,L)$ and $FirstPath(A,B,\emptyset,L')$ return different paths.

between 0 and $n-1$, where n is the number of nodes in G. For a path $X = X_1,\ldots,X_k = Y$, the sequence is $[o_1,\ldots,o_{k-1},0,\ldots,0]$, where o_i is the position of X_{i+1} in the adjacency list associated to X_i. We order the sequences lexicographically, which results in the desired order.

We now prove that if there exists a path between X and Y in G that is not blocked by $\mathbf{Z}$, then $FirstPath(X,Y,\mathbf{Z},L)$ returns the first such a path in the order described above. Let $X = X_1,\ldots,X_k = Y$ denote the first such a path. It suffices to prove that at some point P contains (from the bottom to the top) the nodes $X_1,\ldots,X_j$ for any $1 \leq j \leq k$. We prove the result by induction over j.

The result is immediate for $j = 1$. Assume as induction hypothesis that the result holds for all $i < j$. We now prove the result for j. By the induction hypothesis, at some point t, P contains (from the bottom to the top) the nodes $X_1,\ldots X_{j-1}$. Pushing into P a node that is adjacent to X_{j-1} and appears before X_j in the adjacency list associated to X_{j-1} cannot lead to a path between X_1 and X_k, because that would contradict the assumption that $X = X_1,\ldots,X_k = Y$ is the first path between X and Y in G that is not blocked by $\mathbf{Z}$. Therefore, the algorithm will eventually pop from P every element pushed after time t and, then, X_j will be pushed into P.

Finally, note that if we reverse the order of the nodes in each adjacency list in L, then we also reverse the order of the paths described above. Consequently, $X = X_1,\ldots,X_k = Y$ is the last path in the order considered by $FirstPath(X,Y,\mathbf{Z},L')$. Thus, $FirstPath(X,Y,\mathbf{Z},L')$ does not return $X = X_1,\ldots,X_k = Y$ unless that is the only path between X and Y in G that is not blocked by $\mathbf{Z}$. ■

The algorithm that jointly implements sep and con is as follows. If $FirstPath(X,Y,(\mathbf{X}\setminus X)(\mathbf{Y}\setminus X)\mathbf{Z},L)$ and $FirstPath(X,Y,(\mathbf{X}\setminus X)(\mathbf{Y}\setminus X)\mathbf{Z},L')$ return the same path for some $X \in \mathbf{X}$ and $Y \in \mathbf{Y}$, then $con(\mathbf{X},\mathbf{Y}|\mathbf{Z})$. On the other hand, if $FirstPath(X,Y,(\mathbf{X}\setminus X)(\mathbf{Y}\setminus X)\mathbf{Z})$ returns no path for all $X \in \mathbf{X}$ and $Y \in \mathbf{Y}$, then $sep(\mathbf{X},\mathbf{Y}|\mathbf{Z})$. Finally, if neither of the two previous conditions is met, then neither $con(\mathbf{X},\mathbf{Y}|\mathbf{Z})$ nor $sep(\mathbf{X},\mathbf{Y}|\mathbf{Z})$. The algorithm is considered to require at most $O(n^2(e+n))$ time.

7. An Application to Bioinformatics

Our end-goal is to apply the results in this paper to our project on atherosclerosis gene expression data analysis in order to learn dependencies between genes. We believe that it is not unrealistic to assume that the probability distribution underlying our data satisfies strict positivity and weak transitivity and, thus, that it is a WT graphoid. We base this belief on the following argument. The cell is the functional unit of all the organisms and includes all the information necessary to regulate its function. This information is encoded in the DNA of the cell, which is divided into a set of genes, each coding for one or more proteins. Proteins are required for practically all the functions in the cell. The amount of protein produced depends on the expression level of the coding gene which, in turn, depends on the amount of proteins produced by other genes. Therefore, a dynamic BN seems to be a relatively accurate model of the cell: The nodes represent the genes and proteins, and the edges and parameters represent the causal relations between the gene expression levels and the protein amounts. As a matter of fact, dynamic BNs have become very popular models of gene networks for the last few years (Bernard and Hartemink, 2005; Friedman et al., 1998; Husmeier, 2003; Kim et al., 2003; Murphy and Mian, 1999; Ong et al., 2002; Perrin et al., 2003; Zou and Conzen, 2005). It is important that the BN is dynamic because a gene can regulate some of its regulators and even itself with some time delay. Since the technology for measuring the state of the protein nodes is not widely available yet, the data in most projects on gene expression data analysis is a finite sample of the probability distribution represented by the dynamic BN after marginalizing the protein nodes out. The probability distribution with no node marginalized out is, in the measure-theoretic sense discussed in Section 3, almost surely faithful to the dynamic BN (Meek, 1995) and, thus, it satisfies weak transitivity (Pearl, 1988) and, thus, so does the probability distribution after marginalizing the protein nodes out (see Theorem 2). The assumption that the probability distribution sampled is strictly positive is justified because measuring the state of the gene nodes involves a series of complex wet-lab and computer-assisted steps that introduces noise in the measurements (Sebastiani et al., 2003). Obviously, the reasoning above can be extended to include any other molecules that, in addition to proteins, regulate gene expression but are not measured.

In the rest of this section we focus on Gaussian graphical models (GGMs) of gene networks, which have received increasing attention from the bioinformatics community as a means to gain insight into gene networks (Castelo and Roverato, 2006; Dobra et al., 2004; Kishino and Waddell, 2000; Li and Gui, 2006; Schäfer and Strimmer, 2005a,b; Toh and Horimoto, 2002; Waddell and Kishino, 2000; Wang et al., 2003; Wu et al., 2003). Assume that each random variable in $\mathbf{U}$ represents (the expression level of) a gene in the network under study. Assume also that $\mathbf{U}$ follows a regular Gaussian probability distribution $\mathcal{N}(\mu, \Sigma)$. This is a ubiquitous assumption in bioinformatics. The GGM of the gene network is nothing else but the MUI map of $\mathcal{N}(\mu, \Sigma)$ (Lauritzen, 1996; Whittaker, 1990). Therefore, two genes X and Y are adjacent in the GGM iff $X \not\perp\!\!\!\perp Y | \mathbf{U} \setminus (XY)$ or, equivalently, iff $(\Sigma^{-1})_{XY} \neq 0$ (Lauritzen, 1996). In practice, Σ is unknown and the only information about it that is available is a finite sample from $\mathcal{N}(\mu, \Sigma)$. The usual way of proceeding in practice consists of two steps: First, estimating Σ from the sample and, then, making two genes adjacent in the GGM iff the corresponding entry in the inverse of the estimate of Σ significantly differs from zero. We refer the interested reader to the works cited above for different solutions to these two steps. Recall that regular Gaussian probability distributions are WT graphoids and, thus, that the results obtained in the previous sections of this paper apply to GGMs of gene networks.

The GGM of a gene network is a powerful tool for discovering gene independencies, because *sep* is sound and complete (in the sense discussed in Section 4) for reading independencies from it. However, bioinformaticians are typically more interested in discovering gene dependencies, because these provide contexts in which the expression level of some genes is informative about that of some other genes which, in some cases, can lead to hypothesize dependencies, functional

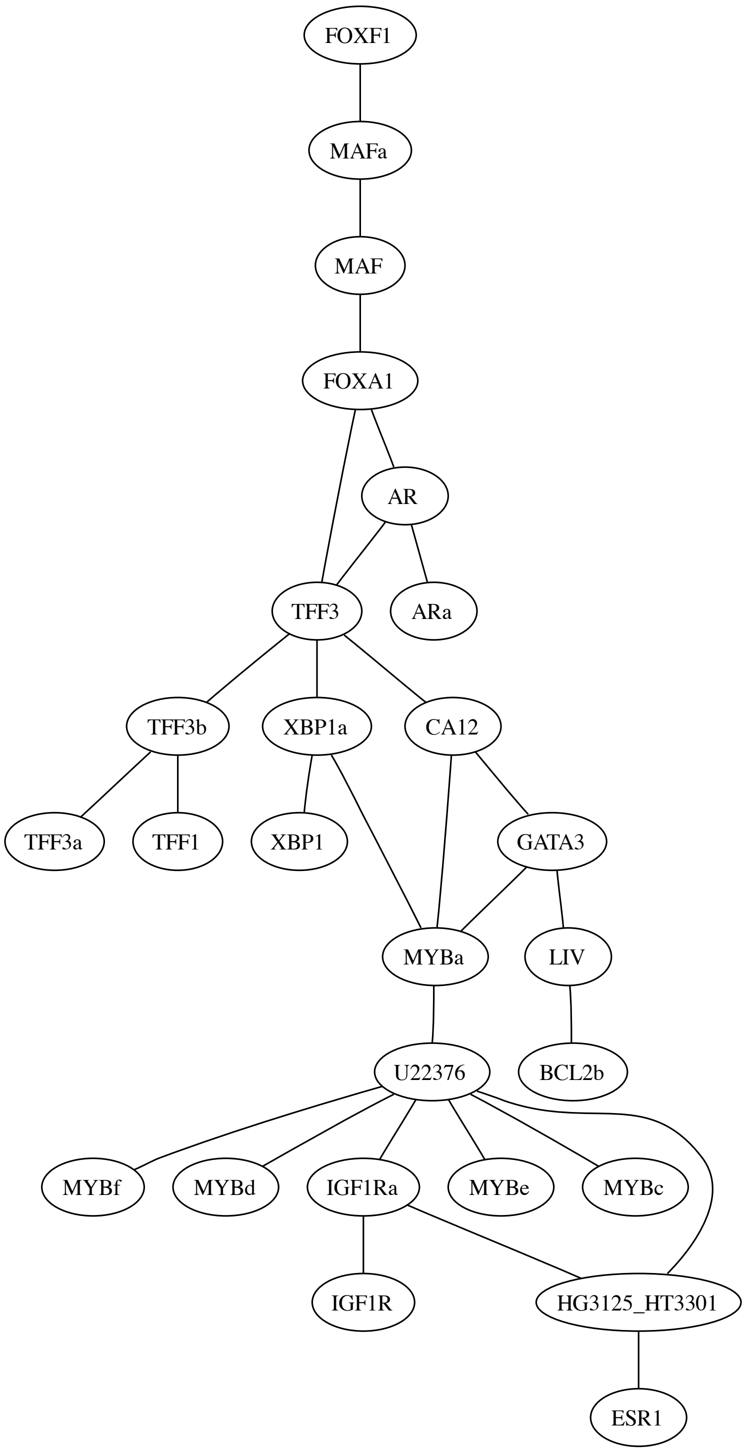


Figure 4: Estrogen receptor pathway.

relations, causal relations, the effects of manipulation experiments, etc. Thanks to *con*, which is sound and complete (in the sense discussed in Section 5) for reading dependencies from the GGM of a gene network, such a model is now a powerful tool for discovering gene dependencies too. We illustrate this with a real world example. In Dobra et al. (2004), a GGM over 12558 nodes is learnt from 158 breast cancer samples. Unfortunately, the GGM is neither reported in the paper nor available from the authors. However, the authors do report in the paper the subgraph of the GGM that is induced by some genes that are known to be related to the estrogen receptor (ER) and TFF1 genes. ER is a transcription factor that plays a key role in breast cancer, and TFF1 is a target of ER. This subgraph, depicted in Figure 4, suffices for our illustrative purposes. Note that the graph in the figure is not necessarily the GGM over the nodes in the graph. It is also worth mentioning that the nodes in the subgraph do not correspond to genes but to probe sets. A probe set is a collection of probes designed to measure the abundance of a particular DNA sequence. Since this sequence is for technical reasons usually shorter than that of a gene, a gene may have several probe sets associated with it, each measuring the abundance of a different subsequence of the gene. This aims at measuring gene expression more accurately. The following genes have multiple probe sets (nodes) in Figure 4: ER (ESR1, HG3125-HT3301), MYB (U22376, MYBa, MYBc, MYBd, MYBe, MYBf), AR (AR, ARa), c-MAF (MAF, MAFa), TFF3 (TFF3, TFF3a, TFF3b), XBP (XBP1, XBP1a), and IGF1R (IGF1R, IGF1Ra). According to Dobra et al. (2004), it is known that ER regulates TFF1, FOXA1 regulates TFF1, GATA3 possibly regulates TFF1, AR regulates ER, MAF inhibits MYB, FOXF1 possibly interacts with ER, and AR regulates IGF1R. Had these relations been unknown, we could have obtained principled clues about them by just applying *con* to the subgraph in Figure 4. For instance, if $\mathbf{U}$ denotes all the 12558 probe sets in the GGM, then *con* enables us to conclude that the following gene dependencies hold in the underlying probability distribution: ER is conditionally dependent on TFF1 since $con(\text{HG3125-HT3301}, \text{TFF1}|\mathbf{U} \setminus \{\text{HG3125-HT3301}, \text{U22376}, \text{MYBa}, \text{XBP1a}, \text{TFF3}, \text{TFF3b}, \text{TFF1}\})$, FOXA1 is conditionally dependent on TFF1 since $con(\text{FOXA1}, \text{TFF1}|\mathbf{U} \setminus \{\text{FOXA1}, \text{TFF3}, \text{TFF3b}, \text{TFF1}\})$, GATA3 is conditionally dependent on TFF1 since $con(\text{GATA3}, \text{TFF1}|\mathbf{U} \setminus \{\text{GATA3}, \text{CA12}, \text{TFF3}, \text{TFF3b}, \text{TFF1}\})$, AR is conditionally dependent on ER since $con(\text{AR}, \text{HG3125-HT3301}|\mathbf{U} \setminus \{\text{AR}, \text{TFF3}, \text{CA12}, \text{MYBa}, \text{U22376}, \text{HG3125-HT3301}\})$, MAF is conditionally dependent on MYB since $con(\text{MAF}, \text{MYBa}|\mathbf{U} \setminus \{\text{MAF}, \text{FOXA1}, \text{TFF3}, \text{CA12}, \text{MYBa}\})$, FOXF1 is conditionally dependent on ER since $con(\text{FOXF1}, \text{HG3125-HT3301}|\mathbf{U} \setminus \{\text{FOXF1}, \text{MAFa}, \text{MAF}, \text{FOXA1}, \text{TFF3}, \text{CA12}, \text{MYBa}, \text{U22376}, \text{HG3125-HT3301}\})$, and AR is conditionally dependent on IGF1R since $con(\text{AR}, \text{IGF1Ra}|\mathbf{U} \setminus \{\text{AR}, \text{TFF3}, \text{CA12}, \text{MYBa}, \text{U22376}, \text{IGF1Ra}\})$. Furthermore, *con* also enables us to conclude that $X \not\perp\!\!\!\perp Y|\mathbf{U} \setminus \{X, \text{U22376}, Y\}$ with $X, Y \in \{\text{MYBa}, \text{MYBc}, \text{MYBd}, \text{MYBe}, \text{MYBf}\}$ and $X \neq Y$, and that $\text{TFF3} \not\perp\!\!\!\perp \text{TFF3a} \,|\mathbf{U} \setminus \{\text{TFF3}, \text{TFF3b}, \text{TFF3a}\}$. These dependencies make sense as they are between different probe sets of the same gene. Note that none of the dependencies discussed above was used in the construction of the GGM. Note also that each of the dependencies discussed above involves a conditioning set of maximum size. It is very likely that these dependencies also hold for smaller conditioning sets, but we cannot confirm this point without seeing the complete GGM the subgraph in Figure 4 is part of which, as discussed above, is not available. In any case, it is clear that *con* improves the current interpretation of GGMs of gene networks by allowing reading biologically meaningful gene dependencies.

8. Discussion

The MUI map G of an independence model M is typically used to identify independencies that hold in M via vertex separation. However, lack of vertex separation in G does not necessarily imply dependency in M. In this paper, we have studied when lack of vertex separation does imply dependency. This should be relevant for those interested in graphical models, as it allows to infer from MUI maps (i.e., without numerical calculation) not only independencies but also dependencies. Specifically, in this paper we have introduced a graphical criterion called *con* for reading dependencies from G when M is a WT graphoid, that is, a graphoid that satisfies weak transitivity. Specifically, $con(\mathbf{X},\mathbf{Y}|\mathbf{Z})$ holds when there exist some $X_1 \in \mathbf{X}$ and $X_n \in \mathbf{Y}$ such that there exists exactly one path in G between X_1 and X_n that is not blocked by $(\mathbf{X} \setminus X_1)(\mathbf{Y} \setminus X_n)\mathbf{Z}$. We have proven that the criterion is sound and complete, where complete means that it is able to read all the dependencies in M that can be derived by applying the WT graphoid properties to the dependencies used in the construction of G and the independencies obtained from G by vertex separation. Note that our criterion remains inconclusive if there are several unblocked paths between any node in $\mathbf{X}$ and any node in $\mathbf{Y}$, though $\mathbf{X}$ and $\mathbf{Y}$ may be dependent given $\mathbf{Z}$ in M. However, we have shown in Section 5 that neither our criterion nor any other sound criterion can identify all the dependencies in M.

Note that our criterion is antimonotone in the following sense: If some edges are added to G then some *con* statements may not longer hold, whereas if some edges are removed from G then some new *con* statements may hold. This antimonotone property should be taken into account if one is to remove "weak" edges from G to make it sparser, because this may result in false dependencies being identified. However, this has nothing to do with the correctness of our criterion, something that we have proved, but with the fact that after removing "weak" edges G is an approximation to the true MUI map. One may consider to extend our work with a measure of confidence in the dependencies identified by our criterion. Such a measure could be a function of the confidence in the dependencies used in the construction of G. We have not pursued this idea further.

A work that is closely related to ours is Bouckaert (1995), which introduces the following graphical criterion, denoted *bou* here, for reading dependencies from the MUI map G of a graphoid M: $bou(\mathbf{X},\mathbf{Y}|\mathbf{Z})$ holds when there exist some $X_1 \in \mathbf{X}$ and $X_n \in \mathbf{Y}$ such that $X_1 \in MB(X_n)$ and either $MB(X_1) \setminus X_n \subseteq (\mathbf{X} \setminus X_1)(\mathbf{Y} \setminus X_n)\mathbf{Z}$ or $MB(X_n) \setminus X_1 \subseteq (\mathbf{X} \setminus X_1)(\mathbf{Y} \setminus X_n)\mathbf{Z}$. In other words, $bou(\mathbf{X},\mathbf{Y}|\mathbf{Z})$ holds when there exist two nodes, one in $\mathbf{X}$ and the other in $\mathbf{Y}$, that are neighbors in G and, moreover, the rest of the neighbors of one of them are among the rest of the nodes in the statement. The criterion is proven to be sound and complete, where complete means that it is able to identify all the dependencies in M that can be derived by applying the graphoid properties to the dependencies used in the construction of G and the independencies obtained from G by vertex separation.

Although *bou* is tailored to the case where M is a graphoid, it can obviously be applied when M is a WT graphoid. However, there are two main differences between *bou* and our criterion that make the latter more powerful when M is a WT graphoid. First, our criterion does not require that X_1 is adjacent to X_n in G, that is, there can be a path of length greater than one between X_1 and X_n. Second, our criterion does not require than all the nodes adjacent to either X_1 or X_n in G are among $\mathbf{XYZ}$, that is, all the paths between X_1 and X_n but one can be blocked by nodes that are neither adjacent to X_1 nor X_n. Consequently, our criterion represents an advantage over *bou* when M is a WT graphoid, as it allows us to identify a superset of the dependencies identified by *bou*. Interestingly, WT graphoids are a rich subclass of graphoids, including any regular Gaussian distribution, any probability distribution that is faithful to some UG or DAG, and all the marginals and conditionals of almost all the strictly positive discrete probability distributions that factorize according to a DAG. We believe that these probability distributions are encountered in many applications and, thus, that the work presented in this paper is of interest to the machine learning community. For instance, regular Gaussian distributions are ubiquitous. Likewise, it is rather usual to assume that the probability distribution underlying the domain at hand is strictly positive and faithful to some DAG, though one may be forced to work with a marginal of it because only a subset of the nodes in the DAG are observable. The strict positivity assumptions is usually justified by measurement errors, whereas the DAG faithfulness assumption is usually justified by the fact that many domains have a causality structure. For a more concrete case example, recall our discussion on bioinformatics applications in Section 7.

A problem that remains open is whether our criterion is complete in the sense that it identifies all the dependencies shared by all the WT graphoids for which G is the MUI map. This is a problem that we are currently studying though, as we have argued in Section 5, we incline to think that our criterion only identifies a proper subset of those dependencies. To the best of our knowledge, there has not been any attempt to solve this problem. As a matter of fact, one can think of an analogous problem for *bou* and the graphoids for which G is the MUI map. However, such a problem is not even mentioned in Bouckaert (1995).

Another problem we are currently working on is the development of a graphical criterion for reading dependencies from the minimal directed independence maps of WT graphoids. As a first step, we have derived such a criterion for the case where the minimal directed independence maps are polytrees (Peña, 2007).

Finally, it is worth recalling that, as an intermediate step in the derivation of our criterion, we have proved that for any UG G there exists a strictly positive discrete probability distribution that is faithful to G for any sample spaces (with at least two possible states) of the random variables in $\mathbf{U}$. For any DAG, an analogous result follows from Meek (1995). Both results are subsumed by our recent work in Peña (2009), which proves an analogous result for chain graphs. Note that such a result is stronger than Theorem 7.2 in Studený and Bouckaert (1998), which proves the result for some sample spaces (with at least two possible states) of the random variables in $\mathbf{U}$ but not for any such sample spaces.

Acknowledgments

This work is funded by the Swedish Research Council (ref. VR-621-2005-4202), the Wallenberg Foundation, the Ph.D. Programme in Medical Bioinformatics, the Swedish Foundation for Strategic Research, Clinical Gene Networks AB, and Linköping University. We thank the editor and the anonymous referees for their insightful comments.

References

Ann Becker, Dan Geiger and Christopher Meek. Perfect tree-like Markovian distributions. In *Proceedings on the Sixteenth Conference on Uncertainty in Artificial Intelligence*, pages 19-23, 2000.

Allister Bernard and Alexander J. Hartemink. Informative structure priors: Joint learning of dynamic regulatory networks from multiple types of data. In *Pacific Symposium on Biocomputing 2005*, pages 459-470, 2005.

Remco R. Bouckaert. *Bayesian belief networks: From construction to inference*. PhD Thesis, University of Utrecht, 1995.

Robert Castelo and Alberto Roverato. A robust procedure for Gaussian graphical model search from microarray data with p larger than n. *Journal of Machine Learning Research*, 7:2621-2650, 2006.

David M. Chickering and Christopher Meek. Finding optimal Bayesian networks. In *Proceedings of the Eighteenth Conference on Uncertainty in Artificial Intelligence*, pages 94-102, 2002.

Adrian Dobra, Chris Hans, Beatrix Jones, Joseph R. Nevins, Guang Yao and Mike West. Sparse graphical models for exploring gene expression data. *Journal of Multivariate Analysis*, 90:196-212, 2004.

Nir Friedman, Kevin Murphy and Stuart Russell. Learning the structure of dynamic probabilistic networks. In *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence*, pages 139-147, 1998.

Dan Geiger and Judea Pearl. Logical and algorithmic properties of conditional independence and graphical models. *The Annals of Statistics*, 21:2001-2021, 1993.

Paul R. Halmos. *Measure Theory*. Van Nostrand, 1966.

Dirk Husmeier. Sensitivity and specificity of inferring genetic regulatory interactions from microarray experiments with dynamic Bayesian networks. *Bioinformatics*, 19:2271-2282, 2003.

Sun Yong Kim, Seiya Imoto and Satoru Miyano. Dynamic Bayesian network and nonparametric regression for nonlinear modeling of gene networks from time series gene expression data. In *Proceedings of the First International Workshop on Computational Methods in Systems Biology*, pages 104-113, 2003.

Hirohisa Kishino and Peter J. Waddell. Correspondence analysis of genes and tissue types and finding genetic links from microarray data. *Genome Informatics*, 11:83-95, 2000.

Steffen L. Lauritzen. *Graphical Models*. Oxford University Press, 1996.

Hongzhe Li and Jiang Gui. Gradient directed regularization for sparse Gaussian concentration graphs, with applications to inference of genetic networks. *Biostatistics*, 7:302-317, 2006.

Radim Lněnička and František Matúš. On Gaussian conditional independence structures. *Kybernetika*, 43:327-342, 2007.

Christopher Meek. Strong completeness and faithfulness in Bayesian networks. In *Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence*, pages 411-418, 1995.

Kevin Murphy and Saira Mian. Modelling gene expression data using dynamic Bayesian networks. Technical Report, University of California, 1999.

Richard E. Neapolitan. *Learning Bayesian Networks*. Prentice Hall, 2003.

Masashi Okamoto. Distinctness of the eigenvalues of a quadratic form in a multivariate sample. *The Annals of Statistics*, 1:763-765, 1973.

Irene M. Ong, Jeremy D. Glasner and David Page. Modelling regulatory pathways in E. Coli from time series expression profiles. *Bioinformatics*, 18:S241-S248, 2002.

Judea Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, 1988.

Jose M. Peña. Faithfulness in chain graphs: The discrete case. *International Journal of Approximate Reasoning*. Under review, 2009.

Jose M. Peña, Roland Nilsson, Johan Björkegren and Jesper Tegnér. Identifying the relevant nodes without learning the model. In *Proceedings of the Twentysecond Conference on Uncertainy in Artificial Intelligence*, pages 367-374, 2006.

Jose M. Peña. Reading dependencies from polytree-like Bayesian networks. In *Proceedings of the Twentythird Conference on Uncertainty in Artificial Intelligence*, pages 303-309, 2007.

Bruno-Edouard Perrin, Liva Ralaivola, Aurélien Mazurie, Samuele Bottani, Jacques Mallet and Florence d'Alché-Buc. Gene networks inference using dynamic Bayesian networks. *Bioinformatics*, 19:ii138-ii148, 2003.

Juliane Schäfer and Korbinian Strimmer. Learning large-scale graphical Gaussian models from genomic data. In *Science of Complex Networks: From Biology to the Internet and WWW*, 2005a.

Juliane Schäfer and Korbinian Strimmer. An empirical Bayes approach to inferring large-scale gene association networks. *Bioinformatics*, 21:754-764, 2005b.

Juliane Schäfer and Korbinian Strimmer. A shrinkage approach to large-scale covariance matrix estimation and implications for functional genomics. *Statistical Applications in Genetics and Molecular Biology*, 4, 2005c.

Paola Sebastiani, Emanuela Gussoni, Isaac S. Kohane and Marco F. Ramoni. Statistical challenges in functional genomics (with discussion). *Statistical Science*, 18:33-60, 2003.

P. Stein. A note on the volume of a simplex. *The American Mathematical Monthly*, 73:299-301, 1966.

Milan Studený. *Probabilistic Conditional Independence Structures*. Springer, 2005.

Milan Studený and Remco R. Bouckaert. On chain graph models for description of conditional independence structures. *The Annals of Statistics*, 26:1434-1495, 1998.

Hiroyuki Toh and Katsuhisa Horimoto. Inference of a genetic network by a combined approach of cluster analysis and graphical Gaussian modeling. *Bioinformatics*, 18:287-297, 2002.

Peter J. Waddell and Hirohisa Kishino. Cluster inference methods and graphical models evaluated on NCI60 microarray gene expression data. *Genome Informatics*, 11:129-140, 2000.

Junbai Wang, Ola Myklebost and Eivind Hovig. MGraph: Graphical models for microarray analysis. *Bioinformatics*, 19:2210-2211, 2003.

Joe Whittaker. *Graphical Models in Applied Multivariate Statistics*. John Wiley & Sons, 1990.

Xintao Wu, Yong Ye and Kalpathi R. Subramanian. Interactive analysis of gene interactions using graphical Gaussian model. In *Proceedings of the Third ACM SIGKDD Workshop on Data Mining in Bioinformatics*, pages 63-69, 2003.

Min Zou and Suzanne D. Conzen. A new dynamic Bayesian network (DBN) approach for identifying gene regulatory networks from time course microarray data. *Bioinformatics*, 21:71-79, 2005.

Journal of Machine Learning Research 10 (2009) 1095-1129 Submitted 3/08; Revised 11/08; Published 5/09

Universal Kernel-Based Learning with Applications to Regular Languages

Leonid (Aryeh) Kontorovich ARYEH.KONTOROVICH@WEIZMANN.AC.IL
Department of Mathematics
Weizmann Institute of Science
Rehovot, Israel 76100

Boaz Nadler BOAZ.NADLER@WEIZMANN.AC.IL
Department of Computer Science and Applied Mathematics
Weizmann Institute of Science
Rehovot, Israel 76100

Editors: Paolo Frasconi, Kristian Kersting, Hannu Toivonen and Koji Tsuda

Abstract

We propose a novel framework for supervised learning of discrete concepts. Since the 1970's, the standard computational primitive has been to find the most consistent hypothesis in a given complexity class. In contrast, in this paper we propose a new basic operation: for each pair of input instances, count how many concepts of bounded complexity contain both of them.

Our approach maps instances to a Hilbert space, whose metric is induced by a universal kernel coinciding with our computational primitive, and identifies concepts with half-spaces. We prove that all concepts are linearly separable under this mapping. Hence, given a labeled sample and an oracle for evaluating the universal kernel, we can efficiently compute a linear classifier (via SVM, for example) and use margin bounds to control its generalization error. Even though exact evaluation of the universal kernel may be infeasible, in various natural situations it is efficiently approximable.

Though our approach is general, our main application is to regular languages. Our approach presents a substantial departure from current learning paradigms and in particular yields a novel method for learning this fundamental concept class. Unlike existing techniques, we make no structural assumptions on the corresponding unknown automata, the string distribution or the completeness of the training set. Instead, given a labeled sample our algorithm outputs a classifier with guaranteed distribution-free generalization bounds; to our knowledge, the proposed framework is the only one capable of achieving the latter. Along the way, we touch upon several fundamental questions in complexity, automata, and machine learning.

Keywords: grammar induction, regular language, finite state automaton, maximum margin hyperplane, kernel approximation

1. Introduction

We begin by describing the basic problem setting and outlining our approach.

1.1 Background

Perhaps the most fundamental problem in learning from labeled data is to construct a classifier with guaranteed small generalization error. Typically, given labeled data there exists a nested sequence of candidate concept classes with increasing complexity. Of course, a classifier from a rich concept class can easily achieve a low or even zero training error, but is likely to overfit the data and have a poor generalization performance. Hence, a major challenge is to choose the hypothesis class with appropriate complexity. This issue also arises in classical statistics, where it is known as the *model selection* problem (Rissanen, 1989). In Vapnik (1982), the structural risk minimization (SRM) principle was proposed to solve this problem, by quantifying the bias-variance tradeoff between fit to the data and model complexity, as captured by the VC-dimension.

The basic computational primitive in the SRM framework is finding the most consistent hypothesis in a given concept class. Although SRM resolves the information theoretic problem of model selection, it offers no method for finding such a hypothesis. Moreover, for discrete concept classes, finding this hypothesis may be infeasible, as this computational primitive typically requires solving a hard combinatorial optimization problem.

An important instance of this scenario is the problem of learning a regular language from labeled examples. Supervised language learning from examples is one of the most fundamental problems in learning theory, and as such has fascinated philosophers, linguists, computer scientists and mathematicians as a quintessential problem of induction. Insofar as the PAC model (Valiant, 1984) is a natural formalization of learning and the regular languages are the simplest nontrivial class of formal languages, the vast amount of literature devoted to learning regular languages is well justified (see Rivest and Schapire 1987 and de la Higuera 2005 and the references therein).

Applying the standard SRM principle to learning a regular language requires finding the smallest automaton consistent with the given labeled sample. However, it was realized early on that this task is computationally very difficult. Finding the smallest automaton consistent with a set of accepted and rejected strings was shown to be NP-complete by Angluin (1978) and Gold (1978); this was further strengthened in the hardness of approximation result of Pitt and Warmuth (1993), where it was proven that even finding a DFA with a number of states polynomial in the number of states of the minimum solution is NP-complete.

To make matters worse, it turns out that under cryptographic assumptions the class of regular languages is inherently hard to learn, regardless of representation. Thus, assuming the average-case difficulty of the Discrete Cube Root problem, Theorem 7.6 of Kearns and Vazirani (1997) shows how to construct small automata over $\{0,1\}$ and distributions over $\{0,1\}^n$ for which it is intractable to discover a hypothesis with error less than $1/2 - 1/p(n)$, for any fixed polynomial p. A similar construction shows that this task is also at least as hard as factoring.

Nevertheless, a number of positive learnability results were obtained under various restricted settings. Trakhtenbrot and Barzdin (1973) showed that the smallest finite automaton consistent with the input data can be learned exactly from a complete sample of all strings up to a given length. The worst case complexity of their algorithm is exponential in automaton size, but a better average-case complexity can be obtained assuming that the topology and the labeling are selected randomly or even that the topology is selected adversarially (Freund et al., 1993). In a different model of learnability—"identification in the limit" (Gold, 1967)—positive results were obtained for the k-reversible languages (Angluin, 1982) and subsequential transducers (Oncina et al., 1993). Some restricted classes of probabilistic automata such as acyclic probabilistic automata were also shown to be efficiently learnable by Ron et al. (1998). In a modified model of PAC, and with additional language complexity constraints, Clark and Thollard (2004) showed a class of probabilistic finite state automata to be learnable; see also literature review therein.

In light of the strong negative results mentioned above, the prevailing paradigm in formal language learning has been to make structural regularity assumptions about the family of languages and/or the sampling distribution in question and to employ a state-merging heuristic. Indeed, over the years a number of clever and sophisticated combinatorial approaches have been proposed for learning DFAs. Typically, an initial automaton or prefix tree consistent with the sample is first created. Then, starting with the trivial partition with one state per equivalence class, classes are merged while preserving an invariant congruence property. The automaton learned is obtained by merging states according to the resulting classes. Thus, the choice of the congruence determines the algorithm and generalization bounds are obtained from the structural regularity assumptions. This rough summary broadly characterizes the techniques of Angluin (1982); Oncina and Garcia (1992); Ron et al. (1998); Clark and Thollard (2004) and until recently, this appears to have been the only general-purpose technique available for learning finite automata.

Finally, despite the fact that the problems of inducing a regular language, and specifically finding the smallest automaton consistent with a given sample were proven to be theoretically computationally hard, in recent years various heuristic search methods have been developed with reported considerable success in solving these problems for moderately sized DFAs, see for example Lang (1992), and Oliveira and Silva (2001) and Bugalho and Oliveira (2005). Of course, these heuristic methods find some automaton consistent with the given sample, but provide no generalization bounds on its expected performance on new samples.

1.2 Learning Languages Via Hyperplanes

In this paper we propose a novel framework for learning a regular language from a finite sample of positive and negative labeled strings. Rather than attempting to find a *single* small consistent automaton, we embed the strings in a Hilbert space and compute a maximum margin hyperplane, which becomes our classifier for new strings. Hence, we effectively convert a difficult combinatorial problem into a (high dimensional) geometric one, amenable to efficient techniques using linear algebra and convex optimization. In particular, our resulting classifier is a linear combination of potentially smaller automata, each of which is typically not consistent with the training data.

Since the advent of Support Vector Machines (SVMs), maximum margin classifiers have flourished in the machine learning community (Schölkopf and Smola, 2002). Within the context of grammatical inference, the first work to apply this methodology was Kontorovich et al. (2006), where a kernel was used to learn a specific family of regular languages (the piecewise-testable ones). The authors embed the set of all strings onto a high-dimensional space and achieve language induction by constructing a maximum-margin hyperplane. This hinges on every language in a family of interest being *linearly separable* under the embedding, and on the efficient computability of the kernel. This line of research is continued in Cortes et al. (2007), where linear separability properties of rational kernels are investigated.

In this paper, we build upon the approach suggested in Kontorovich (2007), where a *universal* kernel is given that renders *all* regular languages linearly separable. We prove that any linearly separable language necessarily has a positive margin, so standard margin-based generalization guarantees apply. A by-product of this result is a novel characterization of the regular languages as precisely those that are linearly separable under our kernel. A drawback of this kernel is that a brute-force computation is infeasible, while an efficient one is currently unknown (and conjectured not to exist unless P= #P). However, we propose a simple randomized scheme for efficiently obtaining an ε-approximation to our universal kernel. We show that the approximate kernel preserves the distances with low distortion, and therefore may be used as a surrogate for the original one. Random sampling of embedding features to evaluate otherwise intractable kernels seems to first have been employed in Kontorovich (2007); Rahimi and Recht (2007) used a similar idea in a somewhat different setting. Also related are the approaches of Garg et al. (2002) and Balcan et al. (2006), where classifiers and their generalization performance are computed based on low-dimensional random projections. Note the marked difference between our method and theirs: we take a random subset of the embedding features as opposed to projecting the embedding space onto a random subspace; in fact, the setting we consider does not seem to allow the use of random projections. Also, note that in general, projecting the sample onto a low-dimensional subspace does not guarantee linear separability; a somewhat delicate margin analysis is needed, as described in Section 5.

From a practical standpoint, our resulting classifier is the sign of a weighted linear combination of possibly many random DFAs. The fact that a complicated DFA with many states may be written as a weighted sum of much simpler DFAs is illustrated below and provides additional motivation for our approach. Our methodology is intimately connected to boosting (see, for example, Bartlett et al. 1998), since our resulting classifier can also be viewed as a weighted sum of weak classifiers—each being one of the random DFAs. Furthermore, our approach has a natural interpretation in the structural risk minimization (SRM) framework, where we trade the computational problem of finding a small consistent hypothesis for the problem of counting the number of small concepts accepting a given instance. The advantage of the latter is that it admits an efficient approximation in many cases of interest, including the case of DFAs.

In summary, our proposed algorithm is efficient, simple to implement, comes with strong theoretical guarantees, and readily generalizes to many other learning settings, such as context-free languages. We also present preliminary empirical results as a proof of concept. To our knowledge, the framework we propose is the only one capable of inducing unrestricted regular languages from labeled samples, without imposing additional structural assumptions—modulo standard cryptographic limitations. Some of the results in this paper first appeared in the conference version (Kontorovich, 2007).

1.3 Outline of Paper

This paper is organized as follows. In Section 2 we set down the notation and definitions used throughout the paper. The notion of linearly separable concept classes is defined in Section 3, and a general theorem regarding their existence via a canonical embedding is proved in Section 4. In Section 5, the universal kernel oracle is used to obtain an efficient generic learning algorithm, and in Section 6 an efficient approximation scheme is given for computing the universal kernel, with a corresponding learning algorithm. In Section 7, these results are applied to the case of the regular languages, with some experimental results presented in Section 8. Some inherent limitations of such kernel-based techniques are discussed in Section 9, and concluding remarks are made in Section 10.

2. Preliminaries

We refer the reader to Kearns and Vazirani (1997) for standard learning-theoretic definitions such as *instance space* and *concept class*. We likewise use standard set-theoretic terminology, with $|\cdot|$ denoting set cardinalities and $\mathbb{1}_{\{\cdot\}}$ representing the 0-1 truth value of the predicate inside the brackets. We blur the distinctions between subsets $c \subset \mathcal{X}$ and binary functions $c : \mathcal{X} \to \{0,1\}$; when we wish to make the distinction explicit, we will use a $\{\pm 1\}$-valued characteristic function:

$$\chi_c(x) := 2\mathbb{1}_{\{x \in c\}} - 1.$$

Recall that if A and B are two sets, B^A represents the collection of all functions from A to B. For countable sets A, we will consider the vector space $\mathbb{R}^A$ and often index $x \in \mathbb{R}^A$ by elements of A as opposed to by the naturals. For example, for $x, y \in \mathbb{R}^A$, we define their *inner product* as

$$\langle x, y \rangle \quad = \quad \sum_{a \in A} [x]_a [y]_a,$$

whenever the latter sum is well-defined. Square brackets may be omitted from the indexing notation for readability.

For any $x \in \mathbb{R}^{\mathbb{N}}$, we define its *support* to be the set of its nonzero coordinates:

$$\operatorname{supp}(x) \quad = \quad \{i \in \mathbb{N} : x_i \neq 0\}.$$

We define the quasinorm $\|\cdot\|_0$ to be the number of the nonzero coordinates:

$$\|x\|_0 \quad = \quad |\operatorname{supp}(x)|,$$

and the standard ℓ_2 norm by $\|x\|_2^2 = \langle x, x \rangle$.

If V_1 and V_2 are two inner product spaces over $\mathbb{R}$, we define their *direct product* $V_1 \otimes V_2$ to be the vector space

$$V_1 \otimes V_2 \quad = \quad \{(x, y) : x \in V_1, y \in V_2\}$$

which naturally inherits vector addition and scalar multiplication from V_1 and V_2, and whose inner product is defined by

$$\left\langle (x, y), (x', y') \right\rangle \quad = \quad \left\langle x, x' \right\rangle + \left\langle y, y' \right\rangle.$$

We will also write $(x, y) \in V_1 \otimes V_2$ as $x \otimes y$. Note that $\|x \otimes y\|_0 = \|x\|_0 + \|y\|_0$.

3. Linearly Separable Concept Classes

Our main results are actually quite general and most of them make no use of the properties of regular languages and automata. The only property we use is the *countability* of the instance space $\mathcal{X}$ and of the concept class $\mathcal{C}$. Henceforth, $\mathcal{X}$ and $\mathcal{C}$ are always assumed to be (at most) countable, unless noted otherwise. Let $\mathcal{C}$ be a concept class defined over an instance space $\mathcal{X}$; thus, $\mathcal{C} \subseteq 2^{\mathcal{X}}$. Any $c \in \mathcal{C}$ is thus a classifier or a function that induces a $\{0, 1\}$ labeling on the instance space $\mathcal{X}$. Note that a concept $c \subset \mathcal{X}$ may contain infinitely many instances, and an instance $x \in \mathcal{X}$ may belong to an infinite number of concepts.

We will say that a concept $c \in \mathcal{C}$ is *finitely linearly separable* if there exists a mapping $\phi : \mathcal{X} \to \{0, 1\}^{\mathbb{N}}$ and a weight vector $w \in \mathbb{R}^{\mathbb{N}}$, both with *finite support*, that is, $\|w\|_0 < \infty$ and $\|\phi(x)\|_0 < \infty$ for all $x \in \mathcal{X}$, such that

$$c = \{x \in \mathcal{X} : \langle w, \phi(x) \rangle > 0\}.$$

The concept class $\mathcal{C}$ is said to be *finitely linearly separable* if all $c \in \mathcal{C}$ are finitely linearly separable under the *same* mapping ϕ.

Note that the condition $\|\phi(\cdot)\|_0 < \infty$ is important; otherwise, we could define the *embedding by concept* $\phi : \mathcal{X} \to \{0,1\}^{\mathcal{C}}$

$$[\phi(x)]_c \quad = \quad \mathbb{1}_{\{x \in c\}}, \qquad c \in \mathcal{C}$$

and for any target $\hat{c} \in \mathcal{C}$,

$$[w]_c \quad = \quad \mathbb{1}_{\{c = \hat{c}\}}.$$

This construction trivially ensures that

$$\langle w, \phi(x) \rangle \quad = \quad \mathbb{1}_{\{x \in \hat{c}\}}, \qquad x \in \mathcal{X},$$

but may not satisfy $\|\phi(\cdot)\|_0 < \infty$ since certain $x \in \mathcal{X}$ may belong to an infinite number of concepts.

Similarly, we disallow $\|w\|_0 = \infty$ due to the algorithmic impossibility of storing infinitely many numbers and also because it leads to the trivial construction, via *embedding by instance* $\phi : \mathcal{X} \to \{0,1\}^{\mathcal{X}}$

$$[\phi(x)]_u \quad = \quad \mathbb{1}_{\{x = u\}}, \qquad u \in \mathcal{X},$$

whereby for any target $\hat{c} \in \mathcal{C}$, the corresponding vector

$$[w]_u \quad = \quad \mathbb{1}_{\{u \in \hat{c}\}}$$

ensures $\langle w, \phi(x) \rangle = \mathbb{1}_{\{x \in \hat{c}\}}$ without doing anything interesting or useful.

An additional important reason to insist on finite linear separability is that it ensures that for each $c \in \mathcal{C}$ there is a "separability dimension" $D(c) < \infty$ such that c is linearly separable in $\{0,1\}^{D(c)}$ under ϕ (in particular, if w defines a separating hyperplane for c then $D(c) \le \|w\|_0$). Now any linearly separable partition of $\{0,1\}^D$ must necessarily be separable with a strictly positive margin; this is true on any discrete finite space (but false in $\{0,1\}^\infty$). The latter in turn provides generalization guarantees, as spelled out in Section 5. Finally, any embedding ϕ induces a kernel K via

$$K(x,y) \quad = \quad \langle \phi(x), \phi(y) \rangle \tag{1}$$

for $x, y \in \mathcal{X}$. The finite support of ϕ automatically guarantees that K is everywhere well-defined; otherwise, additional assumptions are needed.

In light of the discussion above, from now on the only kind of linear separability we shall consider is in the strong sense defined above, where both the embedding $\phi(\cdot)$ and the hyperplane vector w have finite support. The modifiers "countable" (for instance spaces and concept classes), "finitely" (for linear separability) and "finite support" (for embeddings and hyperplanes) will be implicitly assumed throughout. Since an embedding $\phi : \mathcal{X} \to \{0,1\}^{\mathbb{N}}$ induces a kernel K as in (1), and any positive definite[1] kernel $K : \mathcal{X}^2 \to \mathbb{R}$ induces an embedding $\phi : \mathcal{X} \to \{0,1\}^{\mathcal{X}}$ via

$$\phi(x) \quad = \quad K(x, \cdot)$$

(see Schölkopf and Smola 2002), we shall speak of linear separability by ϕ or by K interchangeably, as dictated by convenience.

An immediate question is whether, under our conventions, every concept class is linearly separable. A construction of the requisite ϕ given any $\mathcal{X}$ and $\mathcal{C}$ would provide a positive answer; an example of $\mathcal{X}$ and $\mathcal{C}$ for which no such embedding exists would resolve the question negatively.

1. In the sense that for any finite set $\{x_i \in \mathcal{X} : 1 \le i \le m\}$, the $m \times m$ matrix $G = (g_{ij})$ given by $g_{ij} = K(x_i, x_j)$ is always positive definite.

4. Every Concept Class is Linearly Separable

The question raised in Section 3 turns out to have an affirmative answer, as we prove in the following theorem.

Theorem 1 *Every countable concept class $\mathcal{C}$ over a countable instance space $\mathcal{X}$ is finitely linearly separable.*

Proof We present a constructive proof, by describing an explicit mapping $\phi : \mathcal{X} \to \{0,1\}^{\mathbb{N}}$ under which all concepts $c \in C$ are finitely linearly separable. Recall that both the embedding by concept and embedding by instance, described in the previous section, rendered all concepts linearly separable, but were possibly not finitely supported. Hence, the "trick" in our construction is the introduction of *size functions* that measure the *complexity* of both individual instances and individual concepts, thus providing a notion of capacity control and adaptive regularization, and leading to finite separability.

We thus begin by defining the notion of a *size function* $f : \mathcal{A} \to \mathbb{N}$, mapping any countable set $\mathcal{A}$ to the naturals. We require a size function to have finite level sets:

$$\left|f^{-1}(n)\right| \quad < \quad \infty, \qquad \forall n \in \mathbb{N}.$$

In words, f assigns at most finitely many elements of $\mathcal{A}$ to any fixed size. Any countable $\mathcal{A}$ has such a size function (in fact, there are infinitely many size functions on $\mathcal{A}$). We denote by $|\cdot|$ and $\|\cdot\|$ some fixed choices of size functions on $\mathcal{X}$ and $\mathcal{C}$, respectively.

Recall that our goal is to construct an embedding $\phi : \mathcal{X} \to \{0,1\}^{\mathbb{N}}$ with $\|\phi(x)\|_0 < \infty$ for all $x \in \mathcal{X}$ such that for all $c \in \mathcal{C}$ there is a $w = w(c) \in \mathbb{R}^{\mathbb{N}}$ with $\|w\|_0 < \infty$ such that

$$c = \{x \in \mathcal{X} : \langle w, \phi(x) \rangle > 0\}.$$

We will construct the requisite *canonical* embedding ϕ as the direct product of two auxiliary embeddings, α and β,

$$\phi(x) = \alpha(x) \otimes \beta(x). \tag{2}$$

First, we define the *embedding by instance* $\alpha : \mathcal{X} \to \{0,1\}^{\mathcal{X}}$ by

$$[\alpha(x)]_u \quad = \quad \mathbb{1}_{\{u=x\}}, \qquad u \in \mathcal{X};$$

obviously, $\|\alpha(x)\|_0 = 1$ for all $x \in \mathcal{X}$ (recall our vector-indexing conventions, set down in Section 2). Second, using the size function as notion of complexity, we define a *regularized embedding by concept* $\beta : \mathcal{X} \to \{0,1\}^{\mathcal{C}}$ by

$$[\beta(x)]_c \quad = \quad \mathbb{1}_{\{x \in c\}} \mathbb{1}_{\{\|c\| \le |x|\}}, \qquad c \in \mathcal{C}; \tag{3}$$

since size functions have finite level sets, in contrast to the standard embedding by concept, we have $\|\beta(x)\|_0 < \infty$ for any $x \in \mathcal{X}$.

We now show that the embedding (2) renders all concepts $c \in \mathcal{C}$ finitely linearly separable, by explicitly constructing hyperplane vectors $w^{\alpha} \in \mathbb{R}^{\mathcal{X}}$ and $w^{\beta} \in \mathbb{R}^{\mathcal{C}}$ corresponding to the mappings α and β, respectively. To this end, it is helpful to keep in mind the dual roles of $\mathcal{X}$ and $\mathcal{C}$.

Pick any $\hat{c} \in C$. Since the embedding α involved no regularization, we introduce a complexity restriction in the corresponding hyperplane vector $w^{\alpha} \in \mathbb{R}^{\mathcal{X}}$ as follows

$$[w^{\alpha}]_u \quad = \quad \mathbb{1}_{\{u \in \hat{c}\}} \mathbb{1}_{\{|u| < \|\hat{c}\|\}}, \qquad u \in \mathcal{X};$$

since size functions have finite level sets, we have $\|w^\alpha\|_0 < \infty$. Thus,

$$\begin{aligned}\langle w^\alpha, \alpha(x)\rangle &= \sum_{u\in\mathcal{X}} [w^\alpha]_u [\alpha(x)]_u \\ &= \sum_{u\in\mathcal{X}} \mathbb{1}_{\{u\in\hat{c}\}} \mathbb{1}_{\{|u|<\|\hat{c}\|\}} \mathbb{1}_{\{x=u\}} \\ &= \mathbb{1}_{\{x\in\hat{c}\}} \mathbb{1}_{\{|x|<\|\hat{c}\|\}}. \end{aligned} \tag{4}$$

For the second embedding β, no further regularization is needed, and we define its corresponding hyperplane vector $w^\beta \in \mathbb{R}^{\mathcal{C}}$ by

$$[w^\beta]_c = \mathbb{1}_{\{c=\hat{c}\}}, \qquad c \in \mathcal{C};$$

note that $\left\|w^\beta\right\|_0 = 1$. Now

$$\begin{aligned}\left\langle w^\beta, \beta(x)\right\rangle &= \sum_{c\in\mathcal{C}} [w^\beta]_c [\beta(x)]_c \\ &= \sum_{c\in\mathcal{C}} \mathbb{1}_{\{c=\hat{c}\}} \mathbb{1}_{\{x\in c\}} \mathbb{1}_{\{\|c\|\le|x|\}} \\ &= \mathbb{1}_{\{x\in\hat{c}\}} \mathbb{1}_{\{|x|\ge\|\hat{c}\|\}}. \end{aligned} \tag{5}$$

Since ϕ is the direct product of the two embeddings α and β, the corresponding hyperplane is the direct product of the two hyperplanes:

$$w = w^\alpha \otimes w^\beta.$$

Note that by construction, both ϕ and w are finite:

$$\|\phi(x)\|_0 = \|\alpha(x)\|_0 + \|\beta(x)\|_0 < \infty \quad \text{and} \quad \|w\|_0 = \|w^\alpha\|_0 + \|w^\beta\|_0 < \infty.$$

Combining (4) and (5), we get

$$\begin{aligned}\langle w, \phi(x)\rangle &= \langle w^\alpha, \alpha(x)\rangle + \left\langle w^\beta, \beta(x)\right\rangle \\ &= \mathbb{1}_{\{x\in\hat{c}\}} \mathbb{1}_{\{|x|<\|\hat{c}\|\}} + \mathbb{1}_{\{x\in\hat{c}\}} \mathbb{1}_{\{|x|\ge\|\hat{c}\|\}} \\ &= \mathbb{1}_{\{x\in\hat{c}\}} \end{aligned}$$

which shows that w is indeed a linear separator (with finite support) for $\hat{c}$. ■

Next, we observe that linearly separable concepts may be described by finitely many instances, via the following "representer theorem". Note that this result holds for *any* separating embedding—not just the canonical one constructed in Theorem 1.

Theorem 2 *If a concept class $\mathcal{C}$ over an instance space $\mathcal{X}$ is linearly separable under ϕ (equivalently, under the induced kernel $K(x,y) = \langle\phi(x), \phi(y)\rangle$), there are $s_i \in \mathcal{X}$ and $\alpha_i \in \mathbb{R}$, $i = 1, \ldots, m$, such that*

$$c = \left\{ x \in \mathcal{X} : \sum_{i=1}^{m} \alpha_i K(s_i, x) > 0 \right\}.$$

Proof Suppose a concept $c \in \mathcal{C}$ is linearly separable with vector w, that is, $c = \{x \in \mathcal{X} : \langle w, \phi(x) \rangle > 0\}$. It follows that c is also separable under the finite dimensional embedding $\phi_D : \mathcal{X} \to \{0,1\}^D$ where $D = \max\{i : w_i \neq 0\}$ and ϕ_D is the truncation of ϕ by D:

$$[\phi_D(x)]_i \quad = \quad [\phi(x)]_i, \qquad i = 1, \ldots, D.$$

Define the equivalence relation $\equiv_D$ on $\mathcal{X}$ via

$$x \equiv_D y \quad \Leftrightarrow \quad \phi_D(x) = \phi_D(y)$$

and notice that $\equiv_D$ partitions $\mathcal{X}$ into finitely many equivalence classes. Let $x_1, x_2, \ldots, x_m$ be chosen arbitrarily, one from each equivalence class, and define $y_i := \chi_c(x_i)$. Let w' be the maximum-margin hyperplane for the labeled sample $(x_i, y_i)_{i=1}^m$ (see (9) for the definition of margin). This construction ensures that w' is a linear separator for c. By the representer theorem of Kimeldorf and Wahba (1971), it follows that w' admits a representation of the form

$$w' = \sum_{i=1}^{m} \alpha_i \phi(x_i),$$

and the claim follows. ■

A few remarks regarding Theorem 1 are in order. First, note that the construction of the canonical embedding depends on the choice of the size functions $|\cdot|$ and $\|\cdot\|$ on $\mathcal{X}$ and $\mathcal{C}$. These size functions induce a natural notion of complexity for instances $x \in \mathcal{X}$ and concepts $c \in \mathcal{C}$. Hence, our construction has analogies to various settings where classifier complexity can adaptively grow with the sample, such as structural risk minimization (SRM) in machine learning (Vapnik, 1982), and minimum description length (MDL) and other information theoretic criteria in statistics (Rissanen, 1989). We shall have more to say about SRM in particular, below.

Observe that any nondecreasing transformation $f : \mathbb{N} \to \mathbb{N}$ of a size function $|\cdot|$ produces another valid size function $|x|' := f(|x|)$. These affect the embedding in the following way. Consider a canonical embedding ϕ induced by $(|\cdot|, \|\cdot\|)$, and consider a transformed embedding ϕ' induced by $(f(|\cdot|), \|\cdot\|)$. If f is a very rapidly growing function then the truncation in (3) becomes ineffective, and in the limit of $f(\cdot) \equiv \infty$, ϕ' essentially becomes a pure embedding by concept. In the other extreme of a very slowly increasing f, the truncation in (3) becomes too restrictive. In the limit of $f(\cdot) \equiv$ const, ϕ' effectively becomes an embedding by instance.

A final note is that the separability result does not preclude the trivial scenario where the only coordinate separating a concept is the concept itself. To parse the last statement, recall that Theorem 1 states that under the canonical embedding, for each concept $c \in \mathcal{C}$ there is a finite $N = N(c)$ such that c is separable in $\{0,1\}^N$. However, it may be the case that the separator w is trivial, in the sense that $w_i \equiv \mathbb{1}_{\{i=c\}}$. In other words, the separability result implies that for each $c \in \mathcal{C}$ there is a finite collection of $\{c_i \in \mathcal{C} : \|c_i\| \leq \|c\|, 1 \leq i \leq m\}$ with coefficients $\alpha_i \in \mathbb{R}$ such that

$$c \quad = \quad \sum_{i=1}^{m} \alpha_i c_i, \tag{6}$$

where the relation above is symbolic shorthand for the statement that

$$c \quad = \quad \{x \in \mathcal{X} : \sum_{i=1}^{m} \alpha_i \mathbb{1}_{\{x \in c_i\}} > 0\}.$$

When a relation of type (6) holds, we say that c is *linearly decomposable* into the concepts $\{c_i\}$. No claims of uniqueness are made, and certainly any concept c is trivially decomposable into the singleton $\{c\}$. However, in many situations, concepts indeed decompose. As a simple example, define the concept c_{10} to be the set of all binary numbers, written as strings in $\{0,1\}^*$, divisible by ten, and define c_1, c_2 and c_5 analogously. Then it is straightforward to verify that

$$c_{10} = c_2 + c_5 - c_1,$$

in the sense of (6), since a number is divisible by 10 iff it is divisible by both 2 and 5.

As far as implications for learnability, Theorem 1 is most relevant when the target concept is linearly decomposable into a *small* collection of *simpler* concepts, as illustrated in the last example. In such cases, our approach can offer a substantial advantage over existing methods.

We may formalize the latter observation as a "universal learnability" theorem:

Theorem 3 *Let $\mathcal{C}$ be a concept class over $\mathcal{X}$, both countable, with size functions $\|\cdot\|$ and $|\cdot|$, respectively. Define the family of kernels:*

$$K_n(x,y) \quad = \quad \sum_{c:\|c\|\le n} \mathbb{1}_{\{x\in c\}}\mathbb{1}_{\{y\in c\}} \tag{7}$$

for $n \in \mathbb{N}$. Then, given an oracle for computing K_n, we can efficiently learn $\mathcal{C}$ from labeled examples.

More explicitly, let $c \in \mathcal{C}$ be a fixed concept, and let S be a sequence of m instances independently sampled from an arbitrary distribution P, and labeled according to c.

There is an efficient algorithm `UnivLearn`*, which inputs a labeled sample and outputs a classifier $\hat{f} : \mathcal{X} \to \{-1,1\}$ which, with probability at least $1-\delta$, achieves a small generalization error:*

$$P\{\hat{f}(x) \ne \chi_c(x)\} \quad \le \quad \frac{2}{m}\left(R(c)\log(8em)\log(32m)+\log(8m/\delta)\right), \quad \text{for } m \ge \theta^{-1}(\|c\|), \tag{8}$$

where $R(c)$ does not depend on the distribution P and $\theta, \theta^{-1} : \mathbb{N} \to \mathbb{N}$ are fixed monotone increasing functions with polynomial growth.

We defer the proof of Theorem 3 to Section 5, as it requires some preliminary definitions and results. The regularization function θ is chosen by the learner and represents his desired tradeoff between computational complexity and generalization error; this is further elucidated in Remark 7. We stress that the size of the target concept, $\|c\|$, is unknown to the learner—otherwise regularization would not be necessary. Finally, Remarks 4 and 5 below point out the non-trivial nature of Theorem 3; we are not aware of any way to obtain it as an immediate corollary of known results.

Remark 4 *We emphasize that the bound in Theorem 3 is distribution-free and recall that a distribution-dependent "learning" bound is trivial to obtain. Define $\hat{f}$ to be the rote memorizer: $\hat{f}(x) = y(x)$ if the example x appears in the sample S with label y and $\hat{f}(x) = -1$ otherwise. In this case, we have*

$$P\{\hat{f}(x) \ne \chi_c(x)\} \le P(x \notin S) = P(\mathcal{X} \setminus S),$$

and the latter quantity clearly tends to 0 as $m \to \infty$. All this bound says is that when we have observed a large enough portion of the world, we can be sure that most test examples will simply be recycled training examples. However, for any concept containing infinitely many instances, one can find an adversarial distribution P so that $P(\mathcal{X} \setminus S)$ will tend to zero arbitrarily slowly.

Remark 5 *The dependence of the required number of samples on the unknown concept c in the generalization bound (8) is inevitable whenever $\mathcal{C}$ has infinite VC-dimension. As a simple example, note that to learn an n-state DFA, one needs to observe at least n strings (at least one string ending at each state).*

5. Margin Bounds

In this section, we recall some classic margin bounds and their implications for learnability. Let $\mathcal{C}$ be a concept class over the instance space $\mathcal{X}$. Suppose there is some family of embeddings $\phi_d : \mathcal{X} \to \{0,1\}^{N_d}$; at this point, we assume nothing about the relationship between $\{\phi_d\}$ and $\mathcal{C}$. Let $S \subset \mathcal{X}$ be a finite sample with labels $Y \in \{-1,1\}^S$ consistent with some $c \in \mathcal{C}$, that is, $Y_x = \chi_c(x) = 2\mathbb{1}_{\{x\in c\}} - 1$ for all $x \in S$.

We define the *sample margin* of (S,Y) under ϕ_d as follows:

$$\hat{\gamma}_d(S,Y) \quad := \quad \sup_{w\in B(N_d)} \min_{x\in S} Y_x \langle w, \phi_d(x)\rangle \tag{9}$$

where $B(k) := \{x \in \mathbb{R}^k : \|x\|_2 \leq 1\}$. The *sample radius* of S under ϕ_d is defined to be

$$\hat{\rho}_d(S) \quad := \quad \max_{x\in S} \|\phi_d(x)\|_2 .$$

Analogously, define the *intrinsic margin* of c under ϕ_d to be

$$\gamma_d(c) \quad = \quad \sup_{w\in B(N_d)} \inf_{x\in \mathcal{X}} \chi_c(x) \langle w, \phi_d(x)\rangle$$

and the *radius* of ϕ_d to be

$$\rho_d \quad = \quad \sup_{x\in \mathcal{X}} \|\phi_d(x)\|_2 .$$

The case where γ_d or $\hat{\gamma}_d$ is negative is not excluded; it arises when ϕ_d fails to separate the given concept or sample. Note that since for $A \subset B$, we have $\inf_A f \geq \inf_B f$ and $\sup_A f \leq \sup_B f$, the relation $S \subset \mathcal{X}$ implies that

$$\hat{\gamma}_d(S,Y) \geq \gamma_d(c) \quad \text{and} \quad \hat{\rho}_d(S) \leq \rho_d . \tag{10}$$

The following theorem is a slight rewording of Shawe-Taylor et al. (1998, Theorem 4.5):

Theorem 6 (Shawe-Taylor et al. 1998) *Suppose m instances are drawn independently from a distribution whose support is contained in a ball in $\mathbb{R}^n$ centered at the origin, of radius ρ. If we succeed in correctly classifying all of them by a hyperplane through the origin with margin γ, then with confidence $1-\delta$ the generalization error will be bounded from above by*

$$\frac{2}{m}\left(R\log(8em/R)\log(32m) + \log(8m/\delta)\right), \tag{11}$$

where $R = R(\rho,\gamma) = \lfloor 577\rho^2/\gamma^2 \rfloor$.

We are now in a position to prove the "universal learnability" theorem:

Proof [of Theorem 3] Recall that (S,Y) is our finite labeled sample consistent with some unknown target concept c. For each $n \in \mathbb{N}$, let $\phi_n : \mathcal{X} \to \{0,1\}^{N_n}$ be the embedding associated with K_n, defined in (7), where

$$N_n = |\mathcal{C}_n| = |\{c \in \mathcal{C} : \|c\| \le n\}|.$$

Let the margins $\gamma_n(c), \hat{\gamma}_n(S,Y)$ and radii $\rho_n, \hat{\rho}_n(S)$ be as defined above. Define the *normalized margins*

$$\Delta_n(c) := \frac{\gamma_n(c)}{\rho_n}, \quad \text{and} \quad \hat{\Delta}_n(S,Y) := \frac{\hat{\gamma}_n(S,Y)}{\hat{\rho}_n(S)}.$$

Additionally, define $n_0 = n_0(c)$ to be the smallest n for which K_n linearly separates c:

$$n_0(c) := \min\{n \in \mathbb{N} : \gamma_n(c) > 0\} \tag{12}$$

and define

$$\Delta_0(c) := \Delta_{n_0(c)}(c).$$

The algorithm `UnivLearn` requires some fixed monotone increasing function $\theta : \mathbb{N} \to \mathbb{N}$, as mentioned in the statement of the theorem (for a concrete choice, one may take $\theta(m) = \lceil m^{1/2} \rceil$). Recall the theorem's condition that $m \ge \theta^{-1}(\|c\|)$ which (by Theorem 1) implies that $K_{\theta(m)}$ indeed separates the unknown concept c. The algorithm proceeds by computing the normalized sample margins $\hat{\Delta}_n(S,Y)$ for $n = 1, \ldots, \theta(m)$ and sets

$$\hat{\Delta}_*(S,Y) := \max\{\hat{\Delta}_n(S,Y) : 1 \le n \le \theta(m),\ \hat{\gamma}_n(S,Y) > 0\}$$

with corresponding $n_* \in \{1, \ldots, \theta(m)\}$. The condition $m \ge \theta^{-1}(\|c\|)$ ensures that the latter is well defined.

By (10) we have that $\hat{\Delta}_n(S,Y) \ge \Delta_n(c)$ for all $\{n \in \mathbb{N} : \gamma_n(c) > 0\}$. In addition, the condition $m \ge \theta^{-1}(\|c\|)$ combined with Theorem 1 ensure that

$$n_0(c) \le \|c\| \le \theta(m).$$

Therefore, we have

$$\hat{\Delta}_*(S,Y) \ge \Delta_0(c). \tag{13}$$

The classifier $\hat{f} : \mathcal{X} \to \{-1,1\}$ which `UnivLearn` outputs is the maximum-margin hyperplane under the embedding ϕ_{n_*}. It follows from Theorem 6 that the bound in (8) holds with $R(c) = \lfloor 577/\Delta_0(c)^2 \rfloor$. Note that in (8) we have opted for a slightly coarser but simpler bounds.

■

Remark 7 *To explain the role of the function $\theta(m)$, let us attempt to simplify the algorithm* `UnivLearn`. *Recall the definition of n_0 in (12), and define $\hat{n}_0$ to be the first n for which we perfectly classify the training set, that is, $\hat{n}_0 = \min_n\{\hat{\gamma}_n(S,Y) > 0\}$ and let $\hat{\Delta}_0(S,Y) := \hat{\Delta}_{\hat{n}_0}(S,Y)$. If it were the case that*

$$\hat{\Delta}_0(S,Y) \ge \Delta_0(c), \tag{14}$$

then the maximum-margin classifier $\hat{f}$ corresponding to $\phi_{\hat{n}_0}$ would satisfy the desired bound in (8). Unfortunately, the assertion in (14) is generally false, since it is possible for a small sample to be separable under K_n with $n < n_0$ and margin $\hat{\gamma}_n(S,Y) \ll \gamma_{n_0}(c)$. Because of this potentially very small margin, stopping prematurely at this n would render our theorem false. There seems to be no way around requiring the learner to try every $n \in \mathbb{N}$. To make the algorithm terminate in finite time for each given sample, we cut off the search at $n = \theta(m)$. Note that the theorem is only claimed to hold for sufficiently large sample sizes (namely, $m \geq \theta^{-1}(\|c\|)$)—but this is a feature of many bounds, especially where the complexity of the target concept is unknown (cf. Theorem 8). Since θ is a monotonically increasing function, we are guaranteed that eventually the sample size m will attain $\theta^{-1}(\|c\|)$, which is what the claim in (13) hinges upon. If θ grows too quickly (say, $\theta(\cdot) = \Omega(\exp(\cdot))$), the algorithm will take too long to run. If θ grows too slowly (say, $\theta(\cdot) = O(\log(\cdot))$), the generalization bound will only hold for huge sample sizes. The suggested rate of $\theta(m) = \lceil m^{1/2} \rceil$) seems a reasonable compromise.

Theorem 3 has a natural interpretation in the structural risk minimization (SRM) framework (Vapnik, 1982). Let us quote a well-known result, as it appears in Shawe-Taylor et al. (1996); for a detailed discussion of VC-dimension, see Vapnik's books (Vapnik, 1982, 1998) or any standard reference on learning theory such as Kearns and Vazirani (1997).

Theorem 8 (Shawe-Taylor et al. 1996) *Let H_i, $i = 1,2,\ldots$ be a sequence of hypothesis classes mapping $\mathcal{X}$ to $\{0,1\}$ such that $\mathrm{VCdim}(H_i) = i$ and let P be a probability distribution on $\mathcal{X}$. Let p_d be any set of positive numbers satisfying $\sum_{d=1}^{\infty} p_d = 1$. With probability $1-\delta$ over m independent examples drawn according to P, for any d for which the learner finds a consistent hypothesis h in H_d, the generalization error of h is bounded above by*

$$\varepsilon(m,d,\delta) = \frac{4}{m}\left(d\log\left(\frac{2em}{d}\right) + \log\left(\frac{1}{p_d}\right) + \log\left(\frac{4}{\delta}\right)\right),$$

provided $m \geq d$.

As a concrete choice of p_d, one may always take $p_d = 2^{-d}$. Viewing Theorems 3 and 8 through a computational lens, the two approaches may be contrasted by their "computational primitives". The SRM approach requires one to find a consistent hypothesis in a given complexity class (if one exists). Whenever the latter problem is efficiently solvable, SRM is quite satisfactory as a learning paradigm. In many interesting cases, however, the latter problem is intractable—as is the case for DFAs, for example (see Introduction). Our approach in Theorem 3 is to consider a different computational primitive—namely, the generic kernel

$$K_n(x,y) = |\{c \in \mathcal{C} : \|c\| \leq n, x \in c, y \in c\}|.$$

The margin-based bound may be significantly sharper than the generic SRM bound since it is sensitive to the target concept and its intrinsic margin. Additionally, the SRM bound is not directly applicable to our setting since the VC-dimension of linear combinations of functions may grow linearly with the number of terms (Anthony and Bartlett, 1999). The roles of θ in Theorem 3 and $\{p_d\}$ in Theorem 8 are analogous in that both encode a prior belief regarding hypothesis complexity.

The performance of `UnivLearn` is readily quantified:

Corollary 9 *If a labeled sample of size m is consistent with some concept $c \in \mathcal{C}$, and there is an oracle for computing $K_n(x,y)$ in time $O(1)$, then there is an algorithm with running time $O(m^2\theta(m))$ that achieves, with probability $1-\delta$, a generalization error of at most*

$$\frac{2}{m}\left(R(c)\log(8em)\log(32m)+\log(8m/\delta)\right),$$

for $m \geq \theta^{-1}(\|c\|)$.

Proof Using the oracle, the nth Gram matrix, defined by $(G_n)_{ij} = K_n(x_i,x_j)$, for $x_i, x_j \in S$, is computable in time $O(m^2)$. For a given Gram matrix, the margin may be computed in time $O(m)$ by a primal algorithm such as Pegasos (Shalev-Shwartz et al., 2007). The different Gram matrices and margins are computed $\theta(m)$ times. ■

Of course, at this point we have simply traded the generally intractable problem of finding a consistent $h \in H_d$ for the problem of evaluating $K_n(x,y)$. The latter is unlikely to have an efficient solution in general, and may well be intractable in many cases of interest. However, as we shall see below, under natural assumptions K_n admits an efficient stochastic approximation $\tilde{K}_n$ and Corollary 9 has a suitable extension in terms of $\tilde{K}_n$.

6. Approximating K_n

In the previous section we showed that the generic kernel family K_n renders all concepts $c \in \mathcal{C}$ linearly separable. However, exact evaluation of this kernel involves the summation in (7), which may contain a super-exponential number of terms. For example, a natural size function on DFAs is the number of states. Since the VC-dimension of n-state DFAs is $\Theta(n\log n)$ (Ishigami and Tani, 1997), there are $2^{\Theta(n\log n)}$ such concepts. Hence, a direct brute-force evaluation of the corresponding K_n is out of the question. Though we consider the complexity of computing K_n for DFAs to be a likely candidate for #P-complete, we have no proof of this; there is also the possibility that some symmetry in the problem will enable a clever efficient computation.

Instead, we propose an efficient randomized approximation to the generic K_n. All we require are oracles for computing $|\mathcal{C}_n|$ and for uniform sampling from $\mathcal{C}_n$, where

$$\mathcal{C}_n := \{c \in \mathcal{C} : \|c\| \leq n\}: \tag{15}$$

Assumption 10

(i) There is a sampling algorithm with running time $T_{\text{SAM}}(\mathcal{C},n) = O(\text{poly}(n))$ that outputs random concepts $c \in \mathcal{C}_n$, each with probability $|\mathcal{C}_n|^{-1}$. Additionally, k distinct concepts may be drawn (i.e., without replacement) in expected time $O(kT_{\text{SAM}}(\mathcal{C},n))$.

(ii) There is an algorithm that computes $|\mathcal{C}_n|$, in time $O(1)$.

We make an additional assumption regarding the time complexity of instance/concept membership queries:

Assumption 11 *For each $x \in \mathcal{X}$ and $c \in \mathcal{C}$, the characteristic function $\chi_c(x) = 2\mathbb{1}_{\{x\in c\}} - 1$ is computable in time $O(|x|)$.*

Note that sampling without replacement is a mild condition. Collisions are likely to occur when sampling from small sets, in which case one could simply enumerate the elements of $\mathcal{C}_n$. When the latter is large, however, uniform sampling is rather unlikely to generate collisions, and various hashing schemes can be employed to ensure this in practice.

For the purpose of approximating the kernel, it is convenient to normalize it as follows:

$$\bar{K}_n(x,y) := |\mathcal{C}_n|^{-1} K_n(x,y), \tag{16}$$

Note that the margin bounds of Section 5 are invariant under rescaling of the kernel by any constant.

Having stated our basic assumptions, we are ready to prove some results. The first one deals with efficient approximation of the generic kernel:

Theorem 12 *Let $S \subset \mathcal{X}$ be a sample of size m and the normalized generic kernel $\bar{K}_n$ be defined as in (16). Then, for any $0 < \varepsilon, \delta < 1$, there is a randomized algorithm with deterministic running time*

$$T \quad = \quad O\left(\frac{1}{\varepsilon^2} T_{\text{SAM}}(\mathcal{C}, n) \log\left(\frac{m}{\delta}\right)\right),$$

that produces a (random) kernel $\tilde{K}_n$ with the following properties

(a) $\tilde{K}_n$ is a positive semidefinite function on $\mathcal{X}^2$

(b)

$$\mathbf{E}[\tilde{K}_n(x,y)] \quad = \quad \bar{K}_n(x,y)$$

for all $x, y \in \mathcal{X}$

(c) with probability at least $1 - \delta$,

$$\left|\bar{K}_n(x,y) - \tilde{K}_n(x,y)\right| \quad \leq \quad \varepsilon,$$

for all $x, y \in S$,

(d) for all $x, y \in \mathcal{X}$, $\tilde{K}_n(x,y)$ is computable in deterministic time

$$O\left(\frac{1}{\varepsilon^2} \log\left(\frac{m}{\delta}\right)(|x| + |y|)\right).$$

Proof The approximation algorithm amounts to sampling T concepts, $\{c_i : 1 \leq i \leq T\}$ uniformly at random (without replacement) from $\mathcal{C}_n$, where Assumption 10 ensures that the latter sampling scheme is computationally feasible.

The random concepts $\{c_i : 1 \leq i \leq T\}$ are used to define the random kernel $\tilde{K}_n$:

$$\tilde{K}_n(x,y) \quad = \quad \frac{1}{T} \sum_{i=1}^{T} \mathbb{1}_{\{x \in c_i\}} \mathbb{1}_{\{y \in c_i\}}, \qquad x, y \in \mathcal{X}. \tag{17}$$

Since $\tilde{K}_n$ is constructed as a (semi) inner product, it is non-negative but still not strictly positive definite, as it may fail to separate distinct points. Hence, (a) is proved.

To prove (b), fix $x, y \in \mathcal{X}$ and define $D_{xy} \subset \mathcal{C}_n$ by

$$D_{xy} \quad = \quad \{c \in \mathcal{C}_n : x, y \in c\};$$

this is the set of those $c \in \mathcal{C}$ with $\|c\| \leq n$ which contain both x and y. Since any c drawn uniformly from $\mathcal{C}_n$ contains both x and y with probability $|D_{xy}| \, / \, |\mathcal{C}_n|$ it follows that the approximate kernel

$$\tilde{K}_n(x,y) = \frac{1}{T} \sum_{i=1}^{T} \mathbb{1}_{\{c_i \in D_{xy}\}} \tag{18}$$

is an unbiased estimator for

$$\bar{K}_n(x,y) \quad = \quad |\mathcal{C}_n|^{-1} \sum_{c \in \mathcal{C}_n} \mathbb{1}_{\{c \in D_{xy}\}} \quad = \quad \frac{|D_{xy}|}{|\mathcal{C}_n|};$$

this proves (b).

Our arguments show that the random variable $\mathbb{1}_{\{c_i \in D_{xy}\}}$ has a Bernoulli distribution with mean $\bar{K}_n(x,y)$. Hence, applying Hoeffding's bound (Hoeffding, 1963), which also holds for variables sampled without replacement, to (18) yields

$$\mathbf{P}\left\{ \left| \tilde{K}_n(x,y) - \bar{K}_n(x,y) \right| > \varepsilon \right\} \quad \leq \quad 2\exp(-2T\varepsilon^2). \tag{19}$$

The latter can be inverted to make the claim in (c) hold for a given x, y whenever $T > (2\varepsilon^2)^{-1} \log(2/\delta)$. Applying the union bound to (19), we have established (c), with

$$T \quad = \quad \frac{1}{2\varepsilon^2} \log\left(\frac{m^2 + m}{\delta} \right).$$

Finally, (d) holds by Assumptions 10 and 11. ■

Our next observation is that perturbing a kernel pointwise by a small additive error preserves linear separability:

Theorem 13 *Let $K : \mathcal{X}^2 \to \mathbb{R}$ be a kernel, and $S \subset \mathcal{X}$ be a finite sample with labels $Y \in \{-1, 1\}^S$. Assume that (S, Y) is separable under the kernel K with margin γ, that is,*

$$\sup_{w \in \mathbb{R}^{\mathbb{N}}} \min_{s \in S} \frac{Y_s \langle w, \phi(s) \rangle}{\|w\|_2} \quad \geq \quad \gamma > 0,$$

where ϕ is the embedding associated with K. Let K' be another kernel, which is uniformly close to K on S:

$$\left| K(s,t) - K'(s,t) \right| \leq \varepsilon \qquad \forall s, t \in S. \tag{20}$$

Then, for $\varepsilon < \gamma^2$, the sample (S, Y) is also separable under K', with a positive margin.

Proof Let $f(x)$ be the maximum-margin hyperplane classifier corresponding to the training set S, under the kernel K. By the representer theorem, it can written as

$$f(x) = \langle \phi(x), w \rangle = \sum_{s \in S} \alpha_s K(x,s).$$

Since (S,Y) is linearly separable, $f(x)$ satisfies the constraint

$$Y_s f(s) \geq 1 \qquad \forall s \in S.$$

By the KKT conditions (Schölkopf and Smola, 2002), its margin is given by

$$\gamma^{-2} \;\; = \;\; \sum_s |\alpha_s|.$$

We now consider the following linear classifier, obtained by replacing K by K':

$$f'(x) = \sum_{s \in S} \alpha_s K'(x,s).$$

By (20), for each $x, s \in S$,

$$K'(x,s) = K(x,s) + \varepsilon\, \eta_{x,s}$$

where $|\eta_{x,s}| \leq 1$. Hence, for all $x \in S$,

$$f'(x) = f(x) + \varepsilon \sum_{s \in S} \eta_{x,s} \alpha_s.$$

Therefore,

$$Y_s f'(s) \geq 1 - \varepsilon \sum_s |\alpha_s| = 1 - \frac{\varepsilon}{\gamma^2},$$

which shows that f' is a linear separator for (S,Y) as long as $\varepsilon < \gamma^2$. ■

The next result quantifies the amount by which the margin may shrink when an approximate kernel is constructed using a random subset of the features:

Theorem 14 *Let a labeled sample (S,Y) be given, with $S \subset \mathcal{X}$. Let the kernel K be induced by the embedding $\phi : \mathcal{X} \to \{0,1\}^F$, where F is a finite feature set, as follows:*

$$K(s,t) := \frac{1}{|F|} \langle \phi(s), \phi(t) \rangle .$$

Consider a feature subset $F' \subset F$. Define $\phi' : \mathcal{X} \to \{0,1\}^{F'}$ to be the restriction of ϕ to $\{0,1\}^{F'}$ and K' to be corresponding kernel:

$$K'(s,t) := \frac{1}{|F'|} \langle \phi'(s), \phi'(t) \rangle .$$

Suppose that (S,Y) is linearly separable under K with margin $\gamma > 0$. If K' is ε-close to K on S in the sense of (20) with $\varepsilon < \gamma^2$, then (S,Y) is linearly separable under the kernel K', with margin γ' bounded from below by

$$\gamma' \geq \frac{|F'|}{|F|} \left(\gamma - \frac{\varepsilon}{\gamma} \right).$$

Proof Let $f : \mathcal{X} \to \mathbb{R}$ be the maximum-margin separator under K:

$$f(x) = \sum_{s \in S} \alpha_s K(x,s) = \frac{1}{|F|} \sum_{s \in S} \alpha_s \langle \phi(x), \phi(s) \rangle = \langle w, \phi(x) \rangle .$$

Let us define f' as f with K replaced by K':

$$f'(x) = \sum_{s \in S} \alpha_s K'(x,s) = \frac{1}{|F'|} \sum_{s \in S} \alpha_s \left\langle \phi'(x), \phi'(s) \right\rangle .$$

By Theorem 13 we have that f' is a linear separator for (S,Y), satisfying

$$Y_s f'(s) \geq 1 - \varepsilon \sum_s \alpha_s = 1 - \frac{\varepsilon}{\gamma^2}, \qquad s \in S.$$

Observe that

$$f'(x) = \frac{|F|}{|F'|} \left\langle w|_{F'}, \phi'(x) \right\rangle$$

where $w|_{F'}$ is the restriction of w to $\mathbb{R}^{F'}$. Thus, letting

$$w' := \frac{|F|}{|F'|} \left(1 - \frac{\varepsilon}{\gamma^2} \right)^{-1} w|_{F'},$$

we have that

$$f''(x) := \left\langle w', \phi'(x) \right\rangle$$

linearly separates S and satisfies and $Y_s f''(s) \geq 1$ for $s \in S$. Since the margin attained by the hyperplane w is given by $1/\|w\|$ (Schölkopf and Smola, 2002), the claim follows. ■

Our final result for this section—and perhaps the highlight of the paper—is the following "approximate" version of Corollary 9:

Theorem 15 *Let $\mathcal{C}$ be a concept class over $\mathcal{X}$. Let $S \subset \mathcal{X}$ be a sequence of m instances sampled independently from an arbitrary distribution P, and labeled according to some unknown $c \in \mathcal{C}$. For any $\delta > 0$, there is a randomized algorithm* `ApproxUnivLearn`*, which outputs a classifier $\hat{f} : \mathcal{X} \to \{-1,1\}$ such that with probability at least $1-\delta$ it achieves a small generalization error,*

$$P\{\hat{f}(x) \neq \chi_c(x)\} \quad \leq \quad \frac{2}{m}\left(4R(c)\log(8em)\log(32m) + \log(16m/\delta)\right) \tag{21}$$

for

$$m \geq \max\{\theta^{-1}(\|c\|), D(c)/2, R(c)^2/8.4 \times 10^4\}, \tag{22}$$

where $0 < D(c), R(c) < \infty$ are constants that depend only on c, and $\theta : \mathbb{N} \to \mathbb{N}$ is discussed in Remark 7.

Furthermore, `ApproxUnivLearn` *has deterministic polynomial complexity, with running time*

$$O\left(m\theta(m)\log(m/\delta)(\ell_{\max} m^2 + T_{\text{SAM}}(\mathcal{C}, \theta(m)))\right),$$

where $\ell_{\max} := \max\{|s| : s \in S\})$.

Proof Let (S,Y) denote a training set consisting of m samples, labeled consistently with the target concept $c \in \mathcal{C}$. We define

$$\begin{aligned} n_0 &:= \min\{n \in \mathbb{N} : \gamma_n(c) > 0\}, \\ \gamma_0(c) &:= \gamma_{n_0}(c), \end{aligned}$$

and

$$\begin{aligned} R(c) &:= \left\lfloor 577/\gamma_0(c)^2 \right\rfloor, \\ D(c) &:= |\mathcal{C}_{n_0}|, \end{aligned}$$

where $\mathcal{C}_n$ is defined in (15) as the set of concepts with complexity at most n.

Given the training set (S,Y), our algorithm constructs $\theta(m)$ different classifiers. According to Theorem 12, each of these classifiers is constructed as follows: For $1 \le n \le \theta(m)$, we randomly sample T features, construct the respective approximate kernel, and calculate its resulting max-margin hyperplane, with sample margin $\tilde{\gamma}_n(S,Y)$. Finally, our algorithm chooses the kernel with the largest empirical margin

$$\tilde{\gamma}_* := \max\{\tilde{\gamma}_n(S,Y) : 1 \le n \le \theta(m), \tilde{\gamma}_n(S,Y) > 0\}. \tag{23}$$

If the latter set is empty, we leave $\tilde{\gamma}_*$ undefined (however, our analysis below will show that under the conditions of the theorem, this is a highly unlikely event).

To prove the theorem, we need to show that with high probability,

$$\tilde{\gamma}_* \ge \text{Const} \times \gamma_0(c). \tag{24}$$

If this equation holds (whp), then the standard margin bound (11) proves the theorem. There are two ways in which the theorem's generalization bound (21) may fail to hold. The first is due to a particularly unlucky sample whereas the second is due to a bad kernel approximation so that (24) does not hold. Hence, we split the confidence parameter δ so that each of these kinds of failure occurs with probability at most $\delta/2$.

The first step is to note that by definition $\tilde{\gamma}_* \ge \tilde{\gamma}_{n_0}$. The next step is to lower bound the approximate sample margin, constructed with a random approximate kernel, in terms of the true sample margin corresponding to the exact kernel. Such a bound is provided by Theorem 14:

$$\tilde{\gamma}_{n_0}(S,Y) \ge \frac{\min\{T, D(c)\}}{D(c)} \left(\hat{\gamma}_{n_0}(S,Y) - \frac{\varepsilon}{\hat{\gamma}_{n_0}(S,Y)} \right).$$

Note that by condition (22) we have

$$m \quad \ge \quad 4/\gamma_0(c)^4 \ge 4/\hat{\gamma}_{n_0}(S,Y)^4, \tag{25}$$

since $\hat{\gamma}_{n_0}(S,Y) \ge \gamma_0(c)$, as in (10).

Furthermore, in constructing the approximate kernels, we set the precision to be $\varepsilon = 1/\sqrt{m}$. This condition, combined with (25) gives

$$\tilde{\gamma}_* \ge \frac{1}{2} \frac{\min\{T, D(c)\}}{D(c)} \hat{\gamma}_{n_0}(S,Y) \ge \frac{1}{2} \frac{\min\{T, D(c)\}}{D(c)} \gamma_0(c).$$

The last step is to bound the factor multiplying $\gamma_0(c)$ in the last equation. To this end, recall that according to Theorem 12, under the condition $\varepsilon = 1/\sqrt{m}$, the number of random concepts T drawn from $\mathcal{C}_n$ is

$$T \;=\; \frac{1}{2\varepsilon^2}\log\left(\frac{m^2+m}{\delta/2}\right) = \frac{m}{2}\log\left(\frac{m^2+m}{\delta/2}\right) > m.$$

Combining this with (22), we have

$$T > m \geq D(c)/2.$$

Hence, $\tilde{\gamma}_* \geq \gamma_0(c)/4$, and applying (11) proves the generalization bound.

It remains to analyze the complexity of `ApproxUnivLearn`. The main loop is executed $\theta(m)$ times. To compute each approximate kernel we sample $O(\varepsilon^{-2}\log(m/\delta))$ concepts from $\mathcal{C}_n$, and evaluate the summation in (17) for each $x,y \in S$. Since $\varepsilon = m^{-1/2}$, the nth Gram matrix is computable in time $O\left(m\log(m/\delta)(\ell_{\max}m^2 + T_{\text{SAM}}(\mathcal{C},n))\right)$. For a given Gram matrix, the margin may be computed in time $O(m)$ by a primal algorithm such as Pegasos (Shalev-Shwartz et al., 2007), which gets absorbed into the preceding $O(m^3\theta(m)\log(m))$ bound.

■

Remark 16 *For* $\theta(m) = \lceil m^{1/2}\rceil$, `ApproxUnivLearn` *has a running time of* $O(m^{3.5}\log m)$. *This may be brought down to* $m^{2+\alpha}$ *for any* $\alpha > 0$*, at the expense of increased sample complexity in (21), by choosing* $\varepsilon = m^{-\beta}$ *with* $\beta \ll 1$. *The role of* $\theta(m)$ *in the tradeoff between running time and sample complexity is discussed in Remark 7;* $\varepsilon(m)$*, taken to be* $m^{-1/2}$ *in the proof above, has an analogous role.*

7. Universal Regular Kernel, New Characterization of Regular Languages

Having developed some general tools for learning countable concept classes, we now apply these to regular languages. For the remainder of the paper, we follow standard formal-language terminology. Thus, Σ will denote a fixed finite alphabet and Σ^* is the free monoid over Σ. Its elements are called *strings* (or *words*) and the length of $x \in \Sigma^*$ is denoted by $|x|$. The latter will be our default size function on Σ^*.

Our definition of a Deterministic Finite-state Automaton (DFA) is a slight modification of the standard one. We define the latter to be a tuple $A = (\Sigma, Q, F, \delta)$ where

- Σ is a finite alphabet
- $Q = \{1,2,\ldots,n\}$ is a finite set of states
- $F \subset Q$ is the set of the accepting states
- $\delta : Q \times \Sigma \to Q$ is the deterministic transition function.

The only difference between our definition and the common one is that we take the initial state q_0 to be fixed at $q_0 = 1$, while most authors allow it to be an arbitrary $q_0 \in Q$. We write $L(A)$ for the language accepted by the automaton A, and denote the number of states in A by size(A) or $\|A\|$, interchangeably.

Recall that a language $L \subseteq \Sigma^*$ is *regular* if and only if it is recognized by some DFA. Lewis and Papadimitriou (1981) and Sipser (2005) are among the standard introductory texts on automata and formal languages.

In order to apply our results to regular languages, we begin by defining the instance space $\mathcal{X} = \Sigma^*$ and concept class $\mathcal{C} = \cup_{n \geq 1} \mathrm{DFA}(n)$, where $\mathrm{DFA}(n)$ is the set of all DFAs on n states; both are countable. Once the size functions on $\mathcal{X}$ and $\mathcal{C}$ have been specified, Theorem 1 furnishes an embedding $\phi : \mathcal{X} \to \{0,1\}^{\mathbb{N}}$ that renders all regular languages linearly separable. It is instructive to verify that the construction of the canonical embedding yields the following kernel, which we call a *universal regular kernel*:

$$\begin{aligned} K_{\mathrm{REG}}(x,y) &= \langle \phi(x), \phi(y) \rangle \\ &= \mathbb{1}_{\{x=y\}} + \sum_{n=1}^{\min\{|x|,|y|\}} \sum_{A \in \mathrm{DFA}(n)} \mathbb{1}_{\{x \in L(A)\}} \mathbb{1}_{\{y \in L(A)\}}. \end{aligned}$$

In fact, we obtain a novel characterization of the regular languages:

Theorem 17 *A language $L \subseteq \Sigma^*$ is regular if and only if there exists a finite set of strings, $s_i \in \Sigma^*$ with corresponding weights $\alpha_i \in \mathbb{R}$, $i = 1, \ldots, m$ such that*

$$L = \left\{ x \in \Sigma^* : \sum_{i=1}^{m} \alpha_i K_{\mathrm{REG}}(s_i, x) > 0 \right\}. \tag{26}$$

Proof Theorem 1 provides an embedding $\phi : \Sigma^* \to \{0,1\}^{\mathbb{N}}$ such that for any regular language L there is a $D \in \mathbb{N}$ so that the sets $\phi_D(L)$ and $\phi_D(\Sigma^* \setminus L)$ are separable by a hyperplane in $\{0,1\}^D$, where ϕ_D is the restriction of ϕ to $\{0,1\}^D$. Thus, by Theorem 2, (26) holds for some finite collection of strings s_i and weights α_i. Conversely, suppose that $L \subseteq \Sigma^*$ is expressible as in (26). Writing $w = \sum_{i=1}^{m} \alpha_i \phi_D(x_i) \in \mathbb{R}^D$, consider the function $f : \Sigma^* \to \mathbb{R}$ defined by

$$f(x) = \sum_{j=1}^{D} w_j [\phi_D(x)]_j$$

and note that $L = \{x : f(x) > 0\}$. Observe that f can only take on finitely many real values $\{r_k : k = 1, \ldots, k_f\}$. Let $L_{r_k} \subseteq \Sigma^*$ be defined by

$$L_{r_k} = f^{-1}(r_k).$$

A subset $I \subseteq \{1, 2, \ldots, N\}$ is said to be *r_k-acceptable* if $\sum_{i \in I} w_i = r_k$. Any such r_k-acceptable set corresponds to a set of strings $L_I \subseteq \Sigma^*$ such that

$$L_I = \left(\bigcap_{i \in I} [\phi_D]_i^{-1}(1) \right) \setminus \left(\bigcup_{i \in \{1,\ldots,N\} \setminus I} [\phi_D]_i^{-1}(1) \right).$$

It remains to show that each $[\phi_D]_i^{-1}(1) \subseteq \Sigma^*$ is a regular language, but this follows immediately from Theorem 1 and our choice of size functions $|\cdot|$ and $\|\cdot\|$. Now each L_{r_k} is the union of finitely many r_k-acceptable L_I's, and L is the union of the L_{r_k} for $r_k > 0$. This means that L is a finite Boolean combination of regular languages and is therefore itself regular. ■

The result above is mainly of theoretical interest. In order to obtain an efficient learning algorithm with generalization bounds, we must stratify K_{REG} into a regularized kernel family $\{K_n\}$ and show that these are efficiently computable (or approximable).

For our purposes, $A \in \text{DFA}(n)$ is a directed multigraph on n labeled vertices (states), with edges labeled by elements in Σ, and some of the vertices designated as accepting. Note that a given regular language has infinitely many representations as various DFAs; we make no attempt to minimize automata or identify isomorphic ones, nor do we exclude degenerate or disconnected ones.

The first order of business is to verify that Assumptions 10 and 11 hold for the present choice of $\mathcal{X}$ and $\mathcal{C}$, with the specified size functions:

Lemma 18 *Let*

$$\mathcal{C}_n = \bigcup_{k=1}^{n} \text{DFA}(k).$$

Then

(i) There is a sampling algorithm with running time $T_{\text{SAM}}(\mathcal{C}, n) = O(|\Sigma| n)$ that outputs random automata $A \in \mathcal{C}_n$, each with probability $|\mathcal{C}_n|^{-1}$

(ii) $|\mathcal{C}_n|$ is computable in time $O(1)$

(iii) for each $x \in \mathcal{X}$ and $A \in \text{DFA}(k)$, the characteristic function $\chi_A(x) = 2\mathbb{1}_{\{x \in L(A)\}} - 1$ is computable in time $O(|x|)$.

Proof Under our definition of $\text{DFA}(k)$, each member is specified by a vertex-labeled directed graph. Equivalently, $A \in \text{DFA}(k)$ is described by the matrix $\delta \in Q^{Q \times \Sigma}$, where $Q = \{1, 2, \ldots, k\}$, and by the set of accepting states $F \subseteq Q$. This immediately implies that

$$|\,\text{DFA}(k)| = 2^k k^{|\Sigma| k}$$

and $|\mathcal{C}_n| = \sum_{k=1}^{n} |\,\text{DFA}(k)|$; thus (ii) is proved. To sample a δ uniformly from $Q^{Q \times \Sigma}$, we may independently draw each $\delta(q, \sigma)$ uniformly at random from Q. Since the latter is achievable in constant time, we have that δ may be sampled in time $O(|\Sigma| k)$. Additionally, for each $q \in Q$ we may flip a fair coin to determine whether $q \in F$; this amounts to a uniform sampling of F. We've established that $A \in \text{DFA}(k)$ may be uniformly sampled in time $O(|\Sigma| k)$.

To sample uniformly from $\mathcal{C}_n$, define first the distribution π on $\{1, \ldots, n\}$ by

$$\pi_k = \frac{|\text{DFA}(k)|}{|\mathcal{C}_n|} = \frac{2^k k^{|\Sigma| k}}{\sum_{i=1}^{n} 2^i i^{|\Sigma| i}}.$$

Sampling a $k \in \{1, \ldots, n\}$ according to π can be done in time $O(n)$ and then A is drawn uniformly from $\text{DFA}(k)$ as described above. The resulting automaton is a uniform draw from $\mathcal{C}_n$, which took $O(|\Sigma| n)$ steps to perform; this proves (i).

To prove (iii) we simply recall that an automaton evaluates a string by reading it from beginning to end, each time evolving from state to state as prescribed by δ. ■

We now state our main learnability result for regular languages. The theorem below is a direct consequence of Theorem 15, where $\theta(m) := \lceil m^{1/2} \rceil$ and $T_{\text{SAM}}(\mathcal{C}, n) = O(n)$, as per Lemma 18:

Theorem 19 *Let $L \subseteq \Sigma^*$ be a regular language and suppose that $S \subset \Sigma^*$ is a sequence of m instances sampled independently from an arbitrary distribution P, and labeled according to L. For any $\delta > 0$, there is a randomized algorithm* `RegLearn`*, which outputs a classifier $\hat{f} : \Sigma^* \to \{-1, 1\}$ such that with probability at least $1 - \delta$ it achieves a small generalization error,*

$$\begin{aligned} P\{\hat{f}(x) \neq \chi_L(x)\} \quad &\leq \quad \frac{2}{m}\left(4R(L)\log(8em)\log(32m) + \log(16m/\delta)\right) \\ &\qquad \textit{for } m > \max\{\text{size}(L)^2, D(L)/2, R(L)^2/8.3 \times 10^4\}, \end{aligned}$$

where $\text{size}(L)$ *is the number of states in the smallest automaton recognizing L and $D(L)$, $R(L)$ are constants depending only on L.*

Furthermore, `RegLearn` *has deterministic polynomial complexity, with running time*

$$O\left(\ell_{\max} m^{5/2} \log(m/\delta)\right),$$

where $\ell_{\max} := \max\{|s| : s \in S\}$).

8. Empirical Results

In this section we present some preliminary empirical results that provide a proof of concept for our approach. We performed several types of experiments.

8.1 Proof of Concept, Comparisons

In this basic setup, we draw a sample S of $m = 300$ strings in $\{0,1\}^*$ uniformly with mean Poisson length 15 (that is, the string length ℓ is a Poisson random variable with mean 15 and the string x is drawn uniformly from $\{0,1\}^\ell$). The target DFA A is uniformly drawn at random from DFA(n) as described in Lemma 18, where its size is chosen uniformly in the interval $3 \leq n \leq 50$. This DFA is then minimized, so that its number of states represents its "true" complexity. The automaton A is run on the sample S to produce the label vector $Y \in \{-1,1\}^S$.

For the learning process, we randomly sample a fixed number $T = 1000$ of "feature" automata A_i, from the set DFA$(1:18) = \cup_{n=1}^{18}$ DFA(n), where we chose $\theta(m) = \sqrt{m}$ (note that $18 \approx \sqrt{300}$). For each $s \in S$, we compute the embedding vector $\phi(s) \in \{0,1\}^T$ by

$$[\phi(s)]_i = \mathbb{1}_{\{s \in L(A_i)\}}$$

and arrange the $\{\phi(s)\}_{s \in S}$ into the columns of the $T \times m$ matrix Φ. Now that (Φ, Y) corresponds to m labeled points in $\mathbb{R}^T$, we can apply some standard classification algorithms:

- `SVM` finds the maximum-margin hyperplane separating the positive and negative examples
- `AdaBoost` (Freund and Schapire, 1996) uses the feature automata $\{A_i : 1 \leq i \leq T\}$ as weak classifiers and combines them into a weighted sum

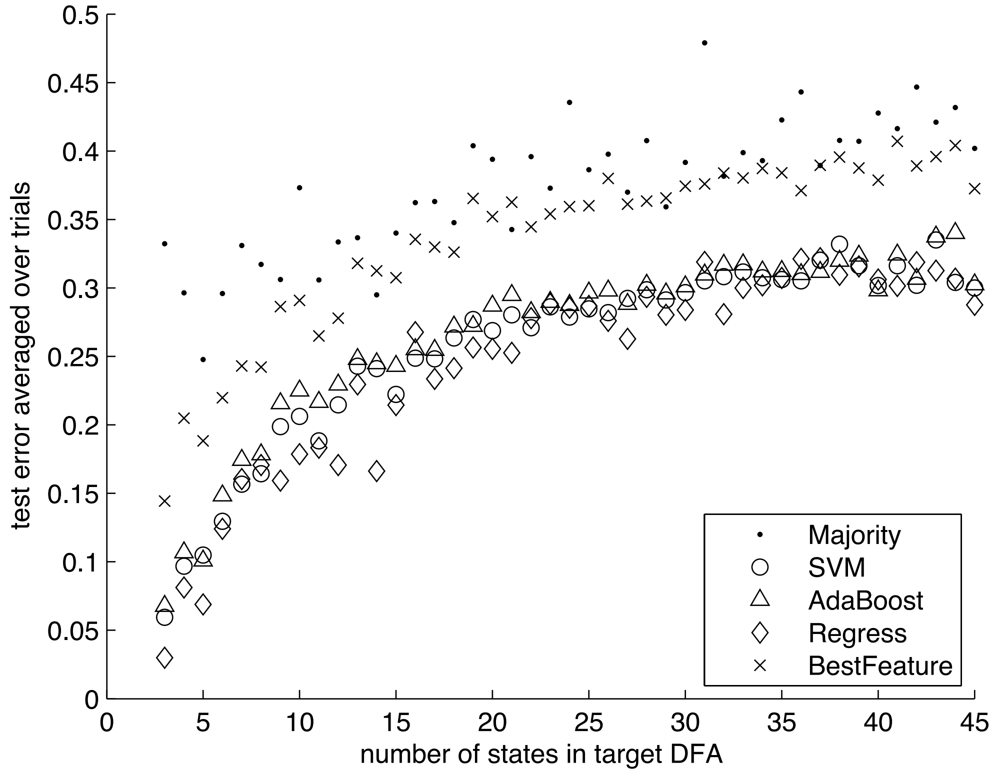


Figure 1: Performance plots of different linear threshold classifiers (sample size = 300, number of features = 1000)

- `Regress` labels a new string $x \in \{0,1\}^*$ as follows:

$$y = \operatorname{sgn}(\phi(x)(\Phi\Phi^{\mathsf{T}})^{-1}\Phi Y)$$

  where $\cdot^{\mathsf{T}}$ denotes the matrix transpose

as well as two "baseline" classifiers:

- `Majority` labels every unseen string $x \in \{0,1\}^*$ by the majority vote of the training labels, disregarding the actual training strings:

$$y = \operatorname{sgn}\left(\sum_{s \in S} Y_s\right)$$

- `BestFeature` picks the single feature automaton A_i with the best empirical performance.

A test set of 100 strings is drawn from the same distribution as S, and the performance of each classifier is compared against the true labels given by A. The results are summarized in Figure 1, averaged over several hundred trials.

As expected, since in this experiment the sample size is kept fixed, the performance degrades with increasing automaton complexity. Furthermore, `SVM`, `AdaBoost` and `Regress` have a similar performance. While the first two methods are margin driven with guaranteed performance bounds, the success of `Regress` is somewhat surprising. We currently have no explanation for why such a seemingly naive approach should work so well. Unsurprisingly, `BestFeature` and `Majority` trail behind the other methods significantly. However, even the latter simplistic approaches perform quite better than chance on moderately sized automata.

Despite the rather pessimistic impossibility results for learning general automata, our results seem to indicate that random automata are efficiently learnable. This is in line with the empirical observations of Lang (1992) and others that random automata are "easy" to learn in practice.

8.2 Role of Adaptive Feature Set

In the second experiment, we demonstrate the importance of having an adaptive feature set. Here, we chose a single fixed target DFA on 20 states, and study the generalization performance of two schemes, one with a fixed number of features T, and one where T is chosen adaptively as a function of sample size. The training and test strings are sampled according to the same scheme as described above. The sample size m varies from 10 to 400; the test set is fixed at 100. In the fixed T scheme we set $T \equiv 500$ while in the adaptive one $T = m \log m$. As expected, the adaptive scheme eventually outperforms the fixed one (see Figure 2).

8.3 Case Study: Parity Languages

For $I = \{i_1 < i_2 \ldots < i_k\} \subset \mathbb{N}$, define the language $L[I] \subset \{0,1\}^*$ by

$$L[I] = \{x \in \{0,1\}^* : |x| \geq i_k, x_{i_1} \oplus x_{i_2} \oplus \ldots \oplus x_{i_k} = 1\}$$

where $\oplus$ is addition modulo 2. Then $L[I]$ is called a *parity language*. The following facts are easily verified

Claim 20 *For $I \subset \mathbb{N}$ and $L[I]$, we have*

(a) the language $L[I]$ is regular

(b) the minimal DFA accepting $L[\{1,k\}]$ has size $2k+1$.

The minimal DFA for the language

$$L[\{1,2\}] = \{x \in \{0,1\}^* : x_1 \oplus x_2 = 1\}$$

is shown in Figure 3. We describe the results of some empirical investigations with parity languages.

8.3.1 EXACT RECOVERY

Recall that the learner is presented with a labeled sample (S,Y), which is consistent with some target language $L_0 \subset \{0,1\}^*$. PAC learning entails producing a hypothesis $\hat{L}$ whose predictions on new strings will be statistically similar to those of L_0. *Exact recovery* is a more stringent demand: we are required to produce a hypothesis that is identical to the target language.

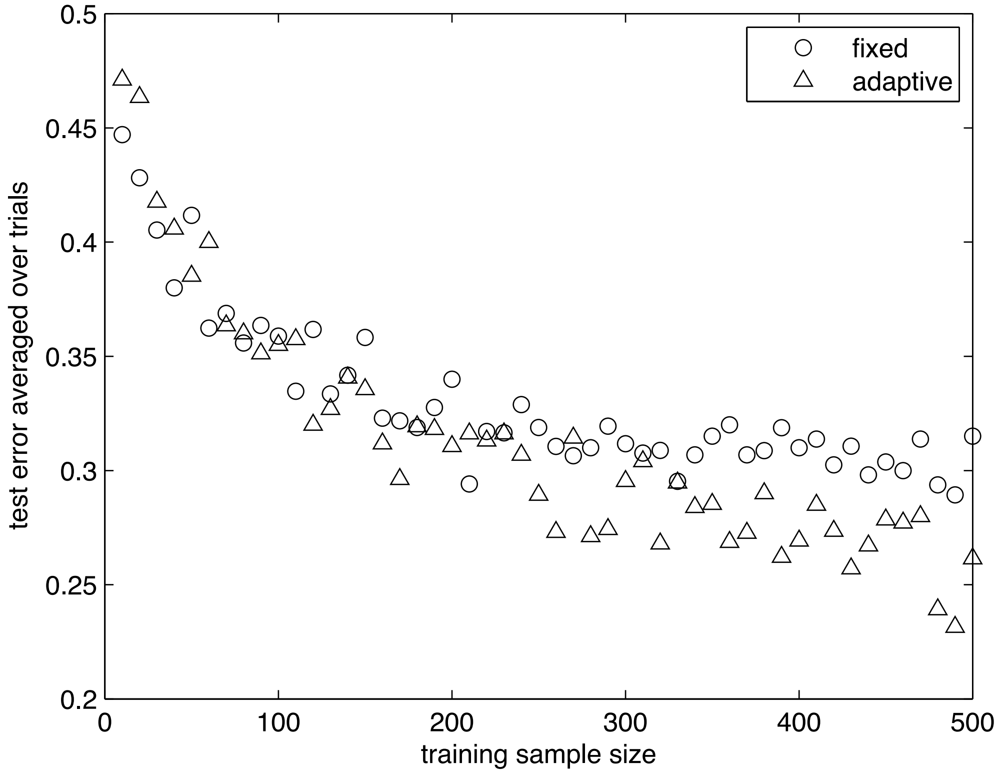


Figure 2: Performance in the fixed and adaptive settings

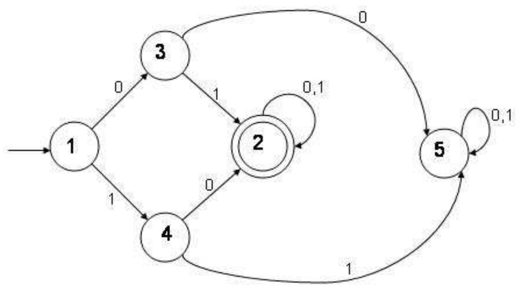


Figure 3: The minimal DFA for the parity language $L[\{1,2\}]$

Note that the output of our algorithm is not an explicit automaton. Instead, we output a collection of DFAs $\{A_1, A_2, \ldots, A_T\}$ with corresponding weights $\alpha_t \in \mathbb{R}$, with the hypothesis language $\hat{L}$ given by

$$\hat{L} = \left\{ x \in \{0,1\}^* : \sum_{t=1}^{T} \alpha_t \mathbb{1}_{\{x \in L(A_t)\}} > 0 \right\}. \tag{27}$$

An interesting question is the relation between $\hat{L}$ and the true unknown language L_0. In principle, the DFA for $\hat{L}$ can be recovered exactly, using the algorithmic construction given in the proof of Theorem 17. However, this procedure has exponential complexity in T and is certainly infeasible for $T \sim 100$. Thus, rather than recovering the *exact* DFA corresponding to $(\{A_t\}, \{\alpha_t\})$, we implement Angluin's algorithm for learning DFAs from membership and equivalence queries (Angluin, 1982). Testing whether a string x belongs to $\hat{L}$ is achievable in time $O(T|x|)$, so membership queries pose no problem. As discussed above, testing whether Angluin's hypothesis language L_{ANGL} is equivalent to $\hat{L}$ is infeasible (at least using the brute-force construction; we do not exclude the possibility of some clever efficient approach). Instead, we only compare L_{ANGL} and $\hat{L}$ on the labeled sample (S, Y). If $\chi_{L_{\text{ANGL}}}(x) = Y_x$ for all $x \in S$, we declare the two equal; otherwise, the first string on which the two disagree is fed as a counterexample into Angluin's algorithm.

We drew a sample of 300 strings in $\{0,1\}^*$ uniformly with mean Poisson length 10, as described in Section 8.1. These strings were labeled consistently with $L[\{1,2\}]$. On this labeled sample we ran the algorithm `RegLearn`, as defined in Theorems 15, and 19, which outputs the language $\hat{L}$, as given in (27). The hypothesis $\hat{L}$ attained an accuracy of 91% on unseen strings and our adaptation of Angluin's algorithm recovered $L[\{1,2\}]$ *exactly* 88% of the time. When this experiment was repeated on $L[\{1,3\}]$, whose minimal DFA has 7 states, prediction accuracy of `RegLearn` dropped to 86% while the target automaton was recovered exactly only 1% of the time. It seems that relatively large sample sizes are needed for exact recovery, though we are not able to quantify (or even conjecture) a rate at this point.

8.3.2 EMPIRICAL MARGIN

We took the "complete" sample $S = \{0,1\}^{\leq 7}$ (i.e., $|S| = 255$) and labeled it with $L[\{1,k\}]$, for $k = 2,3,4,5$. The algorithm `RegLearn` (as a special case of `UnivLearn`) finds the optimal empirical margin $\tilde{\gamma}_*$ as defined in (23). We repeated this experiment a few dozen times—since the kernel is random, the margins obtained are also random. The results are presented in Figure 4, but should be interpreted with caution. As expected, the margin is decreasing with automaton size (the latter given by $2k+1$, as per Claim 20(b)). It is difficult to discern a trend (polynomial/exponential decay) from four points, and extending the experiment for moderate-sized k is computationally expensive.

8.3.3 LONG INPUT STRINGS

One of the claimed advantages of our method is that long training strings do not significantly affect hypothesis complexity. To test this claim empirically, we fixed the target language at $L[\{1,2\}]$ and let the mean string length λ vary from 10 to 100 in increments of 10. For each value of λ, we drew a sample S of 300 strings in $\{0,1\}^*$ uniformly with mean Poisson length λ (as described in Section 8.1) and labeled S consistently with $L[\{1,2\}]$. On this labeled sample we ran the algorithm `RegLearn`, and tested its prediction accuracy on 100 new strings. The results are displayed in Figure 5 and show a graceful degradation of performance. In particular, for strings of mean length

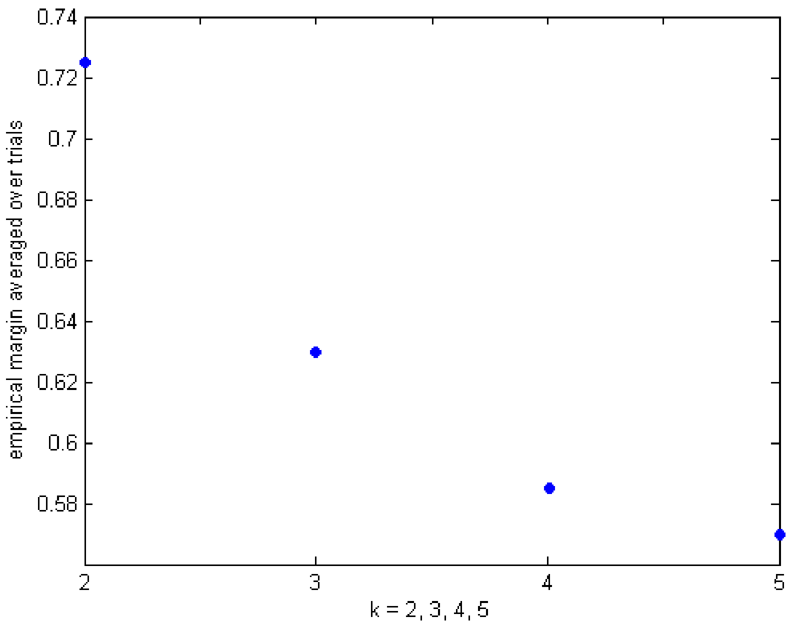


Figure 4: Empirical margins of $L[\{1,k\}]$, for $k = 2,3,4,5$

100, the learner performs significantly better than chance—even though learning parity languages from long strings entails high-dimensional feature selection. This example illustrates the tradeoff between information and computational complexity. If we had unbounded resources, exhaustive enumeration over all automata would most likely find the correct automaton. It is not clear how standard methods, such as the RPNI algorithm (Oncina and Garcia, 1992) would process input strings of such length.

9. Inherent Limitations

An immediate question is, How does the "margin learning rate" $R(\cdot)$ appearing in Theorems 3 and 15 relate to the target concept complexity (say, as measured by $\|\cdot\|$)? One might hope for a bound of the type

$$R(c) \quad = \quad O(\text{poly}(\|c\|)).$$

Unfortunately, under standard cryptographic assumptions this cannot hold in general, as we demonstrate for the case of DFAs. For reference, let us state the Discrete Cube Root (DCR) assumption as it appears in Kearns and Vazirani (1997). In what follows, $N = pq$ is an n-bit number and p,q are randomly chosen primes so that 3 does not divide $(p-1)(q-1)$. The multiplicative group modulo N is denoted by $\mathbb{Z}_N^*$ and x is chosen uniformly random in $\mathbb{Z}_N^*$. DCR states that for every polynomial r, there is no algorithm with running time $r(n)$ that on input N and $y = x^3 \bmod N$ outputs x with probability exceeding $1/r(n)$ (the probability is taken over p,q,x, and the algorithm's internal randomization).

The following theorem is implicit in chapters 6 and 7 of Kearns and Vazirani (1997), and is a combination of results in Pitt and Warmuth (1990) and Kearns and Valiant (1994):

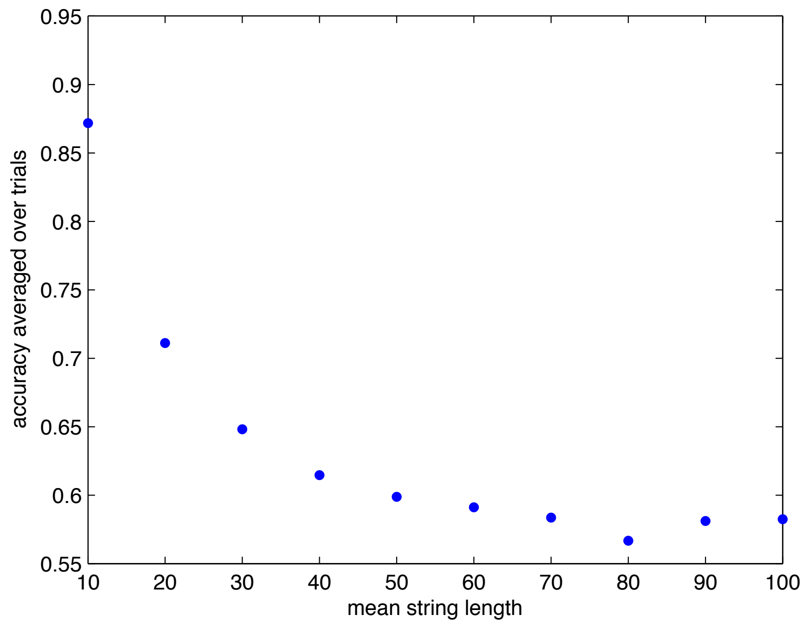


Figure 5: Prediction accuracy when training on long strings

Theorem 21 *For every polynomial p, there is a sequence of distributions P_n on $\{0,1\}^n$ and finite state automata A_n, with $\mathrm{size}(A_n) = \mathrm{poly}(n)$ such that assuming DCR, there is no algorithm with running time $p(n)$ that produces a hypothesis that agrees with A_n on more than $1/2 + 1/p(n)$ of P_n.*

We will refer to the concept-dependent quantity $R(c)$ appearing in theorems 3, 15, and 19 as the *margin-based rate* and the parameter $D(c)$ appearing in Theorems 15, and 19 as the *intrinsic dimension*. Under the DCR assumption, these quantities cannot both grow polynomially in the target automaton size for *all* automata:

Corollary 22 *If DCR is true, there is no efficiently computable (or approximable) universal regular kernel K and polynomial p such that for every finite state automaton A, we have $\max\{R(A), D(A)\} \leq p(\mathrm{size}(A))$ under K.*

Proof We argue by contradiction. Suppose K is an efficiently computable universal regular kernel, and that for some polynomial p we have the margin-based rate $R(A) \leq p(\mathrm{size}(A))$ under K, for every DFA A. Then by Theorem 3, a sample of size $p(\mathrm{size}(A))^2$ suffices to guarantee (with high probability) the existence of a classifier with polynomially small generalization error, and there is an efficient algorithm (e.g., SVM) to discover such a classifier. This contradicts Theorem 21. A similar contradiction is obtained via Theorem 15 if K is efficiently approximable. ■

Still focusing on the special case of DFAs, another natural line of inquiry is the relationship between the true target language L and the hypothesis language $\hat{L}$ induced by our learning algorithm (note that $\hat{L}$ is necessarily regular, by Theorem 17). In particular, it would be desirable to have a handle on the complexity of the minimal DFA accepting $\hat{L}$ in terms of the corresponding minimal

DFA for L. To be more concrete, suppose we have observed a finite sample $S \subset \Sigma^*$ with labels $Y \in \{-1,1\}^S$. We compute a maximum-margin hyperplane (i.e., a set of weights $\alpha \in \mathbb{R}^S$) for this sample, and consider the resulting language

$$L(S,\alpha) \quad = \quad \{x \in \mathcal{X} : \sum_{s \in S} \alpha_s \tilde{K}(s,x) > 0\}$$

where $\tilde{K}$ is the approximate kernel constructed by the algorithm of Theorem 15; note that $L(S,\alpha)$ is regular. Let $A_0(S,Y)$ be the smallest DFA consistent with the labeled sample (S,Y) and let $A_0(S,\alpha)$ denote the smallest DFA recognizing $L(S,\alpha)$. Now approximately recovering $A_0(S,\alpha)$ is feasible via active learning (see Chapter 8 of Kearns and Vazirani 1997). Given a hyperplane representation for a regular language one may efficiently query the resulting classifier on any string in $x \in \Sigma^*$ to see if $x \in L(S,\alpha)$. Using membership and equivalence queries, Angluin's algorithm (Angluin, 1987) recovers $A_0(S,\alpha)$ exactly. If only membership queries are allowed, we can draw enough strings $x \in \Sigma^*$ according to a distribution P to simulate an equivalence query efficiently, to arbitrary P-accuracy. Note that $L(A)$ and $L(A')$ can differ by only one (very long) word, while the two automata may be of exponentially different size. We do not currently have either (a) an efficient method of exactly recovering $A_0(S,\alpha)$ nor can we claim that (b)

$$\mathrm{size}(A_0(S,\alpha)) \quad \leq \quad \mathrm{poly}(\mathrm{size}(A_0(S,Y)));$$

note that if both (a) and (b) were true, that would imply P=NP via the Pitt and Warmuth (1993) hardness of approximation result.

Another limitation of our approach is the possibility of a "Boolean-independent" concept class $\mathcal{C}$, in which no $c \in \mathcal{C}$ may be expressed as a finite Boolean combination of other concepts. One way to construct such a concept class is by taking $\mathcal{X} = \mathbb{N}$ and $\mathcal{C} = \{c_p : p \text{ is prime}\}$, where

$$x \in c_p \quad \Leftrightarrow \quad x \equiv 0 \pmod{p}.$$

That $\mathcal{C}$ is Boolean-independent is seen by taking k distinct primes $\{p_i : 1 \leq i \leq k\}$ and an arbitrary $b \in \{0,1\}^k$. Then the number

$$N_b := \prod_{i=1}^{k} (p_i + b_i)$$

is divisible by p_i iff $b_i = 0$; thus the divisibility of N by $p_1, p_2, \ldots, p_{k-1}$ gives no information regarding its divisibility by p_k. In a Boolean-independent class, no concept may be expressed as a finite linear combination of other concepts (the contrapositive of this claim is established while proving the "if" direction of Theorem 17). Alas, Boolean independence is an inevitable feature of nontrivial concept classes:

Theorem 23 *Any infinite concept class $\mathcal{C}$ contains an infinite Boolean-independent subset.*

Proof Suppose $\mathcal{C}$ contains no infinite Boolean-independent subsets. This means that every $c \in \mathcal{C}$ may be expressed as a finite Boolean combination of elements $\{e_i\}$ from some finite $E \subset \mathcal{C}$. But the Boolean closure of a finite collection of sets is finite, so $\mathcal{C}$ must be finite. ■

The moral of this section is that our approach is certainly vulnerable to the same classical computational hardness reductions as other discrete concept-learning methodologies. One must keep in mind, however, that the hardness reductions are pessimistic, based on carefully contrived target concepts and distributions. As noted in Section 8, "typical" or "random" concepts are often much easier to learn than their worst-case cousins (which is unsurprising given the numerous NP-hard but easy-on-average problems; see, for example Chvátal and Szemerédi 1988 or Flaxman 2003).

10. Discussion

We have presented a new generic technique for learning general countable concept classes and applied it to regular languages. Aside from its generality and ease of implementation, our approach offers certain advantages over existing techniques. Observe that the complexity of the classifier constructed in Theorem 15 does not directly depend on the input string lengths. For the case of learning regular languages, it means that the training string lengths do not affect hypothesis complexity. This is not the case for the methods surveyed in the Introduction, which build a prefix tree acceptor in the initial phase. As already mentioned, we make no structural assumptions on the target automaton (such as acyclic or nearly so), or on the sampling distribution (such as the sample being "structurally complete").

In practice, given m samples, our approach attempts to fit the labeled data by a linear combination of concepts with bounded complexity, $\|c\| \leq \theta(m)$. This method may fail for two possible reasons: Either we don't have sufficient number of samples to learn the unknown target concept, or we got unlucky - this specific concept has a very small margin. Note that these sample size restrictions are inherent also in the SRM framework, for example in Theorem 8.

The experiments presented in Section 8 provide support for our approach. Our basic method could be easily adapted to learning context-free and other grammars. There are obvious computability limitations—thus, if our concept class is all Turing-recognizable languages, the corresponding universal kernel is provably uncomputable (Blocki, 2007). Modulo these limitations, any concept class satisfying the efficient sampling and membership evaluation assumptions (Assumptions 10 and 11) is amenable to our approach.

Our work naturally raises some interesting questions—in particular, regarding learning regular languages. We conclude the paper with a few of them.

1. Can K_{REG} be efficiently computed? Is it a #P-complete problem? Is there *any* efficiently computable universal regular kernel?

2. Can $A_0(S,\alpha)$ be efficiently recovered from its hyperplane representation? Can $\text{size}(A_0(S,\alpha))$ be nontrivially bounded in terms of $\text{size}(A_0(S,Y))$? (The notation is from the previous section.)

3. Corollary 22 seems to thwart attempts to bound the margin-based rate for a regular language in terms of the size of its minimal DFA. Is there a different natural complexity measure for regular languages which does allow a polynomial bound on the margin-based rate?

4. The intractability results in Theorem 21 are highly pessimistic, in that both the automaton and the training/testing distribution are contrived to be pathological. What about the complexity of learning "typical" automata—say, those drawn uniformly from $\text{DFA}(n)$? What can be said

about the margin distribution of such automata? Similarly, what about their sample margin, under some natural class of distributions on the strings Σ^*?

Acknowledgments

AK's work at Carnegie Mellon took place under the guidance of John Lafferty, with part of the support from NSF ITR grant IIS-0205456. The research continued at the Weizmann Institute, where AK is being hosted by Gideon Schechtman, and supported in part by the Israel Science Foundation. BN was supported by the Lord Sieff of Brimpton Memorial fund. Various key concepts were crystallized during AK's many valuable discussions with Corinna Cortes and Mehryar Mohri. Thanks also to Jeremiah Blocki, Avrim Blum, Manuel Blum, Daniel Golovin, Uriel Feige, Nati Linial, Alex Smola, Ofer Zeitouni and Noam Zeilberger for helpful and insightful input. We also thank the referees for constructive comments which improved the paper.

References

Dana Angluin. On the complexity of minimum inference of regular sets. *Information and Control*, 3(39):337–350, 1978.

Dana Angluin. Inference of reversible languages. *Journal of the ACM (JACM)*, 3(29):741–765, 1982.

Dana Angluin. Learning regular sets from queries and counterexamples. *Inf. Comput.*, 75(2):87–106, 1987. ISSN 0890-5401. doi: http://dx.doi.org/10.1016/0890-5401(87)90052-6.

Martin Anthony and Peter L. Bartlett. *Neural Network Learning: Theoretical Foundations*. Cambridge University Press, 1999.

Maria-Florina Balcan, Avrim Blum, and Santosh Vempala. Kernels as Features: On Kernels, Margins, and Low-dimensional Mappings. *Machine Learning*, 65(1):79–94, 2006.

Peter Bartlett, Yoav Freund, Wee Sun Lee, and Robert E. Schapire. Boosting the margin: a new explanation for the effectiveness of voting methods. *Ann. Statist.*, 26(5):1651–1686, 1998.

Jeremiah Blocki. Turing machine kernel is not computable, 2007. Private communication/unpublished note.

Miguel Bugalho and Arlindo L. Oliveira. Inference of regular languages using state merging algorithms with search. *Pattern Recognition*, 38:1457, 2005.

Vašek Chvátal and Endre Szemerédi. Many hard examples for resolution. *J. ACM*, 35(4):759–768, 1988. ISSN 0004-5411. doi: http://doi.acm.org/10.1145/48014.48016.

Alexander Clark and Franck Thollard. Pac-learnability of probabilistic deterministic finite state automata. *Journal of Machine Learning Research (JMLR)*, 5:473–497, 2004.

Corinna Cortes, Leonid Kontorovich, and Mehryar Mohri. Learning languages with rational kernels. *Computational Learning Theory (COLT)*, 2007.

Colin de la Higuera. A bibligraphical study of grammatical inference. *Pattern Recognition*, 38: 1332, 2005.

Abraham Flaxman. A spectral technique for random satisfiable 3cnf formulas. In *SODA '03: Proceedings of the fourteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 357–363, Philadelphia, PA, USA, 2003. Society for Industrial and Applied Mathematics. ISBN 0-89871-538-5.

Yoav Freund and Robert E. Schapire. Predicting $\{0,1\}$-Functions on Randomly Drawn Points. In *Proceedings of the Ninth Annual Conference on Computational Learning Theory (COLT 1996)*, pages 325–332, 1996.

Yoav Freund, Michael Kearns, Dana Ron, Ronitt Rubinfeld, Robert E. Schapire, and Linda Sellie. Efficient learning of typical finite automata from random walks. In *STOC '93: Proceedings of the Twenty-fifth Annual ACM Symposium on Theory of Computing*, pages 315–324, New York, NY, USA, 1993. ACM Press.

Ashutosh Garg, Sariel Har-Peled, and Dan Roth. On generalization bounds, projection profile, and margin distribution. In *ICML '02: Proceedings of the Nineteenth International Conference on Machine Learning*, pages 171–178, San Francisco, CA, USA, 2002. Morgan Kaufmann Publishers Inc. ISBN 1-55860-873-7.

E. Mark Gold. Language identification in the limit. *Information and Control*, 50(10):447–474, 1967.

E. Mark Gold. Complexity of automaton identification from given data. *Information and Control*, 3(37):302–420, 1978.

Wassily Hoeffding. Probability inequalities for sums of bounded random variables. *American Statistical Association Journal*, 58:13–30, 1963.

Yoshiyasu Ishigami and Sei'ichi Tani. Vc-dimensions of finite automata and commutative finite automata with k letters and n states. *Discrete Applied Mathematics*, 74(3):229–240, 1997.

Michael J. Kearns and Leslie G. Valiant. Cryptographic limitations on learning boolean formulae and finite automata. *Journal of the ACM (JACM)*, 41(1):67–95, 1994.

Micheal Kearns and Umesh Vazirani. *An Introduction to Computational Learning Theory*. The MIT Press, 1997.

George Kimeldorf and Grace Wahba. Some results on Tchebycheffian spline functions. *J. Math. Anal. Appl.*, 33:82–95, 1971. ISSN 0022-247x.

Leonid Kontorovich. A universal kernel for learning regular languages. In *The 5th International Workshop on Mining and Learning with Graphs (MLG 2007)*, Florence, Italy, 2007.

Leonid Kontorovich, Corinna Cortes, and Mehryar Mohri. Learning linearly separable languages. In *Proceedings of The 17th International Conference on Algorithmic Learning Theory (ALT 2006)*, volume 4264 of *Lecture Notes in Computer Science*, pages 288–303, Barcelona, Spain, October 2006. Springer, Heidelberg, Germany.

Kevin J. Lang. Random dfa's can be approximately learned from sparse uniform samples. In *Fifth Conference on Computational Learning Theory*, page 45. ACM, 1992.

Harry R. Lewis and Christos H. Papadimitriou. *Elements of the Theory of Computation*. Prentice Hall, 1981.

Arlindo L. Oliveira and Joao P.M. Silva. Efficient algorithms for the inference of minimum size dfas. *Machine Learning*, 44:93, 2001.

José Oncina and Pedro Garcia. Inferring regular languages in polynomial update time. In *Pattern Recognition and Image Analysis*, page 49. World Scientific, 1992.

José Oncina, Pedro García, and Enrique Vidal. Learning subsequential transducers for pattern recognition interpretation tasks. *IEEE Trans. Pattern Anal. Mach. Intell.*, 15(5):448–458, 1993.

Leonard Pitt and Manfred Warmuth. Prediction-preserving reducibility. *Journal of Computer and System Sciences*, 41(3):430–467, 1990.

Leonard Pitt and Manfred Warmuth. The minimum consistent DFA problem cannot be approximated within any polynomial. *Journal of the Assocation for Computing Machinery*, 40(1):95–142, 1993.

Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. In *NIPS 2007*. MIT Press, 2007.

Jorma Rissanen. *Stochastic Complexity in Statistical Inquiry Theory*. World Scientific Publishing Co., Inc., 1989.

Ronald L. Rivest and Robert E. Schapire. Diversity-based inference of finite automata. In *Proc. 28th Annu. IEEE Sympos. Found. Comput. Sci.*, pages 78–87. IEEE Computer Society Press, Los Alamitos, CA, 1987.

Dana Ron, Yoram Singer, and Naftali Tishby. On the learnability and usage of acyclic probabilistic finite automata. *Journal of Computer and System Sciences*, 56(2):133–152, 1998.

Bernhard Schölkopf and Alex Smola. *Learning with Kernels*. MIT Press, 2002.

Shai Shalev-Shwartz, Yoram Singer, and Nathan Srebro. Pegasos: Primal estimated sub-gradient solver for svm. In *ICML '07: Proceedings of the 24th International Conference on Machine learning*, pages 807–814, New York, NY, USA, 2007. ACM. ISBN 9781595937933. doi: 10.1145/1273496.1273598. URL `http://portal.acm.org/citation.cfm?id=1273598`.

John Shawe-Taylor, Peter L. Bartlett, Robert C. Williamson, and Martin Anthony. A framework for structural risk minimisation. In *COLT*, pages 68–76, 1996.

John Shawe-Taylor, Peter L. Bartlett, Robert C. Williamson, and Martin Anthony. Structural risk minimization over data-dependent hierarchies. *IEEE transactions on Information Theory*, 44: 1926–1940, 1998.

Michael Sipser. *Introduction to the Theory of Computation*. Course Technology, 2005.

Boris A. Trakhtenbrot and Janis M. Barzdin. *Finite Automata: Behavior and Synthesis*, volume 1 of *Fundamental Studies in Computer Science*. North-Holland, Amsterdam, 1973.

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

Vladimir N. Vapnik. *Estimation of Dependences Based on Empirical Data*. Springer-Verlag, 1982.

Vladimir N. Vapnik. *Statistical Learning Theory*. John Wiley & Sons, 1998.

Journal of Machine Learning Research 10 (2009) 1131-1186 Submitted 7/08; Revised 3/09; Published 5/09

Multi-task Reinforcement Learning in Partially Observable Stochastic Environments

Hui Li HL1@EE.DUKE.EDU
Xuejun Liao XJLIAO@EE.DUKE.EDU
Lawrence Carin LCARIN@EE.DUKE.EDU
Department of Electrical and Computer Engineering
Duke University
Durham, NC 27708-0291, USA

Editor: Carlos Guestrin

Abstract

We consider the problem of multi-task reinforcement learning (MTRL) in multiple partially observable stochastic environments. We introduce the regionalized policy representation (RPR) to characterize the agent's behavior in each environment. The RPR is a parametric model of the conditional distribution over current actions given the history of past actions and observations; the agent's choice of actions is directly based on this conditional distribution, without an intervening model to characterize the environment itself. We propose off-policy batch algorithms to learn the parameters of the RPRs, using episodic data collected when following a behavior policy, and show their linkage to policy iteration. We employ the Dirichlet process as a nonparametric prior over the RPRs across multiple environments. The intrinsic clustering property of the Dirichlet process imposes sharing of episodes among similar environments, which effectively reduces the number of episodes required for learning a good policy in each environment, when data sharing is appropriate. The number of distinct RPRs and the associated clusters (the sharing patterns) are automatically discovered by exploiting the episodic data as well as the nonparametric nature of the Dirichlet process. We demonstrate the effectiveness of the proposed RPR as well as the RPR-based MTRL framework on various problems, including grid-world navigation and multi-aspect target classification. The experimental results show that the RPR is a competitive reinforcement learning algorithm in partially observable domains, and the MTRL consistently achieves better performance than single task reinforcement learning.

Keywords: reinforcement learning, partially observable Markov decision processes, multi-task learning, Dirichlet processes, regionalized policy representation

1. Introduction

Planning in a partially observable stochastic environment has been studied extensively in the fields of operations research and artificial intelligence. Traditional methods are based on partially observable Markov decision processes (POMDPs) and assume that the POMDP models are given (Sondik, 1971; Smallwood and Sondik, 1973). Many POMDP planning algorithms (Sondik, 1971, 1978; Cheng, 1988; Lovejoy, 1991; Hansen, 1997; Kaelbling et al., 1998; Poupart and Boutilier, 2003; Pineau et al., 2003; Spaan and Vlassis, 2005; Smith and Simmons, 2005; Li et al., 2006a,b) have been proposed, addressing problems of increasing complexity as the algorithms become progressively more efficient. However, the assumption of knowing the underlying POMDP model is often difficult to meet in practice. In many cases the only knowledge available to the agent are experiences, that is, the observations and rewards, resulting from interactions with the environment, and the agent must learn the behavior policy based on such experience. This problem is known as reinforcement learning (RL) (Sutton and Barto, 1998). Reinforcement learning methods generally fall into two broad categories: model-based and model-free. In model-based methods, one first builds a POMDP model based on experiences and then exploits the existing planning algorithms to find the POMDP policy. In model-free methods, one directly infers the policy based on experiences. The focus of this paper is on the latter, trying to find the policy for a partially observable stochastic environment without the intervening stage of environment-model learning.

In model-based approaches, when the model is updated based on new experiences gathered from the agent-environment interaction, one has to solve a new POMDP planing problem. Solving a POMDP is computationally expensive, which is particularly true when one takes into account the model uncertainty; in the latter case the POMDP state space grows fast, often making it inefficient to find even an approximate solution (Wang et al., 2005). Recent work (Ross et al., 2008) gives a relatively efficient approximate model-based method, but still the computation time grows exponentially with the planning horizon. By contrast, model-free methods update the policy directly, without the need to update an intervening POMDP model, thus saving time and eliminating the errors introduced by approximations that may be made when solving the POMDP.

Model-based methods suffer particular computational inefficiency in multi-task reinforcement learning (MTRL), the problem being investigated in this paper, because one has to repeatedly solve multiple POMDPs due to frequent experience-updating arising from the communications among different RL tasks. The work in Wilson et al. (2007) assumes the environment states are perfectly observable, reducing the POMDP in each task to a Markov decision process (MDP); since a MDP is relatively efficient to solve, the computational issue is not serious there. In the present paper, we assume the environment states are partially observable, thus manifesting a POMDP associated with each environment. If model-based methods are pursued, one would have to solve multiple POMDPs for each update of the task clusters, which entails a prohibitive computational burden.

Model-free methods are consequently particularly advantageous for MTRL in partially observable domains. The regionalized policy representation (RPR) proposed in this paper, which yields an efficient parametrization for the policy governing the agent's behavior in each environment, lends itself naturally to a Bayesian formulation and thus furnishes a posterior distribution of the policy. The policy posterior allows the agent to reason and plan under uncertainty about the policy itself. Since the ultimate goal of reinforcement learning is the policy, the policy's uncertainty is more direct and relevant to the learning goal than the POMDP model's uncertainty as considered in Ross et al. (2008).

The MTRL problem considered in this paper shares similar motivations as the work in Wilson et al. (2007)—that is, in many real-world settings there may be multiple environments for which policies are desired. For example, a single agent may have collected experiences from previous environments and wishes to borrow from previous experience when learning the policy for a new environment. In another case, multiple agents are distributed in multiple environments, and they wish to communicate with each other and share experiences such that their respective performances are enhanced. In either case the experiences in one environment should be properly exploited to benefit the learning in another (Guestrin et al., 2003). Appropriate experience sharing among multiple environments and joint learning of multiple policies save resources, improve policy quality, and enhance generalization to new environments, especially when the experiences from each individual environment are scarce (Thrun, 1996). Many problems in practice can be formulated as an MTRL problem, with one example given in Wilson et al. (2007). The application we consider in the experiments (see Section 6.2.3) is another example, in which we make the more realistic assumption that the states of the environments are partially observable.

To date there has been much work addressing the problem of inferring the sharing structure between general learning tasks. Most of the work follows a hierarchical Bayesian approach, which assumes that the parameters (models) for each task are sampled from a common prior distribution, such as a Gaussian distribution specified by unknown hyper-parameters (Lawrence and Platt, 2004; Yu et al., 2003). The parameters as well as the hyper-parameters are estimated simultaneously in the learning phase. In Bakker and Heskes (2003) a single Gaussian prior is extended to a Gaussian mixture; each task is given a corresponding Gaussian prior and related tasks are allowed to share a common Gaussian prior. Such a formulation for information sharing is more flexible than a single common prior, but still has limitations: the form of the prior distribution must be specified *a priori*, and the number of mixture components must also be pre-specified.

In the MTRL framework developed in this paper, we adopt a nonparametric approach by employing the Dirichlet process (DP) (Ferguson, 1973) as our prior, extending the work in Yu et al. (2004) and Xue et al. (2007) to model-free policy learning. The nonparametric DP prior does not assume a specific form, therefore it offers a rich representation that captures complicated sharing patterns among various tasks. A nonparametric prior drawn from the DP is almost surely discrete, and therefore a prior distribution that is drawn from a DP encourages task-dependent parameter clustering. The tasks in the same cluster share information and are learned collectively as a group. The resulting MTRL framework automatically learns the number of clusters, the members in each cluster as well as the associated common policy.

The nonparametric DP prior has been used previously in MTRL (Wilson et al., 2007), where each task is a Markov decision process (MDP) assuming perfect state observability. To the authors' knowledge, this paper represents the first attempt to apply the DP prior to reinforcement learning in multiple partially observable stochastic environments. Another distinction is that the method here is model-free, with information sharing performed directly at the policy level, without having to learn a POMDP model first; the method in Wilson et al. (2007) is based on using MDP models.

This paper contains several technical contributions. We propose the regionalized policy representation (RPR) as an efficient parametrization of stochastic policies in the absence of a POMDP model, and develop techniques of learning the RPR parameters based on maximizing the sum of discounted rewards accrued during episodic interactions with the environment. An analysis of the techniques is provided, and relations are established to the expectation-maximization algorithm and the POMDP policy improvement theorem. We formulate the MTRL framework by placing multiple RPRs in a Bayesian setting and employ a draw from the Dirichlet process as their common nonparametric prior. The Dirichlet process posterior is derived, based on a nonconventional application of Bayes law. Because the DP posterior involves large mixtures, Gibbs sampling analysis is inefficient. This motivates a hybrid Gibbs-variational algorithm to learn the DP posterior. The proposed techniques are evaluated on four problem domains, including the benchmark Hallway2 (Littman et al., 1995), its multi-task variants, and a remote sensing application. The main theoretical results in the paper are summarized in the form of theorems and lemmas, the proofs of which are all given in the Appendix.

The RPR formulation in this paper is an extension of the work in Li (2006) and Liao et al. (2007). All other content in the paper is extended from the work in Li (2006).

2. Partially Observable Markov Decision Processes

The partially observable Markov decision process (POMDP) (Sondik, 1971; Lovejoy, 1991; Kaelbling et al., 1998) is a mathematical model for the optimal control of an agent situated in a partially observable stochastic environment. In a POMDP the state dynamics of the agent are governed by a Markov process, and the state of the process is not completely observable but is inferred from observations; the observations are probabilistically related to the state. Formally, the POMDP can be described as a tuple $(\mathcal{S}, \mathcal{A}, T, O, \Omega, R)$, where $\mathcal{S}$, $\mathcal{A}$, O respectively denote a finite set of states, actions, and observations; T are state-transition matrices with $T_{ss'}(a)$ the probability of transiting to state s' by taking action a in state s; Ω are observation functions with $\Omega_{s'o}(a)$ the probability of observing o after performing action a and transiting to state s'; and R is a reward function with $R(s,a)$ the expected immediate reward received by taking action a in state s.

The optimal control of a POMDP is represented by a policy for choosing the best action at any time such that the future expected reward is maximized. Since the state in a POMDP is only partially observable, the action choice is based on the belief state, a sufficient statistic defined as the probability distribution of the state s given the history of actions and observations (Sondik, 1971). It is important to note that computation of the belief state requires knowing the underlying POMDP model.

The belief state constitutes a continuous-state Markov process (Smallwood and Sondik, 1973). Given that at time t the belief state is b and the action a is taken, and the observation received at time $t+1$ is o, then the belief state at time $t+1$ is computed by Bayes rule

$$b_o^a(s') = \frac{\sum_{s\in\mathcal{S}} b(s) T_{ss'}^a \Omega_{s'o}^a}{p(o|b,a)}, \tag{1}$$

where the superscript a and the subscript o are used to indicate the dependence of the new belief state on a and o, and

$$p(o|b,a) = \sum_{s'\in\mathcal{S}} \sum_{s\in\mathcal{S}} b(s) T_{ss'}^a \Omega_{s'o}^a \tag{2}$$

is the probability of transiting from b to b' when taking action a.

Equations (1) and (2) imply that, for any POMDP, there exists a corresponding Markov decision process (MDP), the state of which coincides with the belief state of the POMDP (hence the term "belief-state MDP"). Although the belief state is continuous, their transition probabilities are discrete : from any given b, one can only make a transition to a finite number of new belief states $\{b_o^a : a \in \mathcal{A}, o \in O\}$, assuming $\mathcal{A}$ and O are discrete sets with finite alphabets. For any action $a \in \mathcal{A}$, the belief state transition probabilities are given by

$$p(b'|b,a) = \begin{cases} p(o|b,a), & \text{if } b' = b_o^a \\ 0, & \text{otherwise} \end{cases} . \tag{3}$$

The expected reward of the belief-state MDP is given by

$$R(b,a) = \sum_{s\in\mathcal{S}} b(s) R(s,a). \tag{4}$$

In summary, the belief-state MDP is completely defined by the action set $\mathcal{A}$, the space of belief state

$$\mathcal{B} = \left\{ b \in \mathbb{R}^{|\mathcal{S}|} : b(s) \geq 0, \sum_{s\in\mathcal{S}} b(s) = 1 \right\},$$

along with the belief state transition probabilities in (3) and the reward function in (4).

The optimal control of the POMDP can be found by solving the corresponding belief-state MDP. Assume that at any time there are infinite steps remaining for the POMDP (infinite horizon), the future rewards are discounted exponentially with a factor $0 < \gamma < 1$, and the action is drawn from $p^{\Pi}(a|b)$, then the expected reward accumulated over the infinite horizon satisfies the Bellman equation (Bellman, 1957; Smallwood and Sondik, 1973)

$$V^{\Pi}(b) = \sum_{a \in \mathcal{A}} p^{\Pi}(a|b) \left[R(b,a) + \gamma \sum_{o \in \mathcal{O}} p(o|b,a) V^{\Pi}(b_o^a) \right],$$

where $V^{\Pi}(b)$ is called the value function. Sondik (1978) showed that, for a finite-transient deterministic policy,[1] there exists a Markov partition $\mathcal{B} = \mathcal{B}_1 \cup \mathcal{B}_2 \cup \cdots$ satisfying the following two properties :

(a) There is a unique optimal action a_i associated with subset $\mathcal{B}_i$, $i = 1, 2, \cdots$. This implies that the optimal control is represented by a deterministic mapping from the Markov partition to the set of actions.

(b) Each subset maps completely into another (or itself), that is, $\{b_o^a : b \in \mathcal{B}_i, a = \Pi(b), o \in \mathcal{O}\} \subseteq \mathcal{B}_j$ (i may equal j).

The Markov partition yields an equivalent representation of the finite-transient deterministic policy. Sondik noted that an arbitrary policy Π is not likely to be finite-transient, and for it one can only construct a partition where one subset maps partially into another (or itself), that is, there exists $b \in \mathcal{B}_i$ and $o \in \mathcal{O}$ such that $b_o^{\Pi(b)} \notin \mathcal{B}_j$. Nevertheless, the Markov partition provides an approximate representation for non-finite-transient policies and Sondik gave an error bound of the difference between the true value function and approximate value function obtained by the Markov partition. Based on the Markov partition, Sondik also proposed a policy iteration algorithm for POMDPs, which was later improved by Hansen (1997) and the improved algorithm is referred to as finite state controller (the partition is finite).

3. Regionalized Policy Representation

We are interested in model-free policy learning, that is, we assume the model of the POMDP is *unknown* and aim to learn the policy directly from the experiences (data) collected from agent-environment interactions. One may argue that we do in fact learn a model, but our model is directly at the policy level, constituting a *probabilistic* mapping from the space of action-observation histories to the action space.

Although the optimal control of a POMDP can be obtained via solving the corresponding belief-state MDP, this is not true when we lack an underlying POMDP model. This is because, as indicated above, the observability of the belief-state depends on the availability of the POMDP model. When the model is unknown, one does not have access to the information required to compute the belief state, making the belief state *unobservable*.

1. Let Π be a deterministic policy, that is, $p^{\Pi}(a|b) = \begin{cases} 1, & \text{if } a = \Pi(b) \\ 0, & \text{otherwise} \end{cases}$. Let S_{Π}^n be the set of all possible belief-states when Π has been followed for n consecutive steps by starting from any initial belief-state. The Π is finite transient if and only if there exists $n < \infty$ such that S_{Π}^n is disjoint with $\{b : \Pi(b) \text{ is discontinuous at } b\}$ (Sondik, 1978).

In this paper, we treat the belief-state as a hidden (latent) variable and marginalize it out to yield a stochastic POMDP policy that is purely dependent on the observable history, that is, the sequence of previous actions and observations. The belief-state dynamics, as well as the optimal control in each state, is learned empirically from experiences, instead of being computed from an underlying POMDP model. Although it may be possible to learn the dynamics and control in the continuous space of belief state, the exposition in this paper is restricted to the discrete case, that is, the case for which the continuous belief-state space is quantized into a finite set of disjoint regions. The quantization can be viewed as a stochastic counterpart of the Markov partition (Sondik, 1978), discussed at the end of Section 2. With the quantization, we learn the dynamics of belief regions and the local optimal control in each region, both represented stochastically. The stochasticity manifests the uncertainty arising from the belief quantization (the policy is parameterized in terms of latent belief *regions*, not the precise belief state). The stochastic policy reduces to a deterministic one when the policy is finitely transient, in which case the quantization becomes a Markov partition. The resulting framework is termed *regionalized policy representation* to reflect the fact that the policy of action selection is expressed through the dynamics of belief regions as well as the local controls in each region. We also use *decision state* as a synonym of *belief region*, in recognition of the fact that each belief region is an elementary unit to encode the decisions of action selection.

3.1 Formal Framework

Definition 1 *A* regionalized policy representation *(RPR) is a tuple $\langle \mathcal{A}, \mathcal{O}, \mathcal{Z}, W, \mu, \pi \rangle$ specified as follows. The $\mathcal{A}$ and $\mathcal{O}$ are respectively a finite set of actions and observations. The $\mathcal{Z}$ is a finite set of decision states (belief regions). The W are decision-state transition matrices with $W(z,a,o',z')$ denoting the probability of transiting from z to z' when taking action a in decision state z results in observing o'. The μ is the initial distribution of decision states with $\mu(z)$ denoting the probability of initially being in decision state z. The π are state-dependent* stochastic *policies with $\pi(z,a)$ denoting the probability of taking action a in decision state z.*

The stochastic formulation of W and π in Definition 1 is fairly general and subsumes two special cases.

1. If z shrinks down to a single belief-state b, $z = b$ becomes a sufficient statistic of the POMDP (Smallwood and Sondik, 1973) and there is a unique action associated with it, thus $\pi(z,a)$ is deterministic and the local policy can be simplified as $a = \pi(b)$.

2. If the belief regions form a Markov partition of the belief-state space (Sondik, 1978), that is, $\mathcal{B} = \cup_{z \in \mathcal{Z}} \mathcal{B}_z$, then the action choice in each region is constant and one region transits completely to another (or itself). In this case, both W and π are deterministic and, moreover, the policy yielded by the RPR (see (8)) is finite transient deterministic. In fact this is the same case as considered in Hansen (1997).

In both of the two special cases, each z has one action choice $a = \pi(z)$ associated with it, and one can write $W(z,a,o',z') = W(z,\pi(z),o',z')$, thus the transition of z is driven solely by o. In general, each z represents multiple individual belief-states, and the belief region transition is driven jointly by a and o. The action-dependency captures the state dynamics of the POMDP, and the observation-dependency reflects the partial observability of the state (perception aliasing).

To make notation simple, the following conventions are observed throughout the paper:

- The elements of $\mathcal{A}$ are enumerated as $\mathcal{A} = \{1,2,\cdots,|\mathcal{A}|\}$, where $|\mathcal{A}|$ denotes the cardinality of $\mathcal{A}$. Similarly, $\mathcal{O} = \{1,2,\cdots,|\mathcal{O}|\}$ and $\mathcal{Z} = \{1,2,\cdots,|\mathcal{Z}|\}$.
- A sequence of actions $(a_0,a_1,\cdots,a_T)$ is abbreviated as $a_{0:T}$, where the subscripts index discrete time steps. Similarly a sequence of observations $(o_1,o_2,\cdots,o_T)$ is abbreviated as $o_{1:T}$, and a sequence of decision states $(z_0,z_1,\cdots,z_T)$ is abbreviated as $z_{0:T}$, etc.
- A history h_t is the set of actions executed and observation received up to time step t, that is, $h_t = \{a_{0:t-1}, o_{1:t}\}$.

Let $\Theta = \{\pi,\mu,W\}$ denote the parameters of the RPR. Given a history of actions and observations, $h_t = (a_{0:t-1}, o_{1:t})$, collected up to time step t, the RPR yields a joint probability distribution of $z_{0:t}$ and $a_{0:t}$

$$p(a_{0:t}, z_{0:t}|o_{1:t},\Theta) = \mu(z_0)\pi(z_0,a_0)\prod_{\tau=1}^{t} W(z_{\tau-1},a_{\tau-1},o_\tau,z_\tau)\pi(z_\tau,a_\tau), \tag{5}$$

where application of local controls $\pi(z_t,a_t)$ at every time step implies that $a_{0:t}$ are all drawn according to the RPR. The decision states $z_{0:t}$ in (5) are hidden variables and we marginalize them to get

$$p(a_{0:t}|o_{1:t},\Theta) = \sum_{z_0,\cdots,z_t=1}^{|\mathcal{Z}|}\left[\mu(z_0)\pi(z_0,a_0)\prod_{\tau=1}^{t} W(z_{\tau-1},a_{\tau-1},o_\tau,z_\tau)\pi(z_\tau,a_\tau)\right]. \tag{6}$$

It follows from (6) that

$$\begin{aligned} p(a_{0:t-1}|o_{1:t},\Theta) &= \sum_{a_t=1}^{|\mathcal{A}|} p(a_{0:t}|o_{1:t},\Theta) \\ &= \sum_{z_0,\cdots,z_{t-1}=1}^{|\mathcal{Z}|}\left[\mu(z_0)\pi(z_0,a_0)\prod_{\tau=1}^{t-1} W(z_{\tau-1},a_{\tau-1},o_\tau,z_\tau)\pi(z_\tau,a_\tau)\right] \\ &\qquad \times \underbrace{\sum_{a_t=1}^{|\mathcal{A}|}\sum_{z_t=1}^{|\mathcal{Z}|} W(z_{t-1},a_{t-1},o_t,z_t)\pi(z_t,a_t)}_{=1} \\ &= p(a_{0:t-1}|o_{1:t-1},\Theta), \end{aligned} \tag{7}$$

which implies that observation o_t does not influence the actions before t, in agreement with expectations. From (6) and (7), we can write the history-dependent distribution of action choices

$$p(a_\tau|h_\tau,\Theta) = p(a_\tau|a_{0:\tau-1},o_{1:\tau},\Theta) = \frac{p(a_{0:\tau}|o_{1:\tau},\Theta)}{p(a_{0:\tau-1}|o_{1:\tau},\Theta)} = \frac{p(a_{0:\tau}|o_{1:\tau},\Theta)}{p(a_{0:\tau-1}|o_{1:\tau-1},\Theta)}, \tag{8}$$

which gives a stochastic RPR policy for choosing the action a_t, given the historical actions and observations. The policy is purely history-dependent, with the unobservable belief regions z integrated out.

The history h_t forms a Markov process with transitions driven by actions and observations: $h_t = h_{t-1} \cup \{a_{t-1}, o_t\}$. Applying this recursively, we get $h_t = \cup_{\tau=1}^{t}\{a_{\tau-1},o_\tau\}$, and therefore

$$\prod_{\tau=0}^{t} p(a_\tau|h_\tau,\Theta) = \left[\prod_{\tau=0}^{t-2} p(a_\tau|h_\tau,\Theta)\right] p(a_{t-1}|h_{t-1},\Theta)p(a_t|h_{t-1},a_{t-1},o_t,\Theta)$$

$$\begin{aligned}
&= \left[\prod_{\tau=0}^{t-2} p(a_\tau|h_\tau,\Theta)\right] p(a_{t-1:t}|h_{t-1},o_t,\Theta) \\
&= \left[\prod_{\tau=0}^{t-3} p(a_\tau|h_\tau,\Theta)\right] p(a_{t-2}|h_{t-2},\Theta)p(a_{t-1:t}|h_{t-2},a_{t-2},o_{t-1},o_t,\Theta) \\
&= \left[\prod_{\tau=0}^{t-3} p(a_\tau|h_\tau,\Theta)\right] p(a_{t-2:t}|h_{t-2},o_{t-1:t},\Theta) \\
&\vdots \\
&= p(a_{0:t}|h_0,o_{1:t},\Theta) \\
&= p(a_{0:t}|o_{1:t},\Theta), \qquad (9)
\end{aligned}$$

where we have used $p(a_\tau|h_\tau,o_{\tau+1:t}) = p(a_\tau|h_\tau)$ and $h_0 = \text{null}$. The rightmost side of (9) is the observation-conditional probability of joint action-selection at multiple time steps $\tau = 0,1,\cdots,t$. Equation (9) can be verified directly by multiplying (8) over $\tau = 0,1,\cdots,t$

$$\begin{aligned}
&\prod_{\tau=0}^{t} p(a_\tau|h_\tau,\Theta) \\
&= p(a_0|\Theta)\frac{p(a_{0:1}|o_1,\Theta)}{p(a_0|\Theta)}\frac{p(a_{0:2}|o_{1:2},\Theta)}{p(a_{0:1}|o_1,\Theta)}\cdots\frac{p(a_{0:t-1}|o_{1:t-1},\Theta)}{p(a_{0:t-2}|o_{1:t-2},\Theta)}\frac{p(a_{0:t}|o_{1:t},\Theta)}{p(a_{0:t-1}|o_{1:t-1},\Theta)} \\
&= p(a_{0:t}|o_{1:t},\Theta). \qquad (10)
\end{aligned}$$

It is of interest to point out the difference between the RPR and previous reinforcement learning algorithms for POMDPs. The reactive policy and history truncation (Jaakkola et al., 1995; Baxter and Bartlett, 2001) condition the action only upon the immediate observation or a truncated sequence of observations, without using the full history, and therefore these are clearly different from the RPR. The U-tree (McCallum, 1995) stores historical information along the branches of decision trees, with the branches split to improve the prediction of future return or utility. The drawback is that the tree may grow intolerably fast with the episode length. The finite policy graphs (Meuleau et al., 1999), finite state controllers (Aberdeen and Baxter, 2002), and utile distinction HMMs (Wierstra and Wiering, 2004) use internal states to memorize the full history, however, their state transitions are driven by observations only. In contrast, the dynamics of decision states in the RPR are driven jointly by actions and observations, the former capturing the dynamics of world-states and the latter reflecting the perceptual aliasing. Moreover, none of the previous algorithms is based on Bayesian learning, and therefore they are intrinsically not amenable to the Dirichlet process framework that is used in the RPR for multi-task examples.

3.2 The Learning Objective

We are interested in empirical learning of the RPR, based on a set of episodes defined as follows.

Definition 2 *(Episode) An episode is a sequence of agent-environment interactions terminated in an absorbing state that transits to itself with zero rewards (Sutton and Barto, 1998). An episode is denoted by* $(a_0^k r_0^k o_1^k a_1^k r_1^k \cdots o_{T_k}^k a_{T_k}^k r_{T_k}^k)$*, where the subscripts are discrete times, k indexes the episodes, and o, a, and r are respectively observations, actions, and immediate rewards.*

Definition 3 *(Sub-episode) A sub-episode is an episode truncated at a particular time step and retaining the immediate reward only at the time step where truncation occurs. The t-th sub-episode of episode $(a_0^k r_0^k o_1^k a_1^k r_1^k \cdots o_{T_k}^k a_{T_k}^k r_{T_k}^k)$ is defined as $(a_0^k o_1^k a_1^k \cdots o_t^k a_t^k r_t^k)$, which yields a total of $T_k + 1$ sub-episodes for this episode.*

The learning objective is to maximize the optimality criterion given in Definition 4. Theorem 5 introduced below establishes the limit of the criterion when the number of episodes approaches infinity.

Definition 4 *(The RPR Optimality Criterion) Let $\mathcal{D}^{(K)} = \{(a_0^k r_0^k o_1^k a_1^k r_1^k \cdots o_{T_k}^k a_{T_k}^k r_{T_k}^k)\}_{k=1}^K$ be a set of episodes obtained by an agent interacting with the environment by following policy Π to select actions, where Π is an arbitrary stochastic policy with action-selecting distributions $p^\Pi(a_t|h_t) > 0$, $\forall$ action a_t, $\forall$ history h_t. The RPR optimality criterion is defined as*

$$\widehat{V}(\mathcal{D}^{(K)};\Theta) \overset{def.}{=} \frac{1}{K}\sum_{k=1}^{K}\sum_{t=0}^{T_k}\frac{\gamma^t r_t^k}{\prod_{\tau=0}^{t} p^\Pi(a_\tau^k|h_\tau^k)}\prod_{\tau=0}^{t} p(a_\tau^k|h_\tau^k,\Theta), \tag{11}$$

where $h_t^k = a_0^k o_1^k a_1^k \cdots o_t^k$ is the history of actions and observations up to time t in the k-th episode, $0 < \gamma < 1$ is the discount, and Θ denotes the parameters of the RPR.

Theorem 5 *Let $\widehat{V}(\mathcal{D}^{(K)};\Theta)$ be as defined in Definition 4, then $\lim_{K\to\infty}\widehat{V}(\mathcal{D}^{(K)};\Theta)$ is the expected sum of discounted rewards within the environment under test by following the RPR policy parameterized by Θ, over an infinite horizon.*

Theorem 5 shows that the optimality criterion given in Definition 4 is the expected sum of discounted rewards in the limit, when the number of episodes approaches infinity. Throughout the paper, we call $\lim_{K\to\infty}\widehat{V}(\mathcal{D}^{(K)};\Theta)$ the value function and $\widehat{V}(\mathcal{D}^{(K)};\Theta)$ the empirical value function. The Θ maximizing the (empirical) value function is the best RPR policy (given the episodes).

It is assumed in Theorem 5 that the behavior policy Π used to collect the episodic data is an arbitrary policy that assigns nonzero probability to any action given any history, that is, Π is required to be a soft policy (Sutton and Barto, 1998). This premise assures a complete exploration of the actions that might lead to large immediate rewards given any history, that is, the actions that might be selected by the optimal policy.

4. Single-Task Reinforcement Learning (STRL) with RPR

We develop techniques to maximize the empirical value function in (11) and the Θ resulting from value maximization is called a Maximum-Value (MV) estimate (related to maximum *likelihood*). An MV estimate of the RPR is preferred when the number of episodes is large, in which case the empirical value function approaches the true value function and the estimate is expected to approach the optimal (assuming the algorithm is not trapped in a local minima). The episodes $\mathcal{D}^{(K)}$ are assumed to have been collected in a single partially observable stochastic environment, which may correspond to a single physical environment or a pool of multiple identical/similar physical environments. As a result, the techniques developed in this section are for single-task reinforcement learning (STRL).

By substituting (6) and (9) into (11), we rewrite the empirical value function

$$\widehat{V}(\mathcal{D}^{(K)};\Theta) = \frac{1}{K}\sum_{k=1}^{K}\sum_{t=0}^{T_k}\tilde{r}_t^k \sum_{z_0^k,\cdots,z_t^k=1}^{|\mathcal{Z}|} p(a_{0:t}^k, z_{0:t}^k | o_{1:t}^k, \Theta), \tag{12}$$

where

$$\tilde{r}_t^k = \frac{\gamma^t r_t^k}{\prod_{\tau=0}^{t} p^{\Pi}(a_\tau^k | h_\tau^k)}$$

is the discounted immediate reward $\gamma^t r_t^k$ weighted by the inverse probability that the behavior policy Π has generated r_t^k. The weighting is a result from importance sampling (Robert and Casella, 1999), and reflects the fact that r_t^k is obtained by following Π but the Monte Carlo integral (i.e., the empirical value function) is with respect to the RPR policy Θ. For simplicity, $\tilde{r}_t^k$ is also referred to as discounted immediate reward or simply reward throughout the paper.

We assume $r_t \geq 0$ (and hence $\widetilde{r}_t \geq 0$), which can always be achieved by adding a constant to r_t; this results in a constant added to the value function (the value function of a POMDP is linear in immediate reward) and does not influence the policy.

Theorem 6 *(Maximum Value Estimation) Let*

$$q_t^k(z_{0:t}^k|\Theta^{(n)}) = \frac{\tilde{r}_t^k}{\widehat{V}(\mathcal{D}^{(K)};\Theta^{(n)})} p(a_{0:t}^k, z_{0:t}^k | o_{1:t}^k, \Theta^{(n)}), \tag{13}$$

for $z_t^k = 1,2,\cdots,|\mathcal{Z}|$, $t = 1,2,\cdots,T_k$, *and* $k = 1,2,\cdots,K$. *Let*

$$\Theta^{(n+1)} = \arg\max_{\widehat{\Theta}\in\mathcal{F}} \frac{1}{K}\sum_{k=1}^{K}\sum_{t=0}^{T_k}\sum_{z_0^k,\cdots,z_t^k=1}^{|\mathcal{Z}|} q_t^k(z_{0:t}^k|\Theta^{(n)}) \ln \frac{\tilde{r}_t^k p(a_{0:t}^k, z_{0:t}^k | o_{1:t}^k, \widehat{\Theta})}{q_t^k(z_{0:t}^k|\Theta^{(n)})}, \tag{14}$$

where

$$\mathcal{F} = \left\{ \Theta = (\mu,\pi,W): \sum_{j=1}^{|\mathcal{Z}|}\widehat{\mu}(j) = 1, \sum_{a=1}^{|\mathcal{A}|}\widehat{\pi}(i,a) = 1, \sum_{j=1}^{|\mathcal{Z}|}\widehat{W}(i,a,o,j) = 1, \right.$$
$$\left. i = 1,2,\cdots,|\mathcal{Z}|, a = 1,2,\cdots,|\mathcal{A}|, o = 1,2,\cdots,|\mathcal{O}| \right\}$$

is the set of feasible parameters for the RPR in question. Let $\{\Theta^{(0)}\Theta^{(1)}\cdots\Theta^{(n)}\cdots\}$ *be a sequence yielded by iteratively applying (13) and (14), starting from* $\Theta^{(0)}$. *Then*

$$\lim_{n\to\infty}\widehat{V}(\mathcal{D}^{(K)};\Theta^{(n)})$$

exists and the limit is a maxima of $\widehat{V}(\mathcal{D}^{(K)};\Theta)$.

To gain a better understanding of Theorem 6, we rewrite (13) to get

$$q_t^k(z_{0:t}^k|\Theta) \quad = \quad \frac{\sigma_t^k(\Theta)}{\widehat{V}(\mathcal{D}^{(K)};\Theta)} p(z_{0:t}^k | a_{0:t}^k, o_{1:t}^k, \Theta), \tag{15}$$

where $p(z_{0:t}^k|a_{0:t}^k,o_{1:t}^k,\Theta)$ is an standard posterior distribution of the latent decision states given the Θ updated in the most recent iteration (the superscript $^{(n)}$ indicating the iteration number has been dropped for simplicity), and

$$\sigma_t^k(\Theta) \overset{Def.}{=} \tilde{r}_t^k p(a_{0:t}^k|o_{1:t}^k,\Theta) \tag{16}$$

is called the *re-computed reward* at time step t in the k-th episode. The re-computed reward represents the discounted immediate reward $\tilde{r}_t^k$ *weighted* by the probability that the action sequence yielding this reward is generated by the RPR policy parameterized by Θ, therefore $\sigma_t^k(\Theta)$ is a function of Θ. The re-computed reward reflects the update of the RPR policy which, if allowed to re-interact with the environment, is expected to accrue larger rewards than in the previous iteration. Recall that the algorithm does not assume real re-interactions with the environment so the episodes themselves cannot update. However, by recomputing the rewards as in (16), the agent is allowed to generate an *internal* set of episodes in which the immediate rewards are modified. The internal episodes represent the *new* episodes that would be collected if the agent followed the updated RPR to *really* re-interact with the environment. In this sense, the reward re-computation can be thought of as virtual re-interactions with the environment.

By (15), $q_t^k(z_{0:t}^k)$ is a weighted version of the standard posterior of $z_{0:t}^k$, with the weight given by the reward recomputed by the RPR in the previous iteration. The normalization constant $\widehat{V}(\mathcal{D}^{(K)};\Theta)$, which is also the empirical value function in (11), can be expressed as the recomputed rewards averaged over all episodes at all time steps,

$$\widehat{V}(\mathcal{D}^{(K)};\Theta) = \frac{1}{K}\sum_{k=1}^{K}\sum_{t=0}^{T_k}\sigma_t^k(\Theta), \tag{17}$$

which ensures

$$\frac{1}{K}\sum_{k=1}^{K}\sum_{t=0}^{T_k}\sum_{z_0^k,\cdots,z_t^k=1}^{|\mathcal{Z}|} q_t^k(z_{0:t}^k|\Theta) = 1.$$

The maximum value (MV) algorithm based on alternately applying (13) and (14) in Theorem 6 bears strong resemblance to the expectation-maximization (EM) algorithms (Dempster et al., 1977) widely used in statistics, with (13) and (14) respectively corresponding to the E-step and M-step in EM. However, the goal in standard EM algorithms is to maximize a likelihood function, while the goal of the MV algorithm is to maximize an empirical value function. This causes significant differences between the MV and the EM. It is helpful to compare the MV algorithm in Theorem 6 to the EM algorithm for maximum likelihood (ML) estimation in hidden Markov models (Rabiner, 1989), since both deal with sequences or episodes. The sequences in an HMM are treated as uniformly important, therefore parameter updating is based solely on the frequency of occurrences of latent states. Here the episodes are not equally important because they have different rewards associated with them, which determine their importance relative to each other. As seen in (15), the posterior of $z_{0:t}^k$ is weighted by the recomputed reward σ_t^k, which means that the contribution of episode k (at time t) to the update of Θ is not solely based on the frequency of occurrences of $z_{0:t}^k$ but also based on the associated σ_t^k. Thus the new parameters $\widehat{\Theta}$ will be adjusted in such a way that the episodes earning large rewards have more "credits" recorded into $\widehat{\Theta}$ and, as a result, the policy parameterized by $\widehat{\Theta}$ will more likely generate actions that lead to high rewards.

The objective function being maximized in (14) enjoys some interesting properties due to the fact that $q_t^k(z_{0:t}^k)$ is a weighted posterior of $z_{0:t}^k$. These properties not only establish a more formal connection between the MV algorithm here and the traditional ML algorithm based on EM, they also shed light on the close relations between Theorem 6 and the policy improvement theorem of POMDP (Blackwell, 1965). To show these properties, we rewrite the objective function in (14) (with the subscript $^{(n)}$ dropped for simplicity) as

$$\begin{aligned}\mathrm{LB}(\widehat{\Theta}|\Theta) &\overset{Def.}{=} \frac{1}{K}\sum_{k=1}^{K}\sum_{t=0}^{T_k}\sum_{z_0^k,\cdots,z_t^k=1}^{|Z|} q_t^k(z_{0:t}^k|\Theta)\ln\frac{\tilde{r}_t^k p(a_{0:t}^k, z_{0:t}^k|o_{1:t}^k,\widehat{\Theta})}{q_t^k(z_{0:t}^k|\Theta)} \\ &= \frac{1}{K}\sum_{k=1}^{K}\sum_{t=0}^{T_k}\frac{\sigma_t^k(\Theta)}{\widehat{V}(\mathcal{D}^{(K)};\Theta)}\sum_{z_0^k,\cdots,z_t^k=1}^{|Z|} p(z_{0:t}^k|a_{0:t}^k,o_{1:t}^k,\Theta)\ln\frac{\tilde{r}_t^k p(a_{0:t}^k, z_{0:t}^k|o_{1:t}^k,\widehat{\Theta})}{\frac{\sigma_t^k(\Theta)}{\widehat{V}(\mathcal{D}^{(K)};\Theta)}p(z_{0:t}^k|a_{0:t}^k,o_{1:t}^k,\Theta)},\end{aligned} \tag{18}$$

where the second equation is obtained by substituting (15) into the left side of it. Since $\frac{1}{K}\sum_{k=1}^{K}\sum_{t=0}^{T_k}\frac{\sigma_t^k(\Theta)}{\widehat{V}(\mathcal{D}^{(K)};\Theta)} = 1$ and $\sum_{z_0^k,\cdots,z_t^k=1}^{|Z|} p(z_{0:t}^k|a_{0:t}^k,o_{1:t}^k,\Theta) = 1$, one can apply Jensen's inequality *twice* to the rightmost side of (18) to obtain two inequalities

$$\begin{aligned}\mathrm{LB}(\widehat{\Theta}|\Theta) &\le \frac{1}{K}\sum_{k=1}^{K}\sum_{t=0}^{T_k}\frac{\sigma_t^k(\Theta)}{\widehat{V}(\mathcal{D}^{(K)};\Theta)}\ln\frac{\tilde{r}_t^k p(a_{0:t}^k|o_{1:t}^k,\widehat{\Theta})}{\frac{\sigma_t^k(\Theta)}{\widehat{V}(\mathcal{D}^{(K)};\Theta)}} \overset{Def.}{=} \Upsilon(\widehat{\Theta}|\Theta) \\ &\le \ln\left[\frac{1}{K}\sum_{k=1}^{K}\sum_{t=0}^{T_k}\tilde{r}_t^k p(a_{0:t}^k|o_{1:t}^k,\widehat{\Theta})\right] = \ln\widehat{V}(\mathcal{D}^{(K)};\widehat{\Theta}),\end{aligned} \tag{19}$$

where the first inequality is with respect to $p(z_{0:t}^k|a_{0:t}^k,o_{1:t}^k,\Theta)$ while the second inequality is with respect to $\left\{\frac{\sigma_t^k(\Theta)}{\widehat{V}(\mathcal{D}^{(K)};\Theta)} : t = 1,\cdots,T_k, k = 1,\cdots,K\right\}$. Each inequality yields a lower bound to the logarithmic empirical value function $\ln\widehat{V}(\mathcal{D}^{(K)};\widehat{\Theta})$. It is not difficult to verify from (18) and (19) that both of the two lower bounds are tight (the respective equality can be reached), that is,

$$\mathrm{LB}(\Theta|\Theta) = \ln\widehat{V}(\mathcal{D}^{(K)};\Theta) = \Upsilon(\Theta|\Theta). \tag{20}$$

The equations in (20) along with the inequalities in (19) show that any $\widehat{\Theta}$ satisfying $\mathrm{LB}(\Theta|\Theta) < \mathrm{LB}(\widehat{\Theta}|\Theta)$ or $\Upsilon(\Theta|\Theta) < \Upsilon(\widehat{\Theta}|\Theta)$ also satisfies $\widehat{V}(\mathcal{D}^{(K)};\Theta) < \widehat{V}(\mathcal{D}^{(K)};\widehat{\Theta})$. Thus one can choose to maximize either of the two lower bounds, $\mathrm{LB}(\widehat{\Theta}|\Theta)$ or $\Upsilon(\widehat{\Theta}|\Theta)$, when trying to improve the empirical value of $\widehat{\Theta}$ over that of Θ. In either case, the maximization is with respect to $\widehat{\Theta}$.

The two alternatives, though both yielding an improved RPR, are quite different in the manner the improvement is achieved. Suppose one has obtained $\Theta^{(n)}$ by applying (13) and (14) for n iterations, and is seeking $\Theta^{(n+1)}$ satisfying $\widehat{V}(\mathcal{D}^{(K)};\Theta^{(n)}) < \widehat{V}(\mathcal{D}^{(K)};\Theta^{(n+1)})$. Maximization of the first lower bound gives $\Theta^{(n+1)} = \arg\max_{\widehat{\Theta}\in\mathcal{F}}\mathrm{LB}(\widehat{\Theta}|\Theta^{(n)})$, which has an analytic solution that will be given in Section 4.2. Maximization of the second lower bound yields

$$\Theta^{(n+1)} = \arg\max_{\widehat{\Theta}\in\mathcal{F}}\Upsilon(\widehat{\Theta}|\Theta^{(n)}). \tag{21}$$

The definition of Υ in (19) is substituted into (21) to yield

$$\Theta^{(n+1)} = \arg\max_{\widehat{\Theta}\in\mathcal{F}}\frac{1}{K}\sum_{k=1}^{K}\sum_{t=0}^{T_k}\frac{\sigma_t^k(\Theta^{(n)})}{\widehat{V}(\mathcal{D}^{(K)};\Theta^{(n)})}\ln\frac{\tilde{r}_t^k p(a_{0:t}^k|o_{1:t}^k,\widehat{\Theta})}{\frac{\sigma_t^k(\Theta^{(n)})}{\widehat{V}(\mathcal{D}^{(K)};\Theta^{(n)})}}$$

$$= \quad \arg\max_{\widehat{\Theta}\in\mathcal{F}} \sum_{k=1}^{K}\sum_{t=0}^{T_k} \sigma_t^k(\Theta^{(n)}) \ln p(a_{0:t}^k|o_{1:t}^k,\widehat{\Theta}), \tag{22}$$

which shows that maximization of the second lower bound is equivalent to maximizing a weighted sum of the log-likelihoods of $\{a_{0:t}^k\}$, with the weights being the rewards recomputed by $\Theta^{(n)}$. Through (22), the connection between the maximum value algorithm in Theorem 6 and the traditional ML algorithm is made more formal and clearer: with the recomputed rewards given and fixed, the MV algorithm is a weighted version of the ML algorithm, with $\Upsilon(\widehat{\Theta}|\Theta^{(n)})$ a weighted log-likelihood function of $\widehat{\Theta}$.

The above analysis also sheds light on the relations between Theorem 6 and the policy improvement theorem in POMDP (Blackwell, 1965). By (19), (20), and (22), we have

$$\begin{aligned}\ln V(\mathcal{D}^{(K)};\Theta^{(n)}) = \Upsilon(\Theta^{(n)}|\Theta^{(n)}) &\leq \Upsilon(\Theta^{(n+1)}|\Theta^{(n)}) \\ &\leq \ln V(\mathcal{D}^{(K)};\Theta^{(n+1)}).\end{aligned}$$

The first inequality, achieved by the weighted likelihood maximization in (22), represents the policy improvement on the old episodes collected by following the previous policy. The second inequality ensures that, if the improved policy is followed to collect new episodes in the environment, the expected sum of newly accrued rewards is no less than that obtained by following the previous policy. This is similar to policy evaluation. Note that the update of episodes is simulated by reward computation. The actual episodes are collected by a fixed behavior policy Π and do not change.

The maximization in (22) can be performed using any optimization techniques. As long as the maximization is achieved, the policy is improved as guaranteed by Theorem 6. Since the latent z variables are involved, it is natural to employ EM to solve the maximization. The EM solution to (22) is obtained by solving a sequence of maximization problems: starting from $\Theta^{(n)(0)} = \Theta^{(n)}$, one successively solves

$$\Theta^{(n)(j)} = \arg\max_{\widehat{\Theta}\in\mathcal{F}} \mathrm{LB}(\widehat{\Theta}|\Theta^{(n)(j-1)}) \text{ subject to } \sigma_t^k(\Theta^{(n)(j-1)}) = \sigma_t^k(\Theta^{(n)}),\ \forall t,k, \tag{23}$$
$$j = 1,2,\cdots,$$

where in each problem one maximizes the first lower bound with an updated posterior of $\{z_t^k\}$ but with the recomputed rewards fixed at $\{\sigma_t^k(\Theta^{(n)})\}$; upon convergence, the solution of (23) is the solution to (22). The EM solution here is almost the same as the likelihood maximization of sequences for hidden Markov models (Rabiner, 1989). The only difference is that here we have a weighted log-likelihood function, but with the weights given and fixed. The posterior of $\{z_t^k\}$ can be updated by employing the dynamical programming techniques similar to those used in HMM, as we discuss below.

It is interesting to note that, with standard EM employed to solve (22), the overall maximum value algorithm is a "double-EM" algorithm, since reward computation constitutes an outer EM-like loop.

4.1 Calculating the Posterior of Latent Belief Regions

To allocate the weights or recomputed rewards and update the RPR as in (14), we do not need to know the full distribution of $z_{0:t}^k$. Instead, a small set of marginals of $p(z_{0:t}^k|a_{0:t}^k,o_{1:t}^k,\Theta)$ are necessary for the purpose, in particular,

$$\xi^k_{t,\tau}(i,j) = p(z^k_\tau = i, z^k_{\tau+1} = j | a^k_{0:t}, o^k_{1:t}, \Theta), \tag{24}$$
$$\phi^k_{t,\tau}(i) = p(z^k_\tau = i | a^k_{0:t}, o^k_{1:t}, \Theta). \tag{25}$$

Lemma 7 *(Factorization of the ξ and ϕ Variables) Let*

$$\begin{aligned} \alpha^k_\tau(i) &= p(z^k_\tau = i | a^k_{0:\tau}, o^k_{1:\tau}, \Theta) \\ &= \frac{p(z^k_\tau = i, a^k_{0:\tau} | o^k_{1:\tau}, \Theta)}{\prod_{\tau'=0}^{\tau} p(a^k_{\tau'} | h^k_{\tau'}, \Theta)}, \end{aligned} \tag{26}$$
$$\beta^k_{t,\tau}(i) = \frac{p(a^k_{\tau+1:t} | z^k_\tau = i, a^k_\tau, o^k_{\tau+1:t}, \Theta)}{\prod_{\tau'=\tau}^{t} p(a^k_{\tau'} | h^k_{\tau'}, \Theta)}. \tag{27}$$

Then

$$\xi^k_{t,\tau}(i,j) = \alpha^k_\tau(i) W(z^k_\tau = i, a^k_\tau, o^k_{\tau+1}, z^k_{\tau+1} = j) \pi(z^k_{\tau+1} = j, a^k_{\tau+1}) \beta^k_{t,\tau+1}(j), \tag{28}$$
$$\phi^k_{t,\tau}(i) = \alpha^k_\tau(i) \beta^k_{t,\tau}(i) p(a^k_\tau | h^k_\tau). \tag{29}$$

The α and β variables in the Lemma 7 are similar to the scaled forward variables and backward variables in hidden Markov models (HMM) (Rabiner, 1989). The scaling factors here are $\prod_{\tau'=0}^{\tau} p(a^k_{\tau'} | h^k_{\tau'}, \Theta)$, which is equal to $p(a^k_{0:\tau} | o^k_{1:\tau}, \Theta)$ as shown in (9) and (10). Recall from Definition 3 that one episode of length T has $T+1$ sub-episodes with each having a different ending time step. For this reason, one must compute the β variables for each sub-episode separately, since the β variables depend on the ending time step. For α variables, one needs to compute them once per episode, since it does not involve the ending time step.

Similar to the forward variables and backward variables in HMM models, the α and β variables can be computed recursively, via dynamical programming,

$$\alpha^k_\tau(i) = \begin{cases} \dfrac{\mu(z^k_0 = i)\pi(z^k_0 = i, a^k_0)}{p(a^k_0 | h^k_0, \Theta)}, & \tau = 0 \\ \dfrac{\sum_{j=1}^{|Z|} \alpha^k_{\tau-1}(j) W(z^k_{\tau-1} = j, a^k_{\tau-1}, o^k_\tau, z^k_\tau = i) \pi(z^k_\tau = i, a^k_\tau)}{p(a^k_\tau | h^k_\tau, \Theta)}, & \tau > 0 \end{cases}, \tag{30}$$
$$\beta^k_{t,\tau}(i) = \begin{cases} \dfrac{1}{p(a^k_t | h^k_t, \Theta)}, & \tau = t \\ \dfrac{\sum_{j=1}^{|Z|} W(z^k_\tau = i, a^k_\tau, o^k_{\tau+1}, z^k_{\tau+1} = j) \pi(z^k_{\tau+1} = j, a^k_{\tau+1}) \beta^k_{t,\tau+1}(j)}{p(a^k_\tau | h^k_\tau, \Theta)}, & \tau < t \end{cases}, \tag{31}$$

for $t = 0, \cdots, T_k$ and $k = 1, \cdots, K$. Since $\sum_{i=1}^{|Z|} \alpha^k_\tau(i) = 1$, it follows from (30) that

$$p(a^k_\tau | h^k_\tau, \Theta) = \begin{cases} \displaystyle\sum_{i=1}^{|Z|} \mu(z^k_0 = i) \pi(z^k_0 = i, a^k_0), & \tau = 0 \\ \displaystyle\sum_{i=1}^{|Z|} \sum_{j=1}^{|Z|} \alpha^k_{\tau-1}(j) W(z^k_{\tau-1} = j, a^k_{\tau-1}, o^k_\tau, z^k_\tau = i) \pi(z^k_\tau = i, a^k_\tau), & \tau > 0 \end{cases}. \tag{32}$$

4.2 Updating the Parameters

We rewrite the lower bound in (18),

$$\begin{aligned}\mathrm{LB}(\widehat{\Theta}|\Theta) &= \frac{1}{K}\sum_{k=1}^{K}\sum_{t=0}^{T_k}\sum_{z_0^k,\cdots,z_t^k=1}^{|\mathcal{Z}|} q_t^k(z_{0:t}^k|\Theta^{(n)})\ln\frac{\tilde{r}_t^k p(a_{0:t}^k,z_{0:t}^k|o_{1:t}^k,\widehat{\Theta})}{q_t^k(z_{0:t}^k|\Theta^{(n)})}\\ &= \frac{1}{K}\sum_{k=1}^{K}\sum_{t=0}^{T_k}\sum_{z_0^k,\cdots,z_t^k=1}^{|\mathcal{Z}|} q_t^k(z_{0:t}^k|\Theta^{(n)})\ln p(a_{0:t}^k,z_{0:t}^k|o_{1:t}^k,\widehat{\Theta})+\text{constant},\end{aligned}$$

where the "constant" collects all the terms irrelevant to $\widehat{\Theta}$. Substituting (5) and (15) gives

$$\begin{aligned}\mathrm{LB}(\widehat{\Theta}|\Theta) &= \frac{1}{K}\sum_{k=1}^{K}\sum_{t=0}^{T_k}\frac{\sigma_t^k}{\widehat{V}(\mathcal{D}^{(K)};\Theta)}\Bigg\{\sum_{i=1}^{|\mathcal{Z}|}\phi_{t,0}^k(i)\ln\widehat{\mu}(i)+\sum_{\tau=0}^{t}\sum_{i=1}^{|\mathcal{Z}|}\phi_{t,\tau}^k(i)\ln\widehat{\pi}(i,a_\tau^k)\\ &\qquad +\sum_{\tau=1}^{t}\sum_{i,j=1}^{|\mathcal{Z}|}\xi_{t,\tau}^k(i,j)\ln\widehat{W}(i,a_{\tau-1}^k,o_\tau^k,j)\Bigg\}+\text{constant}.\end{aligned}$$

It is not difficult to show that $\widehat{\Theta}=\arg\max_{\widehat{\Theta}\in\mathcal{F}}\mathrm{LB}(\widehat{\Theta}|\Theta)$ is given by

$$\widehat{\mu}(i) = \frac{\sum_{k=1}^{K}\sum_{t=0}^{T_k}\sigma_t^k\phi_{t,0}^k(i)}{\sum_{i=1}^{|\mathcal{Z}|}\sum_{k=1}^{K}\sum_{t=0}^{T_k}\sigma_t^k\phi_{t,0}^k(i)}, \tag{33}$$

$$\widehat{\pi}(i,a) = \frac{\sum_{k=1}^{K}\sum_{t=0}^{T_k}\sigma_t^k\sum_{\tau=0}^{t}\phi_{t,\tau}^k(i)\,\delta(a_\tau^k,a)}{\sum_{a=1}^{|\mathcal{A}|}\sum_{k=1}^{K}\sum_{t=0}^{T_k}\sigma_t^k\sum_{\tau=0}^{t}\phi_{t,\tau}^k(i)\,\delta(a_\tau^k,a)}, \tag{34}$$

$$\widehat{W}(i,a,o,j) = \frac{\sum_{k=1}^{K}\sum_{t=0}^{T_k}\sigma_t^k\sum_{\tau=1}^{t-1}\xi_{t,\tau}^k(i,j)\,\delta(a_\tau^k,a)\,\delta(o_{\tau+1}^k,o)}{\sum_{j=1}^{|\mathcal{Z}|}\sum_{k=1}^{K}\sum_{t=0}^{T_k}\sigma_t^k\sum_{\tau=1}^{t-1}\xi_{t,\tau}^k(i,j)\,\delta(a_\tau^k,a)\,\delta(o_{\tau+1}^k,o)}, \tag{35}$$

for $i,j=1,2,\cdots,|\mathcal{Z}|$, $a=1,\cdots,|\mathcal{A}|$, and $o=1,\cdots,|O|$, where $\delta(a,b)=\begin{cases}1, & a=b\\ 0, & a\neq b\end{cases}$, and σ_t^k is the recomputed reward as defined in (16). In computing σ_t^k one employs the equation $p(a_{0:t}^k|o_{1:t}^k,\Theta)=\prod_{\tau=0}^{t}p(a_\tau^k|h_\tau^k,\Theta)$ established in (9) and (10), to get

$$\sigma_t^k(\Theta)\overset{Def.}{=}\tilde{r}_t^k\prod_{\tau=0}^{t}p(a_\tau^k|h_\tau^k,\Theta), \tag{36}$$

with $p(a_\tau^k|h_\tau^k,\Theta)$ computed from the α variables by using (32). Note that the normalization constant, which is equal to the empirical value $\widehat{V}(\mathcal{D}^{(K)};\Theta)$, is now canceled in the update formulae of $\widehat{\Theta}$.

4.3 The Complete Value Maximization Algorithm for Single-Task RPR Learning

The complete value maximization algorithm for single-task RPR learning is summarized in Table 1. In earlier discussions regarding the relations of the algorithm to EM, we have mentioned that reward computation constitutes an outer EM-like loop; the standard EM employed to solve (22) is embedded in the outer loop and constitutes an inner EM loop. The double EM loops are not explicitly shown in Table 1. However, one may separate these two loops by keeping $\{\sigma_t^k\}$ fixed

```
Input: $\mathcal{D}^{(K)}$, $\mathcal{A}$, $\mathcal{O}$, $|\mathcal{Z}|$.
Output: $\Theta = \{\mu, \pi, W\}$.

1. Initialize $\Theta$, $\ell = []$, iteration $= 1$.
2. Repeat
   2.1 Dynamical programming:
       Compute $\alpha$ and $\beta$ variables with Equations (30)-(32).
   2.2 Reward re-computation:
       Calculate $\{\sigma_t^k\}$ using (36) and (32).
   2.3 Convergence check:
       Compute $\ell(\text{iteration}) = \widehat{V}(\mathcal{D}^{(K)};\Theta)$ using (17).
       If the sequence of $\ell$ converges
           Stop the algorithm and exit.
       Else
           iteration := iteration $+ 1$
   2.4 Posterior update for $z$:
       Compute the $\xi$ and $\phi$ variables using Equations (28)-(29).
   2.5 Update of $\Theta$:
       Compute the updated $\Theta$ using (33), (34), and (35).
```

Table 1: The value maximization algorithm for single-task RPR learning

when updating Θ and the posterior of z's, until the empirical value converges; see (23) for details. Once $\{\sigma_t^k\}$ are updated, the empirical value will further increase by continuing updating Θ and the posterior of z's. Note that the $\{\sigma_t^k\}$ used in the convergence check are always updated at each iteration, even though the new $\{\sigma_t^k\}$ may not be used for updating Θ and the posterior of z's.

Given a history of actions and observations $(a_{0:t-1}, o_{1:t})$ collected up to time step t, the single RPR yields a distribution of a_t as given by (8). The optimal choice for a_t can be obtained by either sampling from this distribution or taking the action that maximizes the probability.

4.3.1 TIME COMPLEXITY ANALYSIS

We quantify the time complexity by the number of real number multiplications performed per iteration and present it in the Big-O notation. Since there is no compelling reason for the number of iterations to depend on the size of the input,[2] the complexity per iteration also represents the complexity of the complete algorithm. A stepwise analysis of the time complexity of the value maximization algorithm in Table 1 is given as follows.

- Computation of the α variables with (30) and (32) runs in time $O(|\mathcal{Z}|^2 \sum_{k=1}^K T_k)$.
- Computation of β's with (31) and (32) runs in time $O(|\mathcal{Z}|^2 \sum_{k=1}^K \sum_{t=0, r_t^k \neq 0}^{T_k} (t+1))$, which depends on the degree of sparsity of the immediate rewards $\{r_0^k r_2^k \cdots r_{T_k}^k\}_{k=1}^K$. In the worst case

2. The number of iterations usually depends on such factors as initialization of the algorithm and the required accuracy, etc.

the time is $O(|\mathcal{Z}|^2 \sum_{k=1}^{K} \sum_{t=0}^{T_k}(t+1)) = O(|\mathcal{Z}|^2 \sum_{k=1}^{K} T_k^2)$, which occurs when the immediate reward in each episode is nonzero at every time step. In the best case the time is $O(|\mathcal{Z}|^2 \sum_{k=1}^{K} T_k)$, which occurs when the immediate reward in each episode is nonzero only at a fixed number of time steps (only at the last time step, for example, as is the case of the benchmark problems presented in Section 6).

- The reward re-computation using (36) and (32) requires time $O(\sum_{k=1}^{K} T_k)$ in the worst case and $O(K)$ in the best case, where the worse/best cases are as defined above.

- Update of Θ using (33), (34), and (35), as well as computation of the ξ and ϕ variables using (28) and (29), runs in time $O(|\mathcal{Z}|^2 \sum_{k=1}^{K} T_k^2)$ in the worst case and $O(|\mathcal{Z}|^2 \sum_{k=1}^{K} T_k)$ in the best case, where the worse/best cases are defined above.

Since $\sum_{k=1}^{K} T_k \gg |\mathcal{A}||O|$ in general, the overall complexity of the value maximization algorithm is $O(|\mathcal{Z}|^2 \sum_{k=1}^{K} T_k^2)$ in the worst case and $O(|\mathcal{Z}|^2 \sum_{k=1}^{K} T_k)$ in the best case, depending on the degree of sparsity of the immediate rewards. Therefore the algorithm scales linearly with the number of episodes and to the square of the number of belief regions. The time dependency on the lengths of episodes is between linear and square. The sparser the immediate rewards are, the more the time is towards being linear in the lengths of episodes.

Note that in many reinforcement problems, the agent does not receive immediate rewards at every time step. For the benchmark problems and maze navigation problems considered in Section 6, the agent receives rewards only when the goal state is reached, which makes the value maximization algorithm scale linearly with the lengths of episodes.

5. Multi-Task Reinforcement Learning (MTRL) with RPR

We formulate our MTRL framework by placing multiple RPRs in a Bayesian setting and develop techniques to learn the posterior of each RPR within the context of all other RPRs.

Several notational conventions are observed in this section. The posterior of Θ is expressed in terms of probability density functions. The notation $G_0(\Theta)$ is reserved to denote the density function of a parametric prior distribution, with the associated probability measure denoted by G_0 without a parenthesized Θ beside it. For the Dirichlet process (which is a nonparametric prior), G_0 denotes the base measure and $G_0(\Theta)$ denotes the corresponding density function. The twofold use of G_0 is for notational simplicity; the difference can be easily discerned by the presence or absence of a parenthesized Θ. The δ is a Dirac delta for continuous arguments and a Kronecker delta for discrete arguments. The notation δ_{Θ_j} is the Dirac measure satisfying $\delta_{\Theta_j}(d\Theta_m) = \begin{cases} 1, & \Theta_j \in d\Theta_m \\ 0, & \text{otherwise} \end{cases}$.

5.1 Basic Bayesian Formulation of RPR

Consider M partially observable and stochastic environments indexed by $m = 1, 2 \cdots, M$, each of which is apparently different from the others but may actually share fundamental common characteristics with some other environments. Assume we have a set of episodes collected from each environment, $\mathcal{D}_m^{(K_m)} = \left\{ (a_0^{m,k} r_0^{m,k} o_1^{m,k} a_1^{m,k} r_1^{m,k} \cdots o_{T_{m,k}}^{m,k} a_{T_{m,k}}^{m,k} r_{T_{m,k}}^{m,k}) \right\}_{k=1}^{K_m}$, for $m = 1, 2, \cdots, M$, where $T_{m,k}$ represents the length of episode k in environment m. Following the definitions in Section 3, we write the empirical value function of the m-th environment as

$$\widehat{V}(\mathcal{D}_m^{(K_m)};\Theta_m) = \frac{1}{K_m}\sum_{k=1}^{K_m}\sum_{t=0}^{T_{m,k}} \tilde{r}_t^{m,k} p(a_{0:t}^{m,k}|o_{1:t}^{m,k},\Theta_m), \tag{37}$$

for $m = 1,2,\cdots,M$, where $\Theta_m = \{\pi_m,\mu_m,W_m\}$ are the RPR parameters for the m-th individual environment.

Let $G_0(\Theta_m)$ represent the prior of Θ_m, where $G_0(\Theta)$ is assumed to be the density function of a probability distribution. We define the posterior of Θ_m as

$$p(\Theta_m|\mathcal{D}_m^{(K_m)},G_0) \overset{Def.}{=} \frac{\widehat{V}(\mathcal{D}_m^{(K_m)};\Theta_m)G_0(\Theta_m)}{\widehat{V}_{G_0}(\mathcal{D}_m^{(K_m)})}, \tag{38}$$

where the inclusion of G_0 in the left hand side is to explicitly indicate that the prior being used is G_0, and $\widehat{V}_{G_0}(\mathcal{D}_m^{(K_m)})$ is a normalization constant

$$\widehat{V}_{G_0}(\mathcal{D}_m^{(K_m)}) \overset{Def.}{=} \int \widehat{V}(\mathcal{D}_m^{(K_m)};\Theta_m)G_0(\Theta_m)d\Theta_m, \tag{39}$$

which is also referred to as the *marginal empirical value*,[3] since the parameters Θ_m are integrated out (marginalized). The marginal empirical value $\widehat{V}_{G_0}(\mathcal{D}_m^{(K_m)})$ represents the accumulated discounted reward in the episodes, averaged over infinite RPR policies independently drawn from G_0.

Equation (38) is literally a normalized product of the empirical value function and a prior $G_0(\Theta_m)$. Since $\int p(\Theta_m|\mathcal{D}_m^{(K_m)},G_0)d\Theta_m = 1$, (38) yields a valid probability density, which we call the posterior of Θ_m given the episodes $\mathcal{D}_m^{(K_m)}$. It is noted that (38) would be the Bayes rule if $\widehat{V}(\mathcal{D}_m^{(K_m)};\Theta_m)$ were a likelihood function. Since $\widehat{V}(\mathcal{D}_m^{(K_m)};\Theta_m)$ is a value function in our case, (38) is a somewhat non-standard use of Bayes rule. However, like the classic Bayes rule, (38) indeed gives a posterior whose shape incorporates both the prior information about Θ_m and the empirical information from the episodes.

Equation (38) has another interpretation that may be more meaningful from the perspective of standard probability theory. To see this we substitute (37) into (38) to obtain

$$\begin{aligned} p(\Theta_m|\mathcal{D}_m^{(K_m)},G_0) &= \frac{\frac{1}{K_m}\sum_{k=1}^{K_m}\sum_{t=0}^{T_{m,k}} \tilde{r}_t^{m,k} p(a_{0:t}^{m,k}|o_{1:t}^{m,k},\Theta_m)G_0(\Theta_m)}{\widehat{V}_{G_0}(\mathcal{D}_m^{(K_m)})} && (40)\\ &= \frac{\frac{1}{K_m}\sum_{k=1}^{K_m}\sum_{t=0}^{T_{m,k}} \zeta_t^{m,k} p(\Theta_m|a_{0:t}^{m,k},o_{1:t}^{m,k},G_0)}{\widehat{V}_{G_0}(\mathcal{D}_m^{(K_m)})}, && (41)\end{aligned}$$

where

$$\begin{aligned} \zeta_t^{m,k} &= \tilde{r}_t^{m,k} p(a_{0:t}^{m,k}|o_{1:t}^{m,k},G_0)\\ &= \tilde{r}_t^{m,k} \int p(a_{0:t}^{m,k}|o_{1:t}^{m,k},\Theta_m)G_0(\Theta_m)d\Theta_m\\ &= \int \sigma_t^{m,k}(\Theta_m)G_0(\Theta_m)d\Theta_m, && (42)\end{aligned}$$

3. The term "marginal" is borrowed from the probability theory. Here we use it to indicate that the dependence of the value on the parameter is removed by integrating out the parameter.

with $\sigma_t^{m,k}$ the re-computed reward as defined in (16) and therefore $\zeta_t^{m,k}$ is the averaged re-computed reward, obtained by taking the expectation of $\sigma_t^{m,k}(\Theta_m)$ with respect to $G_0(\Theta_m)$.

In arriving (41), we have used the fact the RPR parameters are independent of the observations, which is true due to the following reasons: RPR is a policy concerning generation of the actions, employing as input the observations (which themselves are generated by the unknown environment); therefore, observations carry no information about the RPR parameters, that is, $p(\Theta|\text{observations}) = p(\Theta) \equiv G_0(\Theta)$.

It is noted that $p(\Theta_m|a_{0:t}^{m,k}, o_{1:t}^{m,k}, G_0)$ in (41) is the standard posterior of Θ_m given the action sequence $a_{0:t}^{m,k}$, and $p(\Theta_m|\mathcal{D}_m^{(K_m)}, G_0)$ is a mixture of these posteriors with the mixing proportion given by $\zeta_t^{m,k}$. The meaning of (40) is fairly intuitive: each action sequence affects the posterior of Θ_m in proportion to its re-evaluated reward. This is distinct from the posterior in the classic hidden Markov model (Rabiner, 1989) where sequences are treated as equally important.

Since $p(\Theta_m|\mathcal{D}_m^{(K_m)}, G_0)$ integrates to one, the normalization constant $\widehat{V}_{G_0}(\mathcal{D}_m^{(K_m)})$ is

$$\widehat{V}_{G_0}(\mathcal{D}_m^{(K_m)}) = \frac{1}{K_m} \sum_{k=1}^{K_m} \sum_{t=0}^{T_{m,k}} \zeta_t^{m,k}. \tag{43}$$

We obtain a more convenient form of the posterior by substituting (6) into (41) to expand the summation over the latent z variables, yielding

$$p(\Theta_m|\mathcal{D}_m^{(K_m)}, G_0) = \frac{\frac{1}{K_m} \sum_{k=1}^{K_m} \sum_{t=0}^{T_{m,k}} \tilde{r}_t^{m,k} \sum_{z_0^{m,k},\cdots,z_t^{m,k}=1}^{|\mathcal{Z}|} p(a_{0:t}^{m,k}, z_{0:t}^{m,k}|o_{1:t}^{m,k}, \Theta_m) G_0(\Theta_m)}{\widehat{V}_{G_0}(\mathcal{D}_m^{(K_m)})}. \tag{44}$$

To obtain an analytic posterior, we let the prior be conjugate to $p(a_{0:t}^{m,k}, z_{0:t}^{m,k}|o_{1:t}^{m,k}, \Theta_m)$. As shown by (5), $p(a_{0:t}^{m,k}, z_{0:t}^{m,k}|o_{1:t}^{m,k}, \Theta_m)$ is a product of multinomial distributions, and hence we choose the prior as a product of Dirichlet distributions, with each Dirichlet representing an independent prior for a subset of parameters in Θ. The density function of such a prior is given by

$$\begin{aligned}
G_0(\Theta_m) &= p(\mu^m|\upsilon)p(\pi^m|\rho)p(W^m|\omega), && (45)\\
p(\mu^m|\upsilon) &= \mathrm{Dir}\Big(\mu^m(1), \cdots, \mu^m(|\mathcal{Z}|)\Big|\upsilon\Big), && (46)\\
p(\pi^m|\rho) &= \prod_{i=1}^{|\mathcal{Z}|} \mathrm{Dir}\Big(\pi^m(i,1), \cdots, \pi^m(i,|\mathcal{A}|)\Big|\rho_i\Big), && (47)\\
p(W^m|\omega) &= \prod_{a=1}^{|\mathcal{A}|}\prod_{o=1}^{|\mathcal{O}|}\prod_{i=1}^{|\mathcal{Z}|} \mathrm{Dir}\Big(W^m(i,a,o,1), \cdots, W^m(i,a,o,|\mathcal{Z}|)\Big|\omega_{i,a,o}\Big), && (48)
\end{aligned}$$

where $\upsilon = \{\upsilon_1, \ldots, \upsilon_{|\mathcal{Z}|}\}$, $\rho = \{\rho_1, \ldots, \rho_{|\mathcal{Z}|}\}$ with $\rho_i = \{\rho_{i,1}, \ldots, \rho_{i,|\mathcal{A}|}\}$, and $\omega = \{\omega_{i,a,o} : i = 1 \ldots |\mathcal{Z}|, a = 1 \ldots |\mathcal{A}|, o = 1 \ldots |\mathcal{O}|\}$ with $\omega_{i,a,o} = \{\omega_{i,a,o,1}, \ldots, \omega_{i,a,o,|\mathcal{Z}|}\}$. Substituting the expression of G_0 into (44), one gets

$$\begin{aligned}
&p(\Theta_m|\mathcal{D}_m^{(K_m)}, G_0)\\
&= \frac{\frac{1}{K_m} \sum_{k=1}^{K_m} \sum_{t=0}^{T_{m,k}} \sum_{z_0^{m,k},\cdots,z_t^{m,k}=1}^{|\mathcal{Z}|} \zeta_t^{m,k}(z_{0:t}^{m,k})\, p(\Theta_m|a_{0:t}^{m,k}, o_{1:t}^{m,k}, z_{0:t}^{m,k}, G_0)}{\widehat{V}_{G_0}(\mathcal{D}_m^{(K_m)})},
\end{aligned}$$

where

$$\begin{aligned}\zeta_t^{m,k}(z_{0:t}^{m,k}) &= \tilde{r}_t^{m,k}\int p(a_{0:t}^{m,k},z_{0:t}^{m,k}|o_{1:t}^{m,k},\Theta_m)G_0(\Theta_m)d\Theta_m\\ &= \tilde{r}_t^{m,k}\frac{\prod_i\Gamma(\widehat{\upsilon}_i^{m,k,t})}{\Gamma(\sum_i\widehat{\upsilon}_i^{m,k,t})}\frac{\Gamma(\sum_i\upsilon_i^{m,k,t})}{\prod_i\Gamma(\upsilon_i^{m,k,t})}\frac{\prod_i\prod_a\Gamma(\widehat{\rho}_{i,a}^{m,k,t})}{\prod_i\Gamma(\sum_a\widehat{\rho}_{i,a}^{m,k,t})}\frac{\prod_i\Gamma(\sum_a\rho_{i,a}^{m,k,t})}{\prod_i\prod_a\Gamma(\rho_{i,a}^{m,k,t})}\\ &\quad\times\frac{\prod_a\prod_o\prod_i\prod_j\Gamma(\widehat{\omega}_{i,a,o,j}^{m,k,t})}{\prod_a\prod_o\prod_i\Gamma(\sum_j\widehat{\omega}_{i,a,o,j}^{m,k,t})}\frac{\prod_a\prod_o\prod_i\Gamma(\sum_j\omega_{i,a,o,j}^{m,k,t})}{\prod_a\prod_o\prod_i\prod_j\Gamma(\omega_{i,a,o,j}^{m,k,t})}\end{aligned}\tag{49}$$

represents the averaged recomputed reward over a given z sequence $z_{0:t}^{m,k}$, and

$$p(\Theta_m|a_{0:t}^{m,k},o_{1:t}^{m,k},z_{0:t}^{m,k},G_0) = p(\mu^m|\widehat{\upsilon}^{m,k,t})p(\pi^m|\widehat{\rho}^{m,k,t})p(W^m|\widehat{\omega}^{m,k,t})$$

is the density of a product of Dirichlet distributions and has the same form as $G_0(\Theta)$ in (45) but with υ,ρ,ω respectively replaced by $\widehat{\upsilon}^{m,k,t}$, $\widehat{\rho}^{m,k,t}$, $\widehat{\omega}^{m,k,t}$ as given by

$$\widehat{\upsilon}_i^{m,k,t} = \upsilon_i^m+\delta(z_0^{m,k}-i),\tag{50}$$

$$\widehat{\rho}_{i,a}^{m,k,t} = \rho_{i,a}^m+\sum_{\tau=0}^{t}\delta(z_\tau^{m,k}-i)\delta(a_\tau^{m,k}-a),\tag{51}$$

$$\widehat{\omega}_{i,a,o,j}^{m,k,t} = \omega_{i,a,o,j}^m+\sum_{\tau=1}^{t}\delta(z_{\tau-1}^{m,k}-i)\delta(a_{\tau-1}^{m,k}-a)\delta(o_\tau^{m,k}-o)\delta(z_\tau^{m,k}-j).\tag{52}$$

The normalization constant $\widehat{V}_{G_0}(\mathcal{D}_m^{(K_m)})$ (which is also the marginal empirical value) can now be expressed as

$$\widehat{V}_{G_0}(\mathcal{D}_m^{(K_m)}) = \frac{1}{K_m}\sum_{k=1}^{K_m}\sum_{t=0}^{T_{m,k}}\sum_{z_0^{m,k},\cdots,z_t^{m,k}=1}^{|\mathcal{Z}|}\zeta_t^{m,k}(z_{0:t}^{m,k}).\tag{53}$$

5.2 The Dirichlet Process Prior

In order to identify related tasks and introduce sharing mechanisms for multi-task learning, we employ the Dirichlet process (Ferguson, 1973; Blackwell and MacQueen, 1973; Antoniak, 1974; Sethuraman, 1994) as a nonparametric prior that is shared by Θ_m, $m=1,2,\cdots,M$. A draw from a DP has the nice property of being almost surely discrete (Blackwell and MacQueen, 1973), which is known to promote clustering (West et al., 1994); therefore, related tasks (as judged by the empirical value function) are encouraged to be placed in the same group and be learned simultaneously by sharing the episodic data across all tasks in the same group. Assuming the prior of Θ_m, $m=1,2,\cdots,M$, is drawn from a Dirichlet process with base measure G_0 and precision α, we have

$$\begin{aligned}\Theta_m|G &\sim G,\\ G|\alpha,G_0 &\sim DP(\alpha,G_0),\end{aligned}$$

where the precision α provides an expected number of dominant clusters, with this driven by the number of samples (West, 1992). It usually suffices to set the precision α using the rule in West (1992). If desired, however, one may also put a Gamma prior on α and draw from its posterior

(Escobar and West, 1995), which yields greater model flexibility. Note the DP precision is denoted by the same symbol as the α variables in (26). The difference is easy to recognize, since the former is a single quantity bearing neither superscripts and nor subscripts while the latter represent a set of variables and always bear superscripts and subscripts.

By marginalizing out G, one obtains the Polya-urn representation of DP (Blackwell and MacQueen, 1973), expressed in terms of density functions,[4]

$$p(\Theta_m|\Theta_{-m},\alpha,G_0)=\frac{\alpha}{\alpha+M-1}G_0(\Theta_m)+\frac{1}{\alpha+M-1}\sum_{\substack{j=1\\j\neq m}}^{M}\delta(\Theta_m-\Theta_j),\;\; m=1,\cdots,M, \tag{54}$$

where the probability is conditioned on $\Theta_{-m}=\{\Theta_1,\Theta_2,\cdots,\Theta_M\}\setminus\{\Theta_m\}$. The Polya-urn representation in (54) gives a set of full conditionals for the joint prior $p(\Theta_1,\Theta_2,\cdots,\Theta_M)$.

The fact that $G\sim DP(\alpha,G_0)$ is almost surely discrete implies that the set $\{\Theta_1,\,\Theta_2,\,\cdots,\,\Theta_M\}$, which are iid drawn from G, can have duplicate elements and the number of distinct elements N cannot exceed M, the total number of environments. It is useful to consider an equivalent representation of (54) based on the distinct elements (Neal, 1998). Let $\overline{\Theta}=\{\overline{\Theta}_1,\overline{\Theta}_2,\cdots,\overline{\Theta}_N\}$ represent the set of distinct elements of $\{\Theta_1,\Theta_2,\cdots,\Theta_M\}$, with $N\le M$. Let $c=\{c_1,c_2,\ldots,c_M\}$ denote the vector of indicator variables defined by $c_m=n$ iff $\Theta_m=\overline{\Theta}_n$ and $c_{-m}=\{c_1,c_2,\cdots,c_M\}\setminus\{c_m\}$. The prior conditional distribution $p(c_m|c_{-m})$ that arises from the Polya-urn representation of the Dirichlet process is as follows (MacEachern, 1994)

$$p(c_m|c_{-m},\alpha)=\frac{\alpha}{\alpha+M-1}\delta(c_m)+\sum_{n=1}^{N}\frac{l_{-m,n}}{\alpha+M-1}\delta(c_m-n), \tag{55}$$

where $l_{-m,n}$ denotes the number of elements in $\{i:c_i=n,i\neq m\}$ and $c_m=0$ indicates a new sample is drawn from the base G_0. Given c_m and $\overline{\Theta}$, the density of Θ_m is given by

$$p(\Theta_m|c_m,\overline{\Theta},G_0)=\delta(c_m)G_0(\Theta_m)+\sum_{n=1}^{N}\delta(c_m-n)\delta(\Theta_m-\overline{\Theta}_n). \tag{56}$$

5.3 The Dirichlet Process Posterior

We take two steps to derive the posterior based on the representation of the DP prior given by (55) and (56). First we write the conditional posterior of c_m, $\forall\, m\in\{1,\cdots,M\}$,

$$p(c_m|c_{-m},\overline{\Theta},\mathcal{D}_m^{(K_m)},\alpha,G_0)=\frac{\int\widehat{V}(\mathcal{D}_m^{(K_m)};\Theta_m)p(\Theta_m|c_m,\overline{\Theta},G_0)p(c_m|c_{-m},\alpha)d\Theta_m}{\sum_{c_m=0}^{N}\int\widehat{V}(\mathcal{D}_m^{(K_m)};\Theta_m)p(\Theta_m|c_m,\overline{\Theta},G_0)p(c_m|c_{-m},\alpha)d\Theta_m},$$

which is rewritten, by substituting (55) and (56) into the righthand side, to yield an algorithmically more meaningful expression

$$p(c_m|c_{-m},\overline{\Theta},\mathcal{D}_m^{(K_m)},\alpha,G_0)=\frac{\alpha\widehat{V}_{G_0}(\mathcal{D}_m^{(K_m)})\,\delta(c_m)+\sum_{n=1}^{N}l_{-m,n}\widehat{V}(\mathcal{D}_m^{(K_m)};\overline{\Theta}_n)\,\delta(c_m-n)}{\alpha\widehat{V}_{G_0}(\mathcal{D}_m^{(K_m)})+\sum_{j=1}^{N}l_{-m,j}\widehat{V}(\mathcal{D}_m^{(K_m)};\overline{\Theta}_j)}, \tag{57}$$

4. The corresponding expression in terms of probability measures (Escobar and West, 1995) is given by

$$\Theta_m|\Theta_{-m},\alpha,G_0\sim\frac{\alpha}{\alpha+M-1}G_0+\frac{1}{\alpha+M-1}\textstyle\sum_{j=1,j\neq m}^{M}\delta_{\Theta_j},\;\; m=1,\cdots,M,$$

where δ_{Θ_j} is the Dirac measure.

where the $\widehat{V}_{G_0}(\mathcal{D}_m^{(K_m)})$ is the marginal empirical value defined in (39) and its expression is given by (53) when the DP base has a density function as specified in (45).

It is observed from (57) that the indicator c_m tends to equal n if $\widehat{V}(\mathcal{D}_m^{(K_m)};\overline{\Theta}_n)$ is large, which occurs when the n-th distinct RPR produces a high empirical value in the m-th environment. If none of the other RPRs produces a high empirical value in the m-th environment, c_m will tend to be equal to zero, which means a new cluster will be generated to account for the novelty. The merit of generating a new cluster is measured by the empirical value weighted by α and averaged with respect to G_0. Therefore the number of distinct RPRs is jointly dictated by the DP prior and the episodes.

Given the indicator variables c, the clusters are formed. Let $I_n(c) = \{m : c_m = n\}$ denote the indices of the environments that have been assigned to the n-th cluster. Given the clusters, we now derive the conditional posterior of $\overline{\Theta}_n, \forall\, n \in \{1,\cdots,N\}$. If $I_n(c)$ is an empty set, there is no empirical evidence available for it to obtain a posterior, therefore one simply removes this cluster. If $I_n(c)$ is nonempty, the density function of the conditional posterior of $\overline{\Theta}_n$ is given by

$$p(\overline{\Theta}_n|\textstyle\bigcup_{m\in I_n(c)}\mathcal{D}_m^{(K_m)},G_0) = \dfrac{\sum_{m\in I_n(c)}\widehat{V}(\mathcal{D}_m^{(K_m)};\overline{\Theta}_n)G_0(\overline{\Theta}_n)}{\int\sum_{m\in I_n(c)}\widehat{V}(\mathcal{D}_m^{(K_m)};\overline{\Theta}_n)G_0(\overline{\Theta}_n)\,d\overline{\Theta}_n} \tag{58}$$

$$= \dfrac{\sum_{m\in I_n(c)}\frac{1}{K_m}\sum_{k=1}^{K_m}\sum_{t=0}^{T_{m,k}}\tilde{r}_t^{m,k}\sum_{z_0^{m,k},\cdots,z_t^{m,k}=1}^{|\mathcal{Z}|}p(a_{0:t}^{m,k},z_{0:t}^{m,k}|o_{1:t}^{m,k},\overline{\Theta}_n)G_0(\overline{\Theta}_n)}{\sum_{m\in I_n(c)}\widehat{V}_{G_0}(\mathcal{D}_m^{(K_m)})}, \tag{59}$$

where (59) results from substituting (12) into the righthand side of (58). Note that $\overline{\Theta}_n$, which represents the set of parameters of the n-th distinct RPR, is conditioned on all episodes aggregated across all environments in the n-th cluster. The posterior in (58) has the same form as the definition in (38) and it is obtained by applying Bayes law to the empirical value function constructed from the aggregated episodes. As before, the Bayes law is applied in a nonstandard manner, treating the value function as if it were a likelihood function.

A more concrete expression of (59) can be obtained by letting the DP base G_0 have a density function as in (45),

$$\begin{aligned}&p(\overline{\Theta}_n|\textstyle\bigcup_{m\in I_n(c)}\mathcal{D}_m^{(K_m)},G_0)\\ &= \dfrac{\sum_{m\in I_n(c)}\frac{1}{K_m}\sum_{k=1}^{K_m}\sum_{t=0}^{T_{m,k}}\tilde{r}_t^{m,k}\sum_{z_0^{m,k},\cdots,z_t^{m,k}=1}^{|\mathcal{Z}|}\zeta_t^{m,k}(z_{0:t}^{m,k})\;p(\overline{\Theta}_n|a_{0:t}^{m,k},o_{1:t}^{m,k},z_{0:t}^{m,k},G_0)}{\sum_{m\in I_n(c)}\widehat{V}_{G_0}(\mathcal{D}_m^{(K_m)})},\end{aligned} \tag{60}$$

where $\widehat{V}_{G_0}(\mathcal{D}_m^{(K_m)})$ is the marginal empirical value given in (53), $\zeta_t^{m,k}(z_{0:t}^{m,k})$ is the average recomputed reward as given in (49), and

$$p(\overline{\Theta}_n|a_{0:t}^{m,k},o_{1:t}^{m,k},z_{0:t}^{m,k},G_0) \quad = \quad p(\overline{\mu}^n|\widehat{\upsilon}^{m,k,t})p(\overline{\pi}^n|\widehat{\rho}^{m,k,t})p(\overline{W}^n|\widehat{\omega}^{m,k,t})$$

is the density of a product of Dirichlet distributions and has the same form as $G_0(\Theta)$ in (45) but with υ,ρ,ω respectively replaced by $\widehat{\upsilon}^{m,k,t}$, $\widehat{\rho}^{m,k,t}$, $\widehat{\omega}^{m,k,t}$ as given by (50), (51), and (52).

It is noted that, conditional on the indicator variables c and the episodes across all environments, the distinct RPRs are independent of each other. The indicator variables cluster the M environments into $N \le M$ groups, each of which is associated with a distinct RPR. Given the clusters, the environments in the n-th group merge their episodes to form a pool, and the posterior of $\overline{\Theta}_n$ is derived based on this pool. Existing clusters may become empty and be removed, and new clusters may be introduced when novelty is detected, thus the pools change dynamically. The dynamic changes are implemented inside the algorithm presented below. Therefore, the number of distinct RPRs is not fixed but is allowed to vary.

5.4 Challenges for Gibbs Sampling

The DP posterior as given by (57) and (60) may be analyzed using the technique of Gibbs sampling (Geman and Geman, 1984; Gelfand and Smith, 1990). The Gibbs sampler successively draws the indicator variables $c_1, c_2, \cdots, c_M$ and the distinct RPRs $\overline{\Theta}_1, \overline{\Theta}_2, \cdots, \overline{\Theta}_N$ according to (57) and (60). The samples are expected to represent the posterior when the Markov chain produced by the Gibbs sampler reaches the stationary distribution. However, the convergence of Gibbs sampling can be slow and a long sequence of samples may be required before the stationary distribution is reached. The slow convergence can generally be attributed to the fact that the Gibbs sampler implements message-passing between dependent variables through the use of samples, instead of sufficient statistics (Jordan et al., 1999). Variational methods have been suggested as a replacement for Gibbs sampling (Jordan et al., 1999). Though efficient, variational methods are known to suffer from bias. A good trade-off is to combine the two, which is the idea of hybrid variational/Gibbs inference in Welling et al. (2008).

In our present case, Gibbs sampling is further challenged by the particular form of the conditional posterior of $\overline{\Theta}_n$ in (60), which is seen to be a large mixture resulting from the summation over environment m, episode k, time step t, and latent z variables. Thus it has a total of $\sum_{m \in I_n} \sum_{k=1}^{K_m} \sum_{t=0}^{T_{m,k}} |\mathcal{Z}|^t$ components and each component is uniquely associated with a single sub-episode and a specific instantiation of latent z variables. To sample from this mixture, one first makes a draw to decide a component and then draws $\overline{\Theta}_n$ from this component. Obviously, any particular draw of $\overline{\Theta}_n$ makes use of one single sub-episode only, instead of simultaneously employing all sub-episodes in the n-th cluster as one would wish.

In essence, mixing with respect to (m,k,t) effectively introduces additional latent indicator variables, that is, those for locating environment m, episode k, and time step t. It is important to note that these new indicator variables play a different role than z's in affecting the samples of $\overline{\Theta}_n$. In particular, the z's are intrinsic latent variables inside the RPR model, while the new ones are extrinsic latent variables resulting from the particular form of the empirical value function in (37). Each realization of the new indicators is uniquely associated with a distinct sub-episode while each realization of z's is uniquely associated specific decision states. Therefore, the update of $\overline{\Theta}_n$ based on one realization of the new indicators employs a single sub-episode, but the update based on one realization of z's employs all sub-episodes.

5.5 The Gibbs-Variational Algorithm for Learning the DP Posterior

The fact that the Gibbs sampler cannot update the posterior RPR samples by using more than one sub-episode motivates us to develop a hybrid Gibbs-variational algorithm for learning the posterior.

We restrict the joint posterior of the latent z variables and the RPR parameters to the variational Bayesian (VB) approximation that assumes a factorized form. This restriction yields a variational approximation to $p(\overline{\Theta}_n | \bigcup_{m \in I_n(c)} \mathcal{D}_m^{(K_m)}, G_0)$ that is a single product of Dirichlet density functions, where the terms associated with different episodes are collected and added up. Therefore, updating of the variational posterior of $\overline{\Theta}_n$ in each Gibbs-variational iteration is based on simultaneously employing all sub-episodes in the n-th cluster. In addition, the variational method yields an approximation of the marginal empirical value $\widehat{V}_{G_0}(\mathcal{D}_m^{(K_m)})$ as given in (39).

The overall Gibbs-variational algorithm is an iterative procedure based on the DP posterior represented by (57) and (58). At each iteration one successively performs the following for $m = 1, 2, \cdots, M$. First, the cluster indicator variable c_m is drawn according to (57), where $\widehat{V}_{G_0}(\mathcal{D}_m^{(K_m)})$ is replaced by its variational Bayesian approximation; accordingly the clusters $I_n = \{m : c_m = n\}, n = 1, \ldots, N$ are updated. For each nonempty cluster n, the associated distinct RPR is updated by drawing from, or finding the mode of, the variational Bayesian approximation of $p(\overline{\Theta}_n | \bigcup_{m \in I_n(c)} \mathcal{D}_m^{(K_m)}, G_0)$. The steps are iterated until the variational approximation of $\sum_{n=1}^{N} \widehat{V}_{G_0}(\bigcup_{m \in I_n(c)} \mathcal{D}_m^{(K_m)})$ converges. Note that the number of clusters is not fixed but changes with the iteration, since existing clusters may become empty and be removed and new clusters may be added in.

5.5.1 VARIATIONAL BAYESIAN APPROXIMATION OF $\widehat{V}_{G_0}(\mathcal{D}^{(K)})$ AND $p(\Theta|\mathcal{D}^{(K)}, G_0)$

In this subsection we drop the variable dependence on environment m, for notational simplicity. The discussion assumes a set of episodes $\mathcal{D}^{(K)} = \{(a_0^k r_0^k o_1^k a_1^k r_1^k \cdots o_{T_k}^k a_{T_k}^k r_{T_k}^k)\}_{k=1}^{K}$, which may come from a single environment or a conglomeration of several environments.

We now derive the variational Bayesian approximation of the marginal empirical value function $\widehat{V}_{G_0}(\mathcal{D}^{(K)})$ as defined in (39). We begin by rewriting (39), using (6) and (37), as

$$\widehat{V}_{G_0}(\mathcal{D}^{(K)}) \quad = \quad \frac{1}{K} \sum_{k=1}^{K} \sum_{t=0}^{T_k} \tilde{r}_t^k \sum_{z_0^k, \cdots, z_t^k = 1}^{|\mathcal{Z}|} \int p(a_{0:t}^k, z_{0:t}^k | o_{1:t}^k, \Theta) G_0(\Theta)\, d\Theta.$$

We follow the general variational Bayesian approach (Jordan et al., 1999; Jaakkola, 2001; Beal, 2003)[5] to find a variational lower bound to $\ln \widehat{V}_{G_0}(\mathcal{D}^{(K)})$, and the variational Bayesian approximation of $\widehat{V}_{G_0}(\mathcal{D}^{(K)})$ is obtained as the exponential of the lower bound. The lower bound is a functional of a set of factorized forms $\{q_t^k(z_{0:t}^k) g(\Theta) : z_t^k \in \mathcal{Z}, t = 1 \ldots T_k, k = 1 \ldots K\}$ that satisfies the following normalization constraints:

$$\sum_{k=1}^{K} \sum_{t=1}^{T_k} \sum_{z_0^k, \cdots, z_t^k = 1}^{|\mathcal{Z}|} q_t^k(z_{0:t}^k) = K \;\text{ and }\; q_t^k(z_{0:t}^k) \geq 0 \, \forall z_{0:t}^k, t, k,$$
$$\int g(\Theta) d\Theta = 1 \;\text{ and }\; g(\Theta) \geq 0 \, \forall \Theta.$$

The lower bound is maximized with respect to $\{q_t^k(z_{0:t}^k) g(\Theta)\}$. As will come clear below, maximization of the lower bound is equivalent to minimization of the Kullback-Leibler (KL) distance between the factorized forms and *weighted* true joint posterior of z's and Θ. In this sense, the optimal $g(\Theta)$ is a variational Bayesian approximation to the posterior $p(\Theta|\mathcal{D}^{(K)}, G_0)$. It should be noted that, as before, the weights result from the empirical value function and are not a part of standard VB (as applied to likelihood functions).

The variational lower bound is obtained by applying Jensen's inequality to $\ln \widehat{V}_{G_0}(\mathcal{D}^{(K)})$,

$$\ln \widehat{V}_{G_0}(\mathcal{D}^{(K)})$$

5. The standard VB applies to a likelihood function. Since we are using a value function instead of a likelihood function, the VB derivation here is not a standard one, just as the Bayes rule in (38) is non-standard.

$$\begin{aligned}
&= \ln \frac{1}{K}\sum_{k=1}^{K}\sum_{t=0}^{T_k}\sum_{z_0^k,\cdots,z_t^k=1}^{|\mathcal{Z}|}\int q_t^k(z_{0:t}^k)g(\Theta)\frac{\tilde{r}_t^k G_0(\Theta)p(a_{0:t}^k,z_{0:t}^k|o_{1:t}^k,\Theta)}{q_t^k(z_{0:t}^k)g(\Theta)}\,d\Theta \\
&\geq \frac{1}{K}\sum_{k=1}^{K}\sum_{t=0}^{T_k}\sum_{z_0^k,\cdots,z_t^k=1}^{|\mathcal{Z}|}\int q_t^k(z_{0:t}^k)g(\Theta)\ln\frac{\tilde{r}_t^k G_0(\Theta)p(a_{0:t}^k,z_{0:t}^k|o_{1:t}^k,\Theta)}{q_t^k(z_{0:t}^k)g(\Theta)}\,d\Theta \\
&= \ln\widehat{V}_{G_0}(\mathcal{D}^{(K)}) - \mathrm{KL}\left(\left\{q_t^k(z_{0:t}^k)g(\Theta)\right\}\Big\|\left\{\frac{\zeta_t^k}{\widehat{V}_{G_0}(\mathcal{D}^{(K)})}p(z_{0:t}^k,\Theta|a_{0:t}^k,o_{1:t}^k)\right\}\right) \\
&\overset{Def.}{=} \mathrm{LB}\left(\left\{q_t^k\right\},g(\Theta)\right), \qquad (61)
\end{aligned}$$

where ζ_t^k is the average recomputed reward as given in (42), and

$$\begin{aligned}
&\mathrm{KL}\left(\left\{q_t^k(z_{0:t}^k)g(\Theta)\right\}\Big\|\left\{\frac{\zeta_t^k}{\widehat{V}_{G_0}(\mathcal{D}^{(K)})}p(z_{0:t}^k,\Theta|a_{0:t}^k,o_{1:t}^k)\right\}\right) \\
&= \frac{1}{K}\sum_{k=1}^{K}\sum_{t=0}^{T_k}\sum_{z_0^k,\cdots,z_t^k=1}^{|\mathcal{Z}|}\int q_t^k(z_{0:t}^k)g(\Theta)\ln\frac{q_t^k(z_{0:t}^k)g(\Theta)}{\frac{\zeta_t^k}{\widehat{V}_{G_0}(\mathcal{D}^{(K)})}p(z_{0:t}^k,\Theta|a_{0:t}^k,o_{1:t}^k)}\,d\Theta,
\end{aligned}$$

with $\mathrm{KL}(q||p)$ denoting the Kullback-Leibler distance.

For any set $\left\{q_t^k(z_{0:t}^k)g(\Theta) : z_t^k \in \mathcal{Z}, t = 1\ldots T_k, k = 1\ldots K\right\}$ satisfying the above normalization constraints, the inequality in (61) holds. In order to obtain the lower bound that is closest to $\ln\widehat{V}(\mathcal{D}^{(K)})$, one maximizes the lower bound by optimizing $\left(\left\{q_t^k\right\},g(\Theta)\right)$ subject to the normalization constraints. Since $\ln\widehat{V}_{G_0}(\mathcal{D}^{(K)})$ is independent of Θ and $\{q_t^k\}$, it is clear that maximization of the lower bound $\mathrm{LB}\left(\left\{q_t^k\right\},g(\Theta)\right)$ is equivalent to minimization of the KL distance between $\left\{q_t^k(z_{0:t}^k)g(\Theta)\right\}$ and the weighted posterior $\left\{\frac{\zeta_t^k}{\widehat{V}_{G_0}(\mathcal{D}^{(K)})}p(z_{0:t}^k,\Theta|a_{0:t}^k,o_{1:t}^k)\right\}$, where the weight for episode k at time step t is $\frac{\zeta_t^k}{\widehat{V}_{G_0}(\mathcal{D}^{(K)})} = K\frac{\zeta_t^k}{\sum_{k=1}^{K}\sum_{t=0}^{T_k}\zeta_t^k}$ (the equation results directly from (43)), that is, K times the fraction that the average recomputed reward ζ_t^k occupies in the total average recomputed reward. Therefore the factorized form $\left\{q_t^k(z_{0:t}^k)g(\Theta)\right\}$ represents an approximation of the weighted posterior when the lower bound reaches the maximum, and the corresponding $g(\Theta)$ is called the approximate variational posterior of Θ.

The lower bound maximization is accomplished by solving $\left\{q_t^k(z_{0:t}^k)\right\}$ and $q(\Theta)$ alternately, keeping one fixed while solving for the other, as shown in Theorem 8.

Theorem 8 *Iteratively applying the following two equations produces a sequence of monotonically increasing lower bounds* $\mathrm{LB}\left(\left\{q_t^k\right\},g(\Theta)\right)$*, which converges to a maxima,*

$$q_t^k(z_{0:t}^k) = \frac{\tilde{r}_t^k}{C_z}\exp\left\{\int g(\Theta)\ln p(a_{0:t}^k,z_{0:t}^k|o_{1:t}^k,\Theta)\,d\Theta\right\}, \qquad (62)$$

$$g(\Theta) = \frac{G_0(\Theta)}{C_\Theta}\exp\left\{\frac{1}{K}\sum_{k=1}^{K}\sum_{t=0}^{T_k}\sum_{z_0^k,\cdots,z_t^k=1}^{|\mathcal{Z}|}q_t^k(z_{0:t}^k)\ln\tilde{r}_t^k p(a_{0:t}^k,z_{0:t}^k|o_{1:t}^k,\Theta)\right\}, \qquad (63)$$

where C_z and C_Θ are normalization constants such that $\int g(\Theta)d\Theta = 1$ and $\sum_{k=1}^{K}\sum_{t=0}^{T_k}\sum_{z_0^k,\cdots,z_t^k=1}^{|\mathcal{Z}|}q_t^k(z_{0:t}^k) = K$.

It is seen from (63) that the variational posterior $g(\Theta)$ takes the form of a product, where each term in the product is uniquely associated with a sub-episode. As will be clear shortly, the terms are properly collected and the associated sub-episodes simultaneously employed in the posterior. We now discuss the computations involved in Theorem 8.

Calculation of $\left\{q_t^k(z_{0:t}^k)\right\}$ We uses the prior of Θ as specified by (45). It is not difficult to verify from (63) that the variational posterior $g(\Theta)$ takes the same form as the prior, that is,

$$g(\Theta) \quad = \quad p(\mu|\widehat{\upsilon})p(\pi|\widehat{\rho})p(W|\widehat{\omega}), \tag{64}$$

where the three factors respectively have the forms of (46),(47), and (48); we have put a hat $\widehat{}$ above the hyper-parameters of $g(\Theta)$ to indicate the difference from those of the prior.

Substituting (5) and (64) into (62), we obtain

$$\begin{aligned}
&q_t^k(z_{0:t}^k)\\
&=\frac{\tilde{r}_t^k}{C_z}\exp\left\{\sum_{\tau=0}^{t}\left\langle\ln\pi(z_\tau^k,a_\tau^k)\right\rangle_{p(\pi|\widehat{\rho})}+\left\langle\ln\mu(z_0^k)\right\rangle_{p(\mu|\widehat{\upsilon})}+\sum_{\tau=1}^{t}\left\langle\ln W(z_{\tau-1}^k,a_{\tau-1}^k,o_\tau^k,z_\tau^k)\right\rangle_{p(W|\widehat{\omega})}\right\}\\
&=\frac{\tilde{r}_t^k}{C_z}\widetilde{\mu}(z_0^k)\widetilde{\pi}(z_0^k,a_0^k)\prod_{\tau=1}^{t}\widetilde{W}(z_{\tau-1}^k,a_{\tau-1}^k,o_\tau^k,z_\tau^k)\widetilde{\pi}(z_\tau^k,a_\tau^k),
\end{aligned}\tag{65}$$

where $\langle\cdot\rangle_{p(\pi|\widehat{\rho})}$ denotes taking expectation with respect to $p(\pi|\widehat{\rho})$, and

$$\begin{aligned}
\widetilde{\mu}(j) &= \exp\left\{\left\langle\ln\mu(j)\right\rangle_{p(\mu|\widehat{\upsilon})}\right\}\\
&= \exp\left\{\psi(\widehat{\upsilon}_j)-\psi(\sum_{j'=1}^{|\mathcal{Z}|}\widehat{\upsilon}_{j'})\right\}, \quad j=1\ldots|\mathcal{Z}|, &(66)\\
\widetilde{\pi}(i,m) &= \exp\left\{\left\langle\ln\pi(i,m)\right\rangle_{p(\pi|\widehat{\rho})}\right\}\\
&= \exp\left\{\psi(\widehat{\rho}_{i,m})-\psi(\sum_{m'=1}^{|\mathcal{A}|}\widehat{\rho}_{i,m'})\right\}, \quad m=1\ldots|\mathcal{A}|, &(67)\\
\widetilde{W}(i,a,o,j) &= \exp\left\{\left\langle\ln W(i,a,o,j)\right\rangle_{p(W|\widehat{\omega})}\right\}\\
&= \exp\left\{\psi(\widehat{\omega}_{i,a,o,j})-\psi(\sum_{j'=1}^{|\mathcal{Z}|}\widehat{\omega}_{i,a,o,j'})\right\}, \quad j=1\ldots|\mathcal{Z}|, &(68)
\end{aligned}$$

each of which is a finite set of nonnegative numbers with a sum less than one. Such a finite set is called under-normalized probabilities in Beal (2003) and used there to perform variational Bayesian learning of hidden Markov models (HMM). The $\psi(\cdot)$ is the digamma function.

It is interesting to note that the product $\widetilde{\mu}(z_0^k)\widetilde{\pi}(z_0^k,a_0^k)\prod_{\tau=1}^{t}\widetilde{W}(z_{\tau-1}^k,a_{\tau-1}^k,o_\tau^k,z_\tau^k)\widetilde{\pi}(z_\tau^k,a_\tau^k)$ on the left side of (65) has exactly the same form as the expression of $p(a_{0:t}^k,z_{0:t}^k|o_{1:t}^k,\Theta)$ in (5), except that the Θ is replaced by $\widetilde{\Theta}=\{\widetilde{\mu},\widetilde{\pi},\widetilde{W}\}$. Therefore, one can nominally rewrite (65) as

$$q_t^k(z_{0:t}^k) \quad = \quad \frac{\tilde{r}_t^k}{C_z}p(a_{0:t}^k,z_{0:t}^k|o_{1:t}^k,\widetilde{\Theta}),$$

with the normalization constant given by

$$C_z = \frac{1}{K}\sum_{k=1}^{K}\sum_{t=0}^{T_k}\sum_{z_0^k,\cdots,z_t^k=1}^{|\mathcal{Z}|} \tilde{r}_t^k p(a_{0:t}^k, z_{0:t}^k | o_{1:t}^k, \widetilde{\Theta}),$$

such that the constraint $\sum_{k=1}^{K}\sum_{t=0}^{T_k}\sum_{z_0^k,\cdots,z_t^k=1}^{|\mathcal{Z}|} q_t^k(z_{0:t}^k) = K$ is satisfied. One may also find that the normalization constant C_z is a nominal empirical value function that has the same form as the empirical value function in (12). The only difference is that the normalized Θ is replaced by the under-normalized $\widetilde{\Theta}$. Therefore, one may write

$$C_z = \widehat{V}(\mathcal{D}^{(K)};\widetilde{\Theta}). \tag{69}$$

Since $\widetilde{\Theta} = \{\widetilde{\mu}, \widetilde{\pi}, \widetilde{W}\}$ are under-normalized, $p(a_{0:t}^k, z_{0:t}^k | o_{1:t}^k, \widetilde{\Theta})$ is not a proper probability distribution. However, one may still write $p(a_{0:t}^k, z_{0:t}^k | o_{1:t}^k, \widetilde{\Theta}) = p(a_{0:t}^k | o_{1:t}^k, \widetilde{\Theta}) p(z_{0:t}^k | a_{0:t}^k, o_{1:t}^k, \widetilde{\Theta})$, where $p(a_{0:t}^k | o_{1:t}^k, \widetilde{\Theta}) = \sum_{z_0^k,\cdots,z_t^k=1}^{|\mathcal{Z}|} p(a_{0:t}^k, z_{0:t}^k | o_{1:t}^k, \widetilde{\Theta})$ and $p(z_{0:t}^k | a_{0:t}^k, o_{1:t}^k, \widetilde{\Theta}) = \frac{p(a_{0:t}^k, z_{0:t}^k | o_{1:t}^k, \widetilde{\Theta})}{p(a_{0:t}^k | o_{1:t}^k, \widetilde{\Theta})}$. Note that $p(z_{0:t}^k | a_{0:t}^k, o_{1:t}^k, \widetilde{\Theta})$ is a proper probability distribution. Accordingly, $q_t^k(z_{0:t}^k)$ can be rewritten as

$$q_t^k(z_{0:t}^k) = \frac{\sigma_t^k(\widetilde{\Theta})}{\widehat{V}(\mathcal{D}^{(K)};\widetilde{\Theta})} p(z_{0:t}^k | a_{0:t}^k, o_{1:t}^k, \widetilde{\Theta}), \tag{70}$$

where

$$\begin{aligned}\sigma_t^k(\widetilde{\Theta}) &= \tilde{r}_t^k p(a_{0:t}^k | o_{1:t}^k, \widetilde{\Theta}) \\ &= \tilde{r}_t^k \prod_{\tau=0}^{t} p(a_\tau^k | h_\tau^k, \widetilde{\Theta})\end{aligned} \tag{71}$$

is called variational re-computed reward, which has the same form as the re-computed reward given in (16) but with Θ replaced by $\widetilde{\Theta}$. The second equality in (71) is based on the equation $p(a_{0:t}^k | o_{1:t}^k, \Theta) = \prod_{\tau=0}^{t} p(a_\tau^k | h_\tau^k, \Theta)$ established in (9) and (10). The nominal empirical value function $\widehat{V}(\mathcal{D}^{(K)};\widetilde{\Theta})$ can now be expressed in terms of $\sigma_t^k(\widetilde{\Theta})$,

$$\widehat{V}(\mathcal{D}^{(K)};\widetilde{\Theta}) = \frac{1}{K}\sum_{k=1}^{K}\sum_{t=0}^{T_k} \sigma_t^k(\widetilde{\Theta}). \tag{72}$$

Equation (70) shows that $q_t^k(z_{0:t}^k)$ is a weighted posterior of $z_{0:t}^k$. The weights, using (72), can be equivalently expressed as $\frac{\sigma_t^k(\widetilde{\Theta})}{\widehat{V}(\mathcal{D}^{(K)};\widetilde{\Theta})} = K\eta_t^k(\widetilde{\Theta})$ where

$$\eta_t^k(\widetilde{\Theta}) \overset{Def.}{=} \frac{\sigma_t^k(\widetilde{\Theta})}{\sum_{k=1}^{K}\sum_{t=0}^{T_k} \sigma_t^k(\widetilde{\Theta})}. \tag{73}$$

The weighted posterior has the same form as (15) used in single-task RPR learning. Therefore we can borrow the techniques developed there to compute the marginal distributions of $p(z_{0:t}^k | a_{0:t}^k, o_{1:t}^k, \widetilde{\Theta})$, particularly those defined in (24) and (25). For clarity, we rewrite these marginal distributions below without re-deriving them, with Θ replaced by $\widetilde{\Theta}$,

$$\xi_{t,\tau}^k(i,j) = p(z_\tau^k = i, z_{\tau+1}^k = j | a_{0:t}^k, o_{1:t}^k, \widetilde{\Theta}), \tag{74}$$

$$\phi_{t,\tau}^k(i) = p(z_\tau^k = i | a_{0:t}^k, o_{1:t}^k, \widetilde{\Theta}). \tag{75}$$

These marginals along with the $\{\eta_t^k(\widetilde{\Theta})\}$ defined in (73) will be used below to compute the variational posterior $g(\Theta)$.

Calculation of the Variational Posterior $g(\Theta)$ To compute $g(\Theta)$, one substitutes (5) and (70) into (63) and performs summation over the latent z variables. Most z variables are summed out, leaving only the marginals in (74) and (75). Employing these marginals and taking into account the weights $\{K\eta_t^k(\widetilde{\Theta})\}$, the variational posterior (with $\eta_t^k(\widetilde{\Theta})$ abbreviated as η_t^k for notational simplicity) is obtained as

$$\begin{aligned} g(\Theta) &= \frac{G_0(\Theta)}{C_\Theta}\exp\left\{\frac{1}{K}\sum_{k=1}^{K}\sum_{t=0}^{T_k}K\eta_t^k\left[\sum_{i=1}^{|\mathcal{Z}|}\phi_{t,0}^k(i)\ln\mu(i)\right.\right.\\ &\quad\left.\left.+\sum_{\tau=1}^{t}\sum_{i=1}^{|\mathcal{Z}|}\phi_{t,\tau}^k(i)\ln\pi(i,a_\tau^k)+\sum_{\tau=1}^{t}\sum_{i,j=1}^{|\mathcal{Z}|}\xi_{t,\tau}^k(i,j)\ln W(i,a_{\tau-1}^k,o_\tau^k,j)\right]\right\}\\ &= \frac{G_0(\Theta)}{C_\Theta}\prod_{k=1}^{K}\prod_{t=0}^{T_k}\left\{\prod_{i=1}^{|\mathcal{Z}|}\left[\mu(i)\right]^{\eta_t^k\phi_{t,0}^k(i)}\prod_{\tau=1}^{t}\prod_{i=1}^{|\mathcal{Z}|}\left[\pi(i,a_\tau^k)\right]^{\eta_t^k\phi_{t,\tau}^k(i)}\right.\\ &\quad\left.\times\prod_{\tau=1}^{t}\prod_{i,j=1}^{|\mathcal{Z}|}\left[W(i,a_{\tau-1}^k,o_\tau^k,j)\right]^{\eta_t^k\xi_{t,\tau-1}^k(i,j)}\right\}\\ &= p(\mu|\widehat{\upsilon})p(\pi|\widehat{\rho})p(W|\widehat{\omega}), \end{aligned}$$

where $p(\mu|\widehat{\upsilon})$, $p(\pi|\widehat{\rho})$, $p(W|\widehat{\omega})$ have the same forms as in (46), (47), and (48), respectively, but with the hyper-parameters replaced by

$$\widehat{\upsilon}_i = \upsilon_i+\sum_{k=1}^{K}\sum_{t=0}^{T_k}\eta_t^k\phi_{t,0}^k(i), \tag{76}$$

$$\widehat{\rho}_{i,a} = \rho_{i,a}+\sum_{k=1}^{K}\sum_{t=0}^{T_k}\sum_{\tau=0}^{t}\eta_t^k\phi_{t,\tau}^k(i)\delta(a_\tau^k,a), \tag{77}$$

$$\widehat{\omega}_{i,a,o,j} = \omega_{i,a,o,j}+\sum_{k=1}^{K}\sum_{t=0}^{T_k}\sum_{\tau=1}^{t}\eta_t^k\xi_{t,\tau-1}^k(i,j)\delta(a_{\tau-1}^k,a)\delta(o_\tau^k,o), \tag{78}$$

for $i,j=1,\ldots,|\mathcal{Z}|$, $a=1,\ldots,|\mathcal{A}|$, $o=1,\ldots,|\mathcal{O}|$. Note that, for simplicity, we have used $\{\widehat{\upsilon},\widehat{\rho},\widehat{\omega}\}$ to denote the hyper-parameters of $g(\Theta)$ for both before and after the updates in (76)-(78) are made. It should be kept in mind that the η's, ϕ's, and ξ's are all based on the numerical values of $\{\widehat{\upsilon},\widehat{\rho},\widehat{\omega}\}$ before the updates in (76)-(78) are made, that is, they are based on the $\{\widehat{\upsilon},\widehat{\rho},\widehat{\omega}\}$ updated in the previous iteration.

It is clear from (76)-(78) that the update of the variational posterior is based on using all episodes at all time steps (i.e., all sub-episodes). The η_t^k can be thought of as a variational soft count at time t of episode k, which is appended to the hyper-parameters (initial Dirichlet counts) of the prior. Each decision state z receives η_t^k in the amount that is proportional to the probability specified by the posterior marginals $\{\phi_{t,\tau}^k\}$ and $\{\xi_{t,\tau-1}^k\}$.

Computation of the Lower Bound To compute the lower bound $\text{LB}(\{q_t^k\},g(\Theta))$ given in (61), one first takes the logarithm of (62) to obtain

$$\begin{aligned} \ln q_t^k(z_{0:t}^k) &= \ln C_z^{-1}\tilde{r}_t^k\exp\left\{\int g(\Theta)\ln p(a_{0:t}^k,z_{0:t}^k|o_{1:t}^k,\Theta)\,d\Theta\right\}\\ &= -\ln C_z+\int g(\Theta)\ln\tilde{r}_t^k p(a_{0:t}^k,z_{0:t}^k|o_{1:t}^k,\Theta)\,d\Theta, \end{aligned}$$

which is then substituted into the right side of (82) in the Appendix to cancel the first term, yielding

$$\begin{aligned}\mathrm{LB}\left(\left\{q_t^k\right\},g(\Theta)\right) &= \ln C_z - \int g(\Theta)\ln\frac{g(\Theta)}{G_0(\Theta)}\,d\Theta \\ &= \ln C_z - \mathrm{KL}\Big(g(\Theta)\|G_0(\Theta)\Big) \\ &= \ln\widehat{V}(\mathcal{D}^{(K)};\widetilde{\Theta}) - \mathrm{KL}\Big(g(\Theta)\|G_0(\Theta)\Big), \end{aligned} \tag{79}$$

where the last equality follows from (69).

The lower bound yields a variational approximation to the logarithm of the marginal empirical value. As variational Bayesian learning proceeds, the lower bound monotonically increases, as guaranteed by Theorem 8, and eventually reaches a maxima, at which point one obtains the best (assuming the maxima is global) variational approximation. By taking exponential of the best lower bound, one gets the approximated marginal empirical value. The lower bound also provides a quantitative measure for monitoring the convergence of variational Bayesian learning.

5.5.2 THE COMPLETE GIBBS-VARIATIONAL LEARNING ALGORITHM

A detailed algorithmic description of the complete Gibbs-variational algorithm is given in Table 2. The algorithm calls the variational Bayesian (VB) algorithm in Table 3 as a sub-routine, to find the variational Bayesian approximations to intractable computations. In particular, the marginal empirical value $\widehat{V}_{G_0}(\mathcal{D}_m^{(K_m)})$ in (57) is approximated by the exponential of the variational lower bound returned from the VB algorithm by feeding the episodes $\mathcal{D}_m^{(K_m)}$. The conditional posterior $p(\overline{\Theta}_n|\bigcup_{m\in I_n(c)}\mathcal{D}_m^{(K_m)},G_0)$ in (58) is approximated by the variational posterior $g(\overline{\Theta}_n)$ returned from the VB algorithm by feeding the episodes $\bigcup_{m\in I_n(c)}\mathcal{D}_m^{(K_m)}$. The variational approximation of $\widehat{V}_{G_0}(\mathcal{D}_m^{(K_m)})$ need be computed only once for each environment m, before the main loop begins, since it solely depends on the DP base G_0 and the episodes, which are assumed given and fixed. The variational posterior $g(\overline{\Theta}_n)$ and $\widehat{V}_{G_0}\left(\bigcup_{m\in I_n(c)}\mathcal{D}_m^{(K_m)}\right)$, however, need be updated inside the main loop, because the clusters $\{I_n(c)\}$ keep changing from iteration to iteration.

Upon convergence of the algorithm, one obtains variational approximations to the posteriors of distinct RPRs $\{g(\overline{\Theta}_n)\}_{n=1}^N$, which along with the cluster indicators $\{c_1,c_2,\cdots,c_M\}$ give the variational posterior $g(\Theta_m)$ for each individual environment m. By simple post-processing of the posterior, we obtain the mean or mode of each Θ_m, which gives a single RPR for each environment and yields the history-dependent policy as given by (8). Alternatively, one may draw samples from the variational posterior and use them to produce an ensemble of RPRs for each environment. The RPR ensemble gives multiple history-dependent policies, that are marginalized (averaged) to yield the final choice for the action.

It should be noted that the VB algorithm in Table 3 can also be used as a stand-alone algorithm to find the variational posterior of the RPR of each environment independently of the RPRs of other environments. In this respect the VB is a Bayesian counterpart of the maximum value (MV) algorithm for single-task reinforcement learning (STRL), presented in Section 4 and Table 1.

Time Complexity Analysis The time complexity of the VB algorithm in Table 3 is given as follows where, as in Section 4.3.1, the complexity is quantified by the number of real number multiplications in each iteration and is presented in the Big-O notation. For the reasons stated in Section 4.3.1, the complexity per iteration also represents the complexity of the complete algorithm.

```
Input: ${\{\mathcal{D}_m^{(K_m)}\}_{m=1}^M}$, $\mathcal{A}$, $\mathcal{O}$, $|\mathcal{Z}|$, $\{\upsilon,\rho,\omega\}$, $\alpha$
Output: $\{\widehat{\upsilon}_n,\widehat{\rho}_n,\widehat{\omega}_n\}_{n=1}^N$ with $N \le M$ and $\{c_1,c_2,\cdots,c_M\}$.

1. Computing the variational approximations of $\{\widehat{V}_{G_0}(\mathcal{D}_m^{(K_m)})\}$:
  1.1 for $m=1$ to $M$
        Call the algorithm in Table 3, with the inputs $\mathcal{D}_m^{(K_m)}$, $\mathcal{A}$, $\mathcal{O}$, $|\mathcal{Z}|$,
        $\{\upsilon,\rho,\omega\}$. Record the returned hyper-parameters as $\{\widehat{\upsilon}_m,\widehat{\rho}_m,\widehat{\omega}_m\}$ and the
        approximate $\widehat{V}_{G_0}(\mathcal{D}_m^{(K_m)})$.
2. Initializations: Let $j=1$, N=M, $\ell=0$.
        Let $\overline{\upsilon}_n=\widehat{\upsilon}_n$, $\overline{\rho}_n=\widehat{\rho}_n$, $\overline{\omega}_n=\widehat{\omega}_n$, for $n=1,\cdots,N$.
3. Repeat
  3.1 For $n=1$ to $N$
        Update $\overline{\Theta}_n$ by drawing from, or finding the mode of, the $G_0$ with hyper-
        parameters $\{\overline{\upsilon}_n,\overline{\rho}_n,\overline{\omega}_n\}$.
  3.2 For $m=1$ to $M$
        Let $c_m^{\text{old}}=c_m$ and draw $c_m$ according to (57).
        If $c_m \neq c_m^{\text{old}}$
          If $c_m=0$, start a new cluster $I_{N+1}(c)=\{m\}$.
          Elseif $c_m \neq 0$, move the element $m$ from $I_{c_m^{\text{old}}}(c)$ to $I_{c_m}(c)$.
          For $n=\{c_m^{\text{old}},c_m\}$
            If $I_n(c)$ is an empty set
              Delete the $n$-th cluster.
            Elseif $I_n(c)$ contain a single element (let it be denoted by $m'$)
              Let $\overline{\upsilon}_n=\widehat{\upsilon}_{m'}$, $\overline{\rho}_n=\widehat{\rho}_{m'}$, $\overline{\omega}_n=\widehat{\omega}_{m'}$. Add $\frac{K_{m'}}{\sum_{i=1}^M K_i}\widehat{V}_{G_0}(\mathcal{D}_{m'}^{(K_{m'})})$ to $\ell(j)$.
            Else
              Call the algorithm in Table 3, with the inputs $\bigcup_{i\in I_n(c)}\mathcal{D}_i^{(K_i)}$,
              $\mathcal{A}$, $\mathcal{O}$, $|\mathcal{Z}|$, $\{\upsilon,\rho,\omega\}$. Record the returned hyper-parameters as
              $\{\overline{\upsilon}_n,\overline{\rho}_n,\overline{\omega}_n\}$. Scale the returned $\widehat{V}_{G_0}\big(\bigcup_{i\in I_n(c)}\mathcal{D}_i^{(K_i)}\big)$
              by $\frac{\sum_{i\in I_n(c)}K_i}{\sum_{i=1}^M K_i}$ and add the result to $\ell(j)$.
            If $I_n(c)$ is not empty
              Draw $\overline{\Theta}_n$ drawn from $G_0$ with hyper-parameters $\{\overline{\upsilon}_n,\overline{\rho}_n,\overline{\omega}_n\}$.
  3.3 Updating $N$:
        Let $N$ be the number of nonempty clusters and renumber the
        nonempty clusters so that their indices are in $\{1,2,\cdots,N\}$.
  3.4 Convergence check:
        If the sequence of $\ell$ converges
          stop the algorithm and exit.
        Otherwise
          Set $j:=j+1$ and $\ell(j)=0$.
```

Table 2: The Gibbs-variational algorithm for learning the DP posterior

```
Input: $\mathcal{D}^{(K)}$, $\mathcal{A}$, $\mathcal{O}$, $|\mathcal{Z}|$, $\{\upsilon, \rho, \omega\}$.
Output: $\{\widehat{\upsilon}, \widehat{\rho}, \widehat{\omega}\}$, $\widehat{V}_{G_0}(\mathcal{D}^{(K)}) \approx \mathrm{LB}(\{q_t^k\}, g(\Theta))$.

1. Initialize $\widehat{\upsilon} = \upsilon$, $\widehat{\rho} = \rho$, $\widehat{\omega} = \omega$, $\ell = []$, iteration $= 1$.
2. Repeat
  2.1 Computing $\widetilde{\Theta}$:
      Compute the set of under-normalized probabilities $\widetilde{\Theta}$ using
      Equations (66)-(68).
  2.2 Dynamical programming:
      Compute $\alpha$ and $\beta$ variables with (30), (31), and (32), with $\Theta$
      replaced by $\widetilde{\Theta}$ in these equations.
  2.3 Reward re-computation:
      Calculate the variational recomputed reward $\{\sigma_t^k(\widetilde{\Theta})\}$ using
      (71) and (32) and compute the weight $\{\eta_t^k(\widetilde{\Theta})\}$ using (73).
  2.4 Lower bound computation:
      Calculate the variational lower bound $\mathrm{LB}(\{q_t^k\}, g(\Theta))$ using
      (79) and (72).
  2.5 Convergence check:
      Let $\ell(\text{iteration}) = \mathrm{LB}(\{q_t^k\}, g(\Theta))$.
      If the sequence of $\ell$ converges
          Stop the algorithm and exit.
      Else
          Set iteration := iteration + 1.
  2.6 Posterior update for $z$:
      Compute the $\xi$ and $\phi$ variables using Equations (28)-(29).
  2.7 Update of hyper-paramters:
      Compute the updated $\{\widehat{\upsilon}, \widehat{\rho}, \widehat{\omega}\}$ using (76), (77), and (78).
```

Table 3: The variational Bayesian learning algorithm for RPR

- The computation of $\widetilde{\Theta}$ based on Equations (66)-(68) runs in time $O(|\mathcal{Z}|)$, $O(|\mathcal{A}||\mathcal{Z}|)$, and $O(|\mathcal{A}||\mathcal{O}||\mathcal{Z}|^2)$, respectively.
- Computation of the α variables with (30) and (32) (with Θ replaced by $\widetilde{\Theta}$) runs in time $O(|\mathcal{Z}|^2 \sum_{k=1}^K T_k)$.
- Computation of the β variables with (31) and (32) (with Θ replaced by $\widetilde{\Theta}$) runs in time $O(|\mathcal{Z}|^2 \sum_{k=1}^K \sum_{t=0, r_t^k \neq 0}^{T_k} (t+1))$, which is $O(|\mathcal{Z}|^2 \sum_{k=1}^K T_k^2)$ in the worst case and is $O(|\mathcal{Z}|^2 \sum_{k=1}^K T_k)$ in the best case, where the worst and best cases are distinguished by the sparseness of immediate rewards, as discussed in Section 4.3.1.
- The reward re-computation using (71), (32), and (73) requires time $O(\sum_{k=1}^K T_k)$ in the worst case and $O(K)$ in the best case.
- Computation of the lower bound using (72) and (79) requires time $O(|\mathcal{A}||\mathcal{O}||\mathcal{Z}|^2)$.

- Update of the hyper-parameters using (76), (77), and (78), as well as computation of the ξ and ϕ variables using (28) and (29), runs in time $O(|\mathcal{Z}|^2 \sum_{k=1}^{K} T_k^2)$ in the worst case and $O(|\mathcal{Z}|^2 \sum_{k=1}^{K} T_k)$ in the best case.

The overall complexity of the VB algorithm is seen to be $O(|\mathcal{Z}|^2 \sum_{k=1}^{K} T_k^2)$ in the worst case and $O(|\mathcal{Z}|^2 \sum_{k=1}^{K} T_k)$ in the best case, based on the fact that $\sum_{k=1}^{K} T_k \gg |\mathcal{A}||\mathcal{O}|$ in general. Thus the VB algorithm has the same time complexity as the value maximization algorithm in Table 1. Note that the time dependency on the lengths of episodes is dictated by the sparseness of the immediate rewards; for most problems considered in Section 6, the agent receives rewards only when the goal state is reached, in which case the VB algorithm scales linearly with the lengths of episodes.

The complexity of the Gibbs-variational algorithm can be easily obtained based on the complexity analysis above for the VB algorithm. At the beginning and before entering the main loop, the Gibbs-variational algorithm calls the VB to compute the variational approximation of the marginal empirical value $\{\widehat{V}_{G_0}(\mathcal{D}_m^{(K_m)})\}$ for each environment m, by feeding the associated episodes $\mathcal{D}_m^{(K_m)}$. These computations are performed only once. For each environment the VB runs until convergence, with a time complexity between $O(|\mathcal{Z}|^2 \sum_{k=1}^{K_m} T_{m,k})$ and $O(|\mathcal{Z}|^2 \sum_{k=1}^{K_m} T_{m,k}^2)$ per iteration, depending on the sparseness of the immediate rewards. Inside the main loop, the Gibbs-variational algorithm calls the VB to compute the variational posterior of distinct RPR for each cluster n, by feeding the merged episodes $\bigcup_{m \in I_n(c)} \mathcal{D}_m^{(K_m)}$. These computations are performed each time the clusters are updated, with a time complexity between $O(|\mathcal{Z}|^2 \sum_{m\in I_n(c)} \sum_{k=1}^{K_m} T_{m,k})$ and $O(|\mathcal{Z}|^2 \sum_{m\in I_n(c)} \sum_{k=1}^{K_m} T_{m,k}^2)$ per iteration for cluster n.

6. Experimental Results

We compare the performance of RPR in multi-task reinforcement learning (MTRL) versus single-task reinforcement learning (STRL), and demonstrate the advantage of MTRL. The RPR for MTRL is implemented by the Gibbs-variational algorithm in Table 2 and the RPR for STRL is implemented by the maximum-value (MV) algorithm in Table 1. The variational Bayesian (VB) algorithm in Table 3, which is a Bayesian counterpart of the MV algorithm, generally performs similar to the MV for STRL and is thus excluded in the comparisons.

Since the MV algorithm is a new technique developed in this paper, we evaluate the performance of the MV before proceeding to the comparison of MTRL and STRL. We also compare the MV to the method of first learning a POMDP model from the episodes and then finding the optimal policy for the POMDP.

6.1 Performance of RPR in Single-Task Reinforcement Learning (STRL)

We consider the benchmark example Hallway2, introduced in Littman et al. (1995). The Hallway2 problem was originally designed to test algorithms based on a given POMDP model, and it has recently been employed as a benchmark for testing model-free reinforcement algorithms (Bakker, 2004; Wierstra and Wiering, 2004).

Hallway2 is a navigation problem where an agent is situated in a maze consisting of a number of rooms and walls that are connected to each other and the agent navigates in the maze with the objective of reaching the goal within a minimum number of steps. The maze is characterized by 92 states, each representing one of four orientations (south, north, east, west) in any of 23 rectangle areas, and four of the states (corresponding to a single rectangle area) represent the goal. The observations consist of $2^4 = 16$ combinations of presence/absence of a wall as viewed when standing in a rectangle facing one of the four orientations, and there is an observation uniquely associated with the goal. There are five different actions that the agent can take: $\{$*stay in place*, *move forward*, *turn right*, *turn left*, *turn around*$\}$. Both state transitions and observations are very noisy (uncertain), except that the goal is fully identified by the unique observation associated with it. The reward function is defined in such a way that a large reward is received when the agent enters the goal from the adjacent states, and zero reward is received otherwise. Thus the reward structure is highly sparse and both the MTRL and STRL algorithms scale linearly with the lengths of episodes in this case, as discussed in Sections 4.3.1 and 5.5.2.

6.1.1 PERFORMANCE AS A FUNCTION OF NUMBER OF EPISODES

We investigate the performance of the RPR, as a function of K the number of episodes used in the learning. For each given K, we learn a RPR from a set of K episodes $\mathcal{D}^{(K)}$ that are generated by following the behavior policy Π, and the learning follows the procedures described in Section 4.

The conditions for the policy Π, as given in Theorem 5, are very mild, and are satisfied by a uniformly random policy. However, a uniformly random agent may take a long time to reach the goal, which makes the learning very slow. To accelerate learning, we use a semi-random policy Π, which is simulated by the rule that, with probability p_{query}, Π chooses an action suggested by the PBVI algorithm (Pineau et al., 2003) and, with probability $1 - p_{\text{query}}$, Π chooses an action uniformly sampled from $\mathcal{A}$. The use of PBVI here is similar to the meta-queries used in Doshi et al. (2008), where a meta-query consults a domain expert (who is assumed to have access to the true POMDP model) for the optimal action at a particular time step. The meta-queries correspond to human-robot interactions in robotics applications. It should be noted that, by implementing the reward re-computation in RPR online, the behavior policy in each iteration simply becomes the RPR in the previous iteration, in which case the use of an external policy like PBVI is eliminated.

In principle, the number of decision states (belief regions) $|\mathcal{Z}|$ can be selected by maximizing the marginal empirical value $\widehat{V}_{G_0}(\mathcal{D}^{(K)}) = \int \widehat{V}(\mathcal{D}^{(K)}) G_0(\Theta) d\Theta$ with respect to $|\mathcal{Z}|$, where an approximation of $\widehat{V}(\mathcal{D}^{(K)})$ can be found by the VB algorithm in Table 3. Because the MV does not employ a prior, we make a nominal prior $G_0(\Theta)$ by letting it take the form of (45) but with all hyper-parameters uniformly set to one. This leads to $G_0(\Theta) \equiv C_{\text{mv}}$, where C_{mv} is a normalization constant. Therefore maximization of $\widehat{V}_{G_0}(\mathcal{D}^{(K)})$ is equivalent to maximization of $\int \widehat{V}(\mathcal{D}^{(K)}; \Theta) d\Theta$, which serves as an evidence of how good the choice of $|\mathcal{Z}|$ fits to the episodes in terms of empirical value. According to the Occam Razor principle (Beal, 2003), the minimum $|\mathcal{Z}|$ fitting the episodes has the best generalization. In practice, letting $|\mathcal{Z}|$ be a multiple of the number of actions is usually a good choice (here $|\mathcal{Z}| = 4 \times 5 = 20$) and we find that the performance of the RPR is quite robust to the choice of $|\mathcal{Z}|$. This may be attributed to the fact that learning of the RPR is a process of allocating counts to the decision states—when more decision states are included, they simply share the counts that otherwise would have been allocated to a single decision state. Provided the sharing of counts is consistent among μ, π, and W, the policy will not change much.

The performance of the RPR is compared against EM-PBVI, the method that first learns a predictive model as in Chrisman (1992) and then learns the policy based on the predictive model. Here the predictive model is a POMDP learned by expectation maximization (EM) based on $\mathcal{D}^{(K)}$ and the PBVI (Pineau et al., 2003) is employed to find the policy given the POMDP. To examine the effect of the behavior policy Π on the RPR's performance, we consider three different Π's, which respectively have a probability $p_{\text{query}} = 5\%, 30\%, 50\%$ of choosing the actions suggested by PBVI, where p_{query} corresponds to the rate of meta-query in Doshi et al. (2008). The episodes used to learn EM-PBVI are collected by the behavior policy with $p_{\text{query}} = 50\%$, which is the highest query rate considered here. Therefore the experiments are biased favorably towards the EM-PBVI, in terms of the number of expert-suggested actions that are employed to generate the training episodes.

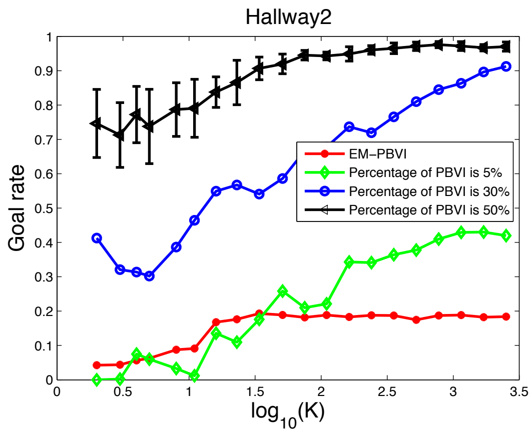
The performance of each RPR, as well as that of EM-PBVI, is evaluated on Hallway2 by following the standard testing procedure as set up in Littman et al. (1995). For each policy, a total of 251 independent trials are performed and each trial is terminated when either the goal is reached or a maximum budget of 251 steps is consumed. Three performance measures are computed based on the 251 trials: (a) the discounted accumulative reward (i.e., the sum of exponentially discounted rewards received over the $N_{\text{te}} \leq 251$ steps) averaged over the 251 trials; (b) the goal rate, that is, the percentage of the 251 trials in which the agent has reached the goal; (c) the number of steps that the agent has actually taken, averaged over the 251 trials.

The results on Hallway2 are summarized in Figure 1, where we present each of the three performance measures plus the learning time, as a function of $\log_{10}$ of the number of episodes K used in the learning. The four curves in each figure correspond to the EM-PBVI, and the three RPRs with a rate of PBVI query 5%, 30%, and 50%, respectively. Each curve results from an average over 20 independently generated $\mathcal{D}^{(K)}$ and the error bars show the standard deviations. For simplicity, the error bars are shown only for the RPR with a 50% query rate.

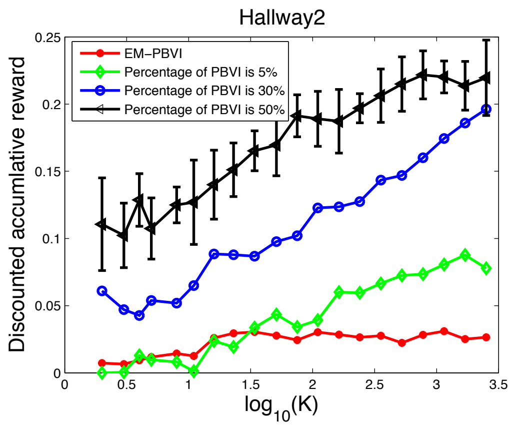
As shown in Figure 1 the performance of the RPR improves as the number of episodes K used to learn it increases, regardless of the behavior policy Π. As recalled from Theorem 5, the empirical value function $\widehat{V}(\mathcal{D}^{(K)};\Theta)$ approaches the exact value function as K goes to infinity. Assuming the RPR has enough memory (decision states) and the algorithm finds the global maxima, the RPR will approach the optimal policy as K increases. Therefore, Figure 1 serves as an experimental verification of Theorem 5. The CPU time shown in Figure 1(d) is exponential in $\log_{10} K$ or, equivalently, is linear in K. The linear time is consistent with the complexity analysis in Section 4.3.1.

The error bars of goal rate exhibits quick shrinkage with K and those of the median number of steps also shrinks relatively fast. In contrast, the discounted accumulative reward has a very slow shrinkage rate for its error bars. The different shrinkage rates show that it is much easier to reach the goal within the prescribed number of steps (251 here) than to reach the goal in relatively less steps. Note that, when the goal is reached at the t-th step, the number of steps is t but the discounted accumulative reward is $\gamma^{-t} r_{\text{goal}}$, where r_{goal} is the reward of entering the goal state. The exponential discounting explains the difference between the number of steps and the discounted accumulative reward regarding the shrinkage rates of error bars.

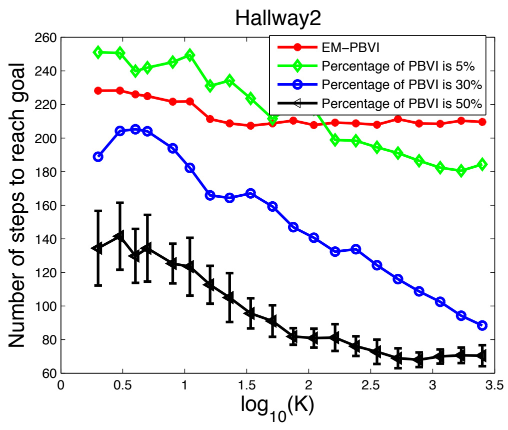
A comparison of the three RPR curves in Figure 1 shows that the rate at which the behavior policy Π uses or queries PBVI influences the RPR's performance and the influence depends on K. When K is small, increasing the query rate significantly improves the performance; whereas, when K gets larger, the influence decreases until it eventually vanishes. The decreased influence is most easily seen between the curves of 30% and 50% query rates. To make the performance not degrade when the query rate decreases to as low as 5%, a much larger K may be required. These experimental results confirm that random actions can accomplish a good exploration of available rewards (the goal state here) by collecting a large number of (lengthy) episodes and the RPR learned from these episodes perform competitively. With a small number of episodes, however, random actions achieve limited exploration and the resulting RPR performs poorly. In the latter case, queries to experts like PBVI plays an important role in improving the exploration and the RPR's performance.



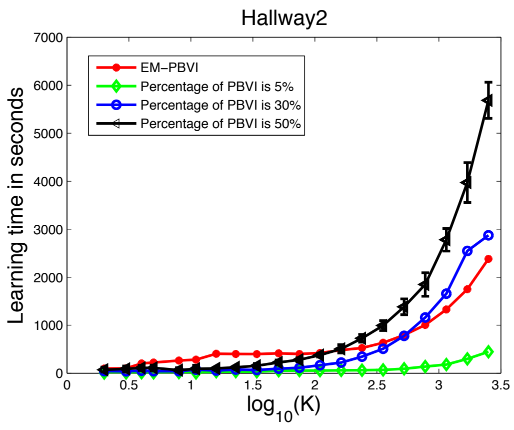
(a) Goal Rate



(b) Reward



(c) Number of steps



(d) Learning time

Figure 1: Performance comparison on the Hallway2 problem. The horizontal axis is $\log_{10}$ of the number of episodes K used in learning the RPR. The horizontal axis in each sub-figure is (a) Goal rate (b) Discounted accumulative reward (c) Number of steps to reach the goal (d) Time in seconds for learning the RPR. The four curves in each figure correspond to the EM-PBVI and the RPR based on a behavior policy Π that queries PBVI with a probability $p_{\text{query}} = 5\%, 30\%, 50\%$, respectively. The EM-PBVI employs EM to learn a POMDP model based on the episodes collected by Π with $p_{\text{query}} = 50\%$ and then uses the PBVI (Pineau et al., 2003) to find the optimal policy based on the learned POMDP. Each curve results from an average over 20 independent runs and, for simplicity, the error bars are shown only for the RPR with a 50% query rate. The performance measures in (a)-(c) are explained in greater detail in text.

It is also seen from Figure 1 that the performance of EM-PBVI is not satisfactory and grows slowly with K. The poor performance is strongly related to insufficient exploration of the environment by the limited episodes. For EM-PBVI, the required amount of episodes is more demanding because the initial objective is to build a POMDP model instead of learning a policy. This is because policy learning is concerned with exploring the reward structure but building a POMDP requires exploration of all aspects of the environment. This demonstrates the drawback of methods that rely on learning an intervening POMDP model, with which a policy is designed subsequently.

6.2 Performance of RPR in Multi-task Reinforcement Learning (MTRL)

We investigate the performance of RPR on three MTRL problems. The first two are robotic navigation in mazes and the last one is multi-aspect classification.

6.2.1 MAZE NAVIGATION

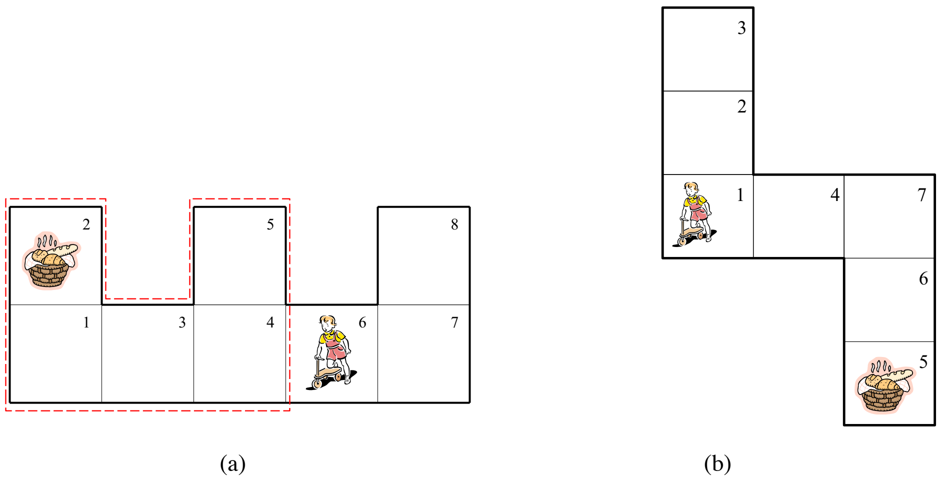
In this problem, there are $M = 10$ environments and each environment is a grid-world, that is, an array of rectangular cells. Of the ten environments, three are distinct and are shown in Figure 2, the remaining are duplicated from the three distinct ones. Specifically, the first three environments are duplicated from the first distinct one, the following three environments are duplicated from the second distinct one, and the last four environments are duplicates from the third distinct one. We assume 10 sets of episodes, with the m-th set collected from the m-th environment.

In each of the distinct environments shown in Figure 2, the agent can take five actions $\{$*move forward*, *move backward*, *move left*, *move right*, *stay*$\}$. In each cell of the grid-world environments, the agent can only observe the openness of the cell in the four directions. The agent then has a total of 16 possible observations indicating the $2^4 = 16$ different combinations of the openness of a cell in the four orientations. The actions (except the action *stay*) taken by the agent are not accurate and have some noise. The probability of arriving at the correct cell by taking a *move* action is 0.7 and the probability of arriving at other neighboring cells is 0.3. The perception is noisy, with a probability 0.8 of correctly observing the openness and the probability 0.2 of making a mistaken observation. The agent receives a unit reward when the goal (indicated by a basket of food in the figures) is reached and zero reward otherwise. The agent does not know the model of any of the environments but only has access to the episodes, that is, sequences of actions, observations, and rewards.

Algorithm Learning and Evaluation For each environment $m = 1, 2, \cdots, 10$, there is a set of K episodes $\mathcal{D}_m^{(K)}$, collected by simulating the agent-environment interaction using the models described above and a behavior policy Π that the agent follows to determine his actions. The behavior policy is the semi-random policy described in Section 6.1, with a probability $p_{\text{query}} = 0.5$ of taking the actions suggested by PBVI.

Reinforcement learning (RL) based on the ten sets of episodes $\{\mathcal{D}_m^{(K)}\}_{m=1}^{10}$ leads to ten RPRs, each associated with one of the ten environments. We consider three paradigms of learning: the MTRL in which the Gibbs-variational algorithm in Table 2 is applied to the ten sets of episodes jointly, the STRL in which the MV algorithm in Table 1 is applied to each of the ten episode sets separately, and pooling in which the MV algorithm is applied to the union of the ten episode sets. The number of decision states is chosen as $|\mathcal{Z}| = 6$ for all environments and all learning paradigms. Other larger $|\mathcal{Z}|$ give similar results and, if desired, the selection of decision states can be accomplished by maximizing the marginal empirical value with respect to $|\mathcal{Z}|$, as discussed above.

The RPR policy learned by any paradigm for any environment is evaluated by executing the policy 1000 times independently, each time starting randomly from a grid cell in the environment and taking a maximum of 15 steps. The performance of the policy is evaluated by two performance



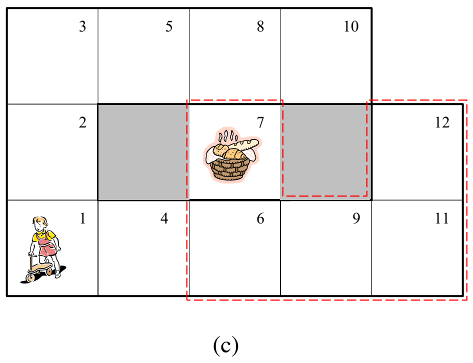
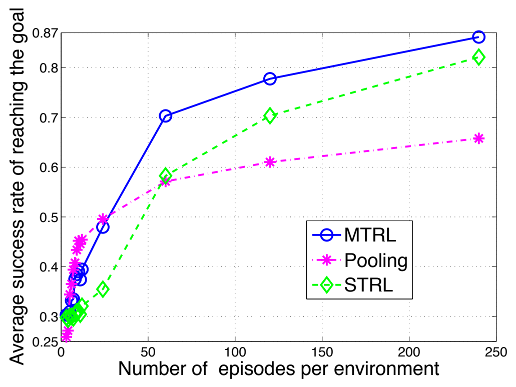


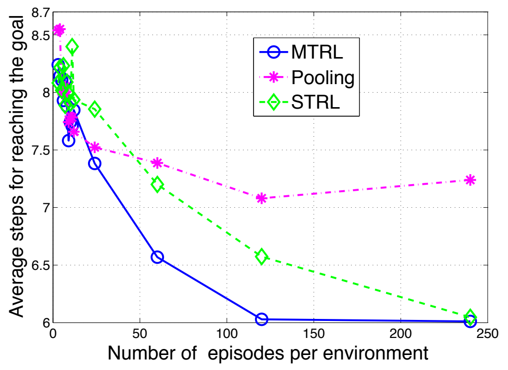
Figure 2: The three distinct grid-world environments, where the goal is designated by the basket of food, each block indicates a cell in the grid world, and the two gray cells are occupied by a wall. The red dashed lines in (a) and (c) indicate the *similar* parts in the two environments. The agent locates himself by observing the openness of a cell in the four orientations. Both the motion and the observation are noisy.

measures: (a) the average success rate at which the agent reaches the goal within 15 steps, and (b) the average number of steps that the agent takes to reach the goal. When the agent does not reach the goal within 15 steps, the number of steps is 15. Each performance measure is computed from the 1000 instances of policy execution, and is averaged over 20 independent trials.

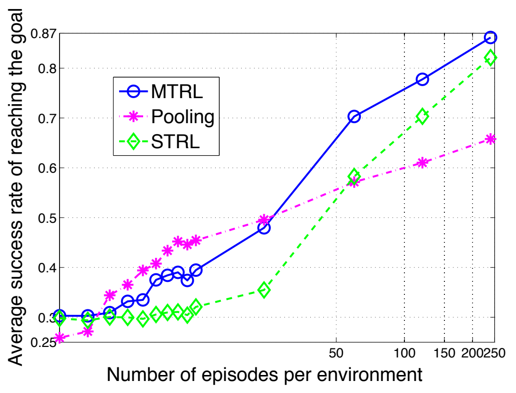
We examine the performance of each learning paradigm for various choices of K, the number of episodes per environment. Specifically we consider 16 different choices: $K = 3,4,5,6,7,8,9,10,11,12,24,60,120,240$. The performances of the three learning paradigms, averaged over 20 independent trials, are plotted in Figure 3 as a function of K. Figures 3(c) and 3(d) are respectively duplicates of Figures 3(a) and 3(b), with the horizontal axis displayed in a logarithmic scale. By (57), the choice of the precision parameter α in Dirichlet process influences the probability of sampling a new cluster; it hence influences the resulting number of distinct RPR parameters $\overline{\Theta}$. According to West (1992), the choice of α is governed by the posterior $p(\alpha|K,N) \propto p(N|K,\alpha)p(\alpha)$, where N is the number of clusters updated in the most recent iteration of the Gibbs-variational algorithm. One



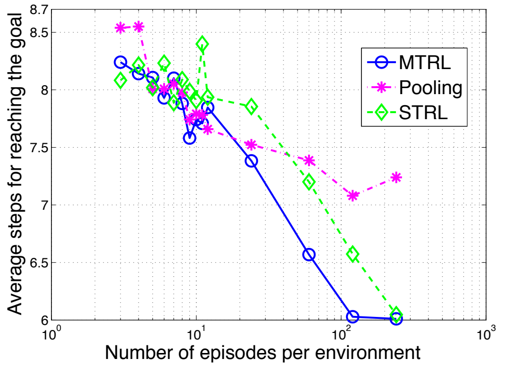
(a)



(b)



(c)



(d)

Figure 3: Comparison of MTRL, STRL, and pooling on the problem of multiple stochastic environments summarized in Figure 2. (a) Average success rate for the agent to reach the goal within 15 steps. (b) Average step for the agent reaching the target. (c) Average success rate for the agent with the horizontal axis in log scale.(d) Average step with the horizontal axis in log scale.

may choose α by sampling from the posterior or finding the mean. When K is large and $N \ll K$ and the prior $p(\alpha)$ is a Gamma distribution, the posterior $p(\alpha|K,N)$ is approximately a Gamma distribution with the mean $\mathbb{E}(\alpha) = O(N\log(K))$. For the different choices of K considered above, we choose $\alpha = 3n$, with $n = 2,3,\ldots,15$ respectively. These choices are based on approximations of $\mathbb{E}(\alpha)$ obtained by fixing N at an initial guess $N = 8$. We find that the results are relatively robust to the initial guess provided the logarithmic dependence on K is employed. The density of the DP base G_0 is of the form in (45), with all hyper-parameters set to one, making the base non-informative.

Figures 3(a) and 3(b) show that the performance of MTRL is generally much better than that of STRL and pooling. The improvement is attributed to the fact that MTRL automatically identifies and enforces appropriate sharing among the ten environments to ensure that information transfer is positive. The improvement over STRL indicates that the number of episodes required for finding the correct sharing is generally smaller that that required for finding the correct policies.

The identification of appropriate sharing is based on information from the episodes. When the number of episodes is very small (say, less than 25 in the examples here), the sharing found by MTRL may not be accurate; in this case, simply pooling the episodes across all ten environments may be a more reasonable alternative. When the number of episodes increases, however, pooling begins to show disadvantages since the environments are not all the same and forcing them to share leads to negative information transfer. The seemingly degraded performance of pooling at the first two points in Figure 3(c) may not be accurate since the results have large variations when the episodes are extremely scarce; much more Monte Carlo runs may be required to obtain accurate results in these cases.

The performance of STRL is poor when the number of episodes is small, because a small set of episodes do not provide enough information for learning a good RPR. However, the STRL performance improves significantly with the increase of episodes, which whittles down the advantage brought about by information transfer and allows STRL to eventually catch up with MTRL in performance.

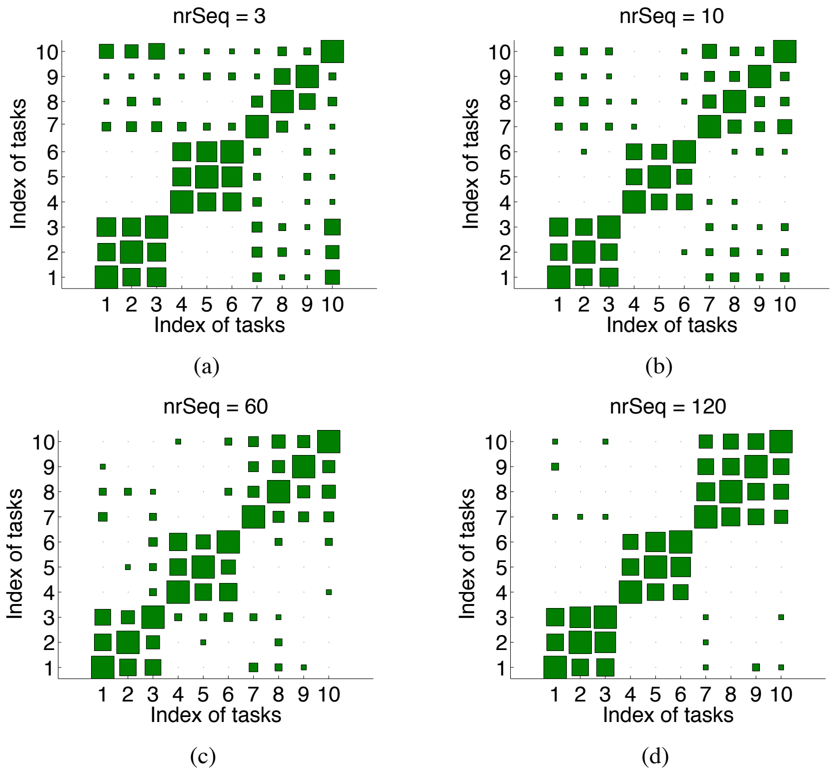


Figure 4: Hinton diagrams of the between-task similarity matrix inferred by the MTRL for the problem of multiple stochastic environments 2. The number of episodes per environment is (a) 3 (b) 10 (c) 60 (d) 120.

Analysis of the Sharing Mechanism We investigate the sharing mechanism of the MTRL by plotting Hinton diagrams. The Hinton diagram (Hinton and Sejnowski, 1986) is a quantitative way of displaying the elements of a data matrix. Each element is represented by a square whose size is proportional to the magnitude. In our case here, the data matrix is the between-task similarity matrix (Xue et al., 2007) learned by the MTRL; it is defined as follows: the between-task similarity matrix is a symmetric matrix of size $M \times M$ (where M denotes the number of tasks and $M = 10$ in the present experiment), the (i, j)-th element measuring the frequency that task i and task j belong to the same cluster (i.e., they result in the same distinct RPR). In order to avoid the bias due to any specific set of episodes, we perform 20 independent trials and average the similarity matrix over the 20 trials. In each trial, if tasks i and j belong to one cluster upon convergence of the Gibbs-variational algorithm, we add one at (i,j) and (j,i) of the matrix. We compute the between-task similarity matrices when the number of episodes is respectively $K = 3, 10, 60, 120$, which represent the typical sharing patterns inferred by the MTRL for the present maze navigation problem. The Hinton diagrams for these four matrices are plotted in Figure 4.

The Hinton diagrams in Figures 4(a) and 4(b) show that when the number of episodes is small, environments 1, 2, 3, 7, 8, 9, 10 have a higher frequency of sharing the same RPR. This sharing can be intuitively justified by first recalling that these environments are duplicates of Figures 2(a) and 2(c), and then noting that the parts circumscribed by red dashed lines in Figures 2(a) and 2(c) are quite similar. Meanwhile the Hinton diagrams also show a weak sharing between environments 4,5,6,7,8,9,10, which are duplicates of Figures 2(b) and 2(c). This is probably because the episodes are very few at this stage, and pooling episodes from environments that are not so relevant to each other could also be helpful. This explains why, in Figure 3(a), the performance of pooling is as good as that of the MTRL when the number of episodes is small.

As the number of episodes progressively increases, the ability of MTRL to identify the correct sharing improves and, as seen in Figures 4(b) and 4(c), only those episodes from relevant environments are pooled together to enhance the performance—a simple pooling of all episodes together deteriorates the performance. This explains why the MTRL outperforms pooling with the increase of episodes. Meanwhile, the STRL does not perform well for limited episodes. However, when there are more episodes from each environment, the STRL learns and performs steadily better until it outperforms the pooling and becomes comparable to the MTRL.

6.2.2 MAZE NAVIGATION 2

We consider six environments, each of which results from modifying the benchmark maze problem Hallway2 (Littman et al., 1995) in the following manner. First the goal state is displaced to a new grid cell and then the unique observation associated with the goal is changed accordingly. For each environment the location of the goal state is shown in Figure 5 as a numbered circle, where the number indicates the index of the environment. Of the six environments the first one is the original Hallway2. It is seen that environments $1,2,3$ have their goal states near the lower right corner while environments $4,5,6$ have their goal states near the upper left corner. Thus we expect that the environments are grouped into two clusters.

For each environment, a set of K episodes are collected by following a semi-random behavior policy Π that executes the actions suggested by PBVI with probability $p_{\text{query}} = 0.3$. As in Section 6.2.1 three versions of RPR are obtained for each environment, based respectively on three paradigms, namely MTRL, STRL, and pooling. The α is chosen as $5\log(K)$ with 5 corresponding to an initial guess of N and G_0 is of the form of (45) with all hyper-parameters close to one (thus the prior is non-informative). The number of decision states is $|\mathcal{Z}| = 20$ as in Section 6.1.1. The performance comparison, in terms of discounted accumulative reward and averaged over 20 independent trials, is summarized in Figure 6, as a function of the number of episodes per environment.

Figure 6(a) shows that the MTRL maintains the overall best performance regardless of the number of episodes K. The STRL and the pooling are sensitive to K, with the pooling outperforming the STRL when $K < 540$ but outperformed by the STRL when $K > 540$. In either case, however, the

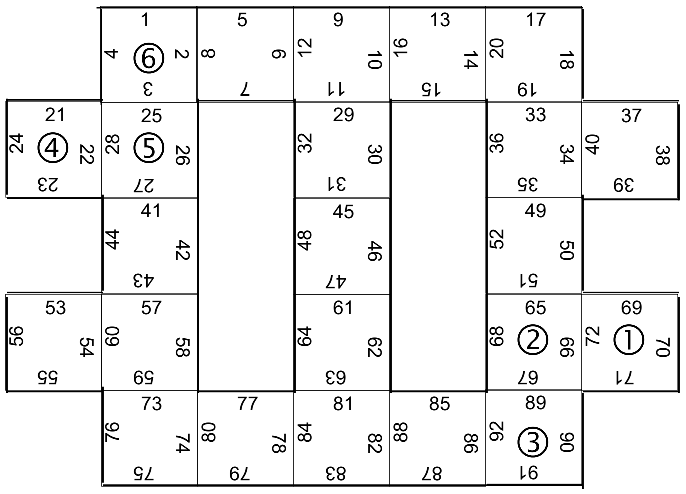
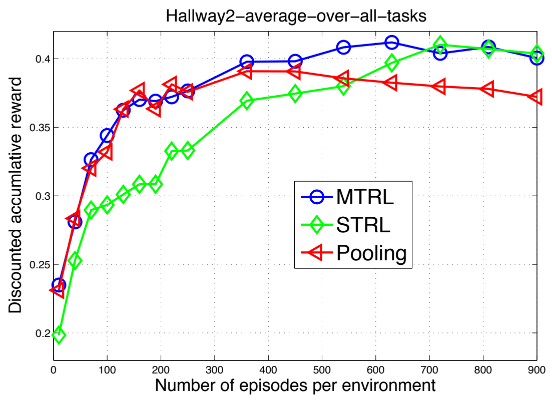
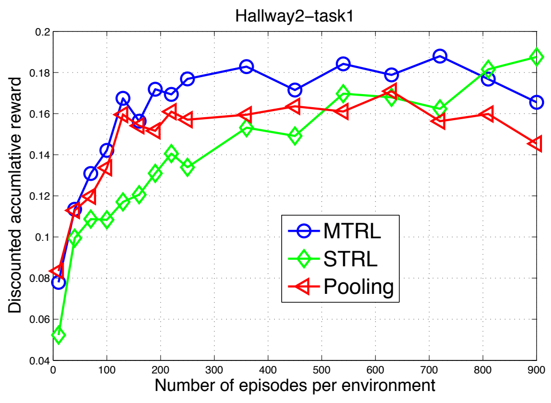


Figure 5: Displacements of goal state in the six environments considered in Maze Navigation 2. Each environment is a variant of the benchmark Hallway2 (Littman et al., 1995) with the goal displaced to a new grid cell designated by a numbered circle and the number indicating the index of the environment. The unique observation associated with the goal is also changed accordingly in each variant.



(*a*)



(*b*)

Figure 6: Performance comparison on the six environments modified from the benchmark problem Hallway2 (Littman et al., 1995). (a) Discounted accumulative reward averaged over the six environments (b) Discounted accumulative reward in the first environment, which is the original Hallway2.

MTRL performs no worse than both. The MTRL consistently performs well because it adaptively adjusts the sharing among tasks as K changes, such that the sharing is appropriate regardless of K. The adaptive sharing can be seen from Figure 7, which shows the Hinton diagram of the between-task similarity matrix learned by the MTRL, for various instances of K. When K is small there is a strong sharing among all tasks, in which case the MTRL reduces to the pooling, explaining why the

MTRL performs similar to the pooling when $K \leq 250$. When K is large, the sharing becomes weak between any two tasks, which reduces the MTRL to the STRL, explaining why the two perform similarly when $K \geq 700$. As the number of episodes approaches to $K = 540$, the performances of the STRL and the pooling tend to become closer and more comparable until they eventually meet at $K = 540$. The range of K near this intersection is also the area in which the MTRL yields the most significant margin of improvements over the STRL and the pooling. This is so because, for this range of K, the correct between-task sharing is complicated (as shown in Figure 7(b)), which can be accurately characterized by the fine sharing patterns provided by the MTRL, but cannot be characterized by the pooling or the STRL.

Figure 6(a) plots the overall performance comparison taking all environments into consideration. As an example of the performances in individual environments, we show in Figure 6(a) the performance comparison in the first environment, which is also the original Hallway2 problem. The change of magnitude in the vertical axis is due to the fact the first environment has the goal in a room (instead in the hallway), which makes it more difficult to reach the goal.

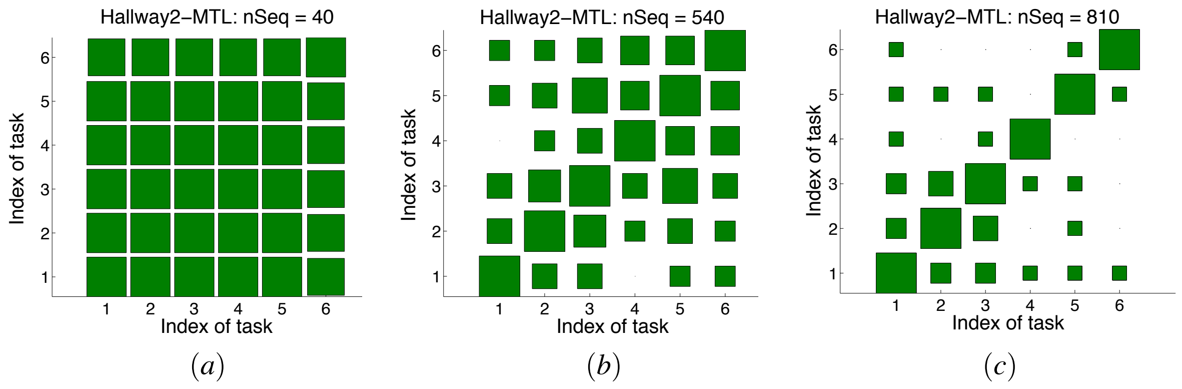


Figure 7: Hinton diagrams of the between-task similarity matrix learned by the MTRL from the six environments modified from the benchmark problem Hallway2 (Littman et al., 1995). The number of episodes is is (a) $K = 40$ (b) $K = 540$ (c) $K = 810$.

6.2.3 MULTI-ASPECT CLASSIFICATION

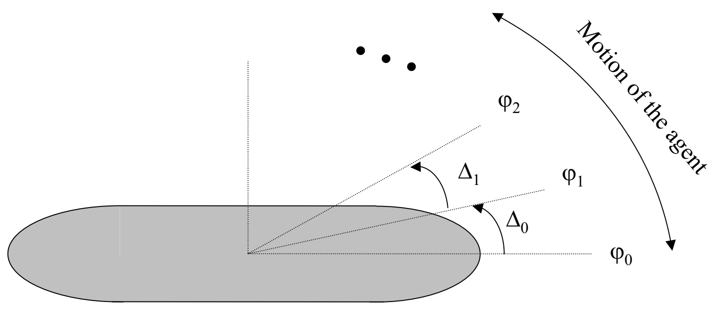


Figure 8: A typical configuration of multi-aspect classification of underwater objects.

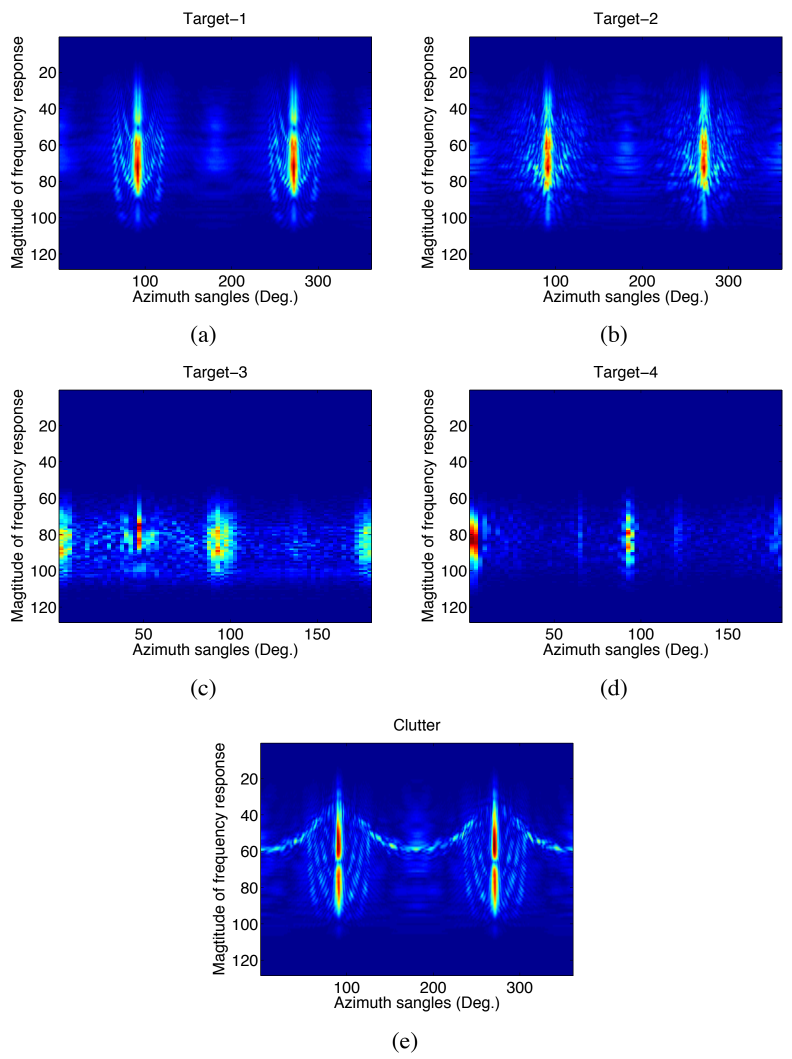


Figure 9: Frequency-domain acoustic responses of the five underwater objects (a) Target-1 (b) Target-2 (c) Target-3 (d) Target-4 (e) Clutter.

Multi-aspect classification refers to the problem of identifying the class label of an object using observations from a sequence of viewing angles. This problem is generally found in applications where the object responds to interrogations in a angle-dependent manner. In such cases, an observation at a single viewing angle carries the information specific to only that angle and the nearby angles, and one requires observations at many viewing angles to fully characterize the object.

More importantly, the observations at different viewing angles are not independent of each other, and are correlated in a complicated and yet useful way. The specific form of the angle-dependency is dictated by the physical constitution of the object as well as the nature of the interrogator—typically electromagnetic or acoustic waves. By carefully collecting and processing observations sampled at densely spaced angles, it is possible to form an image, based on which classification can be performed. An alternative approach is to treat the observations as a sequence and characterize the angle-dependency by a hidden Markov model (HMM) (Runkle et al., 1999).

In this section we consider multi-aspect classification of underwater objects based on acoustic responses of the objects. Figure 8 shows a typical configuration of the problem. The cylinder represents an underwater object of unknown identity y. We assume that the object belongs to a finite set of categories $\mathcal{Y}$ (i.e., $y \in \mathcal{Y}$). The agent aims to discover the unknown y by moving around the object and interrogating it at multiple viewing angles φ. We assume the angular motion is one-dimensional, that is, the agent moves clockwise or counterclockwise on the page, but does not move out of the page. The set of angles that can be occupied by the agent is then $[0°, 360°]$, which in practice is discretized into a finite number of angular sectors denoted by $\mathcal{S}_\varphi$.

In the HMM approach (Runkle et al., 1999), $\mathcal{S}_\varphi$ constitutes the set of hidden states, and the state transitions can be computed using simple geometry (Runkle et al., 1999), under the assumptions that each time the agent moves by a constant angular step and that the specific angles occupied by the agent are uniformly distributed within any given state. Refinement of state transitions and estimation of state emissions can be achieved by maximizing the likelihood function constructed from the training sequences. In the training phase, one trains an HMM for each $y \in \mathcal{Y}$. For an unknown object, one collects a sequence of observations (sensor data) and submit it to the HMM for each $y \in \mathcal{Y}$; the y yielding the maximum likelihood is then declared to be the identity of the unknown object. Obviously the agent must follow a common protocol to collect the data sequences in both the training and test phases, to ensure that their statistics are consistent. Since such a protocol is not part of the HMMs, a question arises as to how to specify the protocol.

From the perspective of sequential decision-making, multi-aspect classification can be formulated as a reinforcement learning problem, with a state space $\mathcal{S} = \mathcal{S}_\varphi \times \mathcal{Y}$, where $\times$ is a Cartesian product. Both $\mathcal{S}_\varphi$ and $\mathcal{Y}$ are only partially observable (through sensor data). The RL approach possesses several conspicuous advantages over the HMM approach. First, the sensor data are now collected in an active manner, under the control of agent actions. When two data sequences are collected by following the same policy of choosing actions, they are automatically ensured to be consistent in statistics, hence there is no need to specify a separate common protocol for collecting the sequential data. Second, unlike maximizing the data likelihood (under a given data collection protocol), the agent is now free to choose a more flexible learning objective by setting an appropriate reward structure. Third, unlike building a HMM for each $y \in \mathcal{Y}$, the different categories are now coalesced into a single RPR (details are presented below), making the RL a discriminative approach vis-a-via the generative HMM approach.

In our experiment, there are a total of five objects, four of them are targets of interest and one of them represents the clutter. The frequency-domain acoustic responses of these objects are shown in Figure 9, for a full coverage of angles from 0° to 360°; the data are real measurements as described in Runkle et al. (1999). We aim to distinguish each target from clutter and this gives four tasks, where task i is defined by the problem of distinguishing target-i from clutter, $i = 1, 2, 3, 4$, and the targets and clutter are as shown in Figure 9. Each task is a multi-aspect classification problem.[6] From the data in Figure 9 targets 1 and 2 have similar angle-dependent scattering phenomena, and therefore Tasks 1 and 2 are expected to be related. Targets 3 and 4 also appear to have similar angle-dependent scattering characteristics, and therefore Tasks 3 and 4 are expected to also be related. In fact, although the target details are too involved to detail here, targets 1 and 2 are both of a cylindrical form (like those in Runkle et al., 1999), while targets 3 and 4 are more irregular in shape.

6. The data are available at `http://www.ee.duke.edu/~lcarin/ShellData.zip`.

The RPR for Multi-aspect Classification In applying the RPR to multi-aspect classification, our approach is distinct from an HMM construction (Runkle et al., 1999) in two important respects. First, the RPR is a control model and it aims to optimize the value function, instead of the likelihood function. Since the RPR takes into account a reward structure, it can be more flexible in specifying the learning objective. Second, the RPR embraces all objects in the same representation, instead of having a separate model for each individual object. As a result, it is a discriminative model instead of a generative model (this may be viewed as a discriminative extension of the traditional generative HMM).

The RPR does not manipulate the angular states—it works directly with observations. Since classification is treated as a control problem in the RPR, we need two extra components, actions and rewards, to complete the specification. We consider four actions, that is, $\mathcal{A}=\{$*declare as target*, *declare as clutter*, *move clockwise and sense*, *move counterclockwise and sense*$\}$. When the agent takes action *move clockwise and sense*, it moves 5° clockwise and collects an observation; when the agent takes action *move counterclockwise and sense*, it moves 5° counterclockwise and collects an observation. The reward structure is specified as follows. A correct declaration receives a reward of 5 units, a false declaration receives a reward of -5, and the actions *move clockwise and sense* and *move counterclockwise and sense* each receives a reward of zero units. The objective, therefore, is to correctly classify the target with the minimal number of sensing actions.

The episodes used in learning the RPR consist of a number of observation sequences, each observation is associated with the action *move clockwise and sense* or *move counterclockwise and sense* and the terminal action in each episode is the correct declaration. The correction declaration is available because the episodes in this problem are the training data in standard classification, hence the ground truth of class labels is known. Note that the training episodes always terminate with a correct declaration, thus the agent never actually receives the penalty -5 during the training phase. Alternatively, one may split each episode into two, respectively terminated with the correct and the false declaration. Recall the false declaration receives the minimum reward which, after an offset of 5 to make all rewards non-negative, is converted to zero. Since a zero reward received at the end of an episode nullifies the entire episode, such an alternative is equivalent to excluding the penalized episodes.

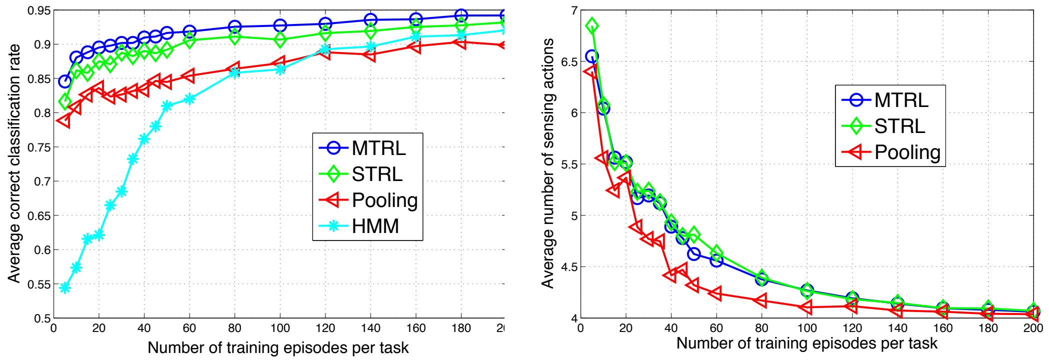
Classification Results The raw data are shown in Figure 9, for the five objects we are considering. Each datum is the response of an object measured at a particular angle and the data set for an object consists of measurements collected at $0^\circ, 1^\circ, \cdots, 359^\circ$. Each raw datum is converted into a feature vector using matching pursuit (McClure and Carin, 1997), and the feature vectors are further discretized by vector quantization (Gersho and Gray, 1992) to produce a finite code-book. As mentioned earlier, we have a total of four tasks, each task being to distinguish each of the four respective targets from the clutter.

Four methods are compared: the MTRL, the STRL, the pooling, and the hidden Markov models (HMM), where the first three are as described in Section 6.2.1 and the last one is the standard hidden Markov model (Rabiner, 1989). The four methods yield four corresponding agents, each following the policy resulting from one of the algorithms.

When the agents collect episodes during the training phase, they start from angles that are uniformly drawn from $\{1^\circ, 2^\circ, \cdots, 360^\circ\}$. For each starting angle, two episodes are collected: the first is obtained by moving clockwise to collect an observation at every 5° and terminating upon the 10-th observation, and the other is the same as the first but the agent moves counterclockwise.

During the testing phase, both the RPR agents and the HMM agent start from angles uniformly drawn from $\{1^\circ, 2^\circ, \cdots, 360^\circ\}$; however, the RPR agents follow one of the three policies (resulting respectively from the MTRL, the STRL, and the pooling) to choose an action from $\mathcal{A}$, while the HMM agent collects n observations by moving consistently clockwise or counterclockwise (either direction is chosen with a probability of 0.5) and then makes a declaration, where n is adaptively set to the *maximum* of the numbers of observations used by the three RPR agents starting from the same angle.

Figure 10 summarizes the performance as a function of the number of training episodes K, where the performance is evaluated by the correct classification rate as well as the average number of sensing actions (i.e., the average number of observations collected) before a declaration is made. Each point in the figures is an average from 20 independent trials.



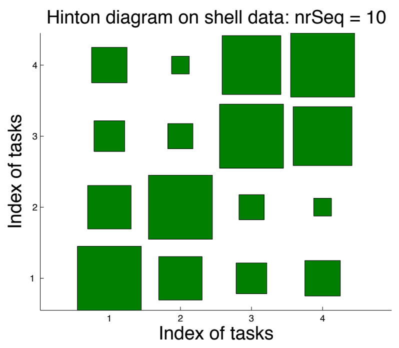
(a) Classification rate

(b) Average number of sensing steps

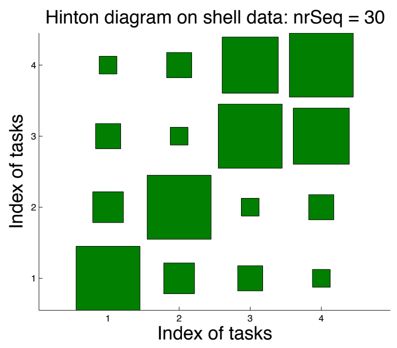
Figure 10: Performance comparison on multi-aspect classification of underwater targets (a) Average classification rate as a function of the number of training episodes per task (b) Average number of sensing actions (average number of observations collected) before a declaration is made, as a function of the number of training episodes per task.

It is seen from Figure 10(a) that the MTRL achieves the highest classification rate regardless of the number of training episodes K. The pooling performs worse than the STRL and the poor performance persists even when K is small. The latter is in contrast with the results on the maze navigation problems in Sections 6.2.1 and 6.2.2, where the pooling performs better than the STRL with a small K. The reason for this will be clear below from the sharing-mechanism analysis. It is noted that all three RPR algorithms perform much better than the HMM, demonstrating the superiority of discriminative models over generative models in classification problems.

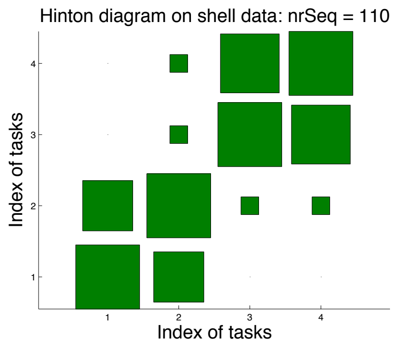
As shown by Figure 10(b), pooling takes the least number of sensing actions, which may be attributed to the over-confidence arising from an abundant set of training data, noting that the pooling agent learns its policy by using the episodes accumulated over all tasks. In contrast, the STRL agent takes the most number of actions. Considering that the STRL agent bases policy learning on the episodes collected from a single task, which may contain inadequate information, it is reasonable that the STRL agent is less confident and would make more observations before coming to a conclusion. The sensing steps taken by the MTRL agent lies in between, since it relies on related tasks, but not all tasks, to provide the episodes for policy learning.



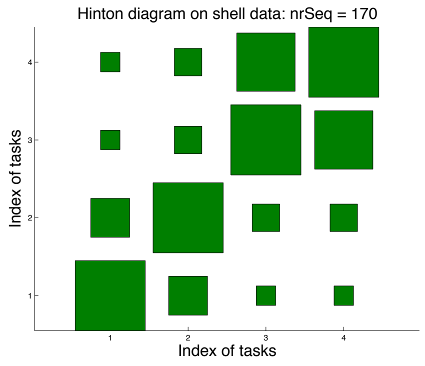
(a)



(b)



(c)



(d)

Figure 11: Sharing mechanism for multi-aspect classification of underwater targets. Each figure is the Hinton diagram of the between similarity matrix, with the number of training episodes per task: (a) 10 (b) 30 (c) 110 (d) 170.

Analysis of the Sharing Mechanism The Hinton diagram of the between-task similarity matrix is shown in Figures 11(a), 11(b), 11(c), 11(d), for the cases when the number of training episodes K is equal to 10, 30, 110, 170, respectively.

It is seen that the sharing patterns are dominated by two clusters, the first consisting of Task 1 and Task 2 and the second consisting of Task 3 and Task 4. The second cluster remains unchanged regardless of K. The first cluster tends to break when $K = 30$, but is resumed later on. The two clusters are consistent with Figure 9 which shows that targets 1 and 2 are similar and so are targets 3 and 4. The fact the two clusters are persistent through the entire range of K implies that the tasks from different clusters are weakly related even when the episodes are scarce, as a result pooling the episodes across all tasks yields poor policies. This explains the poor performance of the pooling in Figure 10(a).

To understand the reason why the cluster of tasks 1 and 2 is less stable, one need delve into some details of the targets. Target 1 and Target 2 both have a cylindrical shape while Task 3 and Task 4 are more irregular in shape. Similar geometry puts Targets 1 and 2 in one cluster and Targets 3 and 4 in another cluster. Moreover, the measurements of Targets 3 and 4 are more noisy than the measurements of Targets 1 and 2, because they are collected under different conditions. The low signal to noise ratio (SNR) increases the similarity between Targets 3 and 4 since their distinctive features are buried in the noise. The more noise-free measurements of Target 1 and 2 yields a more faithful representation of these targets, which tends to magnify their differences and make them appear less similar.

7. Conclusions

We have presented a multi-task reinforcement learning (MTRL) framework for *partially observable* stochastic environments. To our knowledge, this is the first framework proposed for MTRL in the partially observable domain.

A key element in our MTRL framework is the regionalized policy representation (RPR), which yields a history-dependent stochastic policy for environments characterized by a partially observable Markov decision process (POMDP). Learning of the RPR is based on episodic experiences collected from the environment, without requiring the environment's model. We have developed two algorithms for learning the RPR, one based on maximum-value estimation and the other based on the variational Bayesian paradigm. The latter offers the ability for selecting the number of decision states based on the Occam Razor principle and the possibility of transferring experience between related environments.

Built upon the basic RPR, the proposed MTRL framework consists of multiple RPRs, each for an environment, coupled by a common Dirichlet process (DP) that is used to produce the nonparametric prior over all RPRs. By virtue of the discreteness of the nonparametric prior, the environments are clustered into groups, with each group consisting of a subset of environments that are related in some manner. The number of groups as well as the associated environments are automatically identified, and the experiences are shared among the related environments to increase their respective exploration. A hybrid Gibbs-variational algorithm is presented for learning multiple RPRs simultaneously under the unified MTRL framework, based on selective use of the experiences collected across all environments.

Experimental results demonstrate that the proposed MTRL consistently yields superior performance regardless of the amount of experiences used in learning. The two competitors, one based on single-task reinforcement learning (STRL) and other based on simple pooling, are shown to be sensitive to the amount of experiences. The superior performance is attributed to the ability of the MTRL to automatically identify useful experiences from related environments to enhance the exploration. The MTRL adaptively adjusts sharing patterns to offset the changes in the experience and hence has addressed the problem of how to positively transfer the experience from one environment to the benefit of improving learning in another. In addition, we have also presented experimental results on benchmark problems demonstrating the RPR as a powerful stand-alone algorithm for single-task reinforcement learning.

The work presented in this paper mainly focuses on off-policy batch learning, assuming the learning is based on a fixed set of episodic experiences collected by following an external behavior policy. In the off-policy batch learning mode, the policy improvement is implemented without actually re-interacting with the environment; instead the improvement is implemented through virtual "reward re-computation" (discussed after (15)), which simulates the re-interaction with the environment. By taking reward re-computation out of the algorithm and implementing it via real re-interaction, we can learn the RPRs in an on-policy online manner. In this case, the need for an external behavior policy is eliminated and the previous version RPR is employed as the behavior policy. In the next phase of this work, we will focus on on-policy online learning of RPRs and investigate how each environment can be better explored via multi-task reinforcement learning. In this on-policy MTRL setting, multi-task learning will have two aspects: co-exploitation (already addressed in the present paper) and co-exploration (not explicitly addressed here). It is of interest to investigate how much benefit can be gained by simultaneous co-exploitation and co-exploration.

Although the experiments considered in the paper mainly involve robot navigation in grid-worlds, there are many other interesting practical problems to which the proposed algorithms are immediately applicable. The multi-aspect classification serves as a preliminary example of such applications. Other examples include using RPRs as policies to control and coordinate a set of sub-models such that the collective performance is optimized and more advanced tasks could be accomplished than by any single sub-model.

For the work presented here, the DP prior is placed directly on Θ. Because of the discrete nature of G, this implies that when parameters Θ are shared between different environments, they are shared exactly. This may be too restrictive for some problems; for two environments that are similar, we may desire the associated parameters to be similar, but not exactly the same. This may be accommodated, for example, via the following modification to the DP prior

$$\begin{aligned}\Theta_m|\Psi_m &\sim H(\Theta_m|\Psi_m),\\ \Psi_m|G &\sim G,\\ G|\alpha,G_0 &\sim DP(\alpha,G_0).\end{aligned}$$

This formulation results in an infinite mixture model for Θ, where each component is of the form H. When two environments share, their parameters share a component of this infinite mixture, but the specific draws will generally differ from each other—this can provide greater flexibility. The above modification brings some challenges to the inference. Recall that Θ is set of probability mass functions (pmf), it is natural to require H to be a product of Dirichlets. The difficulty now lies in choosing G that provides a conjugate prior for the parameters of H, which seems not easy. If G is properly specified, however, the inference should be a straightforward extension of the techniques developed in this paper. An alternative to the above modification that may avoid the inference difficulty is to follow the approach in Liu et al. (2008) to impose soft sharing by replacing the Dirac delta with its soft version.

Acknowledgments

The authors would like to thank Prof. Ronald Parr of Duke University for pointing out the drawback of an early version of the RPR optimality criterion, and for the insightful counter-examples and the long discussions that eventually lead to the optimality criterion defined in Definition 4.

We would also like to thank the anonymous reviewers for their valuable comments and suggestions, which help to improve the paper significantly.

Appendix A. Proof of Theorem 5

According to Kaelbling et al. (1998), the expected sum of exponentially discounted reward (value function) over an infinite horizon can be written as

$$V = \mathbb{E}\left[\sum_{t=0}^{\infty}\gamma^t r_t\right]$$

where $0 < \gamma < 1$ is the discount factor. Let E denote the environment in question and $\mathscr{P}_E$ the corresponding probabilistic model (POMDP). Let Θ be the parameters specifying the RPR, the expectation in our situation here is $\mathbb{E}_{\text{episodes}|E,\Theta}$. Thus

$$\begin{aligned}
V &= \mathbb{E}_{\text{episodes}|E,\Theta}\left[\sum_{t=0}^{\infty}\gamma^t r_t\right] \\
&= \sum_{a_0 r_0 o_1 a_1 r_1 \cdots} p(a_0 r_0 o_1 a_1 r_1 o_2 \cdots | E, \Theta)\left[\sum_{t=0}^{\infty}\gamma^t r_t\right] \\
&= \sum_{a_0 a_1 \cdots} p(a_0 a_1 \cdots|\Theta)\mathbb{E}_{r_0 o_1 r_1 o_2 r_2 \cdots | a_0 a_1 \cdots \sim \mathscr{P}_E}\left[\sum_{t=0}^{\infty}\gamma^t r_t\right] \\
&= \sum_{a_0 a_1 \cdots} p^{\Pi}(a_0 a_1 \cdots)\frac{p(a_0 a_1 \cdots|\Theta)}{p^{\Pi}(a_0 a_1 \cdots)}\mathbb{E}_{r_0 o_1 r_1 o_2 r_2 \cdots | a_0 a_1 \cdots \sim \mathscr{P}_E}\left[\sum_{t=0}^{\infty}\gamma^t r_t\right] \\
&\quad \text{(Importance sampling Robert and Casella, 1999)} \\
&= \sum_{a_0 a_1 \cdots} p^{\Pi}(a_0 a_1 \cdots)\mathbb{E}_{r_0 o_1 r_1 o_2 r_2 \cdots | a_0 a_1 \cdots \sim \mathscr{P}_E}\left[\sum_{t=0}^{\infty}\frac{\gamma^t r_t}{\prod_{\tau=0}^{t} p^{\Pi}(a_\tau|h_\tau)}\prod_{\tau=0}^{t} p(a_\tau|h_\tau,\Theta)\right] \\
&= \mathbb{E}_{a_0 a_1 \cdots \sim p^{\Pi}}\mathbb{E}_{r_0 o_1 r_1 o_2 r_2 \cdots | a_0 a_1 \cdots \sim \mathscr{P}_E}\left[\sum_{t=0}^{\infty}\frac{\gamma^t r_t}{\prod_{\tau=0}^{t} p^{\Pi}(a_\tau|h_\tau)}\prod_{\tau=0}^{t} p(a_\tau|h_\tau,\Theta)\right] \\
&= \lim_{K\to\infty}\frac{1}{K}\sum_{k=1}^{K}\left[\sum_{t=0}^{\infty}\frac{\gamma^t r_t^k}{\prod_{\tau=0}^{t} p^{\Pi}(a_\tau^k|h_\tau^k)}\prod_{\tau=0}^{t} p(a_\tau^k|h_\tau^k,\Theta)\right] \\
&\quad \left(a_0^k a_1^k \cdots \sim p^{\Pi},\ r_0^k o_1^k r_1^k o_2^k r_2^k \cdots | a_0^k a_1^k \cdots \sim \mathscr{P}_E\right) \\
&= \lim_{K\to\infty}\frac{1}{K}\sum_{k=1}^{K}\left[\sum_{t=0}^{T_k}\frac{\gamma^t r_t^k}{\prod_{\tau=0}^{t} p^{\Pi}(a_\tau^k|h_\tau^k)}\prod_{\tau=0}^{t} p(a_\tau^k|h_\tau^k,\Theta)\right] \\
&= \lim_{K\to\infty}\widehat{V}(\mathcal{D}^{(K)};\Theta)
\end{aligned}$$

where the sum over $0 \le t < \infty$ is equal to the sum over $0 \le t \le T_k$ because $r_t^k = 0$ for $t > T_k$ according to Definition 2. Q.E.D.

Appendix B. Proof of Theorem 6

We begin our derivation by writing the empirical value function in its logarithm

$$\begin{aligned}
\ln\widehat{V}(\mathcal{D}^{(K)};\Theta) &= \ln\frac{1}{K}\sum_{k=1}^{K}\sum_{t=0}^{T_k}\tilde{r}_t^k\sum_{z_0^k,\cdots,z_t^k=1}^{|\mathcal{Z}|} p(a_{0:t}^k, z_{0:t}^k|o_{1:t}^k,\Theta) \\
&= \ln\sum_{k=1}^{K}\sum_{t=0}^{T_k}\sum_{z_0^k,\cdots,z_t^k=1}^{|\mathcal{Z}|}\frac{q_t^k(z_{0:t}^k)}{K}\frac{\tilde{r}_t^k p(a_{0:t}^k, z_{0:t}^k|o_{1:t}^k,\Theta)}{q_t^k(z_{0:t}^k)},
\end{aligned} \tag{80}$$

where

$$\begin{aligned}
q_t^k(z_{0:t}^k) &\ge 0, \\
\frac{1}{K}\sum_{k=1}^{K}\sum_{t=0}^{T_k}\sum_{z_0^k,\cdots,z_t^k=1}^{|\mathcal{Z}|} q_t^k(z_{0:t}^k) &= 1.
\end{aligned}$$

Applying Jensen's inequality to (80), we obtain

$$\ln \widehat{V}(\mathcal{D}^{(K)};\Theta) \quad \geq \quad \sum_{k=1}^{K}\sum_{t=0}^{T_k}\sum_{z_0^k,\cdots,z_t^k=1}^{|\mathcal{Z}|} \frac{q_t^k(z_{0:t}^k)}{K} \ln \frac{\tilde{r}_t^k p(a_{0:t}^k, z_{0:t}^k | o_{1:t}^k, \Theta)}{q_t^k(z_{0:t}^k)}. \tag{81}$$

The lower bound is maximized when

$$q_t^k(z_{0:t}^k) \quad = \quad q_t^k(z_{0:t}^k|\Theta) \overset{Def.}{=} \frac{\tilde{r}_t^k}{\widehat{V}(\mathcal{D}^{(K)};\Theta)} p(a_{0:t}^k, z_{0:t}^k | o_{1:t}^k, \Theta),$$

which turns the inequality in into an equality. Define

$$\mathrm{LB}(\widehat{\Theta}|\Theta) = \frac{1}{K}\sum_{k=1}^{K}\sum_{t=0}^{T_k}\sum_{z_0^k,\cdots,z_t^k=1}^{|\mathcal{Z}|} q_t^k(z_{0:t}^k|\Theta) \ln \frac{\tilde{r}_t^k p(a_{0:t}^k, z_{0:t}^k | o_{1:t}^k, \widehat{\Theta})}{q_t^k(z_{0:t}^k|\Theta)}.$$

By (81), $\mathrm{LB}(\widehat{\Theta}|\Theta) \leq \mathrm{LB}(\widehat{\Theta}|\widehat{\Theta}) = \ln \widehat{V}(\mathcal{D}^{(K)};\widehat{\Theta})$ holds for any Θ and $\widehat{\Theta}$. Therefore, when $\widehat{\Theta} = \arg\max_{\widehat{\Theta}\in\mathcal{F}} \mathrm{LB}(\widehat{\Theta}|\Theta)$, we have

$$\ln \widehat{V}(\mathcal{D}^{(K)};\Theta) = \mathrm{LB}(\Theta|\Theta) \leq \mathrm{LB}(\widehat{\Theta}|\Theta) \leq \mathrm{LB}(\widehat{\Theta}|\widehat{\Theta}) = \ln \widehat{V}(\mathcal{D}^{(K)};\widehat{\Theta}).$$

Starting from $\Theta^{(0)}$ we compute

$$\begin{aligned}
\Theta^{(1)} &= \arg\max_{\widehat{\Theta}\in\mathcal{F}} \mathrm{LB}(\widehat{\Theta}|\Theta^{(0)}),\\
\Theta^{(2)} &= \arg\max_{\widehat{\Theta}\in\mathcal{F}} \mathrm{LB}(\widehat{\Theta}|\Theta^{(1)}),\\
&\ \ \vdots \qquad \vdots
\end{aligned}$$

which satisfy $\widehat{V}(\mathcal{D}^{(K)};\Theta^{(0)}) \leq \widehat{V}(\mathcal{D}^{(K)};\Theta^{(1)}) \leq \widehat{V}(\mathcal{D}^{(K)};\Theta^{(2)}) \leq \cdots$. Since the value function is upper bounded, this monotonically increasing sequence must converge, which happens at a maxima of $\widehat{V}(\mathcal{D}^{(K)};\Theta)$. Q.E.D.

Appendix C. Proof of Lemma 7

Substituting (26) and (27), we have

$$\text{Right side of (28)} = \frac{p(z_\tau^k = i, z_{\tau+1}^k = j, a_{0:t}^k | o_{1:t}^k, \Theta)}{\prod_{\tau'=0}^{t} p(a_{\tau'}^k | h_{\tau'}^k)}.$$

Since the denominator is equal to $p(a_{0:t}^k | o_{1:t}^k, \Theta)$ by (9), we have

$$\text{Right side of (28)} = p(z_\tau^k = i, z_{\tau+1}^k = j | a_{0:t}^k, o_{1:t}^k, \Theta) = \xi_{t,\tau}^k(i,j).$$

Similarly,

$$\begin{aligned}
\text{Right side of (29)} &= \frac{p(z_\tau^k = i, a_{0:t}^k | o_{1:t}^k, \Theta)}{\prod_{\tau'=0}^{t} p(a_{\tau'}^k | h_{\tau'}^k)} = \frac{p(z_\tau^k = i, a_{0:t}^k | o_{1:t}^k, \Theta)}{p(a_{0:t}^k | o_{1:t}^k, \Theta)}\\
&= p(z_\tau^k = i | a_{0:t}^k, o_{1:t}^k, \Theta)\\
&= \phi_{t,\tau}^k(i).
\end{aligned}$$

Q.E.D.

Appendix D. Proof of Theorem 8

We rewrite the lower bound in (61) as

$$\begin{aligned}\mathrm{LB}\left(\left\{q_t^k\right\}, g(\Theta)\right) = \frac{1}{K}\sum_{k=1}^{K}\sum_{t=0}^{T_k}\sum_{z_0^k,\cdots,z_t^k=1}^{|\mathcal{Z}|} \int q_t^k(z_{0:t}^k) g(\Theta) \ln \tilde{r}_t^k p(a_{0:t}^k, z_{0:t}^k | o_{1:t}^k, \Theta)\, d\Theta \\ -\frac{1}{K}\sum_{k=1}^{K}\sum_{t=0}^{T_k}\sum_{z_0^k,\cdots,z_t^k=1}^{|\mathcal{Z}|} q_t^k(z_{0:t}^k) \ln q_t^k(z_{0:t}^k) - \int g(\Theta) \ln \frac{g(\Theta)}{G_0(\Theta)}\, d\Theta. \qquad (82)\end{aligned}$$

We alternatively find the $\{q_t^k\}$ and $g(\Theta)$ that maximizes the lower bound, keeping one fixed while finding the other.

Keeping $g(\Theta)$ fixed, we solve $\max_{\{q_t^k\}} \mathrm{LB}\left(\left\{q_t^k\right\}, g(\Theta)\right)$ subject to the normalization constraint for $\{q_t^k\}$. We construct the Lagrangian

$$\ell_q = \mathrm{LB}\left(\left\{q_t^k\right\}, g(\Theta)\right) - \lambda\left(K - \sum_{k=1}^{K}\sum_{t=1}^{T_k}\sum_{z_0^k,\cdots,z_t^k=1}^{|\mathcal{Z}|} q_t^k(z_{0:t}^k)\right),$$

where λ is the Lagrangian multiplier. Differentiating ℓ_q with respect to $q_t^k(z_{0:t}^k)$ and setting the result to zero, we obtain

$$\frac{\partial \ell_q}{\partial\left(q_t^k(z_{0:t}^k)\right)} = \frac{1}{K}\int g(\Theta)\ln \tilde{r}_t^k p(a_{0:t}^k, z_{0:t}^k | o_{1:t}^k, \Theta)\, d\Theta - \frac{1}{K}\ln q_t^k(z_{0:t}^k) - \frac{1}{K} + \lambda = 0,$$

which is solved to give

$$q_t^k(z_{0:t}^k) = e^{K\lambda - 1}\tilde{r}_t^k \exp\left\{\int g(\Theta)\ln p(a_{0:t}^k, z_{0:t}^k | o_{1:t}^k, \Theta)\, d\Theta\right\}.$$

Using the constraint $\sum_{k=1}^{K}\sum_{t=1}^{T_k}\sum_{z_0^k,\cdots,z_t^k=1}^{|\mathcal{Z}|} q_t^k(z_{0:t}^k) = K$, (62) is arrived with $e^{1-K\lambda} = C_z$.

Keeping $\{q_t^k\}$ fixed, we solve $\max_{g(\Theta)} \mathrm{LB}\left(\left\{q_t^k\right\}, g(\Theta)\right)$ subject to the normalization constraint that $\int g(\Theta) d\Theta = 1$. Construct the Lagrangian

$$\ell_g = \mathrm{LB}\left(\left\{q_t^k\right\}, g(\Theta)\right) - \lambda\left(1 - \int g(\Theta) d\Theta\right),$$

where λ is the Lagrangian multiplier. Differentiating ℓ_g with respect to $g(\Theta)$ and setting the result to zero, we obtain

$$\frac{\partial \ell_g}{\partial\left(g(\Theta)\right)} = \frac{1}{K}\sum_{k=1}^{K}\sum_{t=0}^{T_k}\sum_{z_0^k,\cdots,z_t^k=1}^{|\mathcal{Z}|} q_t^k(z_{0:t}^k)\ln \tilde{r}_t^k p(a_{0:t}^k, z_{0:t}^k | o_{1:t}^k, \Theta) - 1 - \ln\frac{g(\Theta)}{G_0(\Theta)} + \lambda = 0,$$

which is solved to give

$$g(\Theta) = \frac{G_0(\Theta)}{e^{1-\lambda}} \exp\left\{\frac{1}{K}\sum_{k=1}^{K}\sum_{t=0}^{T_k}\sum_{z_0^k,\cdots,z_t^k=1}^{|\mathcal{Z}|} q_t^k(z_{0:t}^k)\ln \tilde{r}_t^k p(a_{0:t}^k, z_{0:t}^k | o_{1:t}^k, \Theta)\right\}.$$

By using the constraint $\int g(\Theta) d\Theta = 1$, we arrive at (63) with $e^{1-\lambda} = C_\Theta$. Q.E.D.

References

D. Aberdeen and J. Baxter. Scalable internal-state policy-gradient methods for POMDPs. In *Proceedings of the Nineteenth International Conference on Machine Learning*, pages 3–10. Morgan Kaufmann Publishers Inc., 2002.

C. E. Antoniak. Mixtures of Dirichlet processes with applications to Bayesian nonparametric problems. *The Annals of Statistics*, 2(6):1152–1174, November 1974.

B. Bakker. *The State of Mind: Reinforcement Learning with Recurrent Neural Networks*. PhD thesis, Unit of Cognitive Psychology, Leiden University, 2004.

B. Bakker and T. Heskes. Task clustering and gating for Bayesian multitask learning. *Journal of Machine Learning Research*, 4:83–99, 2003.

J. Baxter and P. L. Bartlett. Infinite-horizon policy-gradient estimation. *Journal of Artificial Intelligence Research*, 15:319–350, 2001.

M. J. Beal. *Variational Algorithms for Approximate Bayesian Inference*. PhD thesis, Gatsby Computational Neuroscience Unit, University College London, 2003.

R. E. Bellman. *Dynamic Programming*. Princeton University Press, 1957.

D. Blackwell. Discounted dynamic programming. *Ann. Math. Stat.*, 36:226–235, 1965.

D. Blackwell and J. MacQueen. Ferguson distributions via Polya urn schemes. *Annals of Statistics*, 1:353–355, 1973.

H.-T. Cheng. *Algorithms for Partially Observable Markov Decision Processes*. PhD thesis, University of British Columbia, Vancouver, BC, 1988.

L. Chrisman. Reinforcement learning with perceptual aliasing: The perceptual distinctions approach. In *Proceedings of the Tenth International Conference on Artificial Intelligence*, pages 183–188. San Jose, California: AAAI Press, 1992.

A. Dempster, N. Laird, and D. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of Royal Statistical Society B*, 39:1–38, 1977.

F. Doshi, J. Pineau, and N. Roy. Reinforcement learning with limited reinforcement: Using Bayes risk for active learning in POMDPs. In *Proceedings of the 25th International Conference on Machine Learning*, pages 256–263. ACM, 2008.

M. Escobar and M. West. Bayesian density estimation and inference using mixtures. *Journal of the American Statistical Association*, 90:577–588, 1995.

T. Ferguson. A Bayesian analysis of some non-parametric problems. *The Annals of Statistics*, 1: 209–230, 1973.

A. Gelfand and A. Smith. Sample based approaches to calculating marginal densities. *Journal of the American Statistical Association*, 85:398–409, 1990.

S. Geman and D. Geman. Stochastic relaxation, Gibbs distributions and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6:721–741, 1984.

A. Gersho and R. M. Gray. *Vector Quantization and Signal Compression*. Kluwer Academic Press/Springer, 1992.

C. Guestrin, D. Koller, C. Gearhart, and N. Kanodia. Generalizing plans to new environments in relational MDPs. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 2003.

E. A. Hansen. An improved policy iteration algorithm for partially observable MDPs. In *Advances in Neural Information Processing Systems*, volume 10, 1997.

G. E. Hinton and T. J. Sejnowski. Learning and relearning in Boltzmann machines. In J. L. McClelland, D. E. Rumelhart, and the PDP Research Group, editors, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, volume 1, pages 282–317. MIT Press, Cambridge, MA, 1986.

T. Jaakkola, S. P. Singh, and M. I. Jordan. Reinforcement learning algorithm for partially observable Markov decision problems. In *Advances in Neural Information Processing Systems*, volume 7. MIT Press, Cambridge, MA., 1995.

T. S. Jaakkola. Tutorial on variational approximation methods. In M. Opper and D. Saad, editors, *Advanced Mean Field Methods: Theory and Practice*, pages 129–160. MIT Press, 2001.

M. I. Jordan, Z. Ghahramani, T. S. Jaakkola, and L. K. Saul. An introduction to variational methods for graphical models. In *Learning in Graphical Models*, pages 105–161, Cambridge, MA, 1999. MIT Press.

L. Kaelbling, M. Littman, and A. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101:99–134, 1998.

N. D. Lawrence and J. C. Platt. Learning to learn with the informative vector machine. In *Proceedings of the 21st International Conference on Machine Learning*, 2004.

H. Li. *Planning and Learning in Partially Observable Stochastic Environments*. PhD thesis, Duke University, 2006. publically available at http://people.ee.duke.edu/˜hl1/.

H. Li, X. Liao, and L. Carin. Incremental least squares policy iteration for POMDPs. In *Proceedings of the Twenty-First National Conference on Artificial Intelligence (AAAI)*, 2006a.

H. Li, X. Liao, and L. Carin. Region-based value iteration for partially observable Markov decision processes. In *Proceedings of the 23nd International Machine Learning Conference*, 2006b.

X. Liao, H. Li, R. Parr, and L. Carin. Regionalized policy representation for reinforcement learning in POMDPs. In *The Snowbird Learning Workshop*, 2007.

M. L. Littman, A. R. Cassandra, and L. P. Kaelbling. Learning policies for partially obsevable environments: Scaling up. In *Proceedings of the Twelfth International Conference on Machine Learning*, pages 362–370, 1995.

Q. Liu, X. Liao, and L. Carin. Semi-supervised multitask learning. In J.C. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems 20*, pages 937–944. MIT Press, Cambridge, MA, 2008.

W. S. Lovejoy. Computationally feasible bounds for partially observed Markov decision processes. *Operations Research*, 39(1):162–175, 1991.

S. N. MacEachern. Estimating normal means with a conjugate style Dirichlet process prior. *Communications in Statistics: Simulation and Computation*, 23:727–741, 1994.

R. A. McCallum. *Reinforcement Learning with Selective Attention and Hidden State*. PhD thesis, Department of Computer Science, University of Rochester, 1995.

M. McClure and L. Carin. Matched pursuits with a wave-based dictionary. *IEEE Trans. Signal Proc.*, 45:2912–2927, Dec. 1997.

N. Meuleau, L. Peshkin, K. Kim, and L. Kaelbling. Learning finite-state controllers for partially observable environments. In *Proceedings of the 15th Annual Conference on Uncertainty in Artificial Intelligence (UAI-99)*, pages 427–43, San Francisco, CA, 1999. Morgan Kaufmann.

R.M. Neal. Markov chain sampling methods for Dirichlet process mixture models. Technical Report 9815, Dept. of Statistics, University of Toronto, 1998.

J. Pineau, G. Gordon, and S. Thrun. Point-based value iteration: An anytime algorithm for POMDPs. In *Proceedings of IJCAI*, pages 1025 – 1032, August 2003.

P. Poupart and C. Boutilier. Bounded finite state controllers. In *Advances in Neural Information Processing Systems (NIPS)*, 2003.

L. R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–285, 1989.

C. P. Robert and G. Casella. *Monte Carlo Statistical Methods*. Springer-Verlag, 1999.

S. Ross, B. Chaib-draa, and J. Pineau. Bayes-adaptive POMDPs. In *Advances in Neural Information Processing Systems (NIPS)*, volume 20, 2008.

P. R. Runkle, P. K. Bharadwaj, L. Couchman, and L. Carin. Hidden Markov models for multiaspect target classification. *IEEE Transactions on Signal Processing*, 47:2035–2040, July 1999.

J. Sethuraman. A constructive definition of Dirichlet priors. *Statistica Sinica*, 4:639–650, 1994.

R. D. Smallwood and E. J. Sondik. The optimal control of partially observable Markov processes over a finite horizon. *Operational Research*, 21:1071–1088, 1973.

T. Smith and R. Simmons. Point-based POMDP algorithms: Improved analysis and implementation. In *Proc. of the Conference on Uncertainty in Artificial Intelligence*, 2005.

E. J. Sondik. *The Optimal Control of Partially Observable Markov Processes*. PhD thesis, Stanford University, 1971.

E. J. Sondik. The optimal control of partially observable Markov processes over the infinite horizon: Discounted costs. *Operations Research*, 26(2):282–304, Mar. 1978.

M. T. J. Spaan and N. Vlassis. Perseus: Randomized point-based value iteration for POMDPs. *Journal of Artificial Intelligence Research*, 24:195220, 2005.

R. Sutton and A. Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, 1998.

S. Thrun. Is learning the *n*-th thing any easier than learning the first? In *Advances in Neural Information Processing Systems (NIPS)*, 1996.

T. Wang, D. Lizotte, M. Bowling, and D. Schuurmans. Bayesian sparse sampling for on-line reward optimization. In *Proceedings of the Twenty-Second International Conference on Machine Learning (ICML)*, pages 961–968, 2005.

M. Welling, Y. W. Teh, and B. Kappen. Hybrid variational/Gibbs collapsed inference in topic models. In *Proceedings of the Conference on Uncertainty in Artifical Intelligence (UAI)*, pages 587–594, 2008.

M. West. Hyperparameter estimation in Dirichlet process mixture models. Technical Report 92-A03, ISDS Discussion Paper, Duke University, 1992.

M. West, P. Muller, and M.D. Escobar. Hierarchical priors and mixture models, with application in regression and density estimation. In A.F.M. Smith and P. Freeman, editors, *Aspects of Uncertainty: A Tribute to D. V. Lindley*, pages 363–386. New York: Wiley, 1994.

D. Wierstra and M. Wiering. Utile distinction hidden Markov models. In *Proceedings of the International Conference on Machine Learning*, 2004.

A. Wilson, A. Fern, S. Ray, and P. Tadepalli. Multi-task reinforcement learning: A hierarchical Bayesian approach. In *Proceedings of the 24st International Conference on Machine Learning*, 2007.

Y. Xue, X. Liao, L. Carin, and B. Krishnapuram. Multi-task learning for classification with Dirichlet process priors. *Journal of Machine Learning Research (JMLR)*, 8:35–63, 2007.

K. Yu, A. Schwaighofer, V. Tresp, W.-Y. Ma, and H. Zhang. Collaborative ensemble learning: Combining collaborative and content-based information filtering via hierarchical Bayes. In *Proceedings of the 19th Conference on Uncertainty in Artificial Intelligence*, 2003.

K. Yu, V. Tresp, and S. Yu. A nonparametric hierarchical Bayesian framework for information filtering. In *Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2004.

Journal of Machine Learning Research 10 (2009) 1187-1238 Submitted 4/08; Revised 11/08; Published 6/09

The Hidden Life of Latent Variables: Bayesian Learning with Mixed Graph Models

Ricardo Silva[*] RICARDO@STATS.UCL.AC.UK
Department of Statistical Science
University College London, WC1E 6BT, UK

Zoubin Ghahramani[†] ZOUBIN@ENG.CAM.AC.UK
Department of Engineering
University of Cambridge, CB2 1PZ, UK

Editor: David Maxwell Chickering

Abstract

Directed acyclic graphs (DAGs) have been widely used as a representation of conditional independence in machine learning and statistics. Moreover, hidden or latent variables are often an important component of graphical models. However, DAG models suffer from an important limitation: the family of DAGs is not closed under marginalization of hidden variables. This means that in general we cannot use a DAG to represent the independencies over a subset of variables in a larger DAG. Directed mixed graphs (DMGs) are a representation that includes DAGs as a special case, and overcomes this limitation. This paper introduces algorithms for performing Bayesian inference in Gaussian and probit DMG models. An important requirement for inference is the specification of the distribution over parameters of the models. We introduce a new distribution for covariance matrices of Gaussian DMGs. We discuss and illustrate how several Bayesian machine learning tasks can benefit from the principle presented here: the power to model dependencies that are generated from hidden variables, but without necessarily modeling such variables explicitly.

Keywords: graphical models, structural equation models, Bayesian inference, Markov chain Monte Carlo, latent variable models

1. Contribution

The introduction of graphical models (Pearl, 1988; Lauritzen, 1996; Jordan, 1998) changed the way multivariate statistical inference is performed. Graphical models provide a suitable language to decompose many complex real-world processes through conditional independence constraints.

Different families of independence models exist. The directed acyclic graph (DAG) family is a particularly powerful representation. Besides providing a language for encoding causal statements (Spirtes et al., 2000; Pearl, 2000), it is in a more general sense a family that allows for non-monotonic independence constraints: that is, models where some independencies can be destroyed by conditioning on new information (also known as the "explaining away" effect — Pearl, 1988), a feature to be expected in many real problems.

*. Part of this work was done while RS was at the Gatsby Computational Neuroscience Unit, UCL, and at the Statistical Laboratory, University of Cambridge.

†. Also affiliated with the Machine Learning Department, Carnegie Mellon University.

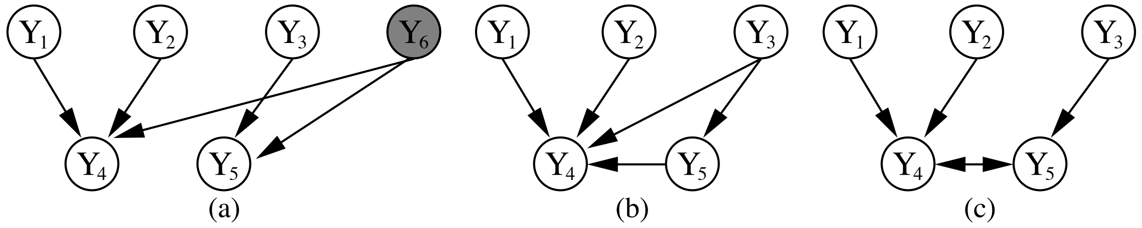


Figure 1: Consider the DAG in (a). Suppose we want to represent the marginal dependencies and independencies that result after marginalizing out Y_6. The simplest resulting DAG (i.e., the one with fewest edges) is depicted in (b). However, notice that this graph does not encode some of the independencies of the original model. For instance, Y_3 and Y_4 are no longer marginally independent in the modified DAGs. A different family of graphical models, encoded with more than one type of edge (directed and *bi-directed*), is the focus of this paper. The graph in (c) depicts the solution using this "mixed" representation.

However, DAG independence models have an undesirable feature: they are not closed under marginalization, as we will illustrate. Consider the regression problem where we want to learn the effect of a cocktail of two drugs for blood pressure, while controlling for a chemotherapy treatment of liver cancer. We refer to Y_1, Y_2 as the dosage for the blood pressure drugs, Y_3 as a measure of chemotherapy dosage, Y_4 as blood pressure, and Y_5 as an indicator of liver status. Moreover, let Y_6 be an hidden physiological factor that affects both blood pressure and liver status. It is assumed that the DAG corresponding to this setup is given by Figure 1(a).

In this problem, predictions concerning Y_6 are irrelevant: what we care is the marginal for $\{Y_1,\dots,Y_5\}$. Ideally, we want to take such irrelevant hidden variables out of the loop. Yet the set of dependencies within the marginal for $\{Y_1,\dots,Y_5\}$ cannot be efficiently represented as a DAG model. If we remove the edge $Y_3 \rightarrow Y_4$ from Figure 1(b), one can verify this will imply a model where Y_3 and Y_4 are independent given Y_5, which is not true in our original model. To avoid introducing unwanted independence constraints, a DAG such as the one in Figure 1(b) will be necessary. Notice that in general this will call for extra dependencies that did not exist originally (such as Y_3 and Y_4 now being marginally dependent). Not only learning from data will be more difficult due to the extra dependencies, but specifying prior knowledge on the parameters becomes less intuitive and therefore more error prone.

In general, it will be the case that variables of interest have hidden common causes. This puts the researcher using DAGs in a difficult position: if she models only the marginal comprising the variables of interest, the DAG representation might not be suitable anymore. If she includes all hidden variables for the sake of having the desirable set of independencies, extra assumptions about hidden variables will have to be taken into account. In this sense, the DAG representation is flawed. There is a need for a richer family of graphical models, for which *mixed graphs* are an answer.

Directed mixed graphs (DMGs) are graphs with directed and bi-directed edges. In particular, acyclic directed mixed graphs (ADMGs) have no directed cycle, that is, no sequence of directed edges $X \rightarrow \cdots \rightarrow X$ that starts and ends on the same node. Such a representation encodes a set of conditional independencies among random variables, which can be read off a graph by using a

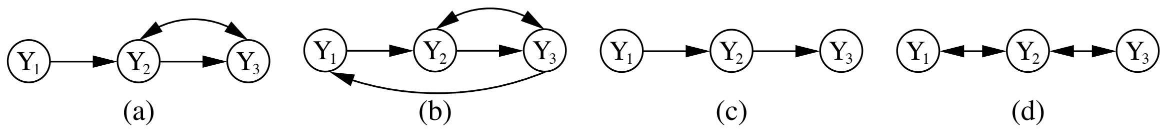


Figure 2: Different examples of directed mixed graphs. The graph in (b) is cyclic, while all others are acyclic. A subgraph of two variables where both edges $Y_1 \rightarrow Y_2$ and $Y_1 \leftrightarrow Y_2$ are present is sometimes known as a "bow pattern" (Pearl, 2000) due to its shape.

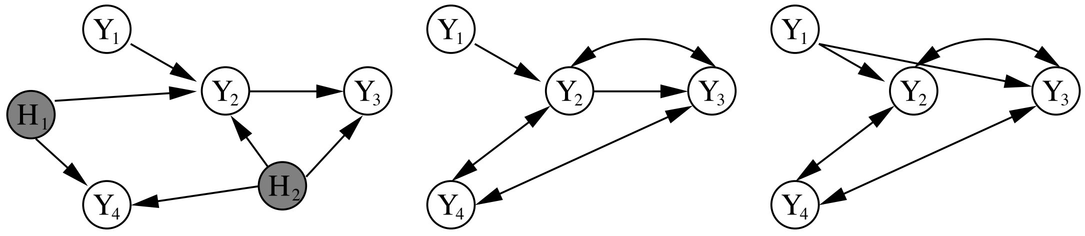


Figure 3: After marginalizing variables H_1 and H_2 from the DAG on the left, one possible DMG representation of the same dependencies is shown by the graph in the middle. Notice that there are multiple DMGs within a same Markov equivalence class, that is, encoding the same set of conditional independencies (Richardson and Spirtes, 2002). The two last graphs above are on the same class.

criterion known as m-separation, a natural extension of the d-separation criterion used for directed acyclic graphs (Richardson, 2003).

In a ADMG, two adjacent nodes might be connected by up to two edges, where in this case one has to be bi-directed and the other directed. A cyclic model can in principle allow for two directed edges of opposite directions. Figure 2 provides a few examples of DMGs. The appeal of this graphical family lies on the representation of the marginal independence structure among a set of observed variables, assuming they are part of a larger DAG structure that includes hidden variables. This is illustrated in Figure 3.[1] More details on DMGs are given in Sections 2 and 8. In our blood pressure\liver status multiple regression problem, the suitable directed mixed graph is depicted in Figure 1(c).

The contribution of this paper is how to perform Bayesian inference on two different families of mixed graph models: Gaussian and probit. Markov chain Monte Carlo (MCMC) and variational approximations will be discussed. Current Bayesian inference approaches for DMG models have limitations, as discussed in Section 2, despite the fact that such models are widely used in several sciences.

The rest of the paper is organized as follows. Section 3 describes a special case of Gaussian mixed graph models, where only bi-directed edges are allowed. Priors and a Monte Carlo algorithm are described. This case will be a building block for subsequent sections, such as Section 4, where

1. Notice that it is not necessarily the case that the probability model itself is closed under marginalization. This will happen to some models, including the Gaussian model treated in this paper. But the basic claim of closure concerns the graph, that is, the representation of independence constraints.

Gaussian DMG models are treated. Section 5 covers a type of discrete distribution for binary and ordinal data that is Markov with respect to an acyclic DMG. In Section 6 we discuss more sophisticated algorithms that are useful for scaling up Bayesian learning to higher-dimensional problems. Section 7 presents several empirical studies. Since the use of mixed graph models in machine learning applications is still in its early stages, we briefly describe in Section 8 a variety of possible uses of such graphs in machine learning applications.

2. Basics of DMGs, Gaussian Models and Related Work

In this section, we describe the Gaussian DMG model and how it complements latent variable models. At the end of the section, we also discuss a few alternative approaches for the Bayesian inference problem introduced in this paper.

2.1 Notation and Terminology

In what follows, we will use standard notions from the graphical modeling literature, such as vertex (node), edge, parent, child, ancestor, descendant, DAG, undirected graph, induced subgraph, Markov condition and d-separation. Refer to Pearl (1988) and Lauritzen (1996) for the standard definitions if needed. Less standard definitions will be given explicitly when appropriate. A useful notion is that of m-separation (Richardson, 2003) for reading off which independencies are entailed by a DMG representation. This can be reduced to d-separation (Pearl, 1988) by the following trick: for each bi-directed edge $Y_i \leftrightarrow Y_j$, introduce a new hidden variable X_{ij} and the edges $X_{ij} \rightarrow Y_i$ and $X_{ij} \rightarrow Y_j$. Remove then all bi-directed edges and apply d-separation to the resulting directed graph.

As usual, we will refer to vertices (nodes) in a graph and the corresponding random variables in a distribution interchangeably. Data points are represented by vectors with an upper index, such as $\mathbf{Y}^{(1)}, \mathbf{Y}^{(2)}, \ldots, \mathbf{Y}^{(n)}$. The variable corresponding to node Y_i in data point $\mathbf{Y}^{(j)}$ is represented by $Y_i^{(j)}$.

2.2 Gaussian Parameterization

The origins of mixed graph models can be traced back to Sewall Wright (Wright, 1921), who used special cases of mixed graph representations in genetic studies. Generalizing Wright's approach, many scientific fields such as psychology, social sciences and econometrics use linear mixed graph models under the name of *structural equation models* (Bollen, 1989). Only recently the graphical and parametrical aspects of mixed graph models have been given a thorough theoretical treatment (Richardson and Spirtes, 2002; Richardson, 2003; Kang and Tian, 2005; Drton and Richardson, 2008a). In practice, many structural equation models today are Gaussian models. We will work under this assumption unless stated otherwise.

For a DMG $\mathcal{G}$ with a set of vertices $\mathbf{Y}$, a standard parameterization of the Gaussian model is given as follows. For each variable Y_i with a (possibly empty) parent set $\{Y_{i1}, ..., Y_{ik}\}$, we define a "structural equation"

$$Y_i = \alpha_i + b_{i1}Y_{i1} + b_{i2}Y_{i2} + \cdots + b_{ik}Y_{ik} + \varepsilon_i$$

where ε_i is a Gaussian random variable with zero mean and variance v_{ii}. Notice that this parameterization allows for cyclic models.

Unlike in standard Gaussian DAG models, the error terms $\{\varepsilon_i\}$ are not necessarily mutually independent. Independence is asserted by the graphical structure: given two vertices Y_i and Y_j, the respective error terms ε_i and ε_j are marginally independent if Y_i and Y_j are not connected by a bi-directed edge.

By this parameterization, each directed edge $Y_i \leftarrow Y_j$ in the graph corresponds to a parameter b_{ij}. Each bi-directed edge $Y_i \leftrightarrow Y_j$ in the graph is associated with a covariance parameter v_{ij}, the covariance of ε_i and ε_j. Each vertex Y_j in the graph is associated with variance parameter v_{jj}, the variance of ε_j. Algebraically, let $\mathbf{B}$ be a $m \times m$ matrix, m being the number of observed variables. This matrix is such that $(\mathbf{B})_{ij} = b_{ij}$ if $Y_i \leftarrow Y_j$ exists in the graph, and 0 otherwise. Let $\mathbf{V}$ be a $m \times m$ matrix, where $(\mathbf{V})_{ij} = v_{ij}$ if $i = j$ or if $Y_i \leftrightarrow Y_j$ is in the graph, and 0 otherwise. Let $\mathbf{Y}$ be the column vector of observed variables, α the column vector of intercept parameters, and ε be the corresponding vector of error terms. The set of structural equations can be given in matrix form as

$$\begin{aligned} & \mathbf{Y} = \mathbf{BY} + \alpha + \varepsilon \Rightarrow \mathbf{Y} = (\mathbf{I} - \mathbf{B})^{-1}(\varepsilon + \alpha) \\ \Rightarrow \quad & \Sigma(\Theta) = (\mathbf{I} - \mathbf{B})^{-1}\mathbf{V}(\mathbf{I} - \mathbf{B})^{-\mathsf{T}} \end{aligned} \tag{1}$$

where $\mathbf{A}^{-\mathsf{T}}$ is the transpose of $\mathbf{A}^{-1}$ and $\Sigma(\Theta)$ is the *implied covariance matrix* of the model, $\Theta \equiv \{\mathbf{B}, \mathbf{V}, \alpha\}$.

2.2.1 COMPLETENESS OF PARAMETERIZATION AND ANCESTRAL GRAPHS

An important class of ADMGs is the directed ancestral graph. Richardson and Spirtes (2002) provide the definition and a thorough account of the Markov properties of ancestral graphs. One of the reasons for the name "ancestral graph" is due to one of its main properties: if there is a directed path $Y_i \rightarrow \cdots \rightarrow Y_j$, that is, if Y_i is an ancestor of Y_j, then there is no bi-directed edge $Y_i \leftrightarrow Y_j$. Thus directed ancestral graphs are ADMGs with this constraint.[2]

In particular, they show that any Gaussian distribution that is Markov with respect to a given ADMG can be represented by some Gaussian ancestral graph model that is parameterized as above. For the ancestral graph family, the given parameterization is *complete*: that is, for each Markov equivalence class, it is always possible to choose an ancestral graph where the resulting parameterization imposes no further constraints on the distribution besides the independence constraints of the class. Since the methods described in this paper apply to general DMG models, they also apply to directed ancestral graphs.

In principle, it is possible to define and parameterize a Gaussian DAG model that entails exactly the same independence constraints encoded in an directed ancestral graph. One possibility, as hinted in the previous Section, is to replace each bi-directed edge $Y_i \leftrightarrow Y_j$ by a new path $Y_i \leftarrow X_{ij} \rightarrow Y_j$. Variables $\{X_{ij}\}$ are "ancillary" hidden variables, in the sense that they are introduced for the sake of obtaining the same independence constraints of an ancestral graph. Standard Bayesian methodology can then be applied to perform inference in this Gaussian DAG model.

However, this parameterization might have undesirable consequences, as discussed in Section 8.6 of Richardson and Spirtes (2002). Moreover, when Markov chain Monte Carlo algorithms are applied to compute posteriors, the "ancillary" hidden variables will have to be integrated out numerically. The resulting Markov chain can suffer from substantial autocorrelation when compared to a model with no ancillary variables. We illustrate this behavior in Section 7.

Further constraints beyond independence constraints are certainly desirable depending on the context. For instance, general ADMGs that are not ancestral graphs may impose other constraints (Richardson and Spirtes, 2002), and such graphs can still be sensible models of, for example, the

2. Notice this rules out the possibility of having both edges $Y_i \rightarrow Y_j$ and $Y_i \leftrightarrow Y_j$ in the same ancestral graph.

causal processes for the problem at hand. When many observed variables are confounded by a same hidden common cause, models based on factor analysis are appropriate (Silva et al., 2006). However, it is useful to be able to build upon independence models that are known to have a complete parameterization. In any case, even the latent variables in any model might have dependencies that arise from other latent variables that were marginalized, and a latent variable ADMG model will be necessary. When it comes to solving a problem, it is up to the modeler (or learning algorithm) to decide if some set of latent variables should be included, or if they should be implicit, living their hidden life through the marginals.

Richardson and Spirtes (2002) provide further details on the advantages of a complete parameterization. Drton and Richardson (2004) provide an algorithm for fitting Gaussian ancestral graph models by maximum likelihood.

2.3 Bayesian Inference

The literature on Bayesian structural equation models is extensive. Scheines et al. (1999) describe one of the first approaches, including ways of testings such models. Lee (2007) provides details on many recent advances. Standard Bayesian approaches for Gaussian DMG models rely on either attempting to reduce the problem to inference with DAG models, or on using rejection sampling.

In an application described by Dunson et al. (2005), the "ancillary latent" trick is employed, and Gibbs sampling for Gaussian DAG models is used. This parameterization has the disadvantages mentioned in the previous section. Scheines et al. (1999) use the complete parameterization, with a single parameter corresponding to each bi-directed edge. However, the global constraint of positive-definiteness in the covariance matrix is enforced only by rejection sampling, which might be inefficient in models with moderate covariance values. The prior is setup in an indirect way. A Gaussian density function is independently defined for each error covariance v_{ij}. The actual prior, however, is the result of multiplying all of such functions and the indicator function that discards non-positive definite matrices, which is then renormalized.

In contrast, the Bayesian approach delineated in the next sections uses the complete parameterization, does not appeal to rejection sampling, makes use of a family of priors which we believe is the natural choice for the problem, and leads to convenient ways of computing marginal likelihoods for model selection. We will also see that empirically they lead to much better behaved Markov chain Monte Carlo samplers when compared to DAGs with ancillary latent variables.

3. Gaussian Models of Marginal Independence

This section concerns priors and sampling algorithms for zero-mean Gaussian models that are Markov with respect to a bi-directed graph, that is, a DMG with no directed edges. Focusing on bi-directed graphs simplifies the presentation, while providing a convenient starting point to solve the full DMG case in the sequel.

Concerning the notation: the distribution we introduce in this section is a distribution over covariance matrices. In the interest of generality, we will refer to the random matrix as Σ. In the context of the previous section, $\Sigma \equiv \Sigma(\Theta) = \mathbf{V}$, since we are assuming $\mathbf{B} = 0, \alpha = 0$.

3.1 Priors

Gaussian bi-directed graph models are sometimes called *covariance graph models*. Covariance graphs are models of marginal independence: each edge corresponds to a single parameter in the covariance matrix (the corresponding covariance); the absence of an edge $Y_i \leftrightarrow Y_j$ is a statement that $\sigma_{Y_iY_j} = 0$, σ_{XY} being the covariance of random variables X and Y. More precisely, if Σ is a random covariance matrix generated by a covariance model, a distribution of Σ is the distribution over the (non-repeated) entries corresponding to variances and covariances of adjacent nodes.[3]

In a model with a fully connected bi-directed graph, this reduces to a space of unrestricted covariance matrices. A common distribution for covariance matrices is the inverse Wishart $IW(\delta, \mathbf{U})$. In this paper, we adopt the following inverse Wishart parameterization:

$$p(\Sigma) \propto |\Sigma|^{-(\delta+2m)/2} \exp\left\{-\frac{1}{2}tr(\Sigma^{-1}\mathbf{U})\right\}, \Sigma \text{ positive definite,}$$

$p(\cdot)$ being the density function, $tr(\cdot)$ the trace function, and m the number of variables (nodes) in our model.[4] We will overload the symbol $p(\cdot)$ wherever it is clear from the context which density function we are referring to. It is assumed that $\delta > 0$ and $\mathbf{U}$ is positive definite.

Following Atay-Kayis and Massam (2005), let $M^+(\mathcal{G})$ be the cone of positive definite matrices such that, for a given bi-directed graph $\mathcal{G}$ and $\Sigma \in M^+(\mathcal{G})$, $\sigma_{ij} = 0$ if nodes Y_i and Y_j are not adjacent in $\mathcal{G}$. It is convenient to choose a distribution that is conjugate to the Gaussian likelihood function, since one can use the same algorithms for performing inference both in the prior and posterior. In a zero-mean Gaussian model, the likelihood function for a fixed data set $\mathcal{D} = \{\mathbf{Y}^{(1)}, \mathbf{Y}^{(2)}, \dots, \mathbf{Y}^{(n)}\}$ is defined by the sufficient statistic $\mathbf{S} = \sum_{d=1}^{n}(\mathbf{Y}^{(d)})(\mathbf{Y}^{(d)})^{\mathsf{T}}$ as follows:

$$\mathcal{L}(\Sigma; \mathcal{D}) = (2\pi)^{-nm/2}|\Sigma|^{-n/2} \exp\left\{-\frac{1}{2}tr(\Sigma^{-1}\mathbf{S})\right\}. \tag{2}$$

We extend the inverse Wishart distribution to the case of constrained covariance matrices in order to preserve conjugacy. This define the following distribution:

$$p(\Sigma) = \frac{1}{I_{\mathcal{G}}(\delta, \mathbf{U})}|\Sigma|^{-(\delta+2m)/2} \exp\left\{-\frac{1}{2}tr(\Sigma^{-1}\mathbf{U})\right\}, \Sigma \in M^+(\mathcal{G}) \tag{3}$$

which is basically a re-scaled inverse Wishart prior with a different support and, consequently, different normalizing constant $I_{\mathcal{G}}(\delta, \mathbf{U})$. An analogous concept exists for undirected graphs, where $\Sigma^{-1} \in M^+(\mathcal{G})$ is given a Wishart-like prior: the "$\mathcal{G}$-Wishart" distribution (Atay-Kayis and Massam, 2005). We call the distribution with density function defined as in Equation (3) the *$\mathcal{G}$-Inverse Wishart* distribution ($\mathcal{G}$-*IW*). It will be the basis of our framework. There are no analytical formulas for the normalizing constant.

3. As such, the density function for Σ is defined with respect to the Lebesgue measure of the non-zero, independent elements of this matrix.
4. We adopt this non-standard parameterization of the inverse Wishart because it provides a more convenient reparameterization used in the sequel. Notice this is the parameterization used by Brown et al. (1993) and Atay-Kayis and Massam (2005), which developed other distributions for covariance matrices.

3.2 The Normalizing Constant

We now derive a Monte Carlo procedure to compute $I_{\mathcal{G}}(\delta, \mathbf{U})$. In the sequel, this will be adapted into an importance sampler to compute functionals of a $\mathcal{G}$-*IW* distribution. The core ideas are also used in a Gibbs sampler to obtain samples from its posterior.

The normalizing constant is essential for model selection of covariance graphs. By combining the likelihood equation (2) with the prior (3), we obtain the joint

$$p(\mathcal{D}, \Sigma \mid \mathcal{G}) = (2\pi)^{-\frac{nm}{2}} I_{\mathcal{G}}(\delta, \mathbf{U})^{-1} \times |\Sigma|^{-\frac{\delta+2m+n}{2}} \exp\left\{-\frac{1}{2} tr[\Sigma^{-1}(\mathbf{S}+\mathbf{U})]\right\}$$

where we make the dependency on the graphical structure $\mathcal{G}$ explicit. By the definition of $I_{\mathcal{G}}$, integrating Σ out of the above equation implies the following marginal likelihood:

$$p(\mathcal{D} \mid \mathcal{G}) = \frac{1}{(2\pi)^{\frac{nm}{2}}} \frac{I_{\mathcal{G}}(\delta + n, \mathbf{S}+\mathbf{U})}{I_{\mathcal{G}}(\delta, \mathbf{U})}$$

from which a posterior $\mathcal{P}(\mathcal{G} \mid \mathcal{D})$ can be easily derived as a function of quantities of the type $I_{\mathcal{G}}(\cdot, \cdot)$.

The normalizing constant $I_{\mathcal{G}}(\delta, \mathbf{U})$ is given by the following integral:[5]

$$I_{\mathcal{G}}(\delta, \mathbf{U}) = \int_{M^+(\mathcal{G})} |\Sigma|^{-\frac{\delta+2m}{2}} \exp\left\{-\frac{1}{2} tr(\Sigma^{-1}\mathbf{U})\right\} d\Sigma. \tag{4}$$

The space $M^+(\mathcal{G})$ can be described as the space of positive definite matrices conditioned on the event that each matrix has zero entries corresponding to non-adjacent nodes in graph $\mathcal{G}$. We will reduce the integral (4) to an integral over random variables we know how to sample from. The given approach follows the framework of Atay-Kayis and Massam (2005) using the techniques of Drton and Richardson (2003).

Atay-Kayis and Massam (2005) show how to compute the marginal likelihood of non-decomposable undirected models by reparameterizing the precision matrix through the Cholesky decomposition. The zero entries in the inverse covariance matrix of this model correspond to constraints in this parameterization, where part of the parameters can be sampled independently and the remaining parameters calculated from the independent ones.

We will follow a similar framework but with a different decomposition. It turns out that the Cholesky decomposition does not provide an easy reduction of (4) to an integral over canonical, easy to sample from, distributions. We can, however, use Bartlett's decomposition to achieve this reduction.

3.2.1 Bartlett's Decomposition

Before proceeding, we will need a special notation for describing sets of indices and submatrices.

Let $\{i\}$ represent the set of indices $\{1, 2, \ldots, i\}$. Let $\Sigma_{i,\{i-1\}}$ be the row vector containing the covariance between Y_i and all elements of $\{Y_1, Y_2, \ldots, Y_{i-1}\}$. Let $\Sigma_{\{i-1\},\{i-1\}}$ be the marginal covariance matrix of $\{Y_1, Y_2, \ldots, Y_{i-1}\}$. Let σ_{ii} be the variance of Y_i. Define the mapping

$$\Sigma \rightarrow \Phi \equiv \{\gamma_1, \mathcal{B}_2, \gamma_2, \mathcal{B}_3, \gamma_3, \ldots, \mathcal{B}_m, \gamma_m\},$$

5. Notice this integral is always finite for any choice of $\delta > 0$ and positive definite $\mathbf{U}$, since it is no greater than the normalizing constant of the inverse Wishart.

such that $\mathcal{B}_i$ is a row vector with $i-1$ entries, γ_i is a scalar, and

$$\begin{array}{rcll} \gamma_1 & = & \sigma_{11}, & \\ \mathcal{B}_i & = & \Sigma_{i,\{i-1\}}\Sigma^{-1}_{\{i-1\},\{i-1\}}, & i>1, \\ \gamma_i & = & \sigma_{ii.\{i-1\},\{i-1\}} \equiv \sigma_{ii} - \Sigma_{i,\{i-1\}}\Sigma^{-1}_{\{i-1\},\{i-1\}}\Sigma_{\{i-1\},i}, & i>1. \end{array} \tag{5}$$

The set Φ provides a parameterization of Σ, in the sense that the mapping (5) is bijective. Given that $\sigma_{11} = \gamma_1$, the inverse mapping is defined recursively by

$$\begin{array}{rcl} \Sigma_{i,\{i-1\}} & = & \mathcal{B}_i\Sigma_{\{i-1\},\{i-1\}},\ i>1, \\ \sigma_{ii} & = & \gamma_i + \mathcal{B}_i\Sigma_{\{i-1\},i},\ i>1. \end{array} \tag{6}$$

We call the set $\Phi \equiv \{\gamma_1, \mathcal{B}_2, \gamma_2, \mathcal{B}_3, \gamma_3, \dots, \mathcal{B}_m, \gamma_m\}$ the *Bartlett parameters* of Σ, since the decomposition (6) is sometimes known as Bartlett's decomposition (Brown et al., 1993).

For a random inverse Wishart matrix, Bartlett's decomposition allows the definition of its density function by the joint density of $\{\gamma_1, \mathcal{B}_2, \gamma_2, \mathcal{B}_3, \gamma_3, \dots, \mathcal{B}_m, \gamma_m\}$. Define $\mathbf{U}_{\{i-1\},\{i-1\}}$, $\mathbf{U}_{\{i-1\},i}$ and $u_{ii.\{i-1\},\{i-1\}}$ in a way analogous to the Σ definitions. The next lemma follows directly from Lemma 1 of Brown et al. (1993):

Lemma 1 *Suppose Σ is distributed as $IW(\delta, \mathbf{U})$. Then the distribution of the corresponding Bartlett parameters $\Phi \equiv \{\gamma_1, \mathcal{B}_2, \gamma_2, \mathcal{B}_3, \gamma_3, \dots, \mathcal{B}_m, \gamma_m\}$ is given by:*

1. *γ_i is independent of $\Phi \backslash \{\gamma_i, \mathcal{B}_i\}$*
2. *$\gamma_i \sim IG((\delta+i-1)/2, u_{ii.\{i-1,i-1\}}/2)$, where $IG(\alpha,\beta)$ is the inverse gamma distribution*
3. *$\mathcal{B}_i \mid \gamma_i \sim N(\mathbf{U}^{-1}_{\{i-1\},\{i-1\}}\mathbf{U}_{\{i-1\},i}, \gamma_i\mathbf{U}^{-1}_{\{i-1\},\{i-1\}})$, where $N(\mathbf{M},\mathbf{C})$ is a multivariate Gaussian distribution and $\mathbf{U}^{-1}_{\{i-1\},\{i-1\}} \equiv (\mathbf{U}_{\{i-1\},\{i-1\}})^{-1}$.*

3.2.2 BARTLETT'S DECOMPOSITION OF MARGINAL INDEPENDENCE MODELS

What is interesting about Bartlett's decomposition is that it provides a simple parameterization of the inverse Wishart distribution with variation independent parameters. This decomposition allows the derivation of new distributions. For instance, Brown et al. (1993) derive a "Generalized Inverted Wishart" distribution that allows one to define different degrees of freedom for different submatrices of an inverse Wishart random matrix. For our purposes, Bartlett's decomposition can be used to reparameterize the $\mathcal{G}$-*IW* distribution. For that, one needs to express the independent elements of Σ in the space of Bartlett parameters.

The original reparameterization maps Σ to $\Phi \equiv \{\gamma_1, \mathcal{B}_2, \gamma_2, \mathcal{B}_3, \gamma_3, \dots, \mathcal{B}_d, \gamma_d\}$. To impose the constraint that Y_i and Y_j are uncorrelated, for $i > j$, is to set $\left(\mathcal{B}_i\Sigma_{\{i-1\},\{i-1\}}\right)_j = \sigma_{Y_iY_j}(\Phi) = 0$. For a fixed $\Sigma_{\{i-1\},\{i-1\}}$, this implies a constraint on $(\mathcal{B}_i)_j \equiv \beta_{ij}$.

Following the terminology used by Richardson and Spirtes (2002), let a *spouse* of node Y in a mixed graph be any node adjacent to Y by a bi-directed edge. The set of spouses of Y_i is denoted by $sp(i)$. The set of spouses of Y_i *according to order* $Y_1, Y_2, \dots, Y_m$ is defined by $sp_\prec(i) \equiv sp(i) \cap \{Y_1, \dots, Y_{i-1}\}$. The set of non-spouses of Y_i is denoted by $nsp(i)$. Analogously, $nsp_\prec(i) \equiv \{Y_1, \dots, Y_{i-1}\} \backslash sp_\prec(i)$. Let $\mathcal{B}_{i,sp_\prec(i)}$ be the subvector of $\mathcal{B}_i$ corresponding to the the respective spouses of Y_i. Define $\mathcal{B}_{i,nsp_\prec(i)}$ analogously.

Given the constraint $\mathcal{B}_i \Sigma_{\{i-1\},nsp_{\prec}(i)} = 0$, it follows that

$$\mathcal{B}_{i,sp_{\prec}(i)}\Sigma_{sp_{\prec}(i),nsp_{\prec}(i)} + \mathcal{B}_{i,nsp_{\prec}(i)}\Sigma_{nsp_{\prec}(i),nsp_{\prec}(i)} = 0 \Rightarrow$$

$$\mathcal{B}_{i,nsp_{\prec}(i)} = -\mathcal{B}_{i,sp_{\prec}(i)}\Sigma_{sp_{\prec}(i),nsp_{\prec}(i)}\Sigma^{-1}_{nsp_{\prec}(i),nsp_{\prec}(i)}. \tag{7}$$

Identity (7) was originally derived by Drton and Richardson (2003). A property inherited from the original decomposition for unconstrained matrices is that $\mathcal{B}_{i,sp_{\prec}(i)}$ is functionally independent of $\Sigma_{\{i-1\},\{i-1\}}$. From (7), we obtain that the free Bartlett parameters of Σ are $\Phi_{\mathcal{G}} \equiv \{\gamma_1, \mathcal{B}_{2,sp_{\prec}(2)}, \gamma_2, \mathcal{B}_{3,sp_{\prec}(3)}, \gamma_3, \dots, \mathcal{B}_{m,sp_{\prec}(m)}, \gamma_m\}$.

Notice that, according to (5), Φ corresponds to the set of parameters of a fully connected, zero-mean, Gaussian DAG model. In such a DAG, Y_i is a child of $\{Y_1, \dots, Y_{i-1}\}$, and

$$Y_i = \mathcal{B}_i \mathbf{Y}_{i-1} + \zeta_j, \qquad \zeta_j \sim N(0, \gamma_j)$$

where $\mathbf{Y}_{i-1}$ is the $(i-1)\times 1$ vector corresponding to $\{Y_1, \dots, Y_{i-1}\}$.

As discussed by Drton and Richardson (2003), this interpretation along with Equation (7) implies

$$Y_i = \mathcal{B}_{i,sp_{\prec}(i)}\mathbf{Z}_i + \zeta_j \tag{8}$$

where the entries in $\mathbf{Z}_i$ are the corresponding residuals of the regression of $sp_{\prec}(i)$ on $nsp_{\prec}(i)$.

The next step in solving integral (4) is to find the Jacobian $J(\Phi_{\mathcal{G}})$ of the transformation $\Sigma \rightarrow \Phi_{\mathcal{G}}$. This is given by the following Lemma:

Lemma 2 *The determinant of the Jacobian for the change of variable $\Sigma \rightarrow \Phi_{\mathcal{G}}$ is*

$$|J(\Phi_{\mathcal{G}})| = \prod_{i=2}^{m}|R_i| = \frac{1}{\prod_{i=2}^{m}|\Sigma_{nsp_{\prec(i)},nsp_{\prec(i)}}|}\prod_{i=1}^{m-1}\gamma_i^{m-i}$$

where $R_i \equiv \Sigma_{sp_{\prec(i)},sp_{\prec(i)}} - \Sigma_{sp_{\prec(i)},nsp_{\prec(i)}}\Sigma^{-1}_{nsp_{\prec(i)},nsp_{\prec(i)}}\Sigma_{nsp_{\prec(i)},sp_{\prec(i)}}$, that is, the covariance matrix of the respective residual $\mathbf{Z}_i$ (as parameterized by $\Phi_{\mathcal{G}}$). If $nsp_{\prec(i)} = \emptyset$, R_i is defined as $\Sigma_{sp_{\prec(i)},sp_{\prec(i)}}$ and $|\Sigma_{nsp_{\prec(i)},nsp_{\prec(i)}}|$ is defined as 1.

The proof of this Lemma is in Appendix C. A special case is the Jacobian of the unconstrained covariance matrix (i.e., when the graph has no missing edges):

$$|J(\Phi)| = \prod_{i=1}^{m-1}\gamma_i^{m-i}. \tag{9}$$

Now that we have the Jacobian, the distribution over Bartlett's parameters given by Lemma 1, and the identities of Drton and Richardson (2003) given in Equation (7), we have all we need to provide a Monte Carlo algorithm to compute the normalizing constant of a $\mathcal{G}$-*IW* with parameters $(\delta, \mathbf{U})$.

Let $\Sigma(\Phi_{\mathcal{G}})$ be the implied covariance matrix given by our set of parameters $\Phi_{\mathcal{G}}$. We start from the integral in (4), and rewrite it as a function of $\Phi_{\mathcal{G}}$. This can be expressed by substituting Σ for $\Sigma(\Phi_{\mathcal{G}})$ and multiplying the integrand by the determinant of the Jacobian. Notice that the parameters in $\Sigma(\Phi_{\mathcal{G}})$ are variation independent: that is, their joint range is given by the product of their individual ranges (positive reals for the γ variables and the real line for the β coefficients). This range will replace the original $M^+(\mathcal{G})$ space, which we omit below for simplicity of notation:

$$I_{\mathcal{G}}(\delta,\mathbf{U}) = \int |J(\Phi_{\mathcal{G}})||\Sigma(\Phi_{\mathcal{G}})|^{-\frac{\delta+2m}{2}} \exp\left\{-\frac{1}{2}tr(\Sigma(\Phi_{\mathcal{G}})^{-1}\mathbf{U})\right\} d\Phi_{\mathcal{G}}.$$

We now multiply and divide the above expression by the normalizing constant of an inverse Wishart $(\delta,\mathbf{U})$, which we denote by $I_{IW}(\delta,\mathbf{U})$:

$$I_{\mathcal{G}}(\delta,\mathbf{U}) = I_{IW}(\delta,\mathbf{U}) \int |J(\Phi_{\mathcal{G}})| \times I_{IW}^{-1}(\delta,\mathbf{U})|\Sigma(\Phi_{\mathcal{G}})|^{-\frac{\delta+2m}{2}} \exp\left\{-\frac{1}{2}tr(\Sigma(\Phi_{\mathcal{G}})^{-1}\mathbf{U})\right\} d\Phi_{\mathcal{G}}. \quad (10)$$

The expression

$$I_{IW}^{-1}(\delta,\mathbf{U})|\Sigma|^{-\frac{\delta+2m}{2}} \exp\left\{-\frac{1}{2}tr(\Sigma^{-1}\mathbf{U})\right\}$$

corresponds to the density function of an inverse Wishart Σ. Lemma 1 allows us to rewrite the inverse Wishart density function as the density of Bartlett parameters, but this is assuming no independence constraints. We can easily reuse the result of Lemma 1 as follows:

1. write the density of the inverse Wishart as the product of gamma-normal densities given in Lemma 1;

2. this expression contains the original Jacobian determinant $|J(\Phi)|$. We have to remove it, since we are plugging in our own Jacobian determinant. Hence, we divide the reparameterized density by the expression in Equation (9).

   This ratio $|J(\Phi_{\mathcal{G}})|/|J(\Phi)|$ can be rewritten as

   $$\frac{|J(\Phi_{\mathcal{G}})|}{|J(\Phi)|} = \prod_{i=1}^{m} \frac{|R_i|}{\gamma_i^{m-i}} = \frac{1}{\prod_{i=2}^{m} |\Sigma_{nsp_{\prec(i)},nsp_{\prec(i)}}|}$$

   where $|\Sigma_{nsp_{\prec(i)},nsp_{\prec(i)}}| \equiv 1$ if $nsp_{\prec}(i) = \emptyset$;

3. substitute each vector $\mathcal{B}_{i,nsp_{\prec}(i)}$, which is not a free parameter, by the corresponding expression $-\mathcal{B}_{i,sp_{\prec}(i)}\Sigma_{sp_{\prec}(i),nsp_{\prec}(i)}\Sigma^{-1}_{nsp_{\prec}(i),nsp_{\prec}(i)}$.

This substitution takes place into the original factors given by Bartlett's decomposition, as introduced in Lemma 1:

$$\begin{array}{rcl} p(\mathcal{B}_i,\gamma_i) & = & (2\pi)^{-(i-1)/2}\gamma_i^{-(i-1)/2}|\mathbf{U}_{\{i-1\},\{i-1\}}|^{1/2} \\ & \times & \exp\left(-\frac{1}{2\gamma_i}(\mathcal{B}_i^{\mathsf{T}} - \mathbf{M}_i)^{\mathsf{T}}\mathbf{U}_{\{i-1\},\{i-1\}}(\mathcal{B}_i^{\mathsf{T}} - \mathbf{M}_i)\right) \\ & \times & \dfrac{(u_{ii.\{i-1\},\{i-1\}}/2)^{(\delta+i-1)/2}}{\Gamma((\delta+i-1)/2)}\gamma_i^{-\left(\frac{\delta+i-1}{2}+1\right)} \exp\left(-\frac{1}{2\gamma_i}u_{ii.\{i-1\},\{i-1\}}\right) \end{array} \quad (11)$$

where $\mathbf{M}_i \equiv \mathbf{U}^{-1}_{\{i-1\},\{i-1\}}\mathbf{U}_{\{i-1\},i}$. Plugging in this in (10) results in

$$I_{\mathcal{G}}(\delta, \mathbf{U}) = I_{IW}(\delta, \mathbf{U}) \int \frac{1}{\prod_{i=2}^{m} |\Sigma_{nsp_{\prec}(i), nsp_{\prec}(i)}|} \times p(\gamma_1) \prod_{i=2}^{m} p(\mathcal{B}_i, \gamma_i)\, d\Phi_{\mathcal{G}}.$$

However, after substitution, each factor $p(\mathcal{B}_i, \gamma_i)$ is not in general a density function for $\{\mathcal{B}_{i,sp_{\prec}(i)}, \gamma_i\}$ and will include also parameters $\{\mathcal{B}_{j,sp_{\prec}(j)}, \gamma_j\}, j < i$. Because of the non-linear relationships that link Bartlett parameters in a marginal independence model, we cannot expect to reduce this expression to a tractable distribution we can easily sample from. Instead, we rewrite each original density factor $p(\mathcal{B}_i, \gamma_i)$ such that it includes all information about $\mathcal{B}_{i,sp_{\prec}(i)}$ and γ_i within a canonical density function. That is, factorize $p(\mathcal{B}_i, \gamma_i)$ as

$$p(\mathcal{B}_i, \gamma_i | \Phi_{i-1}) = p_b(\mathcal{B}_{i,sp_{\prec}(i)} | \gamma_i, \Phi_{i-1}) p_g(\gamma_i | \Phi_{i-1}) \times f_i(\Phi_{i-1}) \tag{12}$$

where we absorb any occurrence of $\mathcal{B}_{i,sp_{\prec}(i)}$ within the sampling distribution and factorize the remaining dependence on previous parameters $\Phi_{i-1} \equiv \{\gamma_1, \gamma_2, \mathcal{B}_{2,sp_{\prec}(2)}, \dots, \gamma_{i-1}, \mathcal{B}_{i-1,sp_{\prec}(i-1)}\}$ into a separate function.[6] We derive the functions $p_b(\cdot), p_g(\cdot)$ and $f_i(\cdot)$ in Appendix A. The result is as follows.

The density $p_b(\mathcal{B}_{i,sp_{\prec}(i)} | \gamma_i, \Phi_{i-1})$ is the density of a Gaussian $N(\mathbf{K}_i \mathbf{m}_i, \gamma_i \mathbf{K}_i)$ such that

$$\begin{aligned}
\mathbf{m}_i &= (\mathbf{U}_{ss} - \mathbf{A}_i \mathbf{U}_{ns}) \mathbf{M}_{sp_{\prec}(i)} + (\mathbf{U}_{sn} - \mathbf{A}_i \mathbf{U}_{nn}) \mathbf{M}_{nsp_{\prec}(i)}, \\
\mathbf{K}_i^{-1} &= \mathbf{U}_{ss} - \mathbf{A}_i \mathbf{U}_{ns} - \mathbf{U}_{sn} \mathbf{A}_i^{\mathsf{T}} + \mathbf{A}_i \mathbf{U}_{nn} \mathbf{A}_i^{\mathsf{T}}, \\
\mathbf{A}_i &= \Sigma_{sp_{\prec}(i), nsp_{\prec}(i)} \Sigma^{-1}_{nsp_{\prec}(i), nsp_{\prec}(i)}
\end{aligned} \tag{13}$$

where

$$\begin{bmatrix} \mathbf{U}_{ss} & \mathbf{U}_{sn} \\ \mathbf{U}_{ns} & \mathbf{U}_{nn} \end{bmatrix} \equiv \begin{bmatrix} \mathbf{U}_{sp_{\prec}(i), sp_{\prec}(i)} & \mathbf{U}_{sp_{\prec}(i), nsp_{\prec}(i)} \\ \mathbf{U}_{nsp_{\prec}(i), sp_{\prec}(i)} & \mathbf{U}_{nsp_{\prec}(i), nsp_{\prec}(i)} \end{bmatrix}. \tag{14}$$

The density $p_g(\gamma_i | \Phi_{i-1})$ is the density of an inverse gamma $IG(g_1, g_2)$ such that

$$\begin{aligned}
g_1 &= \frac{\delta + i - 1 + \#nsp_{\prec}(i)}{2}, \\
g_2 &= \frac{u_{ii.\{i-1\},\{i-1\}} + \mathcal{U}_i}{2}, \\
\mathcal{U}_i &= \mathbf{M}_i^{\mathsf{T}} \mathbf{U}_{\{i-1\},\{i-1\}} \mathbf{M}_i - \mathbf{m}_i^{\mathsf{T}} \mathbf{K}_i \mathbf{m}_i.
\end{aligned}$$

where $u_{ii.\{i-1\},\{i-1\}}$ was originally defined in Section 3.2.1.

Finally,

$$\begin{aligned}
f_i(\Phi_{i-1}) \equiv\ & (2\pi)^{-\frac{(i-1)-\#sp_{\prec}(i)}{2}} |\mathbf{K}_i|^{1/2} |\mathbf{U}_{\{i-1\},\{i-1\}}|^{1/2} \\
\times\ & \frac{(u_{ii.\{i-1\},\{i-1\}}/2)^{(\delta+i-1)/2}}{\Gamma((\delta+i-1)/2)} \frac{\Gamma((\delta+i-1+\#nsp_{\prec}(i))/2)}{((u_{ii.\{i-1\},\{i-1\}} + \mathcal{U}_i)/2)^{(\delta+i-1+\#nsp_{\prec}(i))/2}}.
\end{aligned}$$

6. A simpler decomposition was employed by Silva and Ghahramani (2006) (notice however that paper used an incorrect expression for the Jacobian). The following derivation, however, can be adapted with almost no modification to define a Gibbs sampling algorithm, as we show in the sequel.

Density function $p_b(\mathcal{B}_{i,sp_{\prec}(i)}|\cdot,\cdot)$ and determinant $|\mathbf{K}_i|^{1/2}$ are defined to be 1 if $sp_{\prec}(i) = \emptyset$. $\mathcal{U}_i$ is defined to be zero if $nsp_{\prec}(i) = \emptyset$, and $\mathcal{U}_i = \mathbf{M}_i^{\mathsf{T}}\mathbf{U}_{\{i-1\},\{i-1\}}\mathbf{M}_i$ if $sp_{\prec}(i) = \emptyset$.

The original normalizing constant integral is the expected value of a function of $\Phi_{\mathcal{G}}$ over a factorized inverse gamma-normal distribution. The density function of this distribution is given below:

$$p_{I(\delta,\mathbf{U})}(\Phi_{\mathcal{G}}) = \left(\prod_{i=1}^{m} p_g(\gamma_i|\Phi_{i-1})\right)\left(\prod_{i=2}^{m} p_b(\mathcal{B}_{i,sp_{\prec}(i)}|\gamma_i,\Phi_{i-1})\right).$$

We summarize the main result of this section through the following theorem:

Theorem 3 *Let $\langle f(\mathbf{X})\rangle_{p(\mathbf{X})}$ be the expected value of $f(\mathbf{X})$ where $\mathbf{X}$ is a random vector with density $p(\mathbf{X})$. The normalizing constant of a $\mathcal{G}$-Inverse Wishart with parameters $(\delta,\mathbf{U})$ is given by*

$$I_{\mathcal{G}}(\delta,\mathbf{U}) = I_{IW}(\delta,\mathbf{U}) \times \left\langle \prod_{i=1}^{m} \frac{f_i(\Phi_{i-1})}{|\Sigma_{nsp_{\prec}(i),nsp_{\prec}(i)}|} \right\rangle_{p_{I(\delta,\mathbf{U})}(\Phi_{\mathcal{G}})}.$$

This can be further simplified to

$$I_{\mathcal{G}}(\delta,\mathbf{U}) = \left\langle \prod_{i=1}^{m} \frac{f'_i(\Phi_{i-1})}{|\Sigma_{nsp_{\prec}(i),nsp_{\prec}(i)}|} \right\rangle_{p_{I(\delta,\mathbf{U})}(\Phi_{\mathcal{G}})} \tag{15}$$

where

$$f'_i(\Phi_{i-1}) \equiv (2\pi)^{\frac{\#sp_{\prec}(i)}{2}} |\mathbf{K}_i(\Phi_{i-1})|^{1/2} \frac{\Gamma((\delta+i-1+\#nsp_{\prec}(i))/2)}{((u_{ii.\{i-1\},\{i-1\}}+\mathcal{U}_i)/2)^{(\delta+i-1+\#nsp_{\prec}(i))/2}}$$

which, as expected, reduces $I_{\mathcal{G}}(\delta,\mathbf{U})$ to $I_{IW}(\delta,\mathbf{U})$ when the graph is complete.

A Monte Carlo estimate of $I_{\mathcal{G}}(\delta,\mathbf{U})$ is then given from (15) by obtaining samples $\{\Phi_{\mathcal{G}}^{(1)},\Phi_{\mathcal{G}}^{(2)},\ldots,\Phi_{\mathcal{G}}^{(M)}\}$ according to $p_{I(\delta,\mathbf{U})}(\cdot)$ and computing:

$$I_{\mathcal{G}}(\delta,\mathbf{U}) \approx \frac{1}{M}\sum_{s=1}^{M}\prod_{i=1}^{m} \frac{f'_i(\Phi_{i-1}^{(s)})}{|\Sigma_{nsp_{\prec}(i),nsp_{\prec}(i)}(\Phi_{i-1}^{(s)})|}$$

where here we emphasize that $\Sigma_{nsp_{\prec}(i),nsp_{\prec}(i)}$ is a function of $\Phi_{\mathcal{G}}$ as given by (6).

3.3 General Monte Carlo Computation

If $\mathbf{Y}$ follows a Gaussian $N(0,\Sigma)$ where Σ is given a $\mathcal{G}$-$IW(\delta,\mathbf{U})$ prior, then from a sample $\mathcal{D} = \{\mathbf{Y}^{(1)},\ldots,\mathbf{Y}^{(n)}\}$ with sufficient statistic $\mathbf{S} = \sum_{d=1}^{n}(\mathbf{Y}^{(d)})(\mathbf{Y}^{(d)})^{\mathsf{T}}$, the posterior distribution for Σ given $\mathbf{S}$ will be a $\mathcal{G}$-$IW(\delta+n,\mathbf{U}+\mathbf{S})$. In order to obtain samples from the posterior or to compute its functionals, one can adapt the algorithm for computing normalizing constants. We describe an importance sampler for computing functionals, followed by a Gibbs sampling algorithm that also provides samples from the posterior.

```
Algorithm SAMPLEGIW-1
Input: a $m \times m$ matrix $\mathbf{U}$, scalar $\delta$, bi-directed graph $\mathcal{G}$, an ordering $\prec$

 1. Let $\Sigma$ be a $m \times m$ matrix
 2. Define functions $sp_{\prec}(\cdot), nsp_{\prec}(\cdot)$ according to $\mathcal{G}$ and ordering $\prec$
 3. Sample $\sigma_{11}$ from $IG(\delta/2, u_{11}/2)$
 4. For $i = 2, 3, \dots, m$
 5.     Sample $\gamma_i \sim IG((\delta + i - 1 + \#nsp_{\prec}(i))/2, (u_{ii.\{i-1\},\{i-1\}} + \mathcal{U}_i)/2)$
 6.     Sample $\mathcal{B}_{i,sp_{\prec}(i)} \sim N(\mathbf{K}_i\mathbf{m}_i, \gamma_i\mathbf{K}_i)$
 7.     Set $\mathcal{B}_{i,nsp_{\prec}(i)} = -\mathcal{B}_{i,sp_{\prec}(i)}\Sigma_{sp_{\prec}(i),nsp_{\prec}(i)}\Sigma^{-1}_{nsp_{\prec}(i),nsp_{\prec}(i)}$
 8.     Set $\Sigma^{\mathsf{T}}_{\{i-1\},i} = \Sigma_{i,\{i-1\}} = \mathcal{B}_i\Sigma_{\{i-1\},\{i-1\}}$
 9.     Set $\sigma_{ii} = \gamma_i + \mathcal{B}_i\Sigma_{i,\{i-1\}}$
10. Set $w = \prod_{i=1}^{m} f_i'(\Phi_{i-1})/|\Sigma_{nsp_{\prec}(i),nsp_{\prec}(i)}|$
11. Return $(w, \Sigma)$.
```

Figure 4: A procedure for generating an importance sample Σ and importance weight w for computing functionals of a $\mathcal{G}$-Inverse Wishart distribution. Variables $\{\mathbf{M}_i, \mathbf{m}_i, \mathbf{K}_i, \mathcal{U}_i\}$ and function $f_i'(\Phi_{i-1})$ are defined in Section 3.2.2.

3.3.1 THE IMPORTANCE SAMPLER

One way of computing functionals of the $\mathcal{G}$-IW distribution, that is, functions of the type

$$g(\delta, \mathbf{U}; \mathcal{G}) \equiv \int_{M^+(\mathcal{G})} g(\Sigma)p(\Sigma \mid \delta, \mathbf{U}, \mathcal{G})\, d\Sigma$$

is through the numerical average

$$g(\delta, \mathbf{U}; \mathcal{G}) \approx \frac{\sum_{s=1}^{M} w_s g(\Sigma^{(s)})}{\sum_{s=1}^{M} w_s},$$

where weights $\{w_1, w_2, \dots, w_M\}$ and samples $\{\Sigma^{(1)}, \Sigma^{(2)}, \dots, \Sigma^{(M)}\}$ are generated by an importance sampler. The procedure for computing normalizing constants can be readily adapted for this task using $p_{I(\delta,\mathbf{U})}(\cdot)$ as the importance distribution and the corresponding weights from the remainder factors. The sampling algorithm is shown in Figure 4.

3.3.2 THE GIBBS SAMPLER

While the importance sampler can be useful to compute functionals of the distribution, we will need a Markov chain Monte Carlo procedure to sample from the posterior. In the Gibbs sampling

```
Algorithm SAMPLEGIW-2
Input: a $m \times m$ matrix $\mathbf{U}$, scalar $\delta$, bi-directed graph $\mathcal{G}$, a $m \times m$ matrix $\Sigma^{start}$

 1. Let $\Sigma$ be a copy of $\Sigma^{start}$
 2. Define functions $sp(\cdot), nsp(\cdot)$ according to $\mathcal{G}$
 3. For $i = 1, 2, 3, \dots, m$
 4.     Sample $\gamma_i \sim IG((\delta + (m-1) + \#nsp(i))/2, (u_{ii.\{\backslash i\},\{\backslash i\}} + \mathcal{U}_{\backslash i})/2)$
 5.     Sample $\mathcal{B}_{i,sp(i)}$ from a $N(\mathbf{K}_{\backslash i}\mathbf{m}_{\backslash i}, \gamma_i \mathbf{K}_{\backslash i})$
 6.     Set $\mathcal{B}_{i,nsp(i)} = -\mathcal{B}_{i,sp(i)}\Sigma_{sp(i),nsp(i)}\Sigma^{-1}_{nsp(i),nsp(i)}$
 7.     Set $\Sigma^{\mathsf{T}}_{\{\backslash i\},i} = \Sigma_{i,\{\backslash i\}} = \mathcal{B}_i \Sigma_{\{\backslash i\},\{\backslash i\}}$
 8.     Set $\sigma_{ii} = \gamma_i + \mathcal{B}_i \Sigma_{i,\{\backslash i\}}$
 9. Return $\Sigma$.
```

Figure 5: A procedure for generating a sampled Σ within a Gibbs sampling procedure.

procedure, we sample the whole i-th row of Σ, for each $1 \leq i \leq m$, by conditioning on the remaining independent entries of the covariance matrix as obtained on the previous Markov chain iteration.

The conditional densities required by the Gibbs sampler can be derived from (12), which for a particular ordering $\prec$ implies

$$p(\Sigma; \delta, \mathbf{U}, \mathcal{G}) \propto p_g(\gamma_1) \prod_{i=2}^{m} p_b(\mathcal{B}_{i,sp_{\prec}(i)} | \gamma_i, \Phi_{i-1}) p_g(\gamma_i | \Phi_{i-1}) f_i(\Phi_{i-1}).$$

By an abuse of notation, we used Σ in the left-hand side and the Bartlett parameters in the righ-hand side.

The conditional density of $\{\mathcal{B}_{m,sp_{\prec}(m)}, \gamma_m\}$ given all other parameters is therefore

$$p(\mathcal{B}_{m,sp_{\prec}(m)}, \gamma_m | \Phi_{\mathcal{G}} \backslash \{\mathcal{B}_{m,sp_{\prec}(m)}, \gamma_m\}) = p_b(\mathcal{B}_{m,sp_{\prec}(m)} | \gamma_m, \Phi_{m-1}) p_g(\gamma_m | \Phi_{m-1})$$

from which we can reconstruct a new sample of the m-th row/column of Σ after sampling $\{\mathcal{B}_{m,sp_{\prec}(m)}, \gamma_m\}$. Sampling other rows can be done by redefining a new order where the corresponding target variable is the last one.

More precisely: let $\{\backslash i\}$ denote the set $\{1, 2, \dots, i-1, i+1, \dots, m\}$. The Gibbs algorithm is analogous to the previous algorithms. Instead of $sp_{\prec}(i)$ and $nsp_{\prec}(i)$, we refer to the original $sp(i)$ and $nsp(i)$. Matrices $\Sigma_{\{\backslash i\},\{\backslash i\}}$ and $\mathbf{U}_{\{\backslash i\},\{\backslash i\}}$ are defined by deleting the respective i-th row and i-th columns. Row vector $\Sigma_{i,\{\backslash i\}}$ and scalar $u_{ii.\{\backslash i\}}$ are defined accordingly, as well as any other vector and matrix originally required in the marginal likelihood/importance sampling procedure. The algorithm is described in Figure 5. The procedure can be interpreted as calling a modification of the importance sampler with a dynamic ordering $\prec_i$ which, at every step, moves Y_i to the end of the global ordering $\prec$.

3.4 Remarks

The importance sampler suffers from the usual shortcomings in high-dimensional problems, where a few very large weights dominate the procedure (MacKay, 1998). This can result in unstable estimates of functionals of the posterior and the normalizing constant.

The stability of the importance sampler is not a simple function of the number of variables in the domain. For large but sparse graphs, the number of parameters might be small. For large but fairly dense graphs, the importance distribution might be a good match to the actual distribution since there are few constraints. In Section 7, we performe some experiments to evaluate the sampler.

When used to compute functionals, the Gibbs sampler is more computationally demanding considering the cost per step, but we expect it to be more robust in high-dimensional problems. In problems that require repeated calculations of functionals (such as the variational optimization procedure of Section 4.3), it might be interesting to run a few preliminary comparisons between the estimates of the two samplers, and choose the (cheaper) importance sampler if the estimates are reasonably close.

Naïvely, the Gibbs sampler costs $O(m^4)$ per iteration, since for each step we have to invert the matrix $\Sigma_{nsp\{\backslash i\},nsp\{\backslash i\}}$, which is of size $O(m)$ for sparse graphs. However, this inversion can cost much less than $O(m^3)$ if sparse matrix inversion methods are used. Still, the importance sampler can be even more optimized by using the methods of Section 6.

4. Gaussian Directed Mixed Graph Models

As discussed in Section 2, Gaussian directed mixed graph models are parameterized by the set with parameters $\Theta = \{\mathbf{V}, \mathbf{B}, \alpha\}$. Our prior takes the form $p(\Theta) = p(\mathbf{B})p(\alpha)p(\mathbf{V})$. We assign priors for the parameters of directed edges (non-zero entries of matrix $\mathbf{B}$) in a standard way: each parameter b_{ij} is given a Gaussian $N(c^B_{ij}, s^B_{ij})$ prior, where all parameters are marginally independent in the prior, that is, $p(\mathbf{B}) = \prod_{ij} p(b_{ij})$. The prior for intercept parameters α is analogous, with α_i being a Gaussian $N(c^\alpha_i, s^\alpha_i)$.

Recall from Equation (1) that the implied covariance of the model is given by the matrix $\Sigma(\Theta) = (\mathbf{I}-\mathbf{B})^{-1}\mathbf{V}(\mathbf{I}-\mathbf{B})^{-\mathsf{T}}$. Similarly, we have the implied mean vector $\mu(\Theta) \equiv (\mathbf{I}-\mathbf{B})^{-1}\alpha$. The likelihood function for data set $\mathcal{D} = \{\mathbf{Y}^{(1)}, \mathbf{Y}^{(2)}, \dots, \mathbf{Y}^{(n)}\}$ is defined as

$$\begin{aligned}\mathcal{L}(\Theta;\mathcal{D}) &= |\Sigma(\Theta)|^{-n/2}\textstyle\prod_{d=1}^n \exp\left(-\frac{1}{2}(\mathbf{Y}^{(d)}-\mu(\Theta))^{\mathsf{T}}\Sigma(\Theta)^{-1}(\mathbf{Y}^{(d)}-\mu(\Theta))\right)\\ &= \left\{|(\mathbf{I}-\mathbf{B})^{-1}||\mathbf{V}||(\mathbf{I}-\mathbf{B})^{-\mathsf{T}}|\right\}^{-n/2}\left\{\exp\left(-\frac{1}{2}tr(\mathbf{V}^{-1}(\mathbf{I}-\mathbf{B})\mathbf{S}(\mathbf{I}-\mathbf{B})^{\mathsf{T}})\right)\right\},\end{aligned}$$

where now $\mathbf{S} \equiv \sum_{d=1}^n (\mathbf{Y}^{(d)}-\mu(\Theta))(\mathbf{Y}^{(d)}-\mu(\Theta))^{\mathsf{T}}$.

Given a prior $\mathcal{G}$-$IW(\delta, \mathbf{U})$ for $\mathbf{V}$, it immediately follows that the posterior distribution of $\mathbf{V}$ given the data and other parameters is

$$\mathbf{V} \mid \{\mathbf{B}, \alpha, \mathcal{D}\} \sim \mathcal{G}\text{-}IW(\delta+n, \mathbf{U}+(\mathbf{I}-\mathbf{B})\mathbf{S}(\mathbf{I}-\mathbf{B})^{\mathsf{T}}).$$

Therefore it can be sampled using the results from the previous section. Notice this holds even if the directed mixed graph $\mathcal{G}$ is cyclic.

Sampling α_i given $\{\mathcal{D}, \Theta\backslash\{\alpha_i\}\}$ can also be done easily for both cyclic and acyclic models: the posterior is given by a normal $N(c^{\alpha'}_i/s^{\alpha'}_i, 1/s^{\alpha'}_i)$ where

$$\begin{aligned} s_i^{\alpha'} &\equiv \frac{1}{s_i^{\alpha}} + n(\mathbf{V}^{-1})_{ii}, \\ c_i^{\alpha'} &\equiv \frac{c_i^{\alpha}}{s_i^{\alpha}} - n\sum_{t=1,t\neq i}^{m}(\mathbf{V}^{-1})_{it}\alpha_t + \sum_{d=1}^{n}\sum_{t=1}^{m}(\mathbf{V}^{-1})_{it}\left(Y_t^{(d)} - \sum_{p_t} b_{tp_t}Y_{p_t}^{(d)}\right), \end{aligned}$$

with p_t being an index running over the parents of Y_t in $\mathcal{G}$.

However, sampling the non-zero entries of $\mathbf{B}$ results in two different cases depending whether $\mathcal{G}$ is cyclic or not. We deal with them separately.

4.1 Sampling from the Posterior: Acyclic Case

The acyclic case is simplified by the fact that $\mathbf{I}-\mathbf{B}$ can be rearranged in a way it becomes lower triangular, with each diagonal element being 1. This implies the identity $|(\mathbf{I}-\mathbf{B})^{-1}||\mathbf{V}||(\mathbf{I}-\mathbf{B})^{-\mathsf{T}}| = |\mathbf{V}|$, with the resulting log-likelihood being a quadratic function of the non-zero elements of $\mathbf{B}$. Since the prior for coefficient b_{ij} is Gaussian, its posterior given the data and all other parameters will be the Gaussian $N(c_{ij}^{b'}/s_{ij}^{b'}, 1/s_{ij}^{b'})$ where

$$\begin{aligned} s_{ij}^{b'} &\equiv \frac{1}{s_{ij}^{b}} + (\mathbf{V}^{-1})_{ii}\sum_{d=1}^{n}(Y_j^{(d)})^2, \\ c_{ij}^{b'} &\equiv \frac{c_{ij}^{b}}{s_{ij}^{b}} + \sum_{d=1}^{n}Y_j^{(d)}\sum_{t=1}^{m}(\mathbf{V}^{-1})_{it}\left(Y_t^{(d)} - \sum_{p_t,(t,p_t)\neq(i,j)} b_{tp_t}Y_{p_t}^{(d)} - \alpha_t\right). \end{aligned} \tag{16}$$

As before, p_t runs over the indices of the parents of Y_t in $\mathcal{G}$. Notice that in the innermost summation we exclude $b_{ij}Y_j^{(d)}$. We can then sample b_{ij} accordingly.

It is important to notice that, in practice, better mixing behavior can be obtained by sampling the coefficients (and intercepts) jointly. The joint distribution is Gaussian and can be obtained in a way similar to the above derivation. The derivation of the componentwise conditionals is nevertheless useful in the algorithm for cyclic networks.

4.2 Sampling from the Posterior: Cyclic Case

Cyclic directed graph models have an interpretation in terms of causal systems in equilibrium. The simultaneous presence of directed paths $Y_i \rightarrow \cdots \rightarrow Y_j$ and $Y_j \rightarrow \cdots \rightarrow Y_i$ can be used to parameterize instantaneous causal effects in a feedback loop (Spirtes, 1995). This model appears also in the structural equation modeling literature (Bollen, 1989). In terms of cyclic graphs as families of conditional independence constraints, methods for reading off constraints in linear systems also exist (Spirtes et al., 2000).

The computational difficulty in the cyclic case is that the determinant $|\mathbf{I}-\mathbf{B}|$ is no longer a constant, but a multilinear function of coefficients $\{b_{ij}\}$. Because b_{ij} will appear outside the exponential term, its posterior will no longer be Gaussian.

From the definition of the implied covariance matrix $\Sigma(\Theta)$, it follows that $|\Sigma(\Theta)|^{-n/2} = (|\mathbf{I}-\mathbf{B}||\mathbf{V}|^{-1}|\mathbf{I}-\mathbf{B}|)^{n/2}$. As a function of coefficient b_{ij},

$$|\mathbf{I}-\mathbf{B}| = (-1)^{i+j+1}C_{ij}b_{ij} + \sum_{k=1,k\neq j}^{k=m}(-1)^{i+k+1}C_{ik}b_{ik},$$

where C_{ij} is the determinant of respective co-factor of $\mathbf{I}-\mathbf{B}$, $b_{ik} \equiv 0$ if there is no edge $Y_i \leftarrow Y_k$, and $b_{ii} \equiv -1$. The resulting density function of b_{ij} given $\mathcal{D}$ and $\Theta\backslash\{b_{ij}\}$ is

$$p(b_{ij}|\Theta\backslash\{b_{ij}\}, \mathcal{D}) \propto |b_{ij} - \kappa_{ij}|^n \exp\left\{-\frac{(b_{ij} - c_{ij}^{b'}/s_{ij}^{b'})^2}{2s_{ij}^{b'}}\right\},$$

where

$$\kappa_{ij} \equiv C_{ij}^{-1} \sum_{k=1, k\neq j}^{k=m} (-1)^{k-j+1} C_{ik} b_{ik}$$

and $\{c_{ij}^{b'}, s_{ij}^{b'}\}$ are defined as in Equation (16). Standard algorithms such as Metropolis-Hastings can be applied to sample from this posterior within a Gibbs procedure.

4.3 Marginal Likelihood: A Variational Monte Carlo Approach

While model selection of bi-directed graphs can be performed using a simple Monte Carlo procedure as seen in the previous Section, the same is not true in the full Gaussian DMG case. Approaches such as nested sampling (Skilling, 2006) can in principle be adapted to deal with the full case. For problems where there are many possible candidates to be evaluated, such a computationally demanding sampling procedure might be undesirable (at least for an initial ranking of graphical structures). As an alternative, we describe an approximation procedure for the marginal likelihood $p(\mathcal{D}|\mathcal{G})$ by combining variational bounds (Jordan et al., 1998) with the $\mathcal{G}$-Inverse Wishart samplers, and therefore avoiding a Markov chain over the joint model of coefficients and error covariances. This is described for acyclic DMGs only.

We adopt the following approximation in our variational approach, accounting also for possible latent variables $\mathbf{X}$:

$$p(\mathbf{V}, \mathbf{B}, \alpha, \mathbf{X}|\mathcal{D}) \approx q(\mathbf{V})q(\mathbf{B}, \alpha) \prod_{d=1}^{n} q(\mathbf{X}^{(d)}) \equiv q(\mathbf{V})q(\mathbf{B}, \alpha)q(\mathbf{X})$$

with $q(\mathbf{B}, \alpha)$ being a multivariate Gaussian density of the non-zero elements of $\mathbf{B}$ and α. Function $q(\mathbf{X}^{(d)})$ is also a Gaussian density, and function $q(\mathbf{V})$ is a $\mathcal{G}$-Inverse Wishart density.

From Jensen's inequality, we obtain the following lower-bound (Beal, 2003, p. 47):

$$\begin{array}{rcl} \ln p(\mathcal{D}|\mathcal{G}) & = & \ln \int p(\mathbf{Y}, \mathbf{X}|\mathbf{V}, \mathbf{B}, \alpha)p(\mathbf{V}, \mathbf{B}, \alpha)\, d\mathbf{X}\, d\mathbf{B}\, d\mathbf{V}\, d\alpha \\ & \geq & \langle \ln p(\mathbf{Y}, \mathbf{X}|\mathbf{V}, \mathbf{B}, \alpha)\rangle_{q(\mathbf{V})q(\mathbf{B},\alpha)q(\mathbf{X})} \\ & & + \langle \ln p(\mathbf{V})/q(\mathbf{V})\rangle_{q(\mathbf{V})} \\ & & + \langle \ln p(\mathbf{B}, \alpha)/q(\mathbf{B}, \alpha)\rangle_{q(\mathbf{B},\alpha)} - \langle \ln q(\mathbf{X})\rangle_{q(\mathbf{X})} \end{array} \quad (17)$$

where this lower bound can be optimized with respect to functions $q(\mathbf{V})$, $q(\mathbf{B})$, $q(\mathbf{X})$. This can be done by iterative coordinate ascent, maximizing the bound with respect to a single $q(\cdot)$ function at a time.

The update of $q(\mathbf{V})$ is given by

$$q^{new}(\mathbf{V}) = p_{\mathcal{G}\text{-}IW}(\delta + d, \mathbf{U} + \left\langle (\mathbf{I}-\mathbf{B})\mathbf{S}(\mathbf{I}-\mathbf{B})^{\mathsf{T}} \right\rangle_{q(\mathbf{X})q(\mathbf{B},\alpha)})$$

where $p_{\mathcal{G}\text{-}IW}(\cdot)$ is the density function for a $\mathcal{G}$-Inverse Wishart, and $\mathbf{S}$ is the empirical second moment matrix summed over the completed data set $(\mathbf{X}, \mathbf{Y})$ (hence the expectation over $q(\mathbf{X})$) centered at $\mu(\Theta)$.

The updates for $q(\mathbf{B}, \alpha)$ and $q(\mathbf{X})$ are tedious but straightforward derivations, and described in Appendix B. The relevant fact about these updates is that they are functions of $\langle \mathbf{V}^{-1} \rangle_{q(\mathbf{V})}$. Fortunately, we pay a relatively small cost to obtain these inverses using the Monte Carlo sampler of Figure 4: from the Bartlett parameters, define a lower triangular $m \times m$ matrix $\mathcal{B}$ (by placing on the ith line the row vector $\mathcal{B}_i$, followed by zeroes) and a diagonal matrix Γ from the respective vector of γ_i's. The matrix $\mathbf{V}^{-1}$ can be computed from $(\mathbf{I} - \mathcal{B})^{\mathsf{T}}\Gamma^{-1}(\mathbf{I} - \mathcal{B})$, and the relevant expectation computed according to the importance sampling procedure. For problems of moderate dimensionality,[7] the importance sampler might not be recommended, but the Gibbs sampler can be used.

At the last iteration of the variational maximization, the (importance or posterior) samples from $q(\mathbf{V})$ can then be used to compute the required averages in (17), obtaining a bound on the marginal log-likelihood of the model. Notice that the expectation $\langle \ln p(\mathbf{V})/q(\mathbf{V}) \rangle_{q(\mathbf{V})}$ contains the entropy of $q(\mathbf{V})$, which will require the computation of $\mathcal{G}$-inverse Wishart normalizing constants.

For large problems, the cost of this approximation might still be prohibitive. An option is to partially parameterize $\mathbf{V}$ in terms of ancillary latents and another submatrix distributed as a $\mathcal{G}$-inverse Wishart, but details on how to best do this partition are left as future work (this approximation will be worse but less computationally expensive if ancillary latents are independent of the coefficient parameters in the variational density function $q(\cdot)$). Laplace approximations might be an alternative, which have been successfully applied to undirected non-decomposable models (Roverato, 2002).

We emphasize that the results present in this section are alternatives that did not exist before in previous approaches for learning mixed graph structures through variational methods (e.g., Silva and Scheines, 2006). It is true that the variational approximation for marginal likelihoods will tend to underfit the data, that is, generate models simpler than the true model in simulations. Despite the bias introduced by the method, this is less of a problem for large data sets (Beal and Ghahramani, 2006) and the method has been shown to be useful in model selection applications (Silva and Scheines, 2006), being consistently better than standard scores such as BIC when hidden variables are present (Beal and Ghahramani, 2006). An application in prediction using the variational posterior instead of MCMC samples is discussed by Silva and Ghahramani (2006). It is relevant to explore other approaches for marginal likelihood evaluation of DMG models using alternative methods such as annealed importance sampling (Neal, 2001) and nested sampling (Skilling, 2006), but it is unrealistic to expect that such methods can be used to evaluate a large number of candidate models. A pre-selection by approximations such as variational methods might be essential.

5. Discrete Models: The Probit Case

Constructing a discrete mixed graph parameterization is not as easy as in the Gaussian case. Advances in this area are described by Drton and Richardson (2008a), where a complete parameterization of binary bi-directed graph models is given. In our Bayesian context, inference with the mixed graph discrete models of Drton and Richardson would not to be any computationally easier than the case for Markov random fields, which has been labeled as *doubly-intractable* (Murray et al., 2006).

7. We observed a high ratio of the highest importance weight divided by the median weight in problems with dimensionality as low as 15 nodes. However, notice that in practice the error covariance matrix $\mathbf{V}$ has a block diagonal structure, and only the size of the largest block is relevant. This is explained in more detail in Section 6.

Instead, in this paper we will focus on a class of discrete models that has been widely used in practice: the probit model (Bartholomew and Knott, 1999). This model is essentially a projection of a Gaussian distribution into a discrete space. It also allows us to build on the machinery developed in the previous sections. We will describe the parameterization of the model for acyclic DMGs, and then proceed to describe algorithms for sampling from the posterior distribution.

5.1 Parameterizing Models of Observable Independencies

A probit model for the conditional probability of discrete variable Y_i given a set of variables $\{Y_{i1}, \ldots, Y_{ik}\}$ can be described by the two following relationships:

$$\begin{array}{rcl} Y_i^\star & = & \alpha_i + b_{i1}Y_{i1} + b_{i2}Y_{i2} + \cdots + b_{ik}Y_{ik} + \varepsilon_i \\ \mathcal{P}(Y_i = v_l^i \mid Y_i^\star) & = & 1(\tau_{l-1}^i \leq Y_i^\star < \tau_l^i) \end{array} \tag{18}$$

where $\mathcal{P}(\cdot)$ is the probability mass function of a given random variable, as given by the context, and $1(\cdot)$ is the indicator function. Y_i assumes values in $\{v_1^i, v_1^i, \ldots, v_{\kappa(i)}^i\}$. Thresholds $\{\tau_0^i = -\infty < \tau_1^i < \tau_2^i < \cdots < \tau_{\kappa(i)}^i = \infty\}$ are used to define the mapping from continuous $Y_i^\star$ to discrete Y_i. This model has a sensible interpretation for ordinal and binary values as the discretization of some *underlying latent variable* (UV) $Y_i^\star$. Such a UV is a conditionally Gaussian random variable, which follows by assuming normality of the error term ε_i. This formulation, however, is not appropriate for general discrete variables, which are out of the scope of this paper. Albert and Chib (1993) describe alternative Bayesian treatments of discrete distributions not discussed here.

Given this binary/ordinal regression formulation, the natural step is how to define a graphical model accordingly. As a matter of fact, the common practice does not strictly follow the probit regression model. Consider the following example: for a given graph $\mathcal{G}$, a respective graphical representation of a probit model can be built by first replicating $\mathcal{G}$ as a graph $\mathcal{G}^\star$, where each vertex Y_i is relabeled as $Y_i^\star$. Those vertices represent continuous underlying latent variables (UVs). To each vertex $Y_i^\star$ in $\mathcal{G}^\star$, we then add a single child Y_i. We call this the *Type-I UV model*. Although there are arguments for this approach (see, for instance, the arguments by Webb and Forster (2006) concerning stability to ordinal encoding), this is a violation of the original modeling assumption as embodied by $\mathcal{G}$: if the given graph is a statement of conditional independence constraints, it is expected that such independencies will be present in the actual model. The Type-I formulation does not fulfill this basic premise: by construction there are no conditional independence constraints among the set of variables $\mathbf{Y}$ (the marginal independencies are preserved, though). This is illustrated by Figure 6(b), where the conditional independence of Y_1 and Y_3 given Y_2 disappears.

An alternative is illustrated in Figure 6(c). Starting from the original graph $\mathcal{G}$ (as in Figure 6(a)), the probit graph model $\mathcal{G}^\star$ shown in the Figure is built from $\mathcal{G}$ by the following algorithm:

1. add to empty graph $\mathcal{G}^\star$ the vertices $\mathbf{Y}$ of $\mathcal{G}$, and for each $Y_i \in \mathbf{Y}$, add a respective UV $Y_i^\star$ and the edge $Y_i^\star \rightarrow Y_i$;

2. for each $Y_i \rightarrow Y_j$ in $\mathcal{G}$, add edge $Y_i \rightarrow Y_j^\star$ to $\mathcal{G}^\star$;

3. for each $Y_i \leftrightarrow Y_j$ in $\mathcal{G}$, add edge $Y_i^\star \leftrightarrow Y_j^\star$ to $\mathcal{G}^\star$;

We call this the *Type-II UV model*, which has the following property (the proof is in Appendix C):

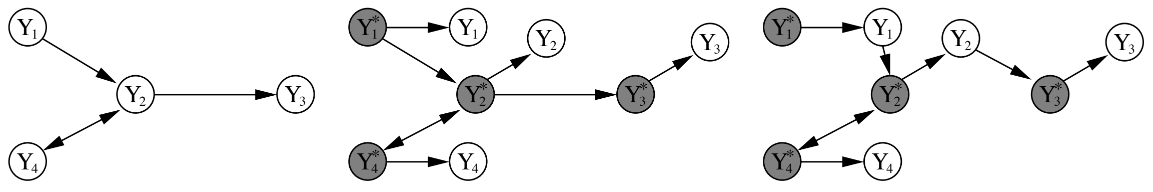


Figure 6: The model in (a) has at least two main representations as a probit network. In (b), the original structure is given to the underlying variables, with observed variables being children of their respective latents. In (c), the underlying variable inherits the parents of the original variable and the underlying latents of the spouses.

Theorem 4 *Suppose $\mathcal{G}$ is acyclic with vertex set* $\mathbf{Y}$. *Y_i and Y_j are m-separated given $\mathbf{Z} \subseteq \mathbf{Y}\backslash\{Y_i, Y_j\}$ in $\mathcal{G}$ if and only if Y_i and Y_j are m-separated given $\mathbf{Z}$ in $\mathcal{G}^\star$.*

The parameterization of the Type-II UV model follows from the definition of probit regression: the conditional distribution Y_i given its parents in $\{Y_{i1}, ..., Y_{ik}\}$ in $\mathcal{G}$ is given as in Equation (18), while the error terms $\{\varepsilon_1, \varepsilon_2, \dots, \varepsilon_m\}$ follow the multivariate Gaussian $N(0, \mathbf{V})$. The entry corresponding to the covariance of ε_i and ε_j is assumed to be zero if there is no bi-directed edge $Y_i \leftrightarrow Y_j$ in $\mathcal{G}$.

In what follows, we discuss algorithms for Type-II models. The approach here described can be easily adapted to cover Type-I models. We say that Type-II models are models of *observable independencies*, since independencies hold even after marginalizing all UVs.

5.2 Algorithm

As before, we provide a Gibbs sampling scheme to sample parameters $\Theta = \{\alpha, \mathbf{B}, \mathbf{V}, \mathcal{T}\}$ from the posterior distribution given data set $\mathcal{D} = \{\mathbf{Y}^{(1)}, \mathbf{Y}^{(2)}, \dots, \mathbf{Y}^{(n)}\}$. The set $\mathcal{T} = \{\mathcal{T}_i\}$ is the set of threshold parameters, $\mathcal{T}_i = \{\tau^i_0 = -\infty < \tau^i_1 < \tau^i_2 < \cdots < \tau^i_{\kappa(i)} = \infty\}$ for each random variable Y_i with $\kappa(i)$ different values. We will not discuss priors and algorithms for sampling $\mathcal{T}$ given the other parameters: this can be done by standard approaches (e.g., Albert and Chib, 1993).[8]

For the purposes of the Gibbs procedure, we augment the data set with the underlying variables $\mathcal{D}^\star = \{\mathbf{Y}^{\star(1)}, \mathbf{Y}^{\star(2)}, \dots, \mathbf{Y}^{\star(n)}\}$ at each sampling step.

From the set of structural equations

$$\mathbf{Y}^{\star(d)} = \alpha + \mathbf{B}\mathbf{Y}^{(d)} + \varepsilon$$

it follows that the conditional distribution of $\mathbf{Y}^{\star(d)}$ given the $\mathcal{D} \cup \Theta$ is a truncated Gaussian with mean $\alpha + \mathbf{B}\mathbf{Y}^{(d)}$ and covariance matrix $\mathbf{V}$. The truncation levels are given by the thresholds and observed data $\mathbf{Y}^{(d)}$: for each $Y^{(d)}_i = v^i_l$, the range for $Y^{\star(d)}_i$ becomes $[\tau^i_{l-1}, \tau^i_l)$. Sampling from a truncated Gaussian is a standard procedure. We used the algorithm of Kotecha and Djuric (1999) in our implementation.

To sample $\mathbf{V}$ from its conditional, we will rely on the following result.

8. In Section 7, we perform experiments with binary data only. In this case, the thresholds are set to fixed values: $\{\tau^i_0 = -\infty, \tau^i_1 = 0, \tau^i_2 = \infty\}$ for all $0 \leq i \leq m$.

Proposition 5 *Let $\mathcal{G}$ be an acyclic DMG, and $(\alpha, \mathbf{B}, \mathbf{V}, \mathcal{T})$ be the respective set of parameters that defines the probit model. For a fixed $(\alpha, \mathbf{B}, \mathcal{T})$, there is a bijective function $f_{\mathbf{B}\alpha\mathcal{T}}(\cdot)$ mapping $\mathbf{Y}^\star$ to ε. This is not true in general if $\mathcal{G}$ is cyclic.*

Proof: If the graph is acyclic, this follows directly by recursively solving the model equations, starting from those corresponding to $Y_j^\star$ vertices with no parents. This results in $\varepsilon = \mathbf{Y}^\star - \alpha - \mathbf{B}\mathbf{Y}$, as expected.

For cyclic graphs, the following model provides a counter-example. Let the graph be $Y_1^\star \rightarrow Y_1 \rightarrow Y_2^\star \rightarrow Y_2 \rightarrow Y_1^\star$. Let the model be $Y_1^\star = Y_2 + \varepsilon_1, Y_2^\star = Y_1 + \varepsilon_2$, that is, $b_{12} = b_{21} = 1$ and $\alpha = 0$. Let the variables be binary, with a threshold at zero ($Y_i = 1$ if and only if $Y_i^\star \geq 0$). Then the two instantiations $(Y_1^\star = -0.8, Y_2^\star = 0)$, $(Y_1^\star = 0.2, Y_2^\star = 1)$ imply the same pair $(\varepsilon_1 = -0.8, \varepsilon_2 = 0)$. $\square$

The negative result for discrete models with cycles is the reason why such models are out of the scope of the paper.

Let $\mathcal{D}_\varepsilon^\star = \{\varepsilon^{(1)}, \dots, \varepsilon^{(n)}\}$, where $\varepsilon^{(d)} = f_{\mathbf{B}\alpha\mathcal{T}}(\mathbf{y}^{(d)\star})$. Due to this bijection (and the determinism mapping $\mathbf{Y}^\star$ to $\mathbf{Y}$), the density $p(\mathbf{V} \mid \Theta\backslash\mathbf{V}, \mathcal{D}, \mathcal{D}^\star) = p(\mathbf{V} \mid \Theta\backslash\mathbf{V}, \mathcal{D}^\star) = p(\mathbf{V} \mid \Theta\backslash\mathbf{V}, \mathbf{y}^{(1)\star}, \dots, \mathbf{y}^{(d)\star})$ is equivalent to

$$\begin{aligned} p(\mathbf{V} \mid \Theta\backslash\mathbf{V}, \mathcal{D}^\star) &= p(\mathbf{V} \mid \alpha, \mathbf{B}, \mathcal{T}, \mathcal{D}^\star, \mathcal{D}_\varepsilon^\star) \\ &= p(\mathbf{V} \mid \alpha, \mathbf{B}, \mathcal{T}, \mathcal{D}_\varepsilon^\star) \\ &\propto p(\mathbf{V} \mid \alpha, \mathbf{B}, \mathcal{T}) p(\mathcal{D}_\varepsilon^\star \mid \alpha, \mathbf{B}, \mathcal{T}, \mathbf{V}) \\ &\propto p(\mathbf{V}) \textstyle\prod_{d=1}^n p(\varepsilon^{(d)} \mid \mathbf{V}). \end{aligned}$$

For the given data set $\mathcal{D} \cup \mathcal{D}^\star$, define $\mathbf{S}^\star$ as the sum of $(\mathbf{Y}^{\star(d)} - \alpha - \mathbf{B}\mathbf{Y}^{(d)})(\mathbf{Y}^{\star(d)} - \alpha - \mathbf{B}\mathbf{Y}^{(d)})^\mathsf{T}$ over all $d \in \{1, 2, \dots, n\}$. Since $p(\varepsilon \mid \mathbf{V})$ is normal with zero mean and covariance matrix $\mathbf{V}$, the posterior for $\mathbf{V}$ given all other parameters and variables is

$$\mathbf{V} \mid \ \{\Theta\backslash\mathbf{V}, \mathcal{D}, \mathcal{D}^\star\} \sim \mathcal{G}\text{-}IW(\delta + n, \mathbf{U} + \mathbf{S}^\star).$$

Sampling $\mathbf{B}$ and α is analogous to the Gaussian case, except that we have to consider that the left-hand side of the structural equations now refer to $\mathbf{Y}^\star$. We give the explicit conditional for α_i, with the conditional for b_{ij} being similarly adapted from Section 4. The posterior for α_i is given by a normal $N((s_i')^{-1} m_i', s_i')$ where

$$\begin{aligned} s_i^{\alpha'} &= \frac{1}{s_i^\alpha} + n(\mathbf{V}^{-1})_{ii}, \\ c_i^{\alpha'} &= \frac{c_i^\alpha}{s_i^\alpha} - n \sum_{t=1, t\neq i}^m (\mathbf{V}^{-1})_{it} \alpha_t + \sum_{d=1}^n \sum_{t=1}^m (\mathbf{V}^{-1})_{it} \left(Y_t^{\star(d)} - \sum_{p_t} b_{tp_t} Y_{p_t}^{(d)} \right). \end{aligned}$$

5.3 A Note on Identifiability

The scale of the underlying latent variables in the probit model is arbitrary. As such, it has been often suggested that such latents should have constant (e.g., unity) variance (Pitt et al., 2006). There are two usual arguments for fixing the variance: improving the interpretability of the model, and improving the mixing of the Markov chain. The interpretability argument is not particularly appealing within the Bayesian setting with proper priors, such as the one proposed in this paper: the posterior distribution of the parameters is well-defined by the prior uncertainty and the data.

The goal of improving the mixing of the chain might be important: if some parameters can assume arbitrary values and still allow for the same model over the observables, then fixing such parameters may help sampling by eliminating largely flat regions from the posterior (which will happen for large data sets and broad priors). In practice, however, scaling UVs might not be advantageous. In some cases it might increase the computational cost of each sampling step, while sampling from the non-scaled model might work just fine. Many MCMC algorithms work well on highly unidentifiable models such as multilayer perceptrons (Neal, 1996). In our experiments, we do not use any scaling.

5.4 Remarks

It is clear that the given approach can be generalized to other generalized linear models by changing the link function that maps underlying latent variables (UVs) to observables. For instance, a model containing discrete and continuous variables can be constructed by using the identity link function instead of probit for the continuous variables. Notice that the continuous variables will not necessarily be marginally Gaussian if some of its parents are discrete. Other link functions will have different parameters besides thresholds, such as in multivalued ("polychotomous") discrete distributions. A Bayesian account of Gaussian copula models is given by Pitt et al. (2006), to which a DMG-based family could in principle be defined. For continuous, marginally non-Gaussian, variables joined by a Gaussian copula, it is possible that all link functions are invertible. In this case, it is easier in principle to define cyclic models through Type-I UV models (e.g., Figure 6(b)) while preserving the observable independencies.

It is important to point out that Type-II probit models with Markov equivalent graphs will not, in general, be likelihood equivalent. A simple example is given by the two-node graphs $Y_1 \rightarrow Y_2$ and $Y_1 \leftrightarrow Y_2$: if Y_1 is binary, then the marginal for Y_2 in the first case is equivalent to having an underlying latent variable that follows a mixture of two Gaussians. While some of these issues can be solved by adopting a mixture of Gaussians marginal independence model to account for bi-directed edges (Silva and Ghahramani, 2009), details need to be worked out. When the goal of model selection is to find causal structures (Spirtes et al., 2000), the usual guarantees of search methods based on Markov equivalence classes do not hold. However, it remains to be seen whether the parametric constraints implied by the Type-II formulation will allow for other consistent approaches for causal discovery, as shown in the case of non-linearities with additive noise (Hoyer et al., 2008).

6. Scaling Up: Factorizations and Perfect Sequences

Each Monte Carlo sampling step for the given mixed graph models is theoretically tractable, but not necessarily practical when the dimensionality m of the data is high. By using clever factorizations of the graph and ordering of the variables, it is possible to sometimes scale to high-dimensional problems. In this section, we describe approaches to minimize the run-time of the marginal likelihood computation for bi-directed graphs, which is also important for computing variational bounds for DMG models. We start, however, with a discussion on factorizations of the posterior density for coefficient parameters $\mathbf{B}$. The context is the Gibbs sampler for acyclic models.

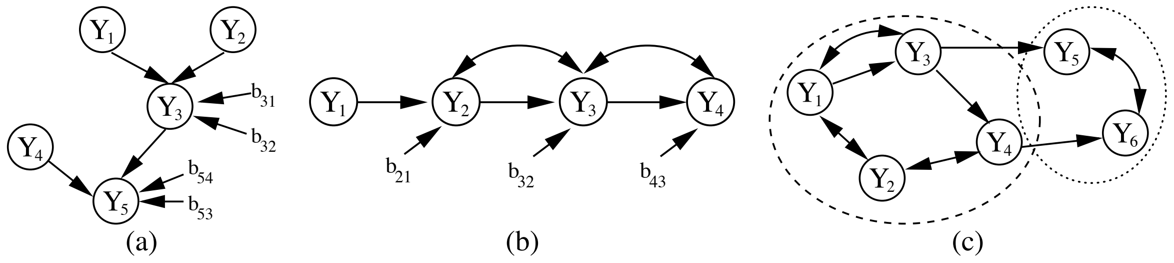


Figure 7: The coefficients b_{31} and b_{32}, represented as nodes in (a), become dependent after conditioning on $\mathbf{Y}$. However, they are still independent of b_{54}. This a general property of DAG models. In DMG models, a sequence of bi-directed edges will connect extra coefficients. In graph (b), coefficients b_{21}, b_{32} and b_{43} will all be dependent given $\mathbf{Y}$. Coefficients into nodes in different districts will still be independent. The graph in (c) has districts $\{Y_1, Y_2, Y_3, Y_4\}$ and $\{Y_5, Y_6\}$.

6.1 Factorizations

Our prior for coefficients $\{b_{ij}\}$ is fully factorized. In directed acyclic graphs, this is particularly advantageous: coefficients corresponding to edges into different nodes are independent in the posterior.[9] One can then jointly sample a whole set of $\{b_{ij}\}$ coefficients with same i index, with no concern for the other coefficients. Figure 7(a) illustrates this factorization. This means that, in Equation (16), the summation over t does not go over all variables, but only for $t = i$. This also follows from the fact that $(\mathbf{V})^{-1}_{it} = 0$ unless $i = t$, since $\mathbf{V}$ is diagonal.

In ADMGs, however, this is not true anymore. For any pair of vertices linked by a path of bi-directed edges, for example, $Y_i \leftrightarrow Y_{i+1} \leftrightarrow \cdots \leftrightarrow Y_t$, one will have in general that $(\mathbf{V})^{-1}_{it} \neq 0$. This can be shown by using the graphical properties of the model when conditioning on some arbitrary datapoint $\mathbf{Y}$:

Proposition 6 *Let $\mathcal{G}$ be an acyclic DMG with vertex set $\mathbf{Y}$, and $\mathcal{G}'$ the DMG obtained by augmenting $\mathcal{G}$ with a vertex for each parameter b_{ij} and a respective edge $b_{ij} \rightarrow Y_i$. Then if there is a bi-directed path $Y_i \leftrightarrow \cdots \leftrightarrow Y_t$ in $\mathcal{G}$, $\{b_{ij}, b_{tv}\}$ are not m-separated given $\mathbf{Y}$ in $\mathcal{G}'$.*

Proof: The joint model for $\{\mathbf{Y}, \mathbf{B}\}$ with independent priors on the non-zero entries of $\mathbf{B}$ is Markov with respect to $\mathcal{G}'$. The sequence of bi-directed edges between Y_i and Y_t implies a path between b_{ij} and b_{tv} where every vertex but the endpoints is a collider in this path. Since every collider is in $\mathbf{Y}$, this path is active. $\Box$

This Proposition is illustrated by Figure 7(b). The practical implication is as follows: m-connection means that there is no further graphical property that would entail $(\mathbf{V})^{-1}_{it} = 0$ (i.e., only particular cancellations on the expression of the inverse, unlikely to happen in practice, would happen to generate such zeroes).

9. Sampling in Gaussian DAG models is still necessary if the model includes latent variables (Dunson et al., 2005).

Consider the maximal sets of vertices in an ADMG such that each pair of elements in this set is connected by a path of bi-directed edges. Following Richardson (2003), we call this a *district*.[10] It follows that is not possible in general to factorize the posterior of $\mathbf{B}$ beyond the set of districts of $\mathcal{G}$. Figure 7(c) illustrates a factorization. Fortunately, for many DMG models with both directed and bi-directed edges found in practical applications (e.g., Bollen, 1989), the maximum district size tends to be considerably smaller than the dimensionality of the problem.

6.2 Perfect Sequences

It is still important to speed up marginal likelihood (or variational bound) computations for models with districts of moderate size, particularly if many models are to be evaluated.

Without loss of generality, assume our graph $\mathcal{G}$ is a bi-directed graph with a single district, since the problem can be trivially separated into the disjoint bi-directed components. We will consider the case where the bi-directed graph is sparse: otherwise there is little to be gained by exploring the graphical structure. In that case, we will assume that the largest number of spouses of any node in $\mathcal{G}$ is bounded by a constant κ that is independent of the total number of nodes, m. The goal is to derive algorithms that are of lower complexity in m than the original algorithms.

The bottleneck of our procedure is the computation of the $\Sigma^{-1}_{nsp_{\prec}(i),nsp_{\prec}(i)}$ matrices, required in the mapping between independent and dependent Bartlett parameters (Equation 7), as well as computing the determinants $|\Sigma_{nsp_{\prec}(i),nsp_{\prec}(i)}|$. Since in sparse districts $nsp_{\prec}(i)$ grows linearly with m, the cost of a naïve algorithm for a single sampling step is $O(m^3)$ per node. Iterating over all nodes implies a cost of $O(m^4)$ for a Monte Carlo sweep. Therefore, our goal is to find a procedure by which such mappings can be computed in less than $O(m^3)$ time. The general framework is reusing previous inverses and determinants instead of performing full matrix inversion and determinant calculation for each Y_i. The difficulty on applying low-rank updates when we traverse the covariance matrix according to $\prec$ is that the sets of non-spouses $nsp_{\prec}(i)$ and $nsp_{\prec}(i+1)$ might differ arbitrarily. We want sensible orderings where such sets vary slowly and allow for efficient low-rank updates, if any.

The foundation of many scaling-up procedures for graphical models is the graph decomposition by clique separators (Tarjan, 1985), usually defined for undirected graphs. The definition for bi-directed graphs is analogous. Such a decomposition identifies overlapping *prime subgraphs* $\{\mathcal{G}_{P(1)}, \mathcal{G}_{P(2)}, \dots, \mathcal{G}_{P(k)}\}$ of the original graph $\mathcal{G}$. A prime graph is a graph that cannot be partitioned into a triple $(\mathbf{Y}', \mathbf{S}, \mathbf{Y}'')$ of non-empty sets such that $\mathbf{S}$ is a complete separator (i.e., $\mathbf{S}$ is a clique and removing $\mathbf{S}$ disconnects the graph). Notice that a clique is also a prime subgraph.

The prime components of a graph can be ordered in a *perfect sequence* $\{\mathbf{Y}_{P(1)}, \dots, \mathbf{Y}_{P(k)}\}$ of subsets of $\mathbf{Y}$ (Roverato, 2002; Lauritzen, 1996). Define $\mathbf{H}_j \equiv \mathbf{Y}_{P(1)} \cup \cdots \cup \mathbf{Y}_{P(j)}$ as the *history* of the perfect sequence up to the j-th subgraph. Let $\mathbf{R}_j \equiv \mathbf{Y}_{P(j)} \backslash \mathbf{H}_{j-1}$ be the *residual* of this history (with $\mathbf{R}_1 \equiv \mathbf{Y}_{P(1)}$), and $\mathbf{S}_j \equiv \mathbf{H}_{j-1} \cap \mathbf{Y}_{P(j)}$ the separator. In a perfect sequence, the triple $(\mathbf{H}_{j-1} \backslash \mathbf{S}_j, \mathbf{S}_j, \mathbf{R}_j)$ forms a decomposition of the subgraph of $\mathcal{G}$ induced by the vertex set $\mathbf{H}_j$.

Surprisingly, although bi-directed and undirected graph models have very different Markov properties (in undirected models, conditioning removes dependencies; in bi-directed models, it adds dependencies), perfect prime graph sequences prove to be also useful, but in an entirely different

10. Kang and Tian (2005) call such structures *c-components* and reserve the word "district" to refer to the function mapping a vertex to its respective c-component, as originally introduced by Richardson (2003). We choose to overload the word and call "district" both the structure and the mapping.

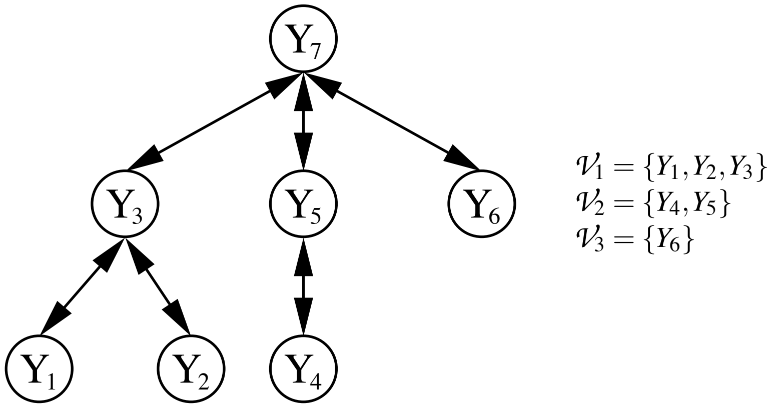

Y7
Y3
Y5
Y6
Y1
Y2
Y4
$\mathcal{V}_1 = \{Y_1, Y_2, Y_3\}$
$\mathcal{V}_2 = \{Y_4, Y_5\}$
$\mathcal{V}_3 = \{Y_6\}$

Figure 8: On the left, we have a bi-directed graph of 7 vertices arranged and ordered such that nodes are numbered by a depth-first numbering starting from "root" Y_7, with $\{Y_1, Y_2, Y_4, Y_6\}$ being leaves. Vertices $\{Y_1, Y_2, \dots, Y_6\}$ can be partitioned as the union $\cup_{t=1}^{3} \mathcal{V}_t$, as illustrated on the right.

way. The next subsection describes the use of prime graph decompositions in a particularly interesting class of bi-directed graphs: the decomposable case. The general case is treated in the sequel.

6.2.1 DECOMPOSABLE MODELS

In a recursively decomposable graph, all prime subgraphs are cliques. We will assume that any perfect sequence in this case contains all and only the (maximal) cliques of the graph. The resulting decomposition can be interpreted as a hypergraph where nodes are the maximal cliques of the original graph, and edges correspond to the separators. In the statistics literature, a decomposable model is defined as a model that is Markov with respect to a recursively decomposable undirected graph (Lauritzen, 1996). Its widespread presence on applications of Markov random fields is due to nice computational properties, with tree-structured distributions being a particular case. Our definition of bi-directed decomposable models is analogous: a model Markov with respect to a recursively decomposable bi-directed graph.

Given the residual sequence $\{\mathbf{R}_1, \mathbf{R}_2, \dots, \mathbf{R}_k\}$ obtained through a perfect sequence of maximal cliques of $\mathcal{G}$, we define a *perfect ordering* $\prec$ by numbering nodes in $\mathbf{R}_t$ before nodes in $\mathbf{R}_1, \dots, \mathbf{R}_{t-1}$, $1 < t \leq k$ and ordering nodes according to this numbering.[11] Any ordering that satisfies this restriction is a perfect ordering. Such an ordering has the following property.

Theorem 7 *Let $\mathcal{G}$ be a recursively decomposable bi-directed graph such that the indexing of its vertices $\mathbf{Y} = \{Y_1, Y_2, \dots, Y_m\}$ follows a perfect ordering $\prec$. Then for each $1 < i \leq m$, the set $\{Y_1, Y_2, \dots, Y_{i-1}\}$ can be partitioned as $\cup_{t=1}^{K(i)} \mathcal{V}_t$ such that:*

1. *each $\mathcal{V}_t$ induces a connected subgraph of $\mathcal{G}$, and for each $Y_t \in \mathcal{V}_t$ and $Y_{t'} \in \mathcal{V}_{t'}$, $t \neq t'$, Y_t is not adjacent to $Y_{t'}$ in $\mathcal{G}$;*

11. Lauritzen (1996) describes other uses of perfect sequences in undirected graphs. Notice that the notion of perfect numbering described by Lauritzen (1996) is not equivalent to our notion of perfect ordering, which is always derived from a perfect sequence of prime graphs.

2. *for each* $\{Y_p, Y_q\} \subseteq \mathcal{V}_t^i$*, if* Y_p *is a spouse of* Y_i*, and* Y_q *is a non-spouse of* Y_i*, then* $p > q$*;*

The proof is in Appendix C. This result is easier to visualize in trees. One can take as a perfect ordering some depth-first ordering for a given choice of root. Then for each vertex Y_i, the set $\{Y_1, Y_2, \dots, Y_{i-1}\}$ is partitioned according to the different branches "rooted" at Y_i. The starting point of each branch is a spouse of Y_i, and all other vertices are non-spouses of Y_i. The ordering result then follows directly from the definition of depth-first traversal, as illustrated in Figure 8.

Let Σ be the covariance matrix of a bi-directed decomposable model with graph $\mathcal{G}$, where Σ follows a $\mathcal{G}$-inverse Wishart distribution. Let $\prec$ be a perfect ordering for $\mathcal{G}$. By the construction of Bartlett's decomposition, mapping between parameters is given by

$$\Sigma_{sp_\prec(i), nsp_\prec(i)} \Sigma^{-1}_{nsp_\prec(i), nsp_\prec(i)},$$

the computational bottleneck being the inversion. Notice this corresponds to the multiple regression coefficients of $sp_\prec(i)$ on $nsp_\prec(i)$. But according to Theorem 7, using a perfect ordering implies that within each $\mathcal{V}_s^i$ for a fixed Y_i, all preceding non-spouses of Y_i are ordered before the preceding spouses. Elements $\{Y_p, Y_q\}$ in different $\mathcal{V}_s^i$ are marginally independent given $\{Y_1, \dots, Y_{i-1}\} \backslash \{Y_p, Y_q\}$. This implies that the regression coefficient of spouse Y_p on non-spouse Y_q will be zero if Y_p and Y_q are on different components $\mathcal{V}_s^i$, and will be identical to the previously computed $\mathcal{B}_{p,q}$ if they are in the same component. Splitting the set $\{Y_1, Y_2, \dots Y_{i-1}\}$ into preceding spouses $\mathbf{Y}_{sp_\prec(i)}$ and non-spouses $\mathbf{Y}_{nsp_\prec(i)}$, we have

$$\begin{aligned}
\mathbf{Y}_{sp_\prec(i)} &= \mathcal{B}_{sp_\prec(i), sp_\prec(i)} \mathbf{Y}_{sp_\prec(i)} + \mathcal{B}_{sp_\prec(i), nsp_\prec(i)} \mathbf{Y}_{nsp_\prec(i)} + \varepsilon_{sp_\prec(i)} \Rightarrow \\
\mathbf{Y}_{sp_\prec(i)} &= (\mathbf{I} - \mathcal{B}_{sp_\prec(i), sp_\prec(i)})^{-1} (\mathcal{B}_{sp_\prec(i), nsp_\prec(i)} \mathbf{Y}_{nsp_\prec(i)} + \varepsilon_{sp_\prec(i)})
\end{aligned}$$

where each ε_j is an independent Gaussian with variance γ_j, and each element (p, q) in $\mathcal{B}_{sp_\prec(i), nsp_\prec(i)}$ corresponds to the known (i.e., previously computed) regression coefficient of the spouse Y_p on the non-spouse Y_q. Matrix $\mathcal{B}_{sp_\prec(i), sp_\prec(i)}$ is defined analogously. Hence, the regression coefficients of $\mathbf{Y}_{sp_\prec(i)}$ on $\mathbf{Y}_{nsp_\prec(i)}$ are given by

$$\Sigma_{sp_\prec(i), nsp_\prec(i)} \Sigma^{-1}_{nsp_\prec(i), nsp_\prec(i)} = (\mathbf{I} - \mathcal{B}_{sp_\prec(i), sp_\prec(i)})^{-1} \mathcal{B}_{sp_\prec(i), nsp_\prec(i)}. \tag{19}$$

No inversion of $\Sigma_{nsp_\prec(i), nsp_\prec(i)}$ is ever necessary. Moreover, the determinant $|\Sigma_{nsp_\prec(i), nsp_\prec(i)}|$ is given by $\prod_{\{q \text{ s.t. } Y_q \in nsp_\prec(i)\}} \gamma_q$, since all non-spouses precede the spouses (which means their marginal covariance matrix is given by the previously computed Bartlett parameters).

Hence, calculating $\mathcal{B}_{i, nsp_\prec(i)}$ for all $1 \leq i \leq m$ according to a perfect ordering has as a bottleneck the inversion (of a triangular matrix) and multiplication in Equation (19), with a cost of $O(\kappa^2 + m\kappa^2)$, κ being the maximum number of spouses for any given node. The cost of the remaining operations for the i-th stage in the importance sampler is $O(\kappa^3)$. As a function of m, the cost of the parameter sampling step falls from $O(m^3)$ to $O(m)$. The cost of computing the weights is dominated by the computation of $\mathbf{K}_i$ from Equation (13), which is $O(\kappa^3 + \kappa m^2) = O(m^2)$. Figure 9 illustrates the derivation of the new ordering in a tree-structured model.

6.2.2 NON-DECOMPOSABLE MODELS

In a non-decomposable model, some prime graphs $\mathbf{Y}_{P(t)}$ will no longer be cliques. In what follows, we once again assume that $\prec$ is a perfect ordering. Unlike in the decomposable case, the product

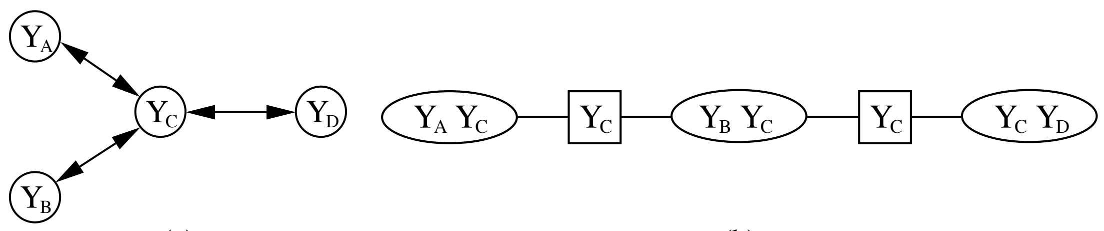


Figure 9: The tree-structured (i.e., cycle-free) bi-directed graph in (a) has as maximal cliques the adjacent pairs. Such cliques can be ordered in a perfect sequence as shown in (b), where rectangles indicate the separators. Notice that $\mathbf{R}_1 = \{Y_A, Y_C\}, \mathbf{R}_2 = \{Y_B\}, \mathbf{R}_3 = \{Y_D\}$. One possible perfect ordering is $\{Y_D, Y_B, Y_C, Y_A\}$.

$\Sigma_{sp_{\prec}(i),nsp_{\prec}(i)}\Sigma^{-1}_{nsp_{\prec}(i),nsp_{\prec}(i)}$ does not simplify in general. Instead we will focus only on fast methods to compute $\Sigma^{-1}_{nsp_{\prec}(i),nsp_{\prec}(i)}$.

As we shall see, the function of the perfect sequence is now to provide a sensible choice of which inverse submatrices $\{\Sigma^{-1}_{\mathbf{W},\mathbf{W}}\}$, $\mathbf{W} \subseteq \mathbf{Y}$, to cache and reuse when computing $\Sigma^{-1}_{nsp_{\prec}(i),nsp_{\prec}(i)}$. The same can be done to compute determinants $|\Sigma_{nsp_{\prec}(i),nsp_{\prec}(i)}|$.

A simple way of reusing the results from the previous section is by triangulating the non-decomposable graph $\mathcal{G}$, transforming it into a decomposable one, $\mathcal{G}'$, which is then used to generate the perfect sequence. We need to distinguish between the "true" spouses of a node Y_i in $\mathcal{G}$ and the artificial spouses in $\mathcal{G}'$ that result from the extra edges added.

Let $nsp_{\prec\mathcal{G}'}(i)$ be the non-spouses of Y_i in $\mathcal{G}'$ that precede it according to $\prec$: by construction, these are also non-spouses of Y_i in $\mathcal{G}$. Let $sp_{\Delta\prec\mathcal{G}'}(i)$ be the spouses of Y_i in $\mathcal{G}'$ that are *not* spouses of Y_i in $\mathcal{G}$. That is, the set of preceding non-spouses of Y_i in $\mathcal{G}$ is given by $nsp_{\prec}(i) = nsp_{\prec\mathcal{G}'}(i) \cup sp_{\Delta\prec\mathcal{G}'}(i)$.

Recall that the inverse of a partitioned matrix can be given by the following identity:

$$\begin{pmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{C} & \mathbf{D} \end{pmatrix}^{-1} = \begin{pmatrix} \mathbf{A}^{-1} + \mathbf{A}^{-1}\mathbf{B}(\mathbf{D} - \mathbf{C}\mathbf{A}^{-1}\mathbf{B})^{-1}\mathbf{C}\mathbf{A}^{-1} & -\mathbf{A}^{-1}\mathbf{B}(\mathbf{D} - \mathbf{C}\mathbf{A}^{-1}\mathbf{B})^{-1} \\ -(\mathbf{D} - \mathbf{C}\mathbf{A}^{-1}\mathbf{B})^{-1}\mathbf{C}\mathbf{A}^{-1} & (\mathbf{D} - \mathbf{C}\mathbf{A}^{-1}\mathbf{B})^{-1} \end{pmatrix}. \quad (20)$$

In order to compute $\Sigma^{-1}_{nsp_{\prec}(i),nsp_{\prec}(i)}$, we consider its partitioned version

$$\Sigma^{-1}_{nsp_{\prec}(i),nsp_{\prec}(i)} = \begin{pmatrix} \Sigma_{nsp_{\prec\mathcal{G}'}(i),nsp_{\prec\mathcal{G}'}(i)} & \Sigma_{nsp_{\prec\mathcal{G}'}(i),sp_{\Delta\prec\mathcal{G}'}(i)} \\ \Sigma_{sp_{\Delta\prec\mathcal{G}'}(i),nsp_{\prec\mathcal{G}'}(i)} & \Sigma_{sp_{\Delta\prec\mathcal{G}'}(i),sp_{\Delta\prec\mathcal{G}'}(i)} \end{pmatrix}^{-1}. \quad (21)$$

Let κ_{nsp} be the maximum number of non-spouses among all Y_i within any prime subgraph induced by $\mathbf{Y}_{P(t)}$. By using relation (20), where we assume for now that we know $\mathbf{A}^{-1} \equiv \Sigma^{-1}_{nsp_{\prec\mathcal{G}'}(i),nsp_{\prec\mathcal{G}'}(i)}$, the cost of computing (21) is $O(m^2\kappa_{nsp}) + O(\kappa^3_{nsp}) = O(m^2\kappa_{nsp})$ (the cost of computing $\mathbf{D} - \mathbf{C}\mathbf{A}^{-1}\mathbf{B}$ is $O(m^2\kappa_{nsp}) + O(\kappa^2_{nsp}) = O(m^2\kappa_{nsp})$, while the cost of inverting it is $O(\kappa^3_{nsp})$). Treating κ_{nsp} as a constant, this reduces the complexity of sampling the i-th row of Σ from $O(m^3)$ to $O(m^2)$. A similar procedure applies to the computation of the determinant $|\Sigma_{nsp_{\prec}(i),nsp_{\prec}(i)}|$, using in this case the relationship (26).

The advantage of using the perfect sequence is to allow for the computation of all $\mathbf{A}^{-1} \equiv \Sigma^{-1}_{nsp_{\prec \mathcal{G}'}(i), nsp_{\prec \mathcal{G}'}(i)}$ at a total cost, across all nodes, of $O(m^3)$: each set $nsp_{\prec \mathcal{G}'}(i)$ is guaranteed to be equal to $\{Y_1, Y_2, \dots, Y_{lns}\}$ where Y_{lns} is the last non-spouse of Y_i in $\mathcal{G}'$ that antecedes Y_i. This follows from the result in the previous section, since all non-spouses of a node in a decomposable graph precede its spouses. Therefore, if we store the inverse covariance matrices for $\{Y_1, Y_2, \dots, Y_i\}$, $1 \leq i \leq m$, we can obtain the required matrices $\mathbf{A}^{-1}$. This requires the storage of $O(m)$ matrices, and each matrix can be obtained by the previous one by a low-rank update (20) with a $O(m^2)$ cost.

Arbitrary orderings do not guarantee such an incremental pattern and, hence, no efficient low-rank updates. Notice that depending on the problem, many of such inverse matrices can be dynamically removed from memory if they are not used by any node placed after a particular position.

6.3 Remarks

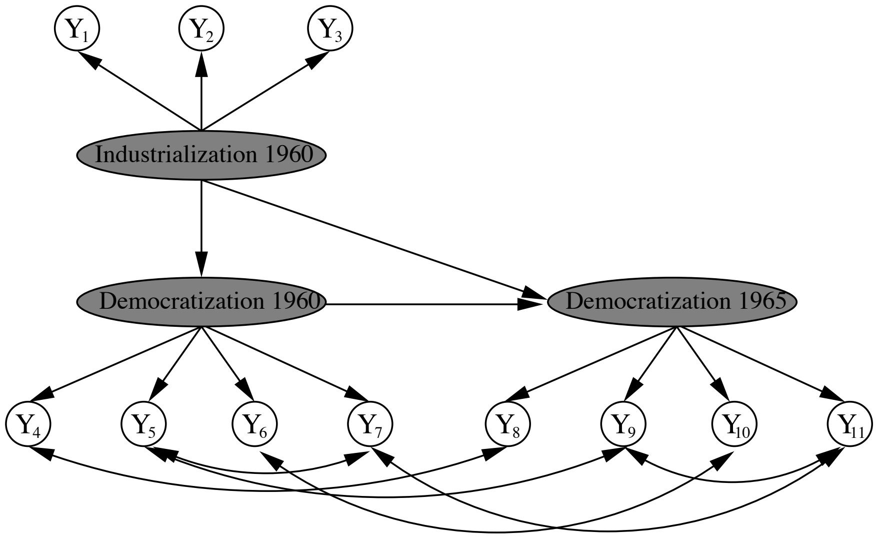
In Gaussian undirected models, the problem of covariance matrix sampling can also be reduced to sampling within each prime graph at the cost of $O(|\mathcal{P}|^4)$, $|\mathcal{P}|$ being the size of the largest prime component (Atay-Kayis and Massam, 2005). Since both κ and κ_{nsp} are $O(|\mathcal{P}|)$, our procedure costs $O(m^2|\mathcal{P}|^2 + |\mathcal{P}|^4)$ per prime graph, plus a cost of $O(m^2)$ per node to compute the importance weights. Considering a number of $m/|\mathcal{P}|$ prime graphs and $|\mathcal{P}| < m$, the total cost is $O(m^3|\mathcal{P}|)$, down from $O(m^4)$. For undirected models, the corresponding cost by sampling step using the perfect ordering decomposition is $O(m|\mathcal{P}|^3)$. The higher-order dependency on m in bi-directed models is to be expected, since the Markov blanket of any node Y_t in a connected bi-directed graph is $\mathbf{V}\backslash\{Y_t\}$. It is clear that inference with a given bi-directed graph model will never scale at the same rate of a undirected model with the same adjacencies, but this does not justify adopting an undirected representation if it is ill-suited to the problem at hand. One has also to consider that in problems with directed and bi-directed edges, the actual maximum district size might be much smaller than the number of variables. For large problems, however, further approximation schemes will be necessary. Drton and Richardson (2008b) describe some reduction techniques for transforming bi-directed edges into directed edges such that the resulting Gaussian model remains the same. As future work, such methods could be adapted to the $\mathcal{G}$-inverse Wishart sampling procedures and combined with the ordering techniques developed here into a single framework. It will also be interesting to develop similar schemes for the Gibbs sampler.

7. Experiments

We now evaluate the advantages of the Gaussian and probit models in Bayesian inference on real problems.

7.1 Industrialization and Democratization Study

Bollen (1989) describes a structural equation model of political and democratization factors within nations. “Democratization” and “industrialization” levels are abstract notions, but nevertheless of clearly observable impact. They are tied to empirical observations through different sets of *indicators*. For instance, an indicator of industrialization level is the gross national product. Hence, democratization and industrialization levels are here defined as scalar latent variables never observed directly, while the observed data is composed of indicators. In this model, there is a total of three indicators of industrialization, and four indicators of democratization. Democratization is



1. Gross national product (GNP) 1960
2. Energy consumption per capita 1960
3. Percentage of labor force in industry 1960
4. Freedom of press 1960
5. Freedom of opposition 1960
6. Fairness of elections 1960
7. Elective nature of legislative body 1960
8. Freedom of press 1965
9. Freedom of opposition 1965
10. Fairness of elections 1965
11. Elective nature of legislative body 1965

Figure 10: A directed mixed graph representing dependencies between 11 observed political and economical indicators and three latent concepts (shaded nodes) (Dunson et al., 2005; Bollen, 1989).

measured in a longitudinal study, where data was collected in two years (1960 and 1965). The indicators of democratization are pooled expert opinions summarized in an ordinal number scaled from 1 to 10. Following Bollen, we will treat the model as multivariate Gaussian, which provides an excellent fit (a p-value greater than 0.3 using a chi-square test) for a sample of 75 countries.

The corresponding mixed graph is depicted in Figure 10, along with a description of all indicators. The graph is taken from Bollen (1989). Other hidden common causes affect the democratization indicators over time, but the nature of such hidden variables is irrelevant to the problem at hand: that is, the bi-directed edges are motivated by unmeasured causes of variability in the observed indicators that exist over time. For instance, the records of freedom of press in 1960 (Y_4) and 1965 (Y_8) co-vary due to other unmeasured factors not accounted by democratization factors.

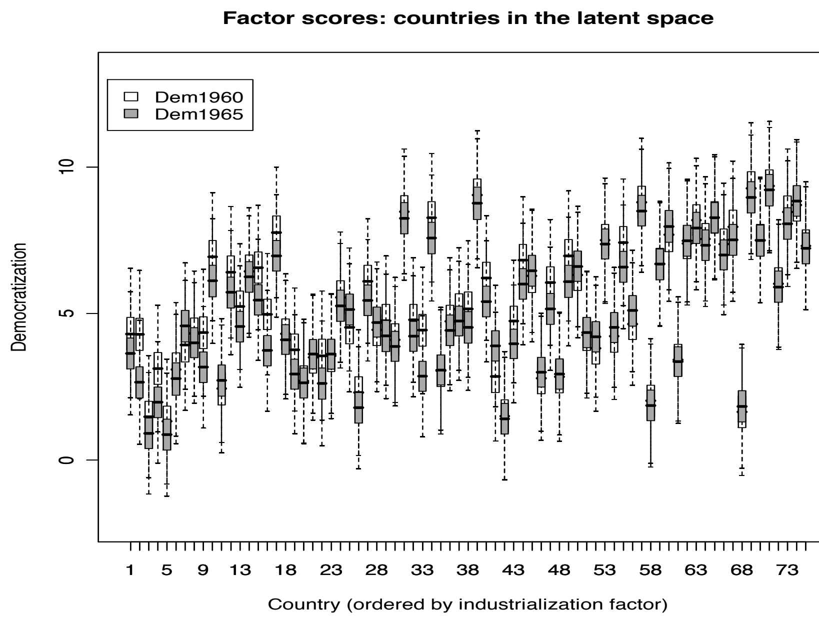


Figure 11: An embedding of 75 countries in a two-dimensional latent space: democratization level in 1960 and 1965. Boxplots of the Bayesian posterior distribution of the projection in the two dimensions are depicted in the vertical axis. Countries are arranged in the horizontal axis by the increasing order of their posterior expected industrialization level. Figure adapted from Dunson et al. (2005).

An example of Bayesian inference application is shown in Figure 11. Boxplots of the posterior values of *Democratization Level 1960* and *Democratization Level 1965* are generated. Dunson et al. (2005) use this information to, for instance, find clusters of countries in the latent space. An example of a cluster is the one formed by the bottom 16 countries in the industrialization level ranking: the growing trend of democratization levels after the first 16 countries is interrupted. This type of analysis might provide new insights to a polical scientist, for example, by revealing particular characteristics for such a group of nations.

7.1.1 EVALUATING THE MCMC ALGORITHM FOR DIFFERENT MODELS

In our analysis, we fix to unity the coefficients corresponding to the edges *Industrialization 1960* $\rightarrow Y_1$, *Democratization 1960* $\rightarrow Y_4$ and *Democratization 1965* $\rightarrow Y_8$, since the scale and sign of the latent variables is arbitrary. The intercept terms of the equations for Y_1, Y_4 and Y_8 are set to zero, since the mean of the latents is also arbitrary. The resulting model is identifiable.

We apply the Gibbs sampling procedure to three different models. The Gaussian DMG model as described in this paper, and two modified DAG models. The first DAG model is the one described by Dunson et al. (2005), where each bi-directed edge is substituted by an “ancillary” latent (as mentioned in Section 2.3). For instance, the pathway corresponding to $Y_4 \leftrightarrow Y_8$ is substituted by the chain $Y_4 \leftarrow D_{48} \rightarrow Y_8$, where D_{48} is unobserved. Dunson et al. further assume that all covariances due to such ancillary latents are positive. As such, the coefficients from D_{ij} into $\{Y_i, Y_j\}$ are set

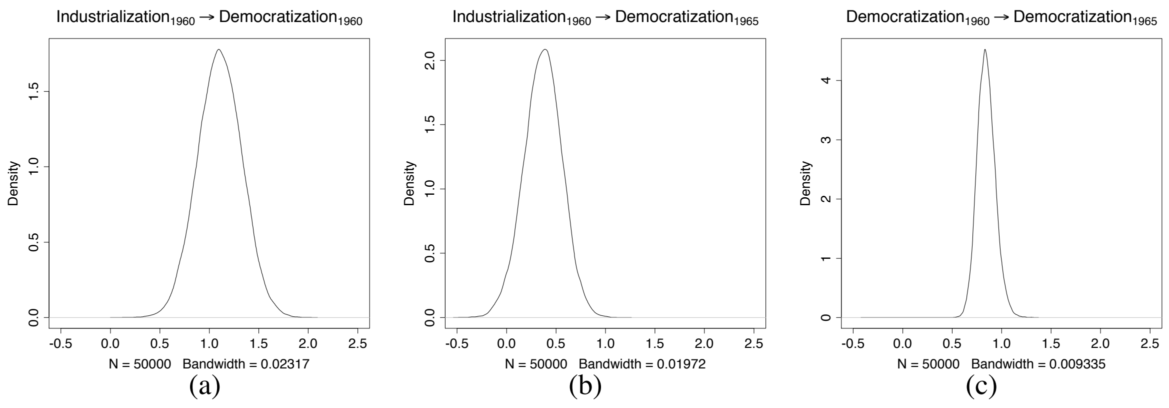


Figure 12: Posterior distribution of parameters associated with the respective edges in the industrialization/democratization domain. Smoothed posterior obtained using the output of our Gibbs sampler and the DENSITY function of R 2.6.0.

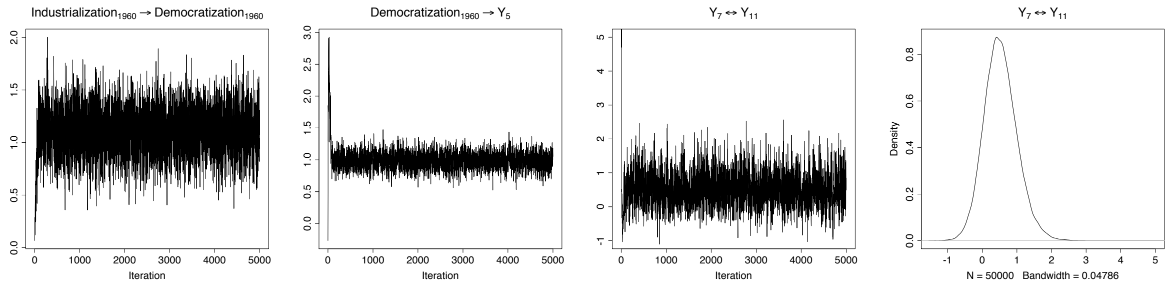
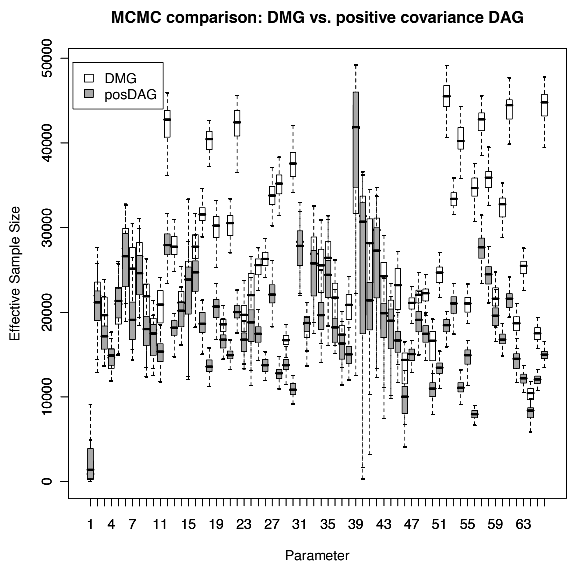


Figure 13: The first three plots show the initial 5,000 iterations of a run of the Gibbs sampling algorithm for the DMG model for three different parameters associated with edges in the graph. The last plot depicts the posterior distribution the error covariance associated with the edge $Y_7 \leftrightarrow Y_{11}$ (smoothed with the kernel density estimator from the statistical software R).

to unity, with the variance of D_{ij} corresponding to the residual covariance of $\{Y_i, Y_j\}$ given their parents. Means of ancillary latents are fixed at zero.

However, even for covariance matrices with positive covariances, this parameterization is not complete. This result is evident from the fact that the variances of Y_i and Y_j will both be larger than their covariance, which is not true of covariance matrices in general. For this particular problem, however, this extra restriction provides no measurable difference in terms of fitness. It does serve as a reminder, however, that "intuitive" parameterizations might hide undesirable constraints.

The second DAG model is an extension of the DAG model suggested by Dunson et al., the only difference being that the coefficients corresponding to edges $D_{ij} \rightarrow Y_i, i < j$, are free to vary (instead of being fixed to 1). In general, there are Gaussian DMG models that cannot be parameterized this way (Richardson and Spirtes, 2002). Notice also that because of chains such as *Democratization 1960* $\rightarrow Y_4 \leftrightarrow Y_8 \leftarrow$ *Democratization 1965*, the set of independence constraints in this graph can only be represented by a DAG if we include the ancillary latents D_{ij}. That is, there is no DAG with



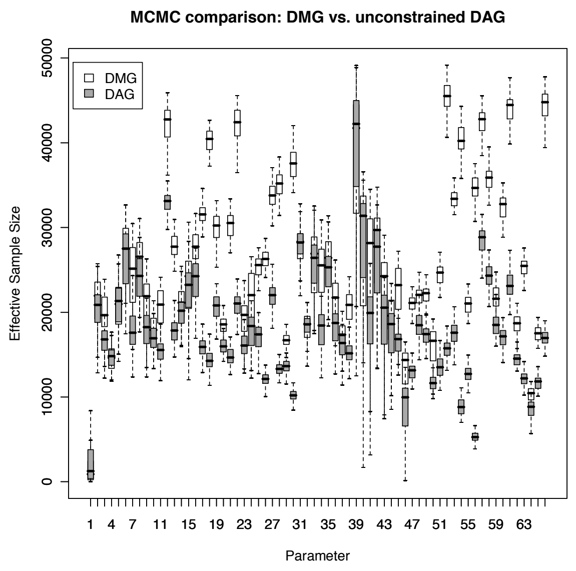


Figure 14: Comparison of the effective sample size of the MCMC algorithm applied to the three models, DMG, DAG with positive covariances (posDAG) and general DAG, as explained in the main text. The horizontal axis is the boxplot for each independent entry of the observed covariance matrix, 66 in total. The boxplots are obtained from 80 independent chains initialized randomly, where each chain runs for 50,000 iterations.

exactly the same set of independence constraints as the given DMG, unless ancillary latent variables are added.

We study the behavior of the MCMC algorithm for these three models.[12] It turns out that the mixing properties of the chain are considerably affected by the choice of model. Recall that, in the Gibbs sampling algorithm for the DMG model, a whole row of the error covariance matrix is sampled jointly conditioning on the other parameters. For the DAG models all entries of the error covariance matrix are independent and can be sampled jointly, but this requires *conditioning* on the ancillary latents, which do not exist in the DMG model and have to be sampled only in the DAG case.

For the majority of the covariance entries, the MCMC procedure mixed quite well, as illustrated in Figure 13. Notice how about 12% of the sampled DMG error covariances for $Y_7 \leftrightarrow Y_{11}$ were under zero, which could raise suspicion over the assumption of positive covariances. Autocorrelation is

12. A few technical notes: we used the priors suggested in Dunson et al. (2005), except that we changed the confidence in the prior of the covariance of the error terms $\mathbf{V}$ to be smaller (in order to minimize the influence of the priors in the models, since in this particular problem the DMG and DAG models are nearly likelihood equivalent but not posterior distribution equivalent – the priors belong to different families). We used 1 degree of freedom in our $\mathcal{G}$-Inverse Wishart, with the matrix parameter being the expected value of Dunson et al.'s prior. For the DAG models, we also used the $\mathcal{G}$-inverse Wishart prior for the error terms, but where all error terms are independent. For the DAG model with a free coefficient per ancillary latent, we assigned a standard Gaussian prior to such coefficients. The chains were initialized randomly by sampling standard Gaussians for the coefficients and latent variables. Error covariance matrices were initialized to diagonal matrices with diagonal entries sampled uniformly in $[1,2]$. Coefficient parameters were sampled jointly given the error covariance matrix and latent variables. Latent variables were also sampled jointly, given the parameters.

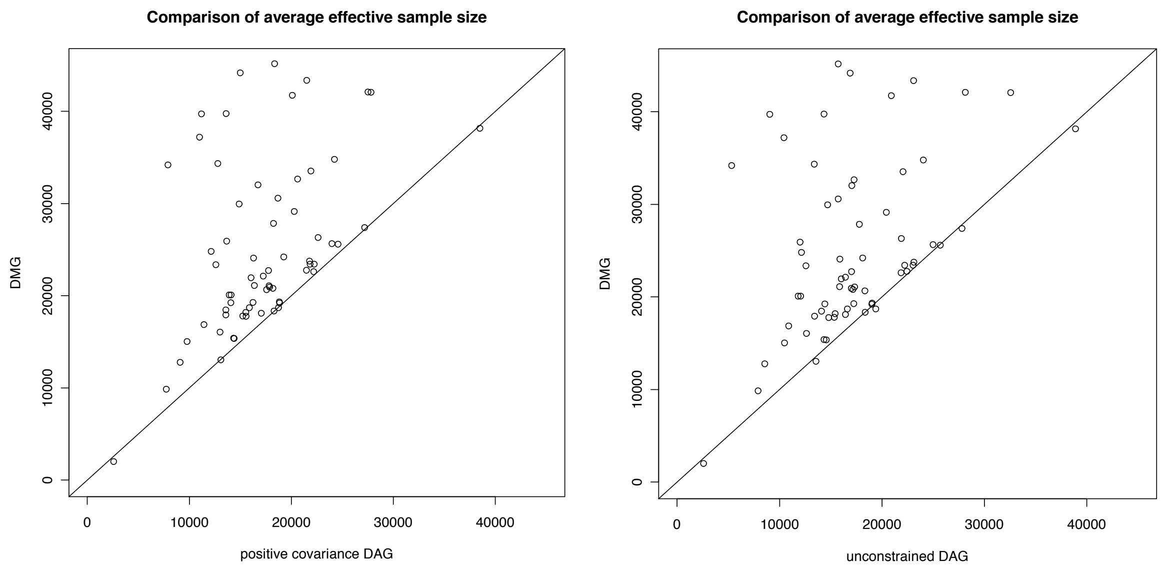


Figure 15: Comparison of the effective sample size of the MCMC algorithm applied to the three models. Here we plot the average effective sample sizes over 80 trials of 50,000 samples for each of the 66 entries of the covariance matrix. Points over the line indicate parameters where the DMG approach performed better.

essentially zero for most parameters at a lag of 50. The degree of autocorrelation, however, varied significantly between the DMG model and each DAG model. The chains for the DMG model mixed considerably better. To summarize such behavior, we calculated the effective sample size of the samples obtained from several chains. The parameters of interest in this comparison are the independent entries in the 11×11 dimensional observed covariance matrix. This is a total of 66 parameters. The effective sample size statistics were obtained by 80 independent chains of 50,000 samples each, for the three models. For each chain and each parameter, we compute the desired statistic using the EFFECTIVESIZE function implemented in the R package CODA, freely available in the Internet.

Results are summarized by boxplots in Figure 14. Parameters are ordered in the x-axis following the upper triangular covariance matrix, scanning it in the order $\{\sigma_{Y_1Y_1}, \sigma_{Y_1Y_2}, \dots, \sigma_{Y_1Y_{11}}, \sigma_{Y_2Y_2}, \dots, \sigma_{Y_{11}Y_{11}}\}$. White boxplots correspond to the distribution of effective sample size statistics with the DMG model across the 80 independent chains. Gray boxplots correspond to the two DAG variants. There is no significant difference between the behaviour of the Gibbs sampling procedure for the two DAG models. The procedure with the DMG model is clearly better behaved. As a summary statistic, the average effective sample size over 80 trials was steadly larger in the DMG outcome than in the positive DAG outcome (61 out of 66 parameters) and unconstrained DAG (59 out of 66). The comparison of averages is illustrated by Figure 15.

By caching the sufficient statistics of the data and factorizing the sampling procedure according to the districts of the graph, the running time for generating 50,000 samples out of the DMG model was of 34 seconds in a dual core Pentium IV 2.0 GHz. Depending on the connectivity of the bi-directed components of the graph and on the implementation of matrix inversion, sampling from the

DAG model might be faster than sampling from the DMG. In this particular study, sampling from the DAG models was substantially slower, an approximate average of 60 seconds for both variants. This can be explained by the fact that sampling latent variables is very expensive, especially considering that in the given DAG models all ancillary latents become dependent when conditioning on the data. To summarize, the DMG approach allowed for a complete parameterization with significantly better mixing properties, while still resulting in a faster MCMC procedure.

7.2 Structure Learning Applications

When trying to find a point estimate of graphical structures (i.e., returning a single graph that explains the data well), simple approaches such as testing for marginal independencies are reasonable learning algorithms under the Gaussian assumption. The Bayesian approach, however, allows one to compute odds and distributions over graphs and graph statistics, for example, the joint probability of small substructures (Friedman and Koller, 2003). Moreover, it is not clear how the independence test procedure controls for the predictive ability of the model, which is not a straightforward function of the edges that are selected due to the quantitative aspects of the dependencies.

We evaluate our Bayesian model selection contribution, focusing on the Monte Carlo sampler for bi-directed models. Jones et al. (2005) propose the following priors for graphs:

$$\mathcal{P}(\mathcal{G}|\beta) = \beta^{|E|}(1-\beta)^{0.5m(m-1)-|E|}$$

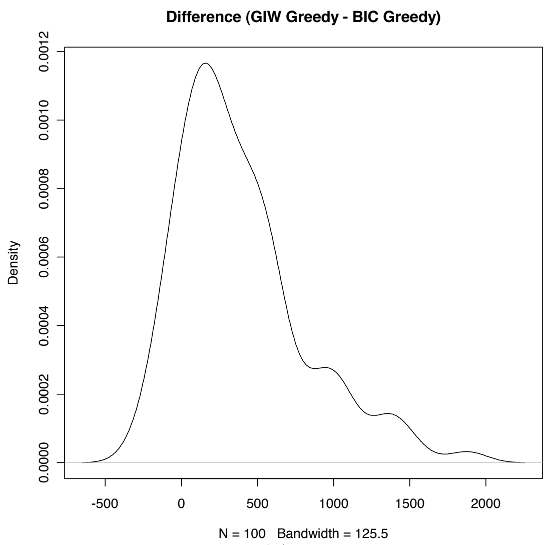
where β is a hyperparameter, $|E|$ is the number of edges in $\mathcal{G}$, and m is the number of nodes. As suggested by Jones et al., we choose $\beta = 0.5/(m-1)$, which puts more mass on graphs with $O(m)$ edges than the uniform prior.

We start with a brief synthetic study to compare the approach against a simple but effective approach based on the BIC approximation.[13] An experiment with gene expression data closes this subsection.

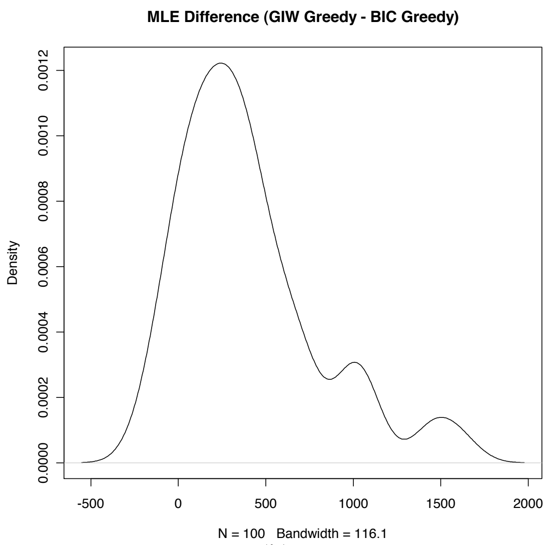
7.2.1 SYNTHETIC STUDIES

As a sanity check for the procedure, we generate synthetic 10-dimensional Gaussian data from models that are Markov with respect to a bi-directed graph. One hundred data sets of 50 datapoints each are generated, each coming from a different model.[14] We initially find a structure by marginal independence tests using the Fisher's Z statistic at a 0.05 level. From this starting point, we perform two searches: one using the BIC score, and the other using the marginal likelihood with a $\mathcal{G}$-*IW* prior.[15] Given the best model for each procedure, we evaluate the predictive log-likelihood on a test set of 2,000 points which are independently sampled for each of the 100 models.

13. The BIC approach is an asymptotically consistent score for selecting the maximum a posteriori Gaussian bi-directed graph model (Richardson and Spirtes, 2002).
14. The details of the simulated data are as follows: we start with DAG with no edges, with observed nodes $\{Y_1, Y_2, \dots, Y_{10}\}$ and hidden nodes $\{X_1, X_2, X_3, X_4\}$. Each individual edge $X_i \rightarrow Y_j$ is added with probability 0.35, and no other edges are allowed. We reject graphs with fewer than 10 edges. All coefficient parameters are sampled from a standard Gaussian, and variances from an uniform distribution in $[0,1]$. The model over $\mathbf{Y}$ corresponds to a bi-directed graph, where the edge $Y_i \leftrightarrow Y_j$ exists if and only if Y_i and Y_j have a common latent parent X_k in the DAG. We then store 50 samples for the $\mathbf{Y}$ variables in a data set. The procedure is repeated 100 times with different parameters and graphical structures each time. The average number of edges in the resulting simulation was of 18.4 edges per graph.
15. In both cases, we center the data at the empirical mean of the training set and assume the data to have been generated from a zero-mean Gaussian. The $\mathcal{G}$-Inverse Wishart is an empirical prior: a diagonal matrix with the training variance



(a)



(b)

Figure 16: The differente in predictive log-likelihood with models learned with the $\mathcal{G}$-*IW* prior and the best BIC models found by greedy search. Although the difference per point is small, it reflects a persistent advantage of the full Bayesian approach. Figure (a) shows the estimated density of the distribution of differences when predicting points using the Bayesian predictive log-likelihood. Since the BIC search method does not atempt to maximize the finite sample posterior distribution, we provide Figure (b) for completeness: in this case, the predictive log-likelihood for the BIC model was calculated using the maximum likelihood estimator. The difference hardly changes, and the fully Bayesian model still wins (density estimates produced by the DENSITY$(\cdot)$ function of R 2.6.0.).

The average difference in log-likelihood prediction[16] between the structure learned with the Bayesian prior and the BIC-induced model is depicted in Figure 16(a). This is computed by conditioning on the learned structures (fully Bayesian vs. BIC maximum a posteriori graphs) and marginalizing over the posterior of the parameters. The parameter priors are those used for the structure learning step. This might be unfair for the BIC procedure, since it is not designed to maximize the finite sample posterior: hence we also show in Figure 16(b) the results obtained when the predictions given the BIC model are obtained by using the maximum likelihood estimators of the

of each variable used as the diagonal. The number of degrees of freedom is set to 1. The search is a standard greedy procedure: we evaluate the marginal log-likelihood or BIC score for each graph that differs from the current candidate by one edge (i.e., graphs with one more or one fewer edge) and pick the one with the highest score. We stop when no improvement is possible.

16. In terms of incorrect edge additions and deletions, the procedures behave about the same: an average of one third of the edges is missed, and 7% of edges are incorrectly added (individual percentages are with respect to total number of possible mistakes in each graph). Unlike BIC, however, our procedure allows for different trade-offs by using different priors. It should also be pointed out that counting edge errors is just one possible measure. A more global quantitative score such as predictive log-likelihood takes into account, indirectly, the magnitude of the errors—although it is not a direct measure of model fitness.

parameters. The average difference in the first case is 400.07, and only slightly less for the second case (389.63). Most of the mass of the difference distribution is positive (85 out of 100 for the first case, 89 out of 100 in the second case), which passes a sign test at a 0.05 level. The difference is still relatively small, suggesting that informative prior knowledge might be necessary to produce substantially better predictions.

7.2.2 GENE EXPRESSION ANALYSIS

To illustrate the use of Bayesian model selection approaches, we analyse the gene expression data previously studied by Drton and Perlman (2007), also as Gaussian bi-directed models. As before, our goal will be to compare the predictive power of models learned by greedy search with BIC and greedy search with the Bayesian posterior.

The data consists of 13 gene expression measurements from a metabolic network. A total of 118 points is available. Using all data, the BIC-induced graph has 39 edges, while the finite sample posterior graph had 44. The same procedure used in the synthetic studies, for initializing graphs and choosing priors and centering the data, was applied in this case with two choices of degrees of freedom δ for the $\mathcal{G}$-*IW* prior: $\delta = 1$ and $\delta = 5$. Preliminary experiments where 90% of the samples are assigned to the training set showed a negligible difference between methods. We then generate 10 random splits of the data, 50% of them assigned to the training set. Predictive results using the MCMC method for evaluating the Bayesian predictions (with half a million samples) are shown in Table 1. The BIC graphs are by definition the same in the three sets of evaluation, but parameters are learned in three different ways (maximum likelihood point estimation and Bayesian averaging with two different priors). There is a steady advantage for the Bayesian approach, although a small one. Notice that using Bayesian averaging over parameters given the BIC graph improves prediction when compared to using the maximum likelihood point estimate, despite the simplistic choice of prior in this study. Notice also that the cases where the Monte Carlo method has small or no advantage over the BIC method were the ones where the maximum likelihood estimators produced their best results.

7.2.3 REMARKS

The procedure based on the sampler is doable for reasonably sized problem on the order of a few dozen variables in desktop machines. Further improvements are necessary for larger problems. One aspect that was not explored here was re-using previous computations when calculating the probability of a new candidate, in a way similar to the local updates in DAG models (Chickering, 2002). How to combine local updates with the ordering-based improved sampler of Section 6 is left as future research. Several practical variations can also be implemented, such as vetoing the inclusion of edges associated with high p-values in the respective independence tests. Such tabu lists can significantly shrink the search space.

It is important to evaluate how the Monte Carlo procedure for computing normalizing constants behaves in practice. For all practical purposes, the procedure is an importance sampler and as such is not guaranteed to work within a reasonable amount of time for problems of high dimensionality (MacKay, 1998). We can, however, exploit the nature of the problem for our benefit. Notice that the procedure depends upon a choice of ordering $\prec$ for the variables. Different orderings correspond in general to *different importance distributions*. We can play with this feature to choose an suitable ordering. Consider the following algorithm for choosing an ordering given a bi-directed graph $\mathcal{G}$:

	MLE	$\delta = 1$		$\delta = 5$	
Folder	BIC	BIC	MC	BIC	MC
1	-6578.44	-6382.45	-6308.19	-6342.82	**-6296.14**
2	-6392.67	-6284.64	**-6277.94**	-6279.54	-6285.26
3	-8194.54	-6567.89	**-6433.51**	-6553.88	-6452.15
4	-6284.00	-6265.16	-6285.77	**-6252.54**	-6258.42
5	-9428.00	-6473.93	**-6400.51**	-6483.43	-6469.45
6	-7111.45	-6573.85	-6572.74	-6528.76	**-6513.02**
7	-6411.43	-6329.53	-6317.18	-6313.05	**-6309.18**
8	-6350.44	-6319.87	**-6295.19**	-6299.53	-6297.80
9	-6374.31	-6307.13	-6308.21	**-6297.47**	-6304.25
10	-7247.82	-6584.96	-6468.51	-6528.61	**-6444.55**

Table 1: Results for the 10 random splits of the gene expression data, with 50% of the points assigned to the training set. The first column shows the predictive log-likelihood for the graph learned with the BIC criterion and parameters fit by maximum likelihood. The next two columns show predictive log-likelihood results for the graphs learned with BIC and the Monte Carlo (MC) marginal likelihood method using a $\mathcal{G}$-*IW* prior with degrees of freedom $\delta = 1$. The last two columns are the results of a prior where $\delta = 5$. Best results in bold.

1. Let $\prec$ be an empty queue.
2. Let $\mathcal{G}'$ be the graph complement of $\mathcal{G}$, that is, the graph where $\{Y_i, Y_j\}$ are neighbors if and only if they are not adjacent in $\mathcal{G}$.
3. Let $\mathcal{C}$ be an arbitrary maximum clique of $\mathcal{G}'$. Add all elements of $\mathcal{C}$ to the end of $\prec$ in any arbitrary order.
4. For each pair $\{Y_i, Y_j\}$, not intersecting $\mathcal{C}$, such that the path $Y_i \leftrightarrow Y_k \leftrightarrow Y_j$ exists in $\mathcal{G}$ and $Y_k \in \mathcal{C}$, add the edge $Y_i \leftrightarrow Y_j$ to $\mathcal{G}$.
5. Remove all elements $Y_k \in \mathcal{C}$ from $\mathcal{G}$, including any edge into Y_k.
6. Iterate Steps 2-5 until $\mathcal{G}$ is an empty graph.

The resulting queue $\prec$ is an ordering that attempts to maximize the number of variables that are marginally independent given their common predecessors. This is just one possibility to simplify the importance sampling distribution: perfect orderings and the approaches for simplifying maximum likelihood estimation described by Drton and Richardson (2008b) could be adapted to provide even better orderings, but we leave this as future work.[17]

17. In our actual implementation used in the experiments in this Section, we implemented an even simpler approach: instead of finding maximum cliques, we start to build a clique from a particular node, "greedily" adding other nodes to the clique according to the column order of the data set. Each node generates a candidate clique, and we pick an arbitrary clique of maximal size to be our new set $\mathcal{C}$.

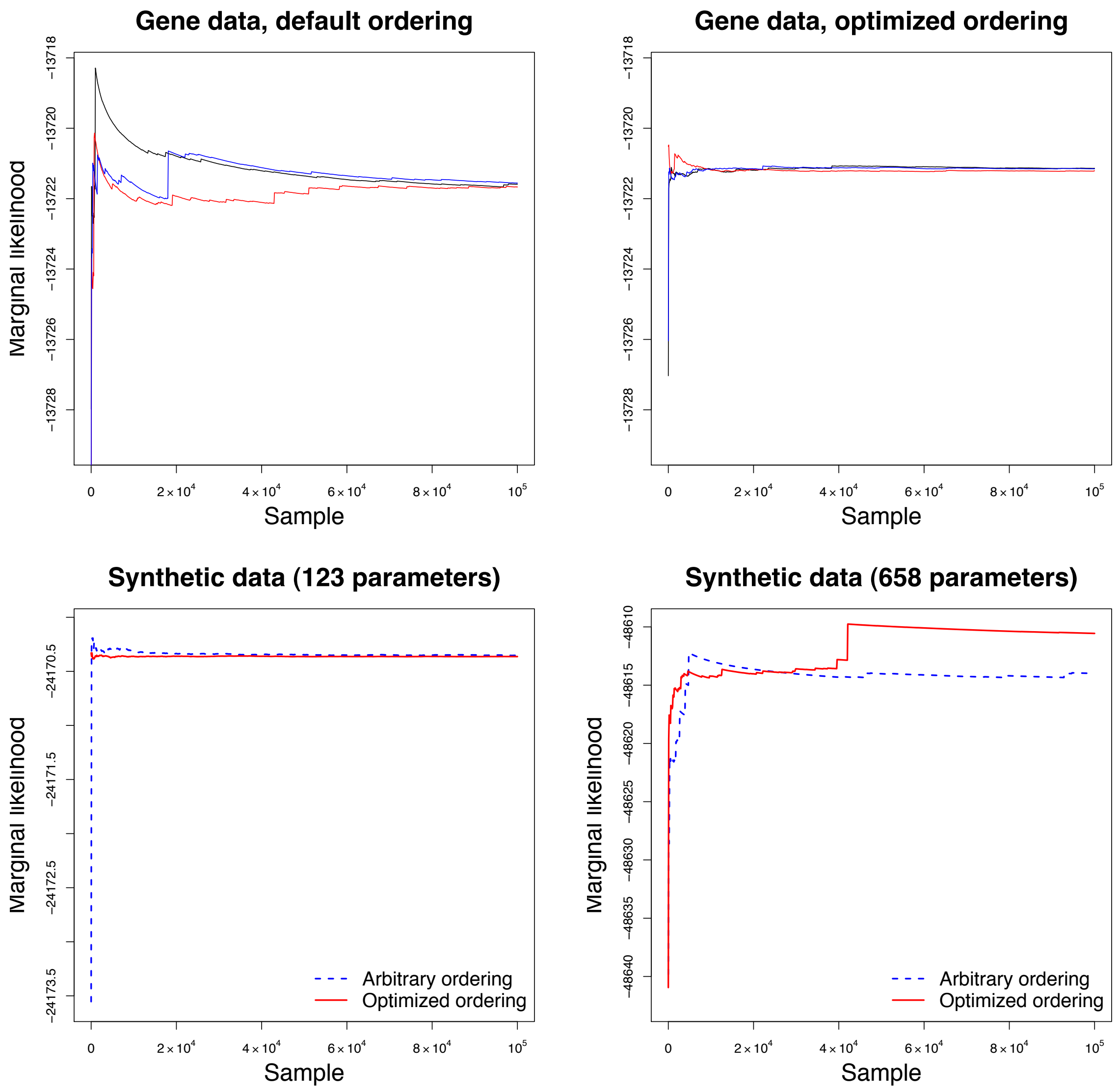


Figure 17: An evaluation on the stability of the Monte Carlo normalizing function procedure. The top row depicts the marginal likelihood estimates for the gene problem using two different distributions implied by two different orderings, as explained in the text. Experiments with synthetic data are shown in the bottom, and the bottom-right figure illustrates major differences.

Figure 17 illustrates the difference that a smart choice of ordering can make. The top left graph in Figure 17 depicts the progress of the marginal likelihood Monte Carlo estimator for the gene expression problem using the graph given by the hypothesis testing procedure. The model has 55 parameters. We obtain three estimates, each using a sample of 100,000 points, which allows us to observe how the estimates change at the initial stages. The variable ordering in this case is the ordering present in the original database (namely, DXPS1, DXPS2, DXPS3, DXR, MCT, CMK, MECPS, HDS, HDR, IPPI1, GPPS, PPDS1 and PPDS2). The top right graph shows three runs using the optimized ordering criterion. Convergence is much faster in this case, and both samplers agree on the normalizing constant estimate.

As an illustration of the power of the procedure and its limitations, we generated a synthetic sample of 1,000 training points from a graph with 25 nodes, using the same procedure of Section 7.2.1. A run of two different samplers is shown at the bottom left of Figure 17. They are seemingly well-behaved, the ratio between the largest and median weight being at the order of one hundred in the "optimally" ordered case. In contrast, the bottom right corner of Figure 17 illustrates the method with a covariance matrix of 50 random variables and 1,000 training points. Notice this is a particularly dense graph. Much bigger jumps are observed in this case and there is no clear sign of convergence at 100,000 iterations.

While there is no foolproof criterion to evaluate the behavior of an importance sampler, the relationship between orderings provides a complementary technique: if the normalizing constant estimators vary substantially for a given set of random permutations of the variables, then the outcomes are arguably not to be trusted even if the respective estimators appear to have converged.

Concerning the choice of priors, in this Section we exploited empirical priors. The $\mathcal{G}$-Inverse Wishart matrix hyperparameter is a diagonal matrix where variance entries are the sample variances. While this adds an extra bias towards diagonal matrices, at least in our experiments we performed close to or better than other approaches—it is however not clear whether we could have done much better. It is still an open question which practical "default" hyperparameters will prove useful for the $\mathcal{G}$-IW. Elicitation of subjective priors in the context of structural equation models can benefit from pre-existing work on Bayesian regression, although again practical matters might be different for the $\mathcal{G}$-IW. Dunson et al. (2005) describe some limitations of default priors for structural equation models. A thorough evaluation of methods for eliciting subjective priors is out of the context of this work, but existing work on inverse Wishart elicitation provides a starting point (Al-Awadhi and Garthwaite, 1998). As in the case of the inverse Wishart, the $\mathcal{G}$-Inverse Wishart has a single hyperparameter for specifying degrees of freedom, a limitation which might motivate new types of priors (Brown et al., 1993).

7.3 Discrete Data Applications

We now show results on learning a discrete distribution that factorizes according to a mixed graph. Drton and Richardson (2008a) describe applications on real-world binary data modeled according to bi-directed graphs. The empirical contingency tables for two studies can be found in the corresponding technical report (Drton and Richardson, 2005). Drton and Richardson used a complete parameterization for bi-directed binary models and a maximum likelihood estimation procedure. In this section, we analyze these two data sets to illustrate the behavior of our Bayesian procedure using the probit model. Our model imposes probit constraints that are not enforced by Drton and Richardson, but it allows us to obtain Bayesian credible intervals and predictions.

The graphs used in the two studies are depicted in Figure 18. The first problem is a study on the dependence between alcoholism and depression, as shown in Figure 18(a). A data point is collected for a given pair of mono-zygotic twins. For each sibling S_i, it is recorded whether S_i is/is not alcoholic (A_i), and whether S_i suffers/does not suffer from depression (D_i). The hypothesis encoded by the graph is that alcoholism and depression do not share a common genetic cause, despite A and

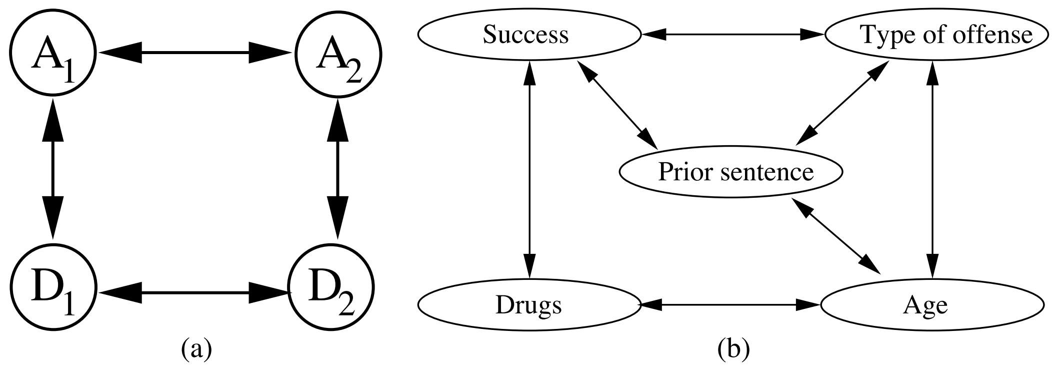


Figure 18: Two learning problems with discrete data. In (a), the graph shows dependencies concerning alcoholism (A_i) and depression (D_i) symptoms for paired twins $\{1,2\}$. In (b), a model for dependencies among features of a study on parole appeals, including the success of the parole, if the type of offense was a person offense or not, and if the offender had a dependency on drugs and was over 25 years old. All variables in these studies are binary and further details and references are provided by Drton and Richardson (2008a).

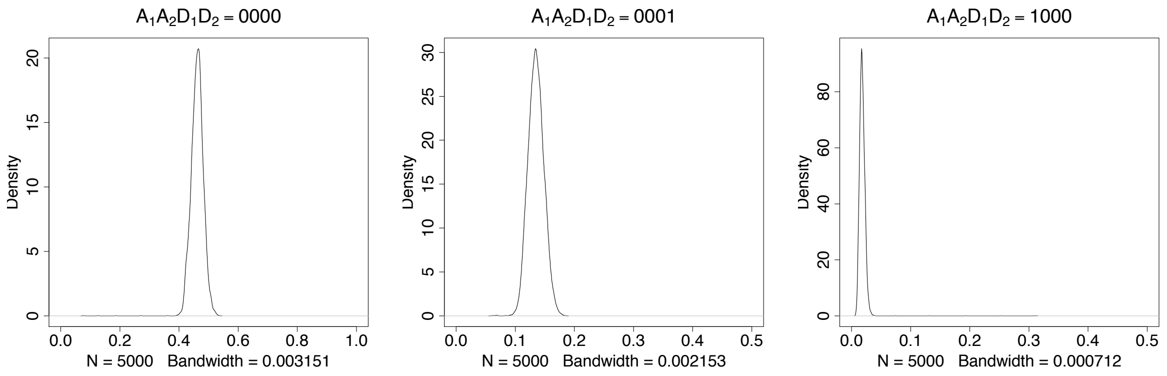


Figure 19: Posterior distribution of some of the marginal contingency table entries for the twin model.

D having some hidden (but different) genetic causes. If A and D did have genetic common causes, one would expect that the edges $A_1 \leftrightarrow D_2$ and $A_2 \leftrightarrow D_1$ would be also required. The compounded hypothesis of marginal independencies for A_i and D_j, $i \neq j$, can be tested jointly by testing a bi-directed model. Notice that no reference to particular genetic hidden causes of alcoholism and depression is necessary, which again illustrates the power of modeling by marginalizing out latent variables.

The second study, as shown in Figure 18(b), concerns the dependencies among several variables in an application for parole. The model implies, for instance, that the success of a parole application (*Success* node, in the Figure) is independent of the age of the offender being under 25 (*Age* node). However, if it is known that the offender had a prior sentence, these two variables become dependent (through the path *Success* $\leftrightarrow$ *Prior sentence* $\leftrightarrow$ *Age*). As reported by Drton and Richardson, their

Entry	Estimates			Entry	Estimates		
$A_1A_2D_1D_2$	E[$\Theta\|\mathcal{D}$]	MLE	uMLE	$A_1A_2D_1D_2$	E[$\Theta\|\mathcal{D}$]	MLE	uMLE
0000	0.461	0.461	0.482	1000	0.018	0.018	0.013
0001	0.136	0.138	0.134	1001	0.003	0.004	0.007
0010	0.157	0.159	0.154	1010	0.021	0.020	0.013
0011	0.097	0.096	0.085	1011	0.009	0.009	0.015
0100	0.032	0.032	0.025	1100	0.008	0.010	0.005
0101	0.022	0.021	0.015	1101	0.003	0.002	0.003
0110	0.007	0.008	0.012	1110	0.003	0.005	0.007
0111	0.012	0.012	0.017	1111	0.006	0.005	0.012

Figure 20: The posterior expected value of the 16 entries in the twin study table ($E[\Theta|\mathcal{D}]$). Results generated with a chain of 5,000 points. We also show the maximum likelihood estimates of Drton and Richardson (MLE) and the maximum likelihood values obtained using an unconstrained model (uMLE). Despite the probit parameterization, in this particular study there is a reasonable agreement between the Bayesian estimator and the estimator of Drton and Richardson.

binary bi-directed model passes a significance test. Drton and Richardson also attempted to learn an undirected (Markov) network structure with this data, but the outcome was a fully connected graph. This is expected, since Markov networks cannot represent marginal independencies unless the graph is disconnected, which would introduce all sorts of other independencies and possibly not fit the data well. If many marginal independencies exist in the data generating process, Markov networks might be a bad choice of representation. For problems with symmetries such as the twin study, DAGs are not a natural choice either.

7.3.1 RESULTS

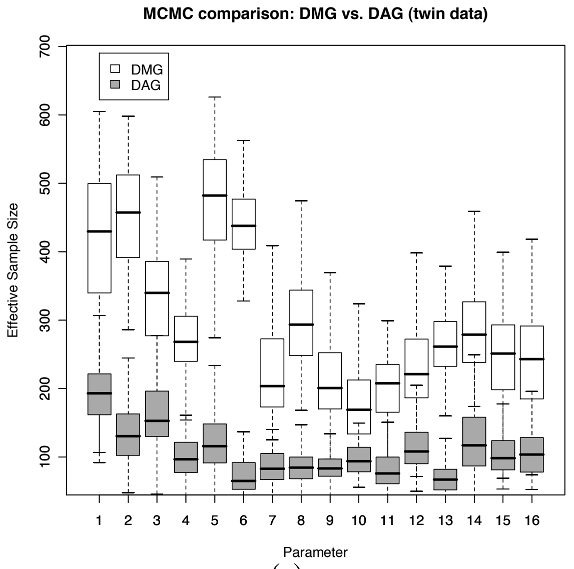
For the twin data problem, we used a simple prior for the covariance matrix of the underlying latent variables: a $\mathcal{G}$-inverse Wishart with 1 degree of freedom and a complete covariance with a value of 2 for each element in the diagonal and 1 outside the diagonals. Thresholds are fixed at zero, since we have binary data. We present the expected posterior values of the contingency table entries in Figure 20. The outcome is essentially identical to the maximum likelihood estimates of Drton and Richardson despite the probit parameterization. Moreover, with our procedure we are able to generate Bayesian confidence intervals, as illustrated in Figure 19. The results are very stable for a chain of 1,000 points.

For the parole data, we used a $\mathcal{G}$-inverse Wishart prior for the covariance matrix of underlying variables $\mathbf{Y}^\star$ with 1 degree of freedom and the identity matrix as hyperparameters. We compare the effective sample size of the Gibbs sampler for our DMG model and the DAG model obtained by using the ancillary latent parameterization of Section 7.1 for the underlying latent variable covariance matrix.[18] Boxplots for the 16 contingency table entries of the twin network and the 32 entries of the parole study are shown in Figure 21. The setup is the same as in the democratization and

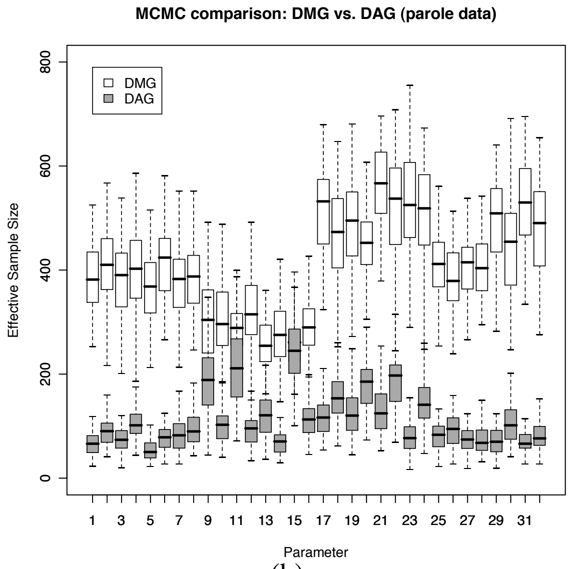
18. The priors used are as follows: the ancillary representation was given a prior with mean 1 and variance 1 for the coefficients $X_{ij} \rightarrow Y_j^\star$, for $j > i$, and set constant to 1, if $i < j$. The means of the ancillary latents were fixed at

industrialization experiment, where we run 80 independent chains and plot the distribution of the effective sample sizes to measure the mixing time. We ran a shorter chain of 2,000 points, since computing the contingency table entries is expensive.

There is a substantial difference in effective sample size for the parole study. Notice that we are comparing MCMC samples for the entries in the contingency table, which in the DAG case requires integrating out not only the underlying latent variables implicit in the probit parameterization, but also the ancillary latents that account for the bi-directed edges. This hierarchy of latent variables, which does not exist in the DMG case, causes a considerable increase on autocorrelation of the chain compared to the DMG model. The standard DMG parameterization can be seen as a way of obtaining a collapsed Gibbs sampler, where the parameterization by construction reflects latent variables that were analytically marginalized.



(a)



(b)

Figure 21: Comparison of effective sample sizes for the twin data (a) and parole data (b). 80 independent chains of 2,000 points were obtained using the Gibbs sampling algorithm, and the respective box-plots shown above. The Markov chain with the DMG approach easily dominates the DAG one. For the parole data, the average effective sample size for the DAG was as low as 60 points.

8. Conclusion

Directed mixed graph models are a generalization of directed graph models. Whenever a machine learning application requires directed graphs, one should first consider whether directed mixed graphs are a better choice of representation instead. DMGs represent conditional independencies of DAGs where hidden variables have been marginalized out. Given that in most applications it is

0. Variance parameters were given $(0.5,0.5)$ inverse gamma priors, which approximately matches the priors in the DMG model.

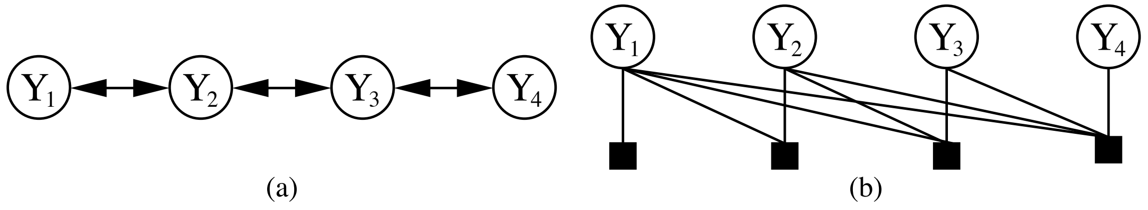


Figure 22: In (a), a simple bi-directed chain with four random variables. In (b), the respective factor graph that is obtained from a Bartlett parameterization using the ordering $\prec \equiv \{Y_1, Y_2, Y_3, Y_4\}$. In this case, the factors are $p(Y_1) \times p(Y_2|Y_1) \times p(Y_3|Y_1, Y_2) \times p(Y_4|Y_1, Y_2, Y_3)$. A different choice of ordering (e.g., the perfect ordering) could provide simpler factors on average, but the presence of a factor linked to all variables is unavoidable.

unlikely that all relevant variables are known, DMGs are a natural representation to use. In this paper, we introduced priors and inference algorithms for Bayesian learning with two popular families of mixed graph models: Gaussian and probit. We discussed some implementations and approximations to scale up algorithms. We showed examples of applications with real data, and demonstrated that Bayesian inference in Gaussian and probit DMG models using MCMC can have substantially faster mixing than in comparable DAGs.

It is part of the machine learning folklore that factor graphs can subsume directed networks. In an important sense, this is known not to be true: undirected and factor graphs only allow for monotonic independence models, where explaining away is ruled out. This excludes a vast number of realistic, non-monotonic, models. While factor graphs are perhaps the *data structures* of choice for general message-passing algorithms (e.g., Yedidia et al., 2005), they are far from being universal *modeling languages* for independencies.

What is true is that for any distribution that is Markov with respect to a DAG or DMG there is at least one corresponding factor graph model, but this is a vacuous claim of little interest: any distribution can be represented by a single-factor model involving all variables. Some will *require* a factor with all variables, even under the presence of a large number of independence constraints. For instance, a factor graph corresponding to any given bi-directed chain will necessarily include a factor node adjacent to all variable nodes, as illustrated in Figure 22. When parameterizing a distribution with many marginal independencies (e.g., a bi-directed tree), the respective factor graph would be no more than an unhelpful drawing. A better strategy for solving real-world problems is to define a family of models according to the (directed/undirected/factor) graphs of choice, and let the inference algorithm decide which re-expression of the model suits the problem. This has been traditional in graphical modeling literature (Lauritzen, 1996). The strategy adopted in this paper followed this spirit.

An alternative has been recently introduced by Huang and Frey (2008). This paper discusses graphical families of marginal independence constraints (essentially identical to bi-directed graphs, although other types of constraints might implicitly follow from the parameterization). Models are parameterized using a very different strategy. The idea is to parameterize cumulative distribution functions (CDFs) instead of densities or probability mass functions. A simple factorization criterion can be defined in the space of CDFs, but densities have to be computed by a novel message-passing scheme. The particular application discussed by Huang and Frey (2008) could in principle be approached using the Gaussian bi-directed probit model of Section 5, but the parameterization in Huang and Frey (2008) does not need to rely on Gaussian distributions. It is not clear, however, how to efficiently perform Bayesian inference in this case and which constraints are implicitly implied by the different choices of parameterization. The different perspective given by products of CDFs is novel and promising. It should point out to new directions in mixed graph modeling.

The structural equation modeling literature also describes several pragmatic ways of specifying non-linearities in the structural equations (Lee, 2007). Less common is the specification of non-Gaussian models for the joint density of the error terms. Silva and Ghahramani (2009) introduce a flexible mixture of Gaussians approach for models of marginal independence. There is a need on how to combine this approach with flexible families of structural equations in a computationally efficient way. Also, models with non-additive error terms remain to be explored.

Current interest in estimating sparse statistical models has lead to approaches that estimate structured covariance matrices (e.g., Bickel and Levina, 2008). This development could also lead to new families of priors. In particular, different matrix decompositions have motivated different ways of specifying priors on covariance matrices. For instance, Chen and Dunson (2003) propose a modified Cholesky decomposition for the covariance matrix of random effect parameters: standard deviations are parameterized separately with a prior that puts positive mass on zero variances (effectively allowing the random effect to be neutralized). Wong et al. (2003) describe a prior for inverse correlation matrices that is uniform conditioned on the number of structural zeros. Metropolis-Hastings schemes are necessary in this case.

Shrinkage methods have also been applied to the estimation of covariance matrices. A common approach, shrinkage towards a diagonal matrix (e.g., Daniels and Kass, 1999), could be generalized towards some sparse matrix corresponding to a bi-directed graph. Although shrinkage will not generate structural zeros in the resulting matrix, allowing for sparse shrinkage matrices other than the identity matrix could be interesting in prediction problems.

Some approaches can exploit an ordering for the variables, which is natural in some domains such as time-series analysis. While the $\mathcal{G}$-Inverse Wishart is invariant to a permutation of the variables, new types of priors that exploit a natural variable ordering should be of interest, as in the original work of Brown et al. (1993) that motivated our approach.

Other directions and applications are suggested by recent papers:

- **learning measurement models:** the industrialization and democratization problem of Section 7.1 provides an example of a measurement model. In such a family of problems, observed variables are children of latent variables, and connections from latents to observables define the measurement model. Sparsity in the measurement can be exploited to allow for more general dependencies connecting latent variables. One role of the bi-directed component is to allow for extra dependencies connecting observed variables that are not accounted by the explicit latent variables in the model. Silva et al. (2006) describes a learning algorithm for mixed graph measurement models using the "ancillary" parameterization. The natural question is which alternative optimization strategies could be used and how to scale them up;

- **structural and relational learning:** in prediction problems where given an input vector $\mathbf{X}$ we have to predict an output vector $\mathbf{Y}$, the dependence structure of $\mathbf{Y}$ given $\mathbf{X}$ can also lie within the directed mixed graph family. Silva et al. (2007) introduces mixed graph models within the context of relational classification, where $\mathbf{Y}$ are labels of different data points not independently distributed. In such a class of problems, novel kinds of parameterization are necessary since the dimensionality of the covariance matrix increases with the sample size. Structural features of the graph are used to propose different parameterizations of the dependencies, and many other alternatives are possible;

- **causal inference:** mixed graphs have been consistently used as a language for representing causal dependencies under unmeasured confounding. Zhang (2008) describes recent advances in identifying causal effects with ancestral graphs. Algorithms for learning mixed graph structures are described by Spirtes et al. (2000) and the recent advances in parameterizing such models should result in new algorithms;

Many challenges remain. For instance, more flexible models for DMG discrete models are being developed (Drton and Richardson, 2008a), but for large graphs they pose a formidable computational problem. An important question is which other less flexible, but more tractable, parameterizations could be used, and which approximation algorithms to develop. The probit family discussed here was a choice among many. The parameterization by Drton and Richardson (2008a) could be a starting point for trading-off flexibility and computational effort. And while it is true that Gaussian copula models (Pitt et al., 2006) can be adapted to generalize the approach introduced here, it remains to be seen if other copula parameterizations easily lead to DMG models.

Acknowledgments

We thank the anonymous reviewers for their suggestions, Kenneth Bollen for providing us with the industrialization and democratization data set, and Robert Gramacy for helpful discussions. An earlier version of this paper (Silva and Ghahramani, 2006) appeared in the proceedings of the Uncertainty in Artificial Intelligence conference. This work was funded by the Gatsby Charitable Foundation and a EPSRC grant #EP/D065704/1.

Appendix A. Deriving the Sampling Distribution for the Monte Carlo Computation of Normalizing Constants

We give here the details on how to derive the sampling distribution used for computing normalizing constants $I_{\mathcal{G}}(\delta, \mathbf{U})$, as described in Section 3.2.2.

Let $\mathbf{A}_i \equiv \Sigma_{sp_{\prec}(i),nsp_{\prec}(i)}\Sigma^{-1}_{nsp_{\prec}(i),nsp_{\prec}(i)}$. Recall from Equation (7) that $\mathcal{B}_{i,nsp_{\prec}(i)} = -\mathcal{B}_{i,sp_{\prec}(i)}\mathbf{A}_i$. The original density $p(\mathcal{B}_i \mid \gamma_i)$, as given by Lemma 1, is a multivariate Gaussian with the following kernel:

$$\exp\left(-\frac{1}{2\gamma_i}\left[\begin{array}{c}\mathcal{B}^{\mathsf{T}}_{i,sp_{\prec}(i)} - \mathbf{M}_{sp_{\prec}(i)} \\ \mathcal{B}^{\mathsf{T}}_{i,nsp_{\prec}(i)} - \mathbf{M}_{nsp_{\prec}(i)}\end{array}\right]^{\mathsf{T}}\left[\begin{array}{cc}\mathbf{U}_{ss} & \mathbf{U}_{sn} \\ \mathbf{U}_{ns} & \mathbf{U}_{nn}\end{array}\right]\left[\begin{array}{c}\mathcal{B}^{\mathsf{T}}_{i,sp_{\prec}(i)} - \mathbf{M}_{sp_{\prec}(i)} \\ \mathcal{B}^{\mathsf{T}}_{i,nsp_{\prec}(i)} - \mathbf{M}_{nsp_{\prec}(i)}\end{array}\right]\right) \tag{22}$$

where $\mathbf{U}_{\{i-1\},\{i-1\}}$ in Lemma 1 was rearranged above as the partitioned matrix in (14). The pair $\{\mathbf{M}_{sp_{\prec}(i)}, \mathbf{M}_{nsp_{\prec}(i)}\}$ corresponds to the respective partition of the mean vector $\mathbf{M}_i$. Plugging in the expression for $\mathcal{B}_{i,nsp_{\prec}(i)}$ in (22), we obtain the modified kernel

$$\exp\left(-\frac{1}{2\gamma_i}\left[\begin{array}{c}\mathcal{B}_{i,sp_{\prec}(i)}^{\mathsf{T}}-\mathbf{M}_{sp_{\prec}(i)}\\ -\mathbf{A}_i^{\mathsf{T}}\mathcal{B}_{i,sp_{\prec}(i)}^{\mathsf{T}}-\mathbf{M}_{nsp_{\prec}(i)}\end{array}\right]^{\mathsf{T}}\left[\begin{array}{cc}\mathbf{U}_{ss} & \mathbf{U}_{sn}\\ \mathbf{U}_{ns} & \mathbf{U}_{nn}\end{array}\right]\left[\begin{array}{c}\mathcal{B}_{i,sp_{\prec}(i)}^{\mathsf{T}}-\mathbf{M}_{sp_{\prec}(i)}\\ -\mathbf{A}_i^{\mathsf{T}}\mathcal{B}_{i,sp_{\prec}(i)}^{\mathsf{T}}-\mathbf{M}_{nsp_{\prec}(i)}\end{array}\right]\right) \qquad (23)$$

which can be rewritten as

$$\begin{array}{rcl} p_b(\mathcal{B}_{i,sp_{\prec}(i)};\mathbf{K}_i\mathbf{m}_i,\gamma_i\mathbf{K}_i) & \times & (2\pi)^{\#sp_{\prec}(i)/2}|\gamma_i|^{\#sp_{\prec}(i)/2}|\mathbf{K}_i(\Phi_{i-1})|^{1/2}\\ & \times & \exp\left\{-\frac{1}{2}\gamma_i^{-1}\mathcal{U}_i\right\}\end{array} \qquad (24)$$

where $\#sp_{\prec}(i)$ is the size of set $sp_{\prec}(i)$, $p_b(\cdot;\alpha,\Sigma)$ is the density function of a multivariate Gaussian distribution with mean α and covariance Σ, $\mathbf{K}_i(\Phi_{i-1})\equiv\mathbf{K}_i$ to emphasize the contribution of previous parameters, and

$$\begin{array}{rcl} \mathbf{m}_i & = & (\mathbf{U}_{ss}-\mathbf{A}_i\mathbf{U}_{ns})\mathbf{M}_{sp_{\prec}(i)}+(\mathbf{U}_{sn}-\mathbf{A}_i\mathbf{U}_{nn})\mathbf{M}_{nsp_{\prec}(i)},\\ \mathbf{K}_i^{-1} & = & \mathbf{U}_{ss}-\mathbf{A}_i\mathbf{U}_{ns}-\mathbf{U}_{sn}\mathbf{A}_i^{\mathsf{T}}+\mathbf{A}_i\mathbf{U}_{nn}\mathbf{A}_i^{\mathsf{T}},\\ \mathcal{U}_i & = & \mathbf{M}_i^{\mathsf{T}}\mathbf{U}_{\{i-1\},\{i-1\}}\mathbf{M}_i-\mathbf{m}_i^{\mathsf{T}}\mathbf{K}_i\mathbf{m}_i.\end{array}$$

If $sp_{\prec}(i)=\emptyset$, it follows that $\mathcal{B}_i=\mathcal{B}_{i,nsp_{\prec}(i)}=0$. The kernel (23) reduces to $\exp(-0.5\mathcal{U}_i/\gamma_i)$, and $\mathcal{U}_i\equiv\mathbf{M}_i^{\mathsf{T}}\mathbf{U}_{\{i-1\},\{i-1\}}\mathbf{M}_i$. If $nsp_{\prec}(i)=\emptyset$, then the expression for the kernel does not change ($\mathcal{U}_i\equiv 0$), and Equation (24) corresponds to the original kernel in Equation (11).

Inserting the re-expressed kernel into the original function (11), we obtain

$$p_b(\mathcal{B}_{i,sp_{\prec}(i)};\mathbf{K}_i\mathbf{m}_i,\gamma_i\mathbf{K}_i)p_g\left(\gamma_i;\frac{\delta+i-1+\#nsp_{\prec}(i)}{2},\frac{u_{ii.\{i-1\},\{i-1\}}+\mathcal{U}_i}{2}\right)f_i(\Phi_{i-1})$$

where $p_g(\cdot;\alpha,\beta)$ is an inverse gamma density function and

$$\begin{array}{rcl} f_i(\Phi_{i-1}) & \equiv & (2\pi)^{-\frac{(i-1)-\#sp_{\prec}(i)}{2}}|\mathbf{K}_i(\Phi_{i-1})|^{1/2}|\mathbf{U}_{\{i-1\},\{i-1\}}|^{1/2}\\ & \times & \dfrac{(u_{ii.\{i-1\},\{i-1\}}/2)^{(\delta+i-1)/2}}{\Gamma((\delta+i-1)/2)}\dfrac{\Gamma((\delta+i-1+\#nsp_{\prec}(i))/2)}{((u_{ii.\{i-1\},\{i-1\}}+\mathcal{U}_i)/2)^{(\delta+i-1+\#nsp_{\prec}(i))/2}}.\end{array}$$

Appendix B. Variational Updates for Gaussian Mixed Graph Models

The variational updates for the coefficient and intercept parameters are essentially identical to their joint conditional distribution given $\mathbf{V}$ and $\mathbf{X}$, where occurrences of $\mathbf{V}$ and $\mathbf{X}$ are substituted by expectations $\langle\mathbf{V}^{-1}\rangle_{q(\mathbf{V})}$ and $\langle\mathbf{X}\rangle_{q(\mathbf{X})}$, respectively. Let $\mathcal{V}_{ij}$ be the ij-th entry of $\langle\mathbf{V}^{-1}\rangle_{q(\mathbf{V})}$. The covariance matrix of $(\mathbf{B},\alpha)$ is the covariance matrix of the vector $vec(\mathbf{B},\alpha)$. Such vector is constructed using all (non-zero) coefficients and intercepts. We denote this covariance matrix by $\Sigma_{\mathbf{B},\alpha}$. For simplicity of notation, we will treat α_i as the coefficient $b_{i(m+1)}$, m being the number of variables. We will also adopt the notation $Y_{m+1}^{(d)}\equiv 1$ in the following derivations. As an abuse of notation, let $\mathbf{Y}$ also refer to latent variables. In this case, if Y_i and Y_j refer to latent variables X_{h_i} and X_{h_j}, then define $Y_i \equiv \langle X_{h_i} \rangle_{q(\mathbf{X})}$, and $Y_iY_j \equiv \left\langle X_{h_i}X_{h_j} \right\rangle_{q(\mathbf{X})}$.

Let b_{ij} and b_{tv} be the r-th and s-th entries of $vec(\mathbf{B}, \alpha)$, respectively. The rs-th entry of the inverse matrix $\Sigma_{\mathbf{B}\alpha}^{-1}$ is given by

$$(\Sigma_{\mathbf{B}\alpha}^{-1})_{rs} \quad = \quad \mathcal{V}_{it} \sum_{d=1}^{n} Y_j^{(d)} Y_v^{(d)} + 1(i = t)1(j = v)\frac{c_{ij}^b}{s_{ij}^b}$$

where $b_{xp_x} \equiv 0$ if no edge $Y_x \leftarrow Y_{p_x}$ exists in the graph, $1(\cdot)$ is the indicator function, and c_{ij}^b, s_{ij}^b are the given prior parameters defined in Section 4. Similarly to the factorization criterion explained in Section 6, the matrix $q(\mathbf{V})$ will in general be block-diagonal, and this summation can be highly simplified.

Define now a vector $\mathbf{c}^b$ analogous to the Gibbs sampling case, where

$$c_r^b \quad = \quad \sum_{t=1}^{m} \mathcal{V}_{it} \sum_{d=1}^{n} Y_j^{(d)} Y_t^{(d)} + \frac{c_{ij}^b}{s_{ij}^b}.$$

The variational distribution $q(\mathbf{B}, \alpha)$ is then a $N(\Sigma_{\mathbf{B},\alpha}\mathbf{c}, \Sigma_{\mathbf{B},\alpha})$. The variational distribution for the latent variables will exactly the same as the Gibbs distribution, except that references to $\mathbf{B}, \alpha, \mathbf{V}^{-1}$ are substituted by $\langle \mathbf{B} \rangle_{q(\mathbf{B},\alpha)}$, $\langle \alpha \rangle_{q(\mathbf{B},\alpha)}$ and $\left\langle \mathbf{V}^{-1} \right\rangle_{q(\mathbf{V})}$.

Appendix C. Proofs

Proof of Lemma 2: Arrange the columns of the Jacobian such that their order corresponds to the sequence $\sigma_{11}, \sigma_{21}, \sigma_{22}, \sigma_{31}, \sigma_{32}, \sigma_{33}, \ldots, \sigma_{mm}$, excluding the entries σ_{ij} that are identically zero by construction. Arrange the rows of the Jacobian such that their order corresponds to the sequence $\gamma_1, \beta_{21}, \gamma_2, \beta_{31}, \beta_{32}, \ldots, \gamma_m$, excluding the entries β_{ij} that are not in $\Phi_{\mathcal{G}}$ (i.e., exclude any β_{ij} corresponding to a pair $\{Y_i, Y_j\}$ that is not adjacent in the graph).

By the definition of Bartlett's decomposition, $\Sigma_{\{i\},\{i\}}$ and β_{st} are functionally independent for $s > i$. The same holds for $\Sigma_{\{i\},\{i\}}$ and γ_s. As such, $\partial\sigma_{ij}/\partial\beta_{st} = 0$ and $\partial\sigma_{ij}/\partial\gamma_s = 0$ for $s > i$. This implies that $J(\Phi_{\mathcal{G}})$ is a (lower) block triangular matrix of $2m - 1$ blocks: for k odd, the k-th block is the singleton $\partial\sigma_{ii}/\partial\gamma_i = 1$, where $i = (k + 1)/2$. For k even, the k-th block is the Jacobian $\partial\Sigma_{i,sp_{\prec(i)}}/\partial\mathcal{B}_{i,sp_{\prec(i)}}$, where $i = 1 + k/2$ and $\Sigma_{i,sp_{\prec(i)}}$ is the vector of covariances of Y_i and its preceding spouses.

From the interpretation given by Equation (8), it follows that $\mathcal{B}_{i,sp_{\prec}(i)}$ can also be defined by the regression of Y_i on $\mathbf{Z}_i$. That is

$$\mathcal{B}_{i,sp_{\prec}(i)} = \Sigma_{Y_i,\mathbf{Z}_i}\Sigma_{\mathbf{Z}_i,\mathbf{Z}_i}^{-1} \equiv \Sigma_{Y_i,\mathbf{Z}_i}R_i^{-1}. \tag{25}$$

However, $\Sigma_{Y_i,\mathbf{Z}_i} = \Sigma_{i,sp_{\prec}(i)}$, since Y_i is independent of its non-spouses. From (25) we get $\Sigma_{i,sp_{\prec}(i)} = \mathcal{B}_{i,sp_{\prec}(i)}R_i$, and as such the submatrix $\partial\Sigma_{i,sp_{\prec(i)}}/\partial\mathcal{B}_{i,sp_{\prec(i)}}$ turns out to be R_i.

Since the determinant of the block triangular Jacobian $J(\Phi_{\mathcal{G}})$ is given by the determinant of the blocks, this implies

$$|J(\Phi_{\mathcal{G}})| = \prod_{i=2}^{m} |R_i|.$$

By the matrix identity

$$\left|\begin{array}{cc}\mathbf{A} & \mathbf{B}\\ \mathbf{C} & \mathbf{D}\end{array}\right| = |\mathbf{A}||\mathbf{D}-\mathbf{C}\mathbf{A}^{-1}\mathbf{B}|, \tag{26}$$

$|\Sigma_{\{i-1\},\{i-1\}}| = |\Sigma_{nsp_{\prec(i)},nsp_{\prec(i)}}||\Sigma_{sp_{\prec(i)},sp_{\prec(i)}} - \Sigma_{sp_{\prec(i)},nsp_{\prec(i)}}\Sigma^{-1}_{nsp_{\prec(i)},nsp_{\prec(i)}}\Sigma_{nsp_{\prec(i)},sp_{\prec(i)}}| \equiv |\Sigma_{nsp_{\prec(i)},nsp_{\prec(i)}}||R_i|$. Since $|\Sigma_{\{i-1\},\{i-1\}}| = \prod_{t=1}^{i-1}\gamma_t$, the second equality holds. □

Proof of Theorem 4: We first describe a mapping from each path in $\mathcal{G}$ to a path in $\mathcal{G}^\star$, and vice-versa (such mappings are not inverse functions of each other, since the number of paths in $\mathcal{G}^\star$ is larger than in $\mathcal{G}$). By construction, all bi-directed edges in $\mathcal{G}^*$ have two UVs as endpoints, with an one-to-one mapping between each $Y_s^\star \leftrightarrow Y_t^\star$ in $\mathcal{G}^\star$ and each $Y_s \leftrightarrow Y_t$ in $\mathcal{G}$. All directed edges in $\mathcal{G}^\star$ are of two types: $Y_s \rightarrow Y_t^\star$, with $s \neq t$, or $Y_s^\star \rightarrow Y_s$. Therefore, one can define an unique path P in $\mathcal{G}$ as a function of a path $P^\star$ in $\mathcal{G}^\star$, obtained by relabeling each $Y^\star$ as Y, and by collapsing any $Y \rightarrow Y$ edges that might result from this relabeling into a single vertex Y. A mapping in the opposite direction is analogous as given by the construction rule of Type-II models.

A collider in a path is any vertex within a head-to-head collision in the path, that is, any vertex Y_t where the preceding and the next vertex in the path are connected to Y_t with an edge (directed or bi-directed) into Y_t. Y_i and Y_j are m-separated by $\mathbf{Z}$ in an acyclic DMG if and only if there is no active path connecting Y_i and Y_j. Like in d-separation, a path is active if all of its colliders have some descendant in $\mathbf{Z}$, and none of its non-colliders is in $\mathbf{Z}$ (Richardson, 2003). The mappings between paths P and $P^\star$ are such that Y_t is a collider in P if and only if Y_t is in $P^\star$ and is a collider, or $Y_t^\star$ is in $P^\star$ and is a collider. Since by construction any $Y_t^\star$ will have the same $\mathbf{Y}$-descendants in $\mathcal{G}^\star$ as Y_t has in $\mathcal{G}$, and $\mathbf{Z} \subset \mathbf{Y}$, the result follows. □

Proof of Theorem 7: The first of the two claims of the theorem trivially holds, since connectivity is a transitive property and as such this partition will always exist (where $K(i) = 1$ is a possibility). We will prove the validity of the second claim by induction. Let $\{\mathbf{R}_1, \dots, \mathbf{R}_k\}$ be the perfect sequence that generated our perfect ordering. The second claim automatically holds for all vertices in $\mathbf{R}_k$, since $\mathbf{R}_k$ is a clique.

Assume the second claim holds for the subsequence $\{\mathbf{R}_{l+1}, \mathbf{R}_{l+2}, \dots, \mathbf{R}_k\}$. Let Y_i be an element of $\mathbf{R}_l$. Assume there is some non-spouse Y_q of Y_i in $\mathbf{R}_{l'}$, and some spouse Y_p of Y_i in $\mathbf{R}_{l''}$, such that $l < l' \leq l''$. We will assume that both Y_q and Y_p belong to the same component $\mathcal{V}_t^i$ and show this leads to a contradiction.

Without loss of generality, we can assume that Y_q and Y_p are adjacent: otherwise, the fact that Y_q and Y_p are in the connected set $\mathcal{V}_t^i$ will imply there is a path connecting Y_q and Y_p in the subgraph induced by $\{\mathbf{R}_{l+1}, \dots, \mathbf{R}_k\}$. We can redefine $\{Y_q, Y_p\}$ to be the endpoints of the first edge in the path containing a non-spouse and a spouse of Y_i. It will still be the case that $q > p$, by the induction hypothesis.

Since $Y_p \in \mathbf{R}_{l''}$, there is a separator $\mathbf{S}_{l''}$ between $\mathbf{H}_{l''} \backslash \mathbf{S}_{l''}$ and $\mathbf{R}_{l''}$. But $Y_i \in \mathbf{H}_{l''}$, and Y_i is adjacent to Y_p, which implies $Y_i \in \mathbf{S}_{l''}$. If $l' < l''$, this will also imply that $Y_q \in \mathbf{S}_{l''}$, which is a contradiction, since $\mathbf{S}_{l''}$ is a complete set. If $l' = l''$, this implies that Y_i and Y_q are both in $\mathbf{Y}_{P(l'')}$, which is also a contradiction since $\mathbf{Y}_{P(l'')}$ is a clique. □

References

A. Al-Awadhi and P. Garthwaite. An elicitation method for multivariate normal distributions. *Communications in Statistics - Theory and Methods*, 27:1123–1142, 1998.

J. Albert and S. Chib. Bayesian analysis of binary and polychotomous response data. *Journal of the American Statistical Association*, 88:669–679, 1993.

A. Atay-Kayis and H. Massam. A Monte Carlo method for computing the marginal likelihood in nondecomposable Gaussian graphical models. *Biometrika*, 92:317–335, 2005.

D. Bartholomew and M. Knott. *Latent Variable Models and Factor Analysis*. Arnold Publishers, 1999.

M. Beal. Variational algorithms for approximate Bayesian inference. *PhD. Thesis, Gatsby Computational Neuroscience Unit, University College London*, 2003.

M. Beal and Z. Ghahramani. Variational Baeysian learning of directed graphical models with hidden variables. *Bayesian Analysis*, 1:793–832, 2006.

P. Bickel and E. Levina. Covariance regularization by thresholding. *Annals of Statistics*, 36:2577–2604, 2008.

K. Bollen. *Structural Equation Models with Latent Variables*. John Wiley & Sons, 1989.

P. Brown, N. Le, and J. Zidek. Inference for a covariance matrix. *In P.R. Freeman, A.F.M. Smith (editors), Aspects of Uncertainty, a tribute to D. V. Lindley*, pages 77–92, 1993.

Z. Chen and D. Dunson. Random effects selection in linear mixed models. *Biometrics*, 59:762–769, 2003.

D. Chickering. Optimal structure identification with greedy search. *Journal of Machine Learning Research*, 3:507–554, 2002.

M. Daniels and R. Kass. Nonconjugate Bayesian estimation of covariance matrices and its use in hierarchical models. *Journal of the American Statistical Association*, 94:1254–1263, 1999.

M. Drton and M. Perlman. Multiple testing and error control in Gaussian graphical model selection. *Statistical Science*, pages 430–449, 2007.

M. Drton and T. Richardson. A new algorithm for maximum likelihood estimation in Gaussian models for marginal independence. *Proceedings of the 19th Conference on Uncertainty in Artificial Intelligence*, 2003.

M. Drton and T. Richardson. Iterative conditional fitting for Gaussian ancestral graph models. *Proceedings of the 20th Conference on Uncertainty in Artificial Intelligence*, 2004.

M. Drton and T. Richardson. Binary models for marginal independence. *Department of Statistics, University of Washington, Tech. report 474*, 2005.

M. Drton and T. Richardson. Binary models for marginal independence. *Journal of the Royal Statistical Society, Series B*, 70:287–309, 2008a.

M. Drton and T. Richardson. Graphical methods for efficient likelihood inference in Gaussian covariance models. *Journal of Machine Learning Research*, pages 893–914, 2008b.

D. Dunson, J. Palomo, and K. Bollen. Bayesian structural equation modeling. *Statistical and Applied Mathematical Sciences Institute, Technical Report #2005-5*, 2005.

N. Friedman and D. Koller. Being Bayesian about network structure: a Bayesian approach to structure discovery in Bayesian networks. *Machine Learning Journal*, 50:95–126, 2003.

P. Hoyer, D. Janzing, J. Mooij, J. Peters, and B. Schölkopf. Nonlinear causal discovery with additive noise models. *Neural Information Processing Systems*, 2008.

J. Huang and B. Frey. Cumulative distribution networks and the derivative-sum-product algorithm. *Proceedings of 24th Conference on Uncertainty in Artificial Intelligence*, 2008.

B. Jones, C. Carvalho, A. Dobra, C. Hans, C. Carter, and M. West. Experiments in stochastic computation for high-dimensional graphical models. *Statistical Science*, 20:388–400, 2005.

M. Jordan. *Learning in Graphical Models*. MIT Press, 1998.

M. Jordan, Z. Ghaharamani, T. Jaakkola, and L. Saul. An introduction to variational methods for graphical models. *In M. Jordan (Ed.), Learning in Graphical Models*, pages 105–162, 1998.

C. Kang and J. Tian. Local Markov property for models satisfying the composition axiom. *Proceedings of 21st Conference on Uncertainty in Artificial Intelligence*, 2005.

J. Kotecha and P. Djuric. Gibbs sampling approach for the generation of truncated multivariate Gaussian random variables. *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 1757–1760, 1999.

S. Lauritzen. *Graphical Models*. Oxford University Press, 1996.

S.-Y. Lee. *Structural Equation Modeling: a Bayesian Approach*. Wiley, 2007.

D. MacKay. Introduction to Monte Carlo methods. *Learning in Graphical Models*, pages 175–204, 1998.

I. Murray, Z. Ghahramani, and D. MacKay. MCMC for doubly-intractable distributions. *Proceedings of 22nd Conference on Uncertainty in Artificial Intelligence*, 2006.

R. Neal. *Bayesian Learning for Neural Networks*. Springer-Verlag, 1996.

R. Neal. Annealed importance sampling. *Statistics and Computing*, 11:125–139, 2001.

J. Pearl. *Probabilistic Reasoning in Expert Systems: Networks of Plausible Inference*. Morgan Kaufmann, 1988.

J. Pearl. *Causality: Models, Reasoning and Inference*. Cambridge University Press, 2000.

M. Pitt, D. Chan, and R. Kohn. Efficient Bayesian inference for Gaussian copula regression models. *Biometrika*, 93:537–554, 2006.

T. Richardson. Markov properties for acyclic directed mixed graphs. *Scandinavian Journal of Statistics*, 30:145–157, 2003.

T. Richardson and P. Spirtes. Ancestral graph Markov models. *Annals of Statistics*, 30:962–1030, 2002.

A. Roverato. Hyper inverse Wishart distribution for non-decomposable graphs and its application to Bayesian inference for Gaussian graphical models. *Scandinavian Journal of Statistics*, 29: 391–411, 2002.

R. Scheines, R. Hoijtink, and A. Boomsma. Bayesian estimation and testing of structural equation models. *Psychometrika*, 64:37–52, 1999.

R. Silva and Z. Ghahramani. Bayesian inference for Gaussian mixed graph models. *Proceedings of 22nd Conference on Uncertainty in Artificial Intelligence*, 2006.

R. Silva and Z. Ghahramani. Factorial mixtures of Gaussians and the marginal independence model. *Artificial Intelligence & Statistics (AISTATS '09)*, 2009.

R. Silva and R. Scheines. Bayesian learning of measurement and structural models. *23rd International Conference on Machine Learning*, 2006.

R. Silva, R. Scheines, C. Glymour, and P. Spirtes. Learning the structure of linear latent variable models. *Journal of Machine Learning Research*, 7:191–246, 2006.

R. Silva, W. Chu, and Z. Ghahramani. Hidden common cause relations in relational learning. *Neural Information Processing Systems (NIPS '07)*, 2007.

J. Skilling. Nested sampling for general Bayesian computation. *Bayesian Analysis*, 1:833–860, 2006.

P. Spirtes. Directed cyclic graphical representations of feedback models. *Proceedings of 11th Conference on Uncertainty in Artificial Intelligence*, 1995.

P. Spirtes, C. Glymour, and R. Scheines. *Causation, Prediction and Search.* Cambridge University Press, 2000.

R. Tarjan. Decomposition by clique separators. *Discrete Mathematics*, 55:221–232, 1985.

E. Webb and J. Forster. Bayesian model determination for multivariate ordinal and binary data. *Technical report, Southampton Statistical Sciences Research Institute*, 2006.

F. Wong, C. Carter, and R. Kohn. Efficient estimation of covariance selection models. *Biometrika*, 90:809–830, 2003.

S. Wright. Correlation and causation. *Journal of Agricultural Research*, pages 557–585, 1921.

J. Yedidia, W. Freeman, and Y. Weiss. Constructing free-energy approximations and generalized belief propagation algorithms. *IEEE Transactions on Information Theory*, 51:2282–2312, 2005.

J. Zhang. Causal reasoning with ancestral graphs. *Journal of Machine Learning Research*, pages 1437–1474, 2008.

Journal of Machine Learning Research 10 (2009) 1239-1262 Submitted 10/01; Revised 10/08; Published 6/09

Incorporating Functional Knowledge in Neural Networks

Charles Dugas DUGAS@DMS.UMONTREAL.CA
Department of Mathematics and Statistic
Université de Montréal
2920 Chemin de la tour, suite 5190
Montreal, Qc, Canada H3T 1J4

Yoshua Bengio BENGIOY@IRO.UMONTREAL.CA
Department of Computer Science and Operations Research
Université de Montréal
2920 Chemin de la tour, suite 2194
Montreal, Qc, Canada H3A 1J4

François Bélisle BELISLE.FRANCOIS@GMAIL.COM
Claude Nadeau CLAUDE_NADEAU@HC-SC.GC.CA
Health Canada
Tunney's Pasture, PL 0913A
Ottawa, On, Canada K1A 0K9

René Garcia GARCIAR@CIRANO.QC.CA
CIRANO
2020 rue University, 25e étage
Montréal, Qc, Canada H3A 2A5

Editor: Peter Bartlett

Abstract

Incorporating prior knowledge of a particular task into the architecture of a learning algorithm can greatly improve generalization performance. We study here a case where we know that the function to be learned is non-decreasing in its two arguments and convex in one of them. For this purpose we propose a class of functions similar to multi-layer neural networks but (1) that has those properties, (2) is a universal approximator of Lipschitz[1] functions with these and other properties. We apply this new class of functions to the task of modelling the price of call options. Experiments show improvements on regressing the price of call options using the new types of function classes that incorporate the *a priori* constraints.

Keywords: neural networks, universal approximation, monotonicity, convexity, call options

1. Introduction

Incorporating *a priori* knowledge of a particular task into a learning algorithm helps reduce the necessary complexity of the learner and generally improves performance, if the incorporated knowledge is relevant to the task and brings enough information about the unknown generating process of the data. In this paper we consider prior knowledge on the positivity of some first and second derivatives of the function to be learned. In particular such constraints have applications to modelling

1. A function f is Lipschitz in Ω if $\exists c > 0,\ \forall x, y \in \Omega,\ |f(y) - f(x)| \leq c|y - x|$ (Delfour and Zolésio, 2001).

the price of stock options. Based on the Black-Scholes formula, the price of a call stock option is monotonically increasing in both the "moneyness" and time to maturity of the option, and it is convex in the "moneyness". Section 4 better explains these terms and stock options. For a function $f(x_1,x_2)$ of two real-valued arguments, this corresponds to the following properties:

$$f \geq 0, \qquad \frac{\partial f}{\partial x_1} \geq 0, \qquad \frac{\partial f}{\partial x_2} \geq 0, \qquad \frac{\partial^2 f}{\partial x_1^2} \geq 0. \tag{1}$$

The mathematical results of this paper (Section 2) are the following: first we introduce a class of one-argument functions that is positive, non-decreasing and convex in its argument. Second, we use this new class of functions as a building block to design another class of functions that is a universal approximator for functions with positive outputs. Third, once again using the first class of functions, we design a third class that is a universal approximator to functions of two or more arguments, with the set of arguments partitioned in two groups: those arguments for which the second derivative is known positive and those arguments for which we have no prior knowledge on the second derivative. The first derivative is positive for any argument. The universality property of the third class rests on additional constraints on cross-derivatives, which we illustrate below for the case of two arguments:

$$\frac{\partial^2 f}{\partial x_1 \partial x_2} \geq 0, \quad \frac{\partial^3 f}{\partial x_1^2 \partial x_2} \geq 0. \tag{2}$$

Thus, we assume that $f \in \mathcal{C}^3$, the set of functions three times continuously differentiable. Comparative experiments on these new classes of functions were performed on stock option prices, showing improvements when using these new classes rather than ordinary feedforward neural networks. The improvements appear to be non-stationary but the new class of functions shows the most stable behavior in predicting future prices. Detailed experimental results are presented in section 6.

2. Theory

Definition 1 *A class of functions $\hat{\mathcal{F}}$ from $\mathbb{R}^n$ to $\mathbb{R}$ is a* **universal approximator** *for a class of functions $\mathcal{F}$ from $\mathbb{R}^n$ to $\mathbb{R}$ if for any $f \in \mathcal{F}$, any compact domain $D \subset \mathbb{R}^n$, and any positive ε, one can find a $\hat{f} \in \hat{\mathcal{F}}$ with* $\sup_{x \in D} |f(x) - \hat{f}(x)| \leq \varepsilon$.

It has already been shown that the class of artificial neural networks with one hidden layer:

$$\hat{\mathcal{N}} = \left\{ f(x) = w_0 + \sum_{i=1}^{H} w_i \cdot h\left(b_i + \sum_j v_{ij} x_j\right) \right\}, \tag{3}$$

for example, with a sigmoid activation function $h(s) = 1/(1+e^{-s})$, is a universal approximator of continuous functions (Cybenko, 1988, 1989; Hornik et al., 1989; Barron, 1993). Furthermore, Leshno et al. (1993) have shown that any non-polynomial activation function will suffice for universal approximation. The number of hidden units H of the neural network is a hyper-parameter that controls the accuracy of the approximation and it should be chosen to balance the trade-off (see also Moody, 1994) between accuracy (bias of the class of functions) and variance (due to the finite sample used to estimate the parameters of the model). Because of this trade-off, in the finite sample case, it may be advantageous to consider a “simpler” class of functions that is appropriate to the task.

Since the sigmoid h is monotonically increasing ($h'(s) = h(s)(1-h(s)) > 0$), it is easy to force the first derivatives with respect to x to be positive by forcing the weights to be positive, for example with the exponential function:

$$\hat{\mathcal{N}}_{+} = \left\{ f(x) = e^{w_0} + \sum_{i=1}^{H} e^{w_i} \cdot h\left(b_i + \sum_j e^{v_{ij}} x_j \right) \right\}. \tag{4}$$

Note that the positivity of $f(x)$ and $f'(x)$ is not affected by the values of the $\{b_i\}$ parameters. Since the sigmoid h has a positive first derivative, its primitive, which we call *softplus*, is convex:

$$\boxed{\zeta(s) = \ln(1+e^s)}$$

where $\ln(\cdot)$ is the natural logarithm operator. Note that $d\zeta(s)/ds = h(s) = 1/(1+e^{-s})$.

2.1 Universality for Functions with Positive Outputs

Using the softplus function introduced above, we define a new class of functions, all of which have positive outputs:

$$\hat{\mathcal{N}}_{>0} = \{f(x) = \zeta(g(x)), g(x) \in \hat{\mathcal{N}}\}.$$

Theorem 2 *Within the set of continuous functions from $\mathbb{R}^n$ to $\mathbb{R}_+ = \{x : x \in \mathbb{R}, x > 0\}$, the class $\hat{\mathcal{N}}_{>0}$ is a universal approximator.*

Proof Consider a positive function $f(x)$, which we want to approximate arbitrarily well. Consider $g(x) = \zeta^{-1}(f(x)) = \ln(e^{f(x)} - 1)$, the inverse softplus transform of $f(x)$. Choose $\hat{g}(x)$ from $\hat{\mathcal{N}}$ such that $\sup_{x \in D} |g(x) - \hat{g}(x)| \le \varepsilon$, where D is any compact domain over $\mathbb{R}^n$ and ε is any positive real number. The existence of $\hat{g}(x)$ is ensured by the universality property of $\hat{\mathcal{N}}$. Set $\hat{f}(x) = \zeta(\hat{g}(x)) = \ln(1+e^{\hat{g}(x)})$. Consider any particular x and define $a = \min(\hat{g}(x), g(x))$ and $b = \max(\hat{g}(x), g(x))$. Since $b - a \le \varepsilon$, we have,

$$\begin{aligned} |\hat{f}(x) - f(x)| &= \ln(1+e^b) - \ln(1+e^a) \\ &= \ln\left(1 + (e^b - e^a)/(1+e^a)\right) \\ &\le \ln\left(1 + (e^\varepsilon - 1)e^a/(1+e^a)\right) \\ &< \varepsilon. \end{aligned}$$

□

Thus, the use of the softplus function to transform the output of a regular one hidden layer artificial neural network ensures the positivity of the final output without hindering the universality property.

2.2 The Class ${}_{c,n}\hat{\mathcal{N}}_{++}$

In this section, we use the softplus function, in order to define a new class of functions with positive outputs, positive first derivatives w.r.t. all input variables and positive second derivatives w.r.t.

some of the input variables. The basic idea is to replace the sigmoid of a sum by a product of either softplus or sigmoid functions over each of the dimensions (using the softplus over the convex dimensions and the sigmoid over the others):

$$ {}_{c,n}\hat{\mathcal{N}}_{++} = \left\{ f(x) = e^{w_0} + \sum_{i=1}^{H} e^{w_i} \left(\prod_{j=1}^{c} \zeta(b_{ij} + e^{v_{ij}} x_j) \right) \left(\prod_{j=c+1}^{n} h(b_{ij} + e^{v_{ij}} x_j) \right) \right\}. \quad (5) $$

One can readily check that the output is necessarily positive, the first derivatives w.r.t. x_j are positive, and the second derivatives w.r.t. x_j for $j \le c$ are positive. However, this class of functions has other properties that are summarized by the following:

$$ \begin{aligned} \frac{\partial^m f}{\partial^{m_1} x_1 \partial^{m_2} x_2 \cdots \partial^{m_n} x_n} &\geq 0, \quad (6) \\ m_j &\in \begin{cases} \{0,1,2\} & 1 \le j \le c \\ \{0,1\} & c+1 \le j \le n, \end{cases} \\ \sum_{j=1}^{n} m_j &= m. \end{aligned} $$

Here, we have assumed that $f \in \mathcal{C}^{c+n}$, the set of functions that are $c+n$ times continuously differentiable. We will also restrict ourselves to Lispschitz functions since the proof of the theorem relies on the fact that the derivative of the function is bounded. The set of functions that respect these derivative conditions will be referred to as ${}_{c,n}\hat{\mathcal{F}}_{++}$. Note that, as special cases we find that f is positive ($m = 0$), and that it is monotonically increasing w.r.t. any of its inputs ($m = 1$), and convex w.r.t. the first c inputs ($m = 2, \exists j : m_j = 2$). Also note that, when applied to our particular case where $n = 2$ and $c = 1$, this set of equations corresponds to Equations (1) and (2). We now state the main universality theorem:

Theorem 3 *Within the set ${}_{c,n}\hat{\mathcal{F}}_{++}$ of Lipschitz functions from $\mathbb{R}^n$ to $\mathbb{R}$ whose set of derivatives as specified by Equation* (6) *are non-negative, the class ${}_{c,n}\hat{\mathcal{N}}_{++}$ is a universal approximator.*

The proof of the theorem is given in Section A.

2.3 Parameter Optimization

In our experiments, conjugate gradient descent was used to optimize the parameters of the model. The backpropagation equations are obtained as the derivatives of $f \in {}_{c,n}\hat{\mathcal{N}}_{++}$ (Equation 5) w.r.t. to its parameters. Let $z_{i,j} = b_{ij} + e^{v_{ij}} x_j$, $u_i = e^{w_i}(\prod_{j=1}^{c} \zeta(z_{i,j}))(\prod_{j=c+1}^{n} h(z_{ij}))$ and $f = e^{w_0} + \sum_{i=1}^{H} u_i$. Then, we have

$$ \begin{aligned} \partial f / \partial w_0 &= e^{w_0}, \\ \partial f / \partial w_i &= u_i, \\ \partial f / \partial b_{i,k} &= \begin{cases} u_i \cdot h(z_{i,k}) / \zeta(z_{i,k}) & 1 \le k \le c \\ u_i \cdot (1 - h(z_{i,k})) & c+1 \le k \le n, \end{cases} \quad (7) \\ \partial f / \partial v_{i,k} &= e^{v_{ik}} x_k \cdot \partial f / \partial b_{i,k}. \end{aligned} $$

Except for terms $h(z_{i,k})$, $k \le c$ of Equation (7), all values are computed through the forward phase, that is, while computing the value of f. Error backpropagation can thus be performed efficiently if careful attention is paid, during the forward phase, to store the values to be reused in the backpropagation phase.

Software implementing parameter optimization of the proposed architecture and the numerical experiments of the following section is available on-line.[2] Code was written using the "R" statistical software package.[3]

3. Experiments with Artificial Data

In this section, we present a series of controlled experiments in order to assess the potential improvements that can be gained from using the proposed architecture in cases where some derivatives of the target function are known to be positive. The emphasis is put on analyzing the evolution of the model bias and model variance values w.r.t. various noise levels and training set sizes.

The function we shall attempt to learn is

$$\begin{aligned} f(\vec{x}) &= \zeta(x_1)\zeta(x_2)\zeta(x_3)h(x_4), \\ y &= f(\vec{x}) + \xi, \end{aligned}$$

where $\zeta(\cdot)$ is the softplus function defined above and $h(\cdot)$ is the sigmoid function. The input values are drawn from a uniform distribution over the [0,1] interval, that is, $x_i \sim \mathcal{U}(0,1)$. The noise term ξ is added to the true function $f(\vec{x})$ to generate the target value y. Finally, $\xi \sim \mathcal{N}(0, \sigma^2)$, that is, we used additive Gaussian noise. Different values for σ have been tested.

For each combination of noise level ($\sigma \in \{$1e-2, 3e-2, 1e-1$\}$) and training set size (25, 50, 100, 200, 400), we chose the best performing combination of number of hidden units and weight decay. In order to perform model selection, 100 models were trained using different random training sets, for each combination. Based on validation set performance, 50 models were retained and their validation set performances were averaged. The best performing combination was chosen based on this average validation performance. Bias and variance were measured using these 50 selected models when applied on another testset of 10000 examples. In each case, the number of training epochs was 10000. The process was repeated for two architectures: the proposed architecture of products of softplus and sigmoid functions over input dimensions with constrained weights (CPSD) and regular unconstrained multi-layered perceptrons with a single hidden layer (UMLP).

In order to compute the bias and variance values, we first computed, for each test example, the average of the $N_D = 50$ model outputs:

$$\bar{g}(\vec{x}_i) = \frac{1}{N_D} \sum_{j=1}^{N_D} g_j(\vec{x}_i),$$

where $g_j(\vec{x}_i)$ is the output of the j^{th} model associated to the i^{th} input vector $\vec{x}_i$.

The variance was unbiasedly approximated as the average over all test examples ($N_i = 10000$), of the sample variance of model outputs $g_j(\vec{x}_i)$ w.r.t. the corresponding mean output $\bar{g}(\vec{x}_i)$:

$$\begin{aligned} \hat{v}(\vec{x}_i) &= \frac{1}{N_D - 1} \sum_{j=1}^{N_D} (g_j(\vec{x}_i) - \bar{g}(\vec{x}_i))^2, \\ \hat{v} &= \frac{1}{N_i} \sum_i \hat{v}(\vec{x}_i). \end{aligned}$$

2. Software can be found at `http://www.dms.umontreal.ca/~dugas/convex/`.
3. Code found at `http://www.r-project.org/`.

The bias was unbiasedly estimated as the average over all test examples, of the squared deviation of the mean output $\bar{g}(\vec{x}_i)$ w.r.t. the known true function value $f(\vec{x}_i)$, less a variance term:

$$\begin{aligned} \hat{b}(\vec{x}_i) &= (\bar{g}(\vec{x}_i) - f(\vec{x}_i))^2 - \hat{v}(\vec{x}_i)/N_D, \\ \hat{b} &= \frac{1}{N_i}\sum_i \hat{b}(\vec{x}_i). \end{aligned}$$

Let $b(\vec{x}_i) = (E_G(g(\vec{x}_i)) - f(\vec{x}_i))^2$ be the true bias, at point $\vec{x}_i$ where $E_G()$ denotes expectation taken over training set distribution, which induces a distribution of the function g produced by the learning algorithm. Let us show that $E_G(\hat{b}(\vec{x}_i)) = b(\vec{x}_i)$:

$$\begin{aligned} E_G(\hat{b}(\vec{x}_i)) &= E_G[(\bar{g}(\vec{x}_i) - f(\vec{x}_i))^2 - \hat{v}(\vec{x}_i)/N_D], \\ &= E_G[(\bar{g}(\vec{x}_i) - g(\vec{x}_i) + g(\vec{x}_i) - f(\vec{x}_i))^2] - v(\vec{x}_i)/N_D, \\ &= E_G[(\bar{g}(\vec{x}_i) - g(\vec{x}_i))^2] + E_G[(g(\vec{x}_i) - f(\vec{x}_i))^2] - v(\vec{x}_i)/N_D, \\ &= v(\vec{x}_i)/N_D + b(\vec{x}_i) - v(\vec{x}_i)/N_D, \\ &= b(\vec{x}_i). \end{aligned}$$

Table 1 reports the results for these simulations. In all cases, the bias and variance are lower for the proposed architecture than for a regular neural network architecture, which is the result we expected. The variance reduction is easy to understand because of the appropriate constraints on the class of functions. The bias reduction, we conjecture to be a side effect of the bias-variance tradeoff being performed by the model selection on the validation set: to achieve a lower validation error, a larger bias is needed with the unconstrained artificial neural network. The improvements are generally more important for smaller sample sizes. A possible explanation is that the proposed architecture helps reduce the variance of the estimator. With small sample sizes, this is very beneficial and becomes less important as the number of points increases.

4. Estimating Call Option Prices

An option is a contract between two parties that entitles the buyer to a claim at a future date T that depends on the future price, S_T of an underlying asset whose price at current time t is S_t. In this paper we consider the very common European call options, in which the buyer (holder) of the option obtains the right to buy the asset at a fixed price K called the strike price. This purchase can only occur at maturity date (time T). Thus, if at maturity, the price of the asset S_T is above the strike price K, the holder of the option can *exercise* his option and buy the asset at price K, then sell it back on the market at price S_T, thus making a profit of $S_T - K$. If, on the other hand, the price of the asset at maturity S_T is below the strike price K, then the holder of the option has no interest in exercising his option (and does not have to) and the option simply expires worthless and unexercised. For this reason, the option is considered to be worth $\max(0, S_T - K)$ at maturity and our goal is to estimate C_t, the value of that worth at current time t.

In the econometric literature, the call function is often expressed in terms of the primary economic variables that influence its value: the actual market price of the security (S_t), the strike price (K), the remaining time to maturity ($\tau = T - t$), the risk free interest rate (r), and the volatility of the return (σ). One important result is that under mild conditions, the call option function is homogeneous of degree one with respect to the strike price and so we can perform dimensionality reduction

Bias and Variance Analysis on Artificial Data

Ntrain	Noise	Architecture	Bias	Variance	Sum
25	1e-02	UMLP	2.31e-04	9.20e-05	3.23e-04
		CPSD	**1.04e-04**	**3.97e-05**	**1.43e-04**
	3e-02	UMLP	1.06e-03	3.46e-04	1.40e-03
		CPSD	**9.37e-04**	**2.30e-04**	**1.17e-03**
	1e-01	UMLP	1.07e-02	2.68e-03	1.33e-02
		CPSD	**1.02e-02**	**2.45e-03**	**1.27e-02**
50	1e-02	UMLP	1.55e-04	9.41e-05	2.49e-04
		CPSD	**1.03e-04**	**1.99e-05**	**1.23e-04**
	3e-02	UMLP	1.05e-03	1.28e-04	1.18e-03
		CPSD	**9.35e-04**	**9.29e-05**	**1.03e-03**
	1e-01	UMLP	1.03e-02	1.22e-03	1.15e-02
		CPSD	**1.02e-02**	**1.11e-03**	**1.13e-02**
100	1e-02	UMLP	1.27e-04	3.98e-05	1.67e-04
		CPSD	**1.02e-04**	**1.01e-05**	**1.12e-04**
	3e-02	UMLP	9.82e-04	2.11e-04	1.19e-03
		CPSD	**9.39e-04**	**4.77e-05**	**9.87e-04**
	1e-01	UMLP	1.04e-02	6.28e-04	1.10e-02
		CPSD	**1.02e-02**	**5.30e-04**	**1.07e-02**
200	1e-02	UMLP	1.07e-04	2.24e-05	1.29e-04
		CPSD	**1.02e-04**	**5.01e-06**	**1.07e-04**
	3e-02	UMLP	9.45e-04	1.10e-04	1.05e-03
		CPSD	**9.15e-04**	**4.31e-05**	**9.58e-04**
	1e-01	UMLP	1.03e-02	3.38e-04	1.07e-02
		CPSD	**1.02e-02**	**3.21e-04**	**1.05e-02**
400	1e-02	UMLP	1.03e-04	1.14e-05	1.15e-04
		CPSD	**1.02e-04**	**2.41e-06**	**1.04e-04**
	3e-02	UMLP	9.32e-04	6.15e-05	9.94e-04
		CPSD	**9.15e-04**	**2.10e-05**	**9.36e-04**
	1e-01	UMLP	1.04e-02	1.75e-04	1.05e-02
		CPSD	**1.02e-02**	**1.43e-04**	**1.03e-02**

Table 1: Comparison of the bias and variance values for two neural network architectures, three levels of noise, and five sizes of training sets (Ntrain), using artificial data. In bold, the best performance between the two models.

by letting our approximating function depend on the "moneyness" ratio ($M = S_t/K$) instead of the current asset price S_t and the strike price K independently. We must then modify the target to be the price of the option divided by the strike price: C_t/K.

Most of the research on call option modelling relies on strong parametric assumptions of the underlying asset price dynamics. Any misspecification of the stochastic process for the asset price will lead to systematic mispricings for any option based on the asset (Hutchinson et al., 1994). The well-known Black-Scholes formula (Black and Scholes, 1973) is a consequence of such specifications and other assumptions:

$$f(M,\tau,r,\sigma) \quad = \quad M\Phi(d_1) - e^{-r\tau}\Phi(d_2),$$

where $\Phi(\cdot)$ is the cumulative Gaussian function evaluated in points

$$d_1, d_2 \quad = \quad \frac{\ln M + (r \pm \sigma^2/2)\tau}{\sigma\sqrt{\tau}},$$

that is, $d_1 = d_2 + \sigma\sqrt{\tau}$. In particular, two assumptions on which this formula relies have been challenged by empirical evidence: the assumed lognormality of returns on the asset and the assumed constance of volatility over time.

On the other hand, nonparametric models such as neural networks do not rely on such strong assumptions and are therefore robust to model specification errors and their consequences on option modelling and this motivates research in the direction of applying nonparametric techniques for option modelling.

Analyzing the primary economic variables that influence the call option price, we note that the risk free interest rate (r) needs to be somehow extracted from the term structure of interest rates and the volatility (σ) needs to be forecasted. This latter task is a field of research in itself. Dugas et al. (2000) have previously tried to feed in neural networks with estimates of the volatility using historical averages but the gains have remained insignificant. We therefore drop these two features and rely on the ones that can be observed (S_t, K, τ) to obtain the following:

$$C_t/K \quad = \quad f(M,\tau).$$

The novelty of our approach is to account for properties of the call option function as stated in Equation (1). These properties derive from simple arbitrage pricing theory.[4] Now even though we know the call option function to respect these properties, we do not know if it does respect the additional cross derivative properties of Equation (2). In order to gain some insight in this direction, we confront the Black-Scholes formula to our set of constraints:

$$\frac{\partial f}{\partial M} \quad = \quad \Phi(d_1), \tag{8}$$

$$\frac{\partial^2 f}{\partial M^2} \quad = \quad \frac{\phi(d_1)}{\sqrt{\tau}M\sigma}, \tag{9}$$

$$\frac{\partial f}{\partial \tau} \quad = \quad e^{-r\tau}\left(\frac{\phi(d_2)\sigma}{2\sqrt{\tau}} + r\Phi(d_2)\right), \tag{10}$$

$$\frac{\partial^2 f}{\partial M \partial \tau} \quad = \quad \frac{\phi(d_1)}{2\sigma\tau^{3/2}}\left((r+\sigma^2/2)\tau - \ln M\right), \tag{11}$$

$$\frac{\partial^3 f}{\partial M^2 \partial \tau} \quad = \quad \frac{\phi(d_1)}{2M\sigma^3\tau^{5/2}}\left(\ln^2 M - \sigma^2\tau - (r+\sigma^2/2)^2\tau^2\right), \tag{12}$$

where $\phi(\cdot)$ is the Gaussian density function. Equations (8), (9) and (10) confirm that the Black-Scholes formula is in accordance with our prior knowledge of the call option function: all three

4. The convexity of the call option w.r.t. the moneyness is a consequence of the butterfly spread strategy (Garcia and Gençay, 1998).

derivatives are positive. Equations (11) and (12) are the cross derivatives which will be positive for any function chosen from ${}_{1,2}\hat{\mathcal{N}}_{++}$. When applied to the Black-Scholes formula, it is less clear whether these values are positive, too. In particular, one can easily see that both cross derivatives can not be simultaneously positive. Thus, the Black-Scholes formula is not within the set ${}_{1,2}\hat{\mathcal{F}}_{++}$. Then again, it is known that the Black-Scholes formula does not adequately represent the market pricing of options, but it is considered as a useful guide for evaluating call option prices. So, we do not know if these constraints on the cross derivatives are present in the true price function.

Nonetheless, even if these additional constraints are not respected by the true function on all of its domain, one can hope that the increase in the bias of the estimator due to the constraints will be offset (because we are searching in a smaller function space) by a decrease in the variance of that estimator and that overall, the mean-squared error will decrease. This strategy has often been used successfully in machine learning (e.g., regularization, feature selection, smoothing).

5. Experimental Setup

As a reference model, we use a simple multi-layered perceptron with one hidden layer (Equation 3). For **UMLP models**, weights are left unconstrained whereas for **CMLP models**, weights are constrained, through exponentiation, to be positive. We also compare our results with a recently proposed model (Garcia and Gençay, 1998) that closely resembles the Black-Scholes formula for option pricing (i.e., another way to incorporate possibly useful prior knowledge):

$$\begin{aligned} y &= \alpha + M \cdot \sum_{i=1}^{n_h} \beta_{1,i} \cdot h(\gamma_{i,0} + \gamma_{i,1} \cdot M + \gamma_{i,2} \cdot \tau) \\ &+ e^{-r\tau} \cdot \sum_{i=1}^{n_h} \beta_{2,i} \cdot h(\gamma_{i,3} + \gamma_{i,4} \cdot M + \gamma_{i,5} \cdot \tau), \end{aligned} \tag{13}$$

with inputs M, τ, parameters r, α, β, γ and hyperparameter n_h (number of hidden units). We shall refer to Equation (13) as the **UBS models**. Constraining the weights of Equation (13) through exponentiation leads to a different architecture we refer to as the **CBS models**.

We evaluate two new architectures incorporating all of the constraints defined in Equation (6). The proposed architecture involves the product of softplus and sigmoid functions over input dimensions, hence the **UPSD models** and **CPSD models** labels for an unconstrained version of the proposed architecture and the proposed constrained architecture, respectively. Finally, we also tested another architecture derived from the proposed one by simply summing, instead of multiplying, softplus and sigmoid functions. For that last architecture (with constrained weights), positivity, monotonicity and convexity properties are respected but in that case, cross-derivatives are all equal to zero. We do not have a universality proof for that specific class of functions. The unconstrained and constrained architectures are labelled as **USSD models** and **CSSD models**, respectively.

We used European call option data from 1988 to 1993. A total of 43518 transaction prices on European call options on the S&P500 index were used. In Section 6, we report results on 1988 data. In each case, we used the first two quarters of 1988 as a training set (3434 examples), the third quarter as a validation set (1642 examples) for model selection and the fourth quarter as a test set (each with around 1500 examples) for final generalization error estimation. In tables 2 and 3, we present results for networks with unconstrained weights on the left-hand side, and weights constrained to positive and monotone functions through exponentiation of parameters on the right-hand side. For each model, the number of hidden units varies from one to nine. The mean squared error results reported were obtained as follows: first, we randomly sampled the parameter space 1000 times. We picked the best (lowest training error) model and trained it up to 1000 more epochs. Repeating this procedure 10 times, we selected and averaged the performance of the best of these 10 models (those with training error no more than 10% worse than the best out of 10). In figure 1, we present tests of the same models on each quarter up to and including 1993 (20 additional test sets) in order to assess the persistence (conversely, the degradation through time) of the trained models.

6. Forecasting Results

As can be seen in tables 2 and 3, unconstrained architectures obtain better training, validation and testing (test 1) results but fail in the extra testing set (test 2). A possible explanation is that constrained architectures capture more fundamental relationships between variables and are more robust to nonstationarities of the underlying process. Constrained architectures therefore seem to give better generalization when considering longer time spans.

The importance in the difference in performance between constrained and unconstrained architectures on the second test set lead us to look even farther into the future and test the selected models on data from later years. In Figure 1, we see that the Black-Scholes similar constrained model performs slightly better than other models on the second test set but then fails on later quarters. All in all, at the expense of slightly higher initial errors our proposed architecture allows one to forecast with increased stability much farther in the future. This is a very welcome property as new derivative products have a tendency to lock in values for much longer durations (up to 10 years) than traditional ones.

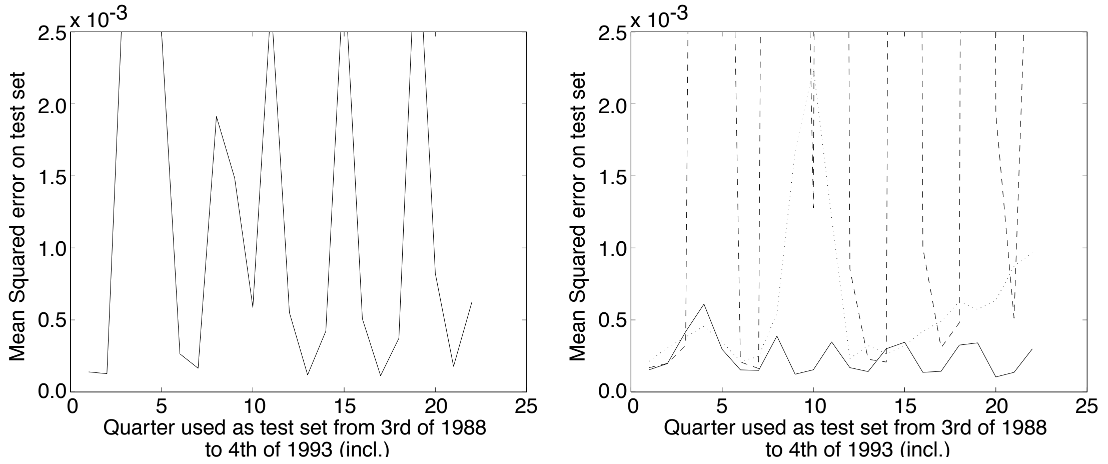


Figure 1: Out-of-sample results from the third quarter of 1988 to the fourth of 1993 (incl.) for models with best validation results. Left: unconstrained models; results for the UBS models. Other unconstrained models exhibit similar swinging result patterns and levels of errors. Right: constrained models. The proposed CPSD architecture (solid) does best. The model with sums over dimensions (CSSD) obtains similar results. Both CMLP (dotted) and CBS (dashed) models obtain poorer results. (dashed).

Mean Squared Error Results on Call Option Pricing ($\times 10^{-4}$)

Hidden Units	**UMLP**				**CMLP**			
	Train	Valid	Test1	Test2	Train	Valid	Test1	Test2
1	2.38	1.92	2.73	6.06	2.67	2.32	3.02	3.60
2	1.68	1.76	1.51	5.70	2.63	2.14	3.08	3.81
3	1.40	1.39	1.27	27.31	2.63	2.15	3.07	3.79
4	1.42	1.44	1.25	27.32	2.65	2.24	3.05	3.70
5	1.40	1.38	**1.27**	**30.56**	2.67	2.29	3.03	3.64
6	1.41	1.43	1.24	33.12	2.63	2.14	**3.08**	**3.81**
7	1.41	1.41	1.26	33.49	2.65	2.23	3.05	3.71
8	1.41	1.43	1.24	39.72	2.63	2.14	3.07	3.80
9	1.40	1.41	1.24	38.07	2.66	2.27	3.04	3.67

Hidden Units	**UBS**				**CBS**			
	Train	Valid	Test1	Test2	Train	Valid	Test1	Test2
1	1.54	1.58	1.40	4.70	2.49	2.17	2.78	3.61
2	1.42	1.42	1.27	24.53	1.90	1.71	2.05	3.19
3	1.40	1.41	1.24	30.83	1.88	1.73	2.00	3.72
4	1.40	1.39	**1.27**	**31.43**	1.85	1.70	1.96	3.15
5	1.40	1.40	1.25	30.82	1.87	1.70	2.01	3.51
6	1.41	1.42	1.25	35.77	1.89	1.70	2.04	3.19
7	1.40	1.40	1.25	35.97	1.87	1.72	1.98	3.12
8	1.40	1.40	1.25	34.68	1.86	1.69	**1.98**	**3.25**
9	1.42	1.43	1.26	32.65	1.92	1.73	2.08	3.17

Table 2: Left: the parameters are free to take on negative values. Right: parameters are constrained through exponentiation so that the resulting function is both positive and monotone increasing everywhere w.r.t. both inputs as in Equation (4). Top: regular feedforward artificial neural networks. Bottom: neural networks with an architecture resembling the Black-Scholes formula as defined in Equation (13). The number of hidden units varies from 1 to 9 for each network architecture. The first two quarters of 1988 were used for training, the third of 1988 for validation and the fourth of 1988 for testing. The first quarter of 1989 was used as a second test set to assess the persistence of the models through time (figure 1). In bold: test results for models with best validation results.

In another series of experiments, we tested the unconstrained multi-layered perceptron against the proposed constrained products of softplus convex architecture using data from years 1988 through 1993 incl. For each year, the first two quarters were used for training, the third quarter for model selection (validation) and the fourth quarter for testing. We trained neural networks for 50000 epochs and with a number of hidden units ranging from 1 through 10. In Table 4, we report training, validation and test results for the two chosen architectures. Model selection was performed using the validation set in order to choose the best number of hidden units, learning rate, learning rate decrease and weight decay. In all cases, except for 1988, the proposed architecture outperformed the multi-layered perceptron model. This might explain why the proposed architecture did

Mean Squared Error Results on Call Option Pricing ($\times 10^{-4}$)

Hidden Units	**UPSD**				**CPSD**			
	Train	Valid	Test1	Test2	Train	Valid	Test1	Test2
1	2.27	2.15	2.35	3.27	2.28	2.14	2.37	3.51
2	1.61	1.58	1.58	14.24	2.28	2.13	2.37	3.48
3	1.51	1.53	1.38	18.16	2.28	2.13	2.36	3.48
4	1.46	1.51	1.29	20.14	1.84	1.54	**1.97**	**4.19**
5	1.57	1.57	1.46	10.03	1.83	1.56	1.95	4.18
6	1.51	1.53	1.35	22.47	1.85	1.57	1.97	4.09
7	1.62	1.67	1.46	7.78	1.86	1.55	2.00	4.10
8	1.55	1.54	1.44	11.58	1.84	1.55	1.96	4.25
9	1.46	1.47	**1.31**	**26.13**	1.87	1.60	1.97	4.12

Hidden Units	**USSD**				**CSSD**			
	Train	Valid	Test1	Test2	Train	Valid	Test1	Test2
1	1.83	1.59	1.93	4.10	2.30	2.19	2.36	3.43
2	1.42	1.45	**1.26**	**25.00**	2.29	2.19	2.34	3.39
3	1.45	1.46	1.32	35.00	1.84	1.58	1.95	4.11
4	1.56	1.69	1.33	21.80	1.85	1.56	1.99	4.09
5	1.60	1.69	1.42	10.11	1.85	1.52	**2.00**	**4.21**
6	1.57	1.66	1.39	14.99	1.86	1.54	2.00	4.12
7	1.61	1.67	1.48	8.00	1.86	1.60	1.98	3.94
8	1.64	1.72	1.48	7.89	1.85	1.54	1.98	4.25
9	1.65	1.70	1.52	6.16	1.84	1.54	1.97	4.25

Table 3: Similar results as in table 2 but for two new architectures. Top: products of softplus along the convex axis with sigmoid along the monotone axis. Bottom: the softplus and sigmoid functions are summed instead of being multiplied. Top right: the fully constrained proposed architecture (CPSD).

not perform as well as other architectures on previous experiments using only data from 1988. Also note that the MSE obtained in 1989 is much higher. This is a possible explanation for the bad results obtained in tables 2 and 3 on the second test set. A hypothesis is that the process was undergoing nonstationarities that affected the forecasting performances. This shows that performance can vary by an order of magnitude from year to year and that forecasting in the presence of nonstationary processes is a difficult task.

7. Conclusions

Motivated by prior knowledge on the positivity of the derivatives of the function that gives the price of European options, we have introduced new classes of functions similar to multi-layer neural networks that have those properties. We have shown universal approximation properties for these classes. On simulation experiments, using artificial data sets, we have shown that these classes of functions lead to a reduction in the variance and the bias of the associated estimators. When applied

Mean Squared Error Results
on Call Option Pricing ($\times 10^{-5}$)

Year	Architecture	Units	Train	Valid	Test
1988	UMLP	9	2.09	1.45	3.28
	CPSD	9	3.86	2.70	5.23
1989	UMLP	4	9.10	28.89	51.39
	CPSD	2	9.31	23.96	48.22
1990	UMLP	9	2.17	4.81	5.61
	CPSD	9	1.58	4.18	5.39
1991	UMLP	9	2.69	1.76	3.41
	CPSD	8	2.62	1.25	2.74
1992	UMLP	8	3.46	1.16	1.52
	CPSD	8	3.27	1.28	1.27
1993	UMLP	9	1.34	1.47	1.76
	CPSD	10	0.68	0.54	0.65

Table 4: Comparison between a simple unconstrained multi-layered architecture (UMLP) and the proposed architecture (CPSD). Data from the first two quarters of each year was used as training set, data from the third quarter was used for validation and the fourth quarter was used for testing. We also report the number of units chosen by the model selection process.

in empirical tests of option pricing, we showed that the architecture from the proposed constrained classes usually generalizes better than a standard artificial neural network.

Appendix A. Proof of the Universality Theorem for Class ${}_{c,n}\hat{\mathcal{N}}_{++}$

In this section, we prove theorem 2.2. In order to help the reader through the formal mathematics, we first give an outline of the proof, that is, a high-level informal overview of the proof, in Section A.1. Then, in Section A.2, we make use of two functions namely, the threshold $\theta(x) = I_{x\geq 0}$ and positive part $x_+ = \max(0,x)$ functions. These two functions are part of the closure of the set ${}_{c,n}\hat{\mathcal{N}}_{++}$ since

$$\begin{aligned} \theta(x) &= \lim_{t\to\infty} h(tx), \\ x_+ &= \lim_{t\to\infty} \zeta(tx). \end{aligned}$$

This extended class of functions that includes $\theta(x)$ and x_+ shall be referred to as ${}_{c,n}\hat{\mathcal{N}}^{\infty}_{++}$. In Section A.3, we give an illustration of the constructive algorithm used to prove universal approximation. Now the proof, as it is stated in Section A.2, only involves functions $\theta(x)$ and x_+, that is, the limit cases of the class ${}_{c,n}\hat{\mathcal{N}}^{\infty}_{++}$ which are actually not part of class ${}_{c,n}\hat{\mathcal{N}}_{++}$. Functions $\theta(x)$ and x_+ assume the use of parameters of infinite value, making the proof without any practical bearing. For this reason, in Section A.4, we broaden the theorem's application from ${}_{c,n}\hat{\mathcal{N}}^{\infty}_{++}$ to ${}_{c,n}\hat{\mathcal{N}}_{++}$, building upon the proof of Section A.2.

A.1 Outline of the Proof

The proof of the first main part (Section A.2) works by construction: we start by setting the approximating function equal to a constant function. Then, we build a grid over the domain of interest and scan through it. At every point of the grid we add a term to the approximating function. This term is a function itself that has zero value at every point of the grid that has already been visited. Thus, this term only affects the current point being visited and some of the points to be visited. The task is therefore to make sure the term being added is such that the approximating function matches the actual function at the point being visited. The functions to be added are chosen from the set ${}_{c,n}\hat{\mathcal{N}}^{\infty}_{++}$ so that each of them individually respects the constraints on the derivatives. The bulk of the work in the proof is to show that, throughout the process, at each scanned point, we need to add a positive term to match the approximating function to the true function. For illustrative purposes, we consider the particular case of call options of Section A.3.

In the second part (Section A.4), we build upon the proof of the first part. The same constructive algorithm is used with the same increment values. We simply consider sigmoidal and softplus functions that are greater or equal, in every point, than their limit counterparts, used in the first part. Products of these softplus and sigmoidal functions are within ${}_{c,n}\hat{\mathcal{N}}_{++}$. Consequently, the function built here is always greater than or equal to its counterpart of the first main part. The main element of the second part is that the difference between these two functions, *at gridpoints*, is capped. This is done by setting the sigmoid and softplus parameter values appropriately. Universality of approximation follows from (1) the capped difference, at gridpoints, between the functions obtained in the first and second parts, (2) the exact approximation obtained at gridpoints in the first part and (3) the bounded function variation between gridpoints.

A.2 Proof of the Universality Theorem for Class ${}_{c,n}\hat{\mathcal{N}}^{\infty}_{++}$

Let D be the compact domain over which we wish to obtain an approximation error below ε in every point. Suppose the existence of an oracle allowing us to evaluate the function in a certain number of points. Let T be the smallest hyperrectangle encompassing D. Let us partition T in hypercubes with sides of length L so that the variation of the function between two arbitrary points of any hypercube is bounded by $\varepsilon/2$. For example, given s, an upper bound on the derivative of the function in any direction, setting $L \leq \frac{\varepsilon}{2s\sqrt{n}}$ would do the trick. Since we have assumed the function to be approximated is Lipschitz, then its derivative is bounded and s does exist. The number of gridpoints is N_1+1 over the x_1 axis, N_2+1 over the x_2 axis, ..., N_n+1 over the x_n axis. Thus, the number of points on the grid formed within T is $H=(N_1+1)\cdot(N_2+1)\cdot\ldots\cdot(N_n+1)$. We define gridpoints $\vec{a}=(a_1,a_2,\cdots,a_n)$ and $\vec{b}=(b_1,b_2,\cdots,b_n)$ as the innermost (closest to origin) and outermost corners of T, respectively. Figure 2 illustrates these values. The points of the grid are defined as:

$$\begin{aligned}
\vec{p}_1 &= a, \\
\vec{p}_2 &= (a_1, a_2, \ldots, a_n + L), \\
\vec{p}_{N_n+1} &= (a_1, a_2, \ldots, b_n), \\
\vec{p}_{N_n+2} &= (a_1, a_2, \ldots a_{n-1} + L, a_n), \\
\vec{p}_{N_n+3} &= (a_1, a_2, \ldots a_{n-1} + L, a_n + L), \ldots, \\
\vec{p}_{(N_n+1)(N_{n-1}+1)} &= (a_1, a_2, \ldots, a_{n-2}, b_{n-1}, b_n), \\
\vec{p}_{(N_n+1)(N_{n-1}+1)+1} &= (a_1, a_2, \ldots, a_{n-2} + L, a_{n-1}, a_n), \ldots, \\
\vec{p}_H &= b. \qquad (14)
\end{aligned}$$

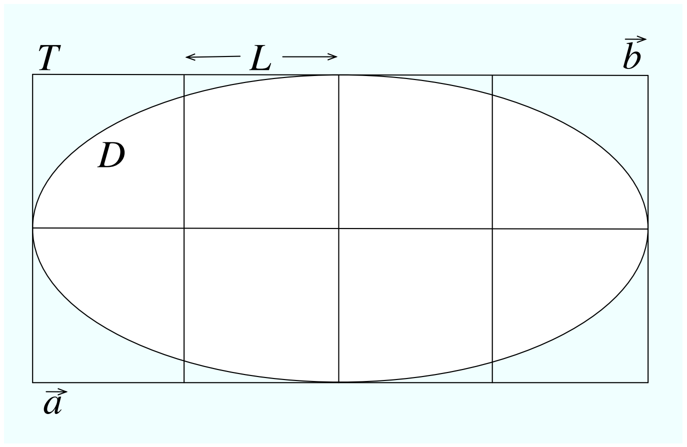


Figure 2: Two dimensional illustration of the proof of universality: ellipse D corresponds to the domain of observation over which we wish to obtain a universal approximator. Rectangle T encompasses D and is partitioned in squares of length L. Points $\vec{a}$ and $\vec{b}$ are the innermost (closest to origin) and outermost corners of T, respectively.

We start with an approximating function $\hat{f}_0 = f(\vec{a})$, that is, the function $\hat{f}_0$ is initially set to a constant value equal to $f(\vec{a})$ over the entire domain. Note that, for the remainder of the proof, notations $\hat{f}_h$, f_h, $\hat{g}_h$, without any argument, refer to the functions themselves. When an argument is present, such as in $f_h(\vec{p})$, we refer to the value of the function f_h evaluated at point $\vec{p}$.

After setting $\hat{f}_0$ to its initial value, we scan the grid according to the order defined in Equation (14). At each point along the grid, we add a term ($\hat{g}_h$, a function) to the current approximating function so that it becomes exact at point $\{\vec{p}_h\}$:

$$\begin{aligned}
\hat{f}_h &= \hat{g}_h + \hat{f}_{h-1}, \\
&= \sum_{k=0}^{h} \hat{g}_k,
\end{aligned}$$

where we have set $\hat{g}_0 = \hat{f}_0$.

The functions $\hat{f}_h$, $\hat{g}_h$ and $\hat{f}_{h-1}$ are defined over the whole domain and the increment function $\hat{g}_h$ must be such that at point $\vec{p}_h$, we have $\hat{f}_h(\vec{p}_h) = f(\vec{p}_h)$. We compute the constant term δ_h as the difference between the value of the function evaluated at point $\vec{p}_h$, $f(\vec{p}_h)$, and the value of the currently accumulated approximating function at the same point $\hat{f}_{h-1}(\vec{p}_h)$:

$$\delta_h = f(\vec{p}_h) - \hat{f}_{h-1}(\vec{p}_h).$$

Now, the function $\hat{g}_h$ must not affect the value of the approximating function at gridpoints that have already been visited. According to our sequencing of the gridpoints, this corresponds to having $\hat{g}_h(\vec{p}_k) = 0$ for $0 < k < h$. Enforcing this constraint ensures that $\forall k \leq h$, $\hat{f}_h(\vec{p}_k) = \hat{f}_k(\vec{p}_k) = f(\vec{p}_k)$. We define

$$\hat{\beta}_h(\vec{p}_k) = \prod_{j=1}^{c} (p_k(j) - p_h(j) + L)_+ / L \cdot \prod_{j=c+1}^{n} \theta(p_k(j) - p_h(j)), \tag{15}$$

where $p_k(j)$ is the j^{th} coordinate of $\vec{p}_k$ and similarly for $\vec{p}_h$. We have assumed, without loss of generality, that the convex dimensions are the first c ones. One can readily verify that $\hat{\beta}_h(\vec{p}_k) = 0$ for $0 < k < h$ and $\hat{\beta}_h(\vec{p}_h) = 1$. We can now define the incremental function as:

$$\hat{g}_h(\vec{p}) = \delta_h \hat{\beta}_h(\vec{p}), \tag{16}$$

so that after all gridpoints have been visited, our final approximation is

$$\hat{f}_H(\vec{p}) = \sum_{h=0}^{H} \hat{g}_h(\vec{p}),$$

with $f(\vec{p}) = \hat{f}_H(\vec{p})$ for all gridpoints.

So far, we have devised a way to approximate the target function as a sum of terms from the set ${}_{c,n}\hat{\mathcal{N}}_{++}^{\infty}$. We know our approximation to be exact in every point of a grid and that the grid is tight enough so that the approximation error is bounded above by $\varepsilon/2$ anywhere within T (thus within D): take any point $\vec{q}$ within a hypercube. Let $\vec{q}_1$ and $\vec{q}_2$ be the innermost (closest to origin) and outermost gridpoints of $\vec{q}$'s hypercube, respectively. Then, we have $f(\vec{q}_1) \leq f(\vec{q}) \leq f(\vec{q}_2)$ and, *assuming* $\delta_h \geq 0\ \forall h$, $f(\vec{q}_1) = \hat{f}_H(\vec{q}_1) \leq \hat{f}_H(\vec{q}) \leq \hat{f}_H(\vec{q}_2) = f(\vec{q}_2)$. Thus, $|\hat{f}_H(\vec{q}) - f(\vec{q})| \leq |f(\vec{q}_2) - f(\vec{q}_1)| \leq Ls\sqrt{n} \leq \varepsilon/2$, since we have set $L \leq \frac{\varepsilon}{2s\sqrt{n}}$. And there remains to be shown that, effectively, $\delta_h \geq 0\ \forall h$. In order to do so, we will express the target function at gridpoint $\vec{p}_h$, $f(\vec{p}_h)$ in terms of the δ_k coefficients ($0 < k \leq h$), then solve for δ_h and show that it is necessarily positive.

First, let $p_k(j) = a(j) + \vec{\iota}_k(j)L$ and define $\vec{\iota}_k = (i_k(1), i_k(2), \ldots, i_k(n))$ so that $\vec{p}_k = \vec{a} + L \cdot \vec{\iota}_k$. Now, looking at Equations (15) and (16), we see that $\hat{g}_k(\vec{p})$ is equal to zero if, for any j, $p_k(j) > p(j)$. Conversely, $\hat{g}_k(\vec{p})$ can only be different from zero if $p_k(j) \leq p(j), \forall j$ or, equivalently, if $i_k(j) \leq i(j), \forall j$.

Next, in order to facilitate the derivations to come, it will be convenient to define some subsets of $\{1, 2, \ldots, H\}$, the indices of the gridpoints of T. Given index h, define $Q_{h,l} \subset \{1, 2, \ldots, H\}$ as

$$Q_{h,l} = \{k : i_k(j) \leq i_h(j) \text{ if } j \leq l \text{ and } i_k(j) = i_h(j) \text{ if } j > l\}.$$

In particular, $Q_{h,n} = \{k : i_k(j) \le i_h(j) \forall j\}$ and $Q_{h,0} = \{h\}$. Thus, we have

$$\begin{aligned}
f(\vec{p}_h) &= \hat{f}_H(\vec{p}_h) \\
&= \sum_{k \in Q_{h,n}} \hat{g}_k(\vec{p}_h) \\
&= \sum_{k \in Q_{h,n}} \delta_k \prod_{j=1}^{c} (i_h(j) - i_k(j) + 1)_+ \prod_{j=c+1}^{n} \theta(i_h(j) - i_k(j)) \\
&= \sum_{k \in Q_{h,n}} \delta_k \prod_{j=1}^{c} (i_h(j) - i_k(j) + 1). && (17)
\end{aligned}$$

Now, let us define the finite difference of the function along the l^{th} axis as

$$\Delta_l f(\vec{p}_h) \quad = \quad f(\vec{p}_h) - f(\vec{p}_{h_l}), \qquad (18)$$

where $\vec{p}_{h_l}$ is the neighbor of $\vec{p}_h$ on T with all coordinates equal except along the l^{th} axis where $i_{h_l}(l) = i_h(l) - 1$. The following relationship shall be useful:

$$\begin{aligned}
Q_{h,l} \backslash Q_{h_l,l} &= \{k : i_k(j) \le i_h(j) \text{ if } j < l, i_k(l) \le i_h(l) \text{ and } i_k(j) = i_h(j) \text{ if } j > l\} \backslash \\
&\quad \{k : i_k(j) \le i_h(j) \text{ if } j < l, i_k(l) < i_h(l) \text{ and } i_k(j) = i_h(j) \text{ if } j > l\} \\
&= \{k : i_k(j) \le i_h(j) \text{ if } j \le l-1 \text{ and } i_k(j) = i_h(j) \text{ if } j > l-1\} \\
&= Q_{h,l-1}. && (19)
\end{aligned}$$

We now have the necessary tools to solve for δ_h by differentiating the target function. Using Equations (17), (18) and (19) we get:

$$\begin{aligned}
\Delta_n f(\vec{p}_h) &= \sum_{k \in Q_{h,n}} \delta_k \prod_{j=1}^{c} (i_h(j) - i_k(j) + 1) \\
&\quad - \sum_{k \in Q_{h_n,n}} \delta_k \prod_{j=1}^{c} (i_{h_n}(j) - i_k(j) + 1).
\end{aligned}$$

Since $i_{h_n}(j) = i_h(j)$ for $j \le c$, then

$$\Delta_n f(\vec{p}_h) \quad = \quad \sum_{k \in Q_{h,n-1}} \delta_k \prod_{j=1}^{c} (i_h(j) - i_k(j) + 1).$$

This process is repeated for non-convex dimensions $n-1, n-2, \ldots, c+1$ until we obtain

$$\Delta_{c+1} \ldots \Delta_n f(\vec{p}_h) \quad = \quad \sum_{k \in Q_{h,c}} \delta_k \prod_{j=1}^{c} (i_h(j) - i_k(j) + 1),$$

at which point we must consider differentiating with respect to convex dimensions:

$$\begin{aligned}\Delta_c \ldots \Delta_n f(\vec{p}_h) &= \sum_{k \in Q_{h,c}} \delta_k \prod_{j=1}^{c} (i_h(j) - i_k(j) + 1) \\ &\quad - \sum_{k \in Q_{h_c,c}} \delta_k \prod_{j=1}^{c} (i_{h_c}(j) - i_k(j) + 1) \\ &= \sum_{k \in Q_{h,c}} \delta_k (i_h(c) - i_k(c) + 1) \prod_{j=1}^{c-1} (i_h(j) - i_k(j) + 1) \\ &\quad - \sum_{k \in Q_{h_c,c}} \delta_k (i_h(c) - i_k(c)) \prod_{j=1}^{c-1} (i_h(j) - i_k(j) + 1).\end{aligned}$$

According to Equation (19), $Q_{h,c} \backslash Q_{h_c,c} = Q_{h,c-1}$ and by definition $i_k(c) - i_h(c) = 0 \; \forall k \in Q_{h,c-1}$. Using this, we subtract a sum of zero terms from the last equation in order to simplify the result:

$$\begin{aligned}\Delta_c \ldots \Delta_n f(\vec{p}_h) &= \sum_{k \in Q_{h,c}} \delta_k (i_h(c) - i_k(c) + 1) \prod_{j=1}^{c-1} (i_h(j) - i_k(j) + 1) \\ &\quad - \sum_{k \in Q_{h_c,c}} \delta_k (i_h(c) - i_k(c)) \prod_{j=1}^{c-1} (i_h(j) - i_k(j) + 1) \\ &\quad - \sum_{k \in Q_{h,c-1}} \delta_k (i_h(c) - i_k(c)) \prod_{j=1}^{c-1} (i_h(j) - i_k(j) + 1) \\ &= \sum_{k \in Q_{h,c}} \delta_k (i_h(c) - i_k(c) + 1) \prod_{j=1}^{c-1} (i_h(j) - i_k(j) + 1) \\ &\quad - \sum_{k \in Q_{h,c}} \delta_k (i_h(c) - i_k(c)) \prod_{j=1}^{c-1} (i_h(j) - i_k(j) + 1) \\ &= \sum_{k \in Q_{h,c}} \delta_k \prod_{j=1}^{c-1} (i_h(j) - i_k(j) + 1).\end{aligned}$$

Differentiating once again with respect to dimension c:

$$\begin{aligned}\Delta_c^2 \ldots \Delta_n f(\vec{p}_h) &= \sum_{k \in Q_{h,c}} \delta_k \prod_{j=1}^{c-1} (i_h(j) - i_k(j) + 1) \\ &\quad - \sum_{k \in Q_{h_c,c}} \delta_k \prod_{j=1}^{c-1} (i_{h_c}(j) - i_k(j) + 1).\end{aligned}$$

and since $i_{h_c}(j) = i_h(c) \forall j \leq c-1$, then

$$\Delta_c^2 \ldots \Delta_n f(\vec{p}_h) = \sum_{k \in Q_{h,c-1}} \delta_k \prod_{j=1}^{c-1} (i_h(j) - i_k(j) + 1).$$

This process of differentiating twice is repeated for all convex dimensions so that

$$\begin{aligned}\Delta_1^2 \dots \Delta_c^2 \Delta_{c+1} \dots \Delta_n f(\vec{p}_h) &= \sum_{k \in Q_{h,0}} \delta_k. \\ &= \delta_h\end{aligned}$$

Now, by definition of the integral operator,

$$\Delta f = \tfrac{f(b)-f(a)}{b-a} = \frac{1}{b-a} \int_a^b f' \, dx,$$

so that if $f' \geq 0$ over the range $[a,b]$, then consequently, $\Delta f \geq 0$. Since, according to Equation (6), we have

$$\frac{\partial^{n+c} f(p_h)}{\partial x_1^2 \partial x_2^2 \dots \partial x_c^2 \partial x_{c+1} \dots \partial x_n} \geq 0,$$

then $\Delta_1^2 \dots \Delta_c^2 \Delta_{c+1} \dots \Delta_n f(\vec{p}_h) \geq 0$ and $\delta_h \geq 0$.

For gridpoints with either $i_h(j) = 1$ for any j or with $i_h(j) = 2$ for any $j \leq c$, solving for δ_h requires fewer than $n+c$ differentiations. Since the positivity of the derivatives of f corresponding to these lower order differentiations are covered by Equation (6), then we also have that $\delta_h \geq 0$ for these gridpoints laying at or near some of the boundaries of T. Thus, ${}_{c,n}\hat{\mathcal{N}}_{++}^{\infty}$ is a universal approximator of ${}_{c,n}\hat{\mathcal{F}}_{++}$.

$\square$

A.3 Illustration of the Constructive Algorithm

In order to give the reader a better intuition regarding the constructive algorithm and as how to solve for δ_h, we apply the developments of the previous subsection to ${}_{1,2}\hat{\mathcal{N}}_{++}$, the set of functions that include call price functions, that is, positive convex w.r.t. the first variable and monotone increasing w.r.t. both variables. Figure 3 illustrates the two dimensional setting of our example with the points of the grid labelled in the order in which they are scanned according the constructive procedure. Here, we will solve δ_6.

For the set ${}_{1,2}\hat{\mathcal{N}}_{++}$, we have,

$$f(\vec{p}_h) = \sum_{k=1}^{H} \delta_k \cdot (p_h(1) - p_k(1) + L)_+ \cdot \theta(p_h(2) - p_k(2)).$$

Applying this to the six gridpoints of Figure 3, we obtain $f(\vec{p}_1) = \delta_1$, $f(\vec{p}_2) = (\delta_1 + \delta_2)$, $f(\vec{p}_3) = (2\delta_1 + \delta_3)$, $f(\vec{p}_4) = (2\delta_1 + 2\delta_2 + \delta_3 + \delta_4)$, $f(\vec{p}_5) = (3\delta_1 + 2\delta_3 + \delta_5)$, $f(\vec{p}_6) = (3\delta_1 + 3\delta_2 + 2\delta_3 + 2\delta_4 + \delta_5 + \delta_6)$.

Differentiating w.r.t. the second variable, then the first, we have:

$$\begin{aligned}\Delta_2 f(\vec{p}_6) &= f(\vec{p}_6) - f(\vec{p}_5) \\ \Delta_1 \Delta_2 f(\vec{p}_6) &= (f(\vec{p}_6) - f(\vec{p}_5)) - (f(\vec{p}_4) - f(\vec{p}_3)) \\ \Delta_1^2 \Delta_2 f(\vec{p}_6) &= (f(\vec{p}_6) - f(\vec{p}_5)) - (f(\vec{p}_4) - f(\vec{p}_3)) \\ &\quad - (f(\vec{p}_4) - f(\vec{p}_3)) + (f(\vec{p}_2) - f(\vec{p}_1)) \\ &= \delta_6.\end{aligned}$$

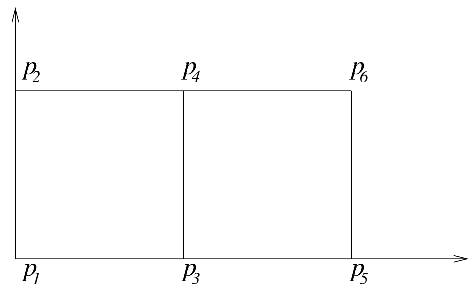


Figure 3: Illustration in two dimensions of the constructive proof. The points are labelled according to the order in which they are visited. The function is known to be convex w.r.t. to the first variable (abscissa) and monotone increasing w.r.t. both variables.

The conclusion associated with this result is that the third finite difference of the function must be positive in order for δ_6 to be positive as well. As stated above, enforcing the corresponding derivative to be positive is a stronger condition which is respected by all element functions of ${}_{c,n}\hat{\mathcal{N}}_{++}$. In the illustration above, other increment terms (δ_1 through δ_5) can be solved for with fewer differentiations. As mention in the previous subsection, derivatives associated to these lower order differentiations are all positive.

A.4 Proof of the Universality Theorem for Class ${}_{c,n}\hat{\mathcal{N}}_{++}$

In Section A.2, we obtained an approximating function $\hat{f}_H \in {}_{c,n}\hat{\mathcal{N}}^{\infty}_{++}$ such that $|\hat{f}_H - f| \leq \varepsilon/2$. Here, we will build a function $\tilde{f}_H \in {}_{c,n}\hat{\mathcal{N}}_{++}$ everywhere greater or equal to $\hat{f}_H$, but we will show how the difference between the two functions can be bounded so that $\tilde{f}_H - \hat{f}_H \leq \varepsilon/2$ at all gridpoints.

We start with an approximating function $\tilde{f}_0 = \hat{f}_0 = f(\vec{a})$, that is, $\tilde{f}_0$ is initially set to a constant value equal to $f(\vec{a})$ over the entire domain. Then, we scan the grid in an orderly manner, according to the definition of the set of points $\{\vec{p}_h\}$. At each point $\vec{p}_h$ along the grid, we add a term $\tilde{g}_h$ (a function) to the current approximating function $\tilde{f}_{h-1}$:

$$\begin{aligned}\tilde{f}_h &= \tilde{g}_h + \tilde{f}_{h-1}\\ &= \sum_{k=1}^{h} \tilde{g}_k\\ &= \sum_{k=1}^{h} \delta_k \tilde{\beta}_k,\end{aligned}$$

where the δ_k are kept equal to the ones found in Section A.2 and we define the set of $\tilde{\beta}_k$ functions as a product of sigmoid and softplus functions, one for each input dimension:

$$\tilde{\beta}_h(\vec{p}) = \prod_{j=1}^{n} \tilde{\beta}_{h,j}(\vec{p}).$$

For each of the convex coordinates, we set:

$$\tilde{\beta}_{h,j}(\vec{p}) = \frac{1}{\alpha}\zeta(\alpha \cdot (p(j) - p_h(j) + L)). \tag{20}$$

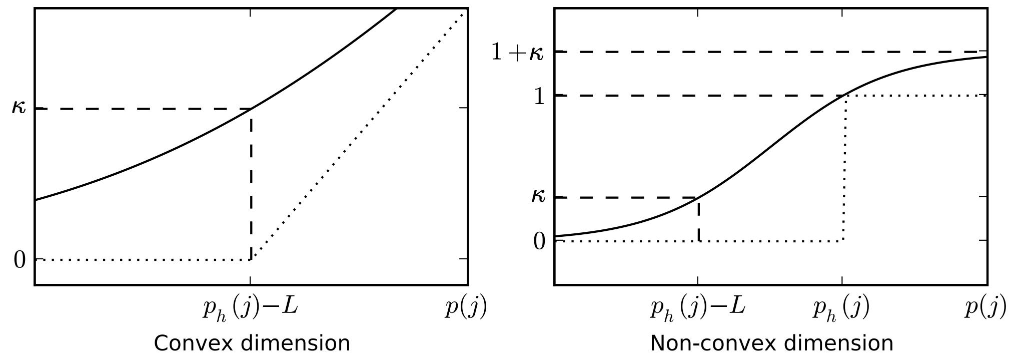


Figure 4: Illustration of the difference between $\tilde{\beta}_{h,j}$ (solid) and $\hat{\beta}_{h,j}$ (dotted) for convex (left) and non-convex (right) dimensions.

where $\alpha > 0$. Now, note that κ, the maximum of the difference between the softplus function of Equation (20) and the positive part function $\hat{\beta}_{h,j}(\vec{p}) = (p(j) - p_h(j) + L)_+$, is attained for $p(j) = p_h(j) - L$ where the difference is $\ln 2/\alpha$. Thus, in order to cap the difference resulting from the approximation along the convex dimensions, we simply need to set κ (α) to a small (large) enough value which we shall soon define. Let us now turn to the non-convex dimensions where we set:

$$\tilde{\beta}_{h,j}(\vec{p}) \quad = \quad (1+\kappa)\, h(\gamma \cdot p(j) + \eta)$$

and add two constraints:

$$\begin{aligned} h(\gamma(p_h(j) - L) + \eta) &= \frac{\kappa}{1+\kappa}, \\ h(\gamma \cdot p_h(j) + \eta) &= \frac{1}{1+\kappa}. \end{aligned}$$

Solving for γ and η, we obtain:

$$\gamma \quad = \quad -\frac{2}{L}\ln\kappa, \tag{21}$$

$$\eta \quad = \quad \left(\frac{2p_h(j)}{L} - 1\right)\ln\kappa. \tag{22}$$

For non-convex dimensions, we have $\hat{\beta}_{h,j}(\vec{p}) = \theta(p(j) - p_h(j))$. Thus, for values of $p(j)$ such that $p_h(j) - L < p(j) < p_h(j)$, we have a maximum difference $\tilde{\beta}_{h,j} - \hat{\beta}_{h,j}$ of 1. For other values of $p(j)$, the difference is capped by κ. In particular, the difference is bounded above by κ for all gridpoints and is zero for gridpoints with $p(j) = p_h(j)$. These values are illustrated in Figure 4.

We now compare incremental terms. Our goal is to cap the difference between $\tilde{g}_h$ and $\hat{g}_h$ by $\varepsilon/2H$. This will lead us to bound the value of κ. At gridpoints, $\hat{\beta}_h$ is equal to

$$\hat{\beta}_h \quad = \quad \prod_{j=1}^{n} m_j,$$

where $m_j \in \{0,1\}$ along non-convex dimensions and m_j is equal to a non-negative integer along the convex dimensions. Also, since $\tilde{\beta}_{h,j} - \hat{\beta}_{h,j} \leq \kappa$ at gridpoints, then

$$\tilde{\beta}_h \leq \prod_{j=1}^{n}(m_j+\kappa).$$

In order find a bound on the value of κ, we need to consider two cases. First, consider the case where $m_j > 0 \,\forall j$:

$$\begin{aligned}
\tilde{g}_h - \hat{g}_h &\leq \delta_h\left(\prod_{j=1}^{n}(m_j+\kappa) - \prod_{j=1}^{n} m_j\right)\\
&\leq \delta_h\left(\prod_{j=1}^{n} m_j(1+\kappa) - \prod_{j=1}^{n} m_j\right)\\
&= \delta_h((1+\kappa)^n - 1)\prod_{j=1}^{n} m_j\\
&\leq \delta_h((1+\kappa)^n - 1)\prod_{j=1}^{n}(N_j+1)\\
&\leq \varepsilon((1+\kappa)^n - 1)H.
\end{aligned}$$

In that case, we have $\tilde{g}_h - \hat{g}_h < \varepsilon/2H$ if

$$\kappa \leq (1/2H^2+1)^{1/n} - 1. \tag{23}$$

Now, consider cases where $\exists j : m_j = 0$. Let $d = \#\{j : m_j = 0\}$. Then,

$$\begin{aligned}
\tilde{g}_h - \hat{g}_h &\leq \delta_h\left(\prod_{j=1}^{n}(m_j+\kappa) - \prod_{j=1}^{n} m_j\right)\\
&\leq \delta_h\kappa^d \prod_{m_j\neq 0}(N_j+1)(1+\kappa)\\
&\leq \delta_h\kappa^d(1+\kappa)^{n-d}H\\
&\leq \varepsilon\kappa^d 2^{n-d}H\\
&\leq \varepsilon\kappa 2^{n-1}H,
\end{aligned}$$

so that here, the bound on κ is:

$$\kappa \leq \frac{1}{2^n H^2}. \tag{24}$$

Depending on the relative values of n and H, one of the two bounds may be effective so that both values of Equations (23) and (24) must be considered in order to set an upper bound on κ:

$$\kappa \leq \min\left((1/2H^2+1)^{1/n} - 1, \frac{1}{2^n H^2}\right).$$

Values for $\alpha = \ln 2/\kappa, \gamma$, and η (Equations 21 and 22) are derived accordingly.

Thus, for any gridpoint, we have:

$$\begin{aligned} \tilde{f}_h - \hat{f}_h &= \sum_{k=1}^{H} \tilde{g}_h - \hat{g}_h \\ &\leq H \cdot \varepsilon/2H \\ &= \varepsilon/2. \end{aligned}$$

In Section A.2, we developed an algorithm such that $\hat{f} = f$ for any gridpoint. In this present subsection, we showed that softplus and sigmoid parameters could be chosen such that $\hat{f} \leq \tilde{f} \leq \hat{f} + \varepsilon/2$ for any gridpoint. Note that $f, \hat{f}$, and $\tilde{f}$ are increasing along each input dimension.

As in Section A.2, consider any point $\vec{q} \in D$. Let $\vec{q}_1$ and $\vec{q}_2$ be the innermost and outermost gridpoints of $\vec{q}$'s encompassing hypercube of side length L. In Section A.2, we showed how a grid could be made tight enough so that $f(\vec{q}_2) - f(\vec{q}_1) \leq \varepsilon/2$.

With these results at hand, we can set upper and lower bounds on $\tilde{f}(\vec{q})$. First, observe that $\tilde{f}_H(\vec{q}) \geq \tilde{f}_H(\vec{q}_1) \geq \hat{f}_H(\vec{q}_1) = f(\vec{q}_1)$, which provides us with a lower bound on $\tilde{f}_H(\vec{q})$. Next, for the upper bound we have: $\tilde{f}_H(\vec{q}) \leq \tilde{f}_H(\vec{q}_2) \leq \hat{f}_H(\vec{q}_2) + \varepsilon/2 = f(\vec{q}_2) + \varepsilon/2 \leq f(\vec{q}_1) + \varepsilon$. Thus, $\tilde{f}_H(\vec{q}) \in [f(\vec{q}_1), f(\vec{q}_1) + \varepsilon]$ and $f(\vec{q}) \in [f(\vec{q}_1), f(\vec{q}_1) + \varepsilon/2] \subset [f(\vec{q}_1), f(\vec{q}_1) + \varepsilon]$. Since both $f(\vec{q})$ and $\tilde{f}(\vec{q})$ are within a range of length ε, then $|\tilde{f}(\vec{q}) - f(\vec{q})| \leq \varepsilon$.

$\square$

References

A. R. Barron. Universal approximation bounds for superpositions of a sigmoidal function. *IEEE Transactions on Information Theory*, 39(3):930–945, 1993.

F. Black and M. Scholes. The pricing of options and corporate liabilities. *Journal of Political Economy*, 81(3):637–654, 1973.

G. Cybenko. Continuous valued neural networks with two hidden layers are sufficient. Technical report, Department of Computer Science, Tufts University, Medford, MA, 1988.

G. Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals, and Systems*, 2:303–314, 1989.

M.C. Delfour and J.-P. Zolésio. *Shapes and Geometries: Analysis, Differential Calculus, and Optimization*. SIAM, 2001.

C. Dugas, O. Bardou, and Y. Bengio. Analyses empiriques sur des transactions d'options. Technical Report 1176, Départment d'informatique et de Recherche Opérationnelle, Université de Montréal, Montréal, Québec, Canada, 2000.

R. Garcia and R. Gençay. Pricing and hedging derivative securities with neural networks and a homogeneity hint. Technical Report 98s-35, CIRANO, Montréal, Québec, Canada, 1998.

K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2:359–366, 1989.

J.M. Hutchinson, A.W. Lo, and T. Poggio. A nonparametric approach to pricing and hedging derivative securities via learning networks. *Journal of Finance*, 49(3):851–889, 1994.

M. Leshno, V. Lin, A. Pinkus, and S. Schocken. Multilayer feedforward networks with a nonpolynomial activation function can approximate any function. *Neural Networks*, 6:861–867, 1993.

J. Moody. *Prediction Risk and Architecture Selection for Neural Networks*. Springer, 1994.

Journal of Machine Learning Research 10 (2009) 1263-1304 Submitted 8/07; Revised 11/08; Published 6/09

Perturbation Corrections in Approximate Inference: Mixture Modelling Applications

Ulrich Paquet ULRICH@CANTAB.NET
Computer Laboratory
University of Cambridge
Cambridge CB3 0FD, United Kingdom

Ole Winther OWI@IMM.DTU.DK
Informatics and Mathematical Modelling
Technical University of Denmark
DK-2800 Lyngby, Denmark

Manfred Opper OPPERM@CS.TU-BERLIN.DE
Computer Science
TU Berlin
D - 10587 Berlin, Germany

Editor: Zoubin Ghahramani

Abstract

Bayesian inference is intractable for many interesting models, making deterministic algorithms for approximate inference highly desirable. Unlike stochastic methods, which are exact in the limit, the accuracy of these approaches cannot be reasonably judged. In this paper we show how low order perturbation corrections to an expectation-consistent (EC) approximation can provide the necessary tools to ameliorate inference accuracy, and to give an indication of the quality of approximation without having to resort to Monte Carlo methods. Further comparisons are given with variational Bayes and parallel tempering (PT) combined with thermodynamic integration on a Gaussian mixture model. To obtain practical results we further generalize PT to temper from arbitrary distributions rather than a prior in Bayesian inference.

Keywords: Bayesian inference, mixture models, expectation propagation, expectation consistent, perturbation correction, variational Bayes, parallel tempering, thermodynamic integration

1. Introduction

Approximate methods for Bayesian inference have recently enjoyed a limelight of attention. These methods can be either deterministic or stochastic. Deterministic methods, which typically turn integration and summation problems of Bayesian marginalization into optimization problems, include the Laplace approximation, mean field (or variational) methods like variational Bayes (VB), expectation propagation (EP), and expectation consistent (EC) and Bethe/Kikuchi approximations (also known as loopy belief propagation or generalized belief propagation). Their attraction lies in the precise but tractable inferences that they typically provide, but their drawback is the lack of a built-in sanity check, as we cannot assess the approximation error. Stochastic methods like Markov chain Monte Carlo (MCMC) algorithms, which give exact estimates in a large enough sample limit, lie orthogonal to deterministic methods. They are normally much slower than their deterministic counterparts, but given a skilled user and enough computational resources stochastic methods are capable of giving more precise answers. Whether inference errors (of unknown size) are acceptable of course depends on the application in question. In statistical applications one might prefer simple models which allow for exact inferences, whereas in communication systems intractability is an inherent property of communication channels and to counter this, one instead designs fault tolerant error-correcting protocols.

The problem under consideration can be stated in general terms: We are presented with a data set of N independent and identically distributed (i.i.d.) examples $\mathcal{D} = \{x_n\}_{n=1}^N$, which we model by a generative model specified by the distribution $p(x|\theta)$, such that $p(\mathcal{D}|\theta) = \prod_n p(x_n|\theta)$. In Bayesian inference we introduce a prior distribution $p(\theta)$ over model parameters θ, and to infer unobserved random variables we compute different averages over the posterior distribution

$$p(\theta|\mathcal{D}) = \frac{1}{Z} p(\mathcal{D}|\theta)p(\theta) \quad \text{with} \quad Z = \int d\theta \, p(\mathcal{D}|\theta)p(\theta) \; . \tag{1}$$

In model selection or model averaging the normalizer (marginal likelihood) $Z = p(\mathcal{D})$ needs to be computed for different models under consideration, that is, $p(\mathcal{D}|\mathcal{M}_m)$, $m = 1, \ldots, |\mathcal{M}|$. Another central inference is about the density at a new (test) example, the so-called predictive density (or distribution):

$$p(x|\mathcal{D}) = \int d\theta \, p(x|\theta)p(\theta|\mathcal{D}) \; . \tag{2}$$

This paper mostly specializes to modelling the density with a mixture model

$$p(x|\theta) = \sum_{k=1}^{K} p(k)p(x|\theta_k)$$

such that mixing proportions $p(k)$ sum to one, and $\theta = \{p(k), \theta_k\}_{k=1}^K$. A mixture of Gaussians (MoG) corresponds to $p(x|\theta_k)$ being Gaussian. The prior distribution and the likelihood term for each component term $p(k)p(x|\theta_k)$ are chosen to be conjugate, such that their product is in the same distribution family as the prior and thus tractable. Intractability for the mixture model arises not because integration is intractable, but because the number of terms in the marginal likelihood is K^N.

This paper starts from the vantage point of an expectation consistent (EC) approximation (Opper and Winther, 2005) (and its algorithmic realisation by expectation propagation (EP) Minka 2001a) and substantiates these main contributions and findings:

1. We express the exact posterior distribution by an approximating distribution which is given by EC plus a series of error terms with increasing complexity. When low order corrections are small, one might hope that the remaining contributions will also decrease with the order, suggesting that the approximation can be improved by retaining only the lowest orders in the series. One can thus expect corrections to improve an already good approximation, but not a poor one. On the other hand, large lower order terms may indicate a poor approximation, providing an error check on the approximation without having to resort to Monte Carlo methods.

2. We derive corrections both for the marginal likelihood and the predictive distribution in the form of an expansion in terms of "clusters" of likelihood terms of the posterior. This expansion resembles the loop series expansions which were derived for correcting loopy belief propagation (LBP) (Chertkov and Chernyak, 2006, Gómez et al., 2007, Sudderth et al.,

2008).[1] All these methods hold in common that the correction terms are expressed as averages over the approximating solution and can thus be calculated *after* the convergence of the EP or LBP iterative scheme.

3. We show that our first order correction to the posterior can be simply expressed by quantities already computed by the EP algorithm. No further averages are needed. In contrast, the lowest non-trivial correction to the marginal likelihood is of second order, with the number of terms growing as $O(N^2)$. Corrections to the marginal likelihood can be tractably computed, for example, for models where the likelihood is a mixture distribution. Each of error terms contains the original K-component mixture, such that a correction up to order j requires the computation of $O((NK)^j)$ terms.

4. When the true distribution is multi-modal, EP will in most cases provide a local (single) mode approximation, with lower-order corrections also being local. One such example is the $K!$-fold labelling symmetry of the latent space of mixture models, which may cause $O(K!)$ separated modes in the posterior distribution. While the predictive distribution is invariant to this symmetry, the log marginal likelihood usually has to be further corrected by a factor of $O(\log K!)$, a correction that is typically much larger than a low-order perturbation correction.

5. Thorough empirical tests of EP validate its precision, and show errors that do not scale with N. The perturbation corrected predictions are almost uniformly more precise than EP. As a tool for improving inference accuracy, we show in a practical example that the first nontrivial correction term to the marginal likelihood approximation can make a clear difference in predicting which K maximizes the marginal likelihood, compared to when the correction was not used.

In this paper EC or EP and its resulting corrections are compared with variational Bayes (VB), Minka's α-divergence message passing scheme, and a gold standard benchmark of parallel tempering (PT) and thermodynamic integration (TI). PT is a Markov chain Monte Carlo (MCMC) method whose Markov chain operates on a "tempered posterior" and has very good convergence properties. Contrary to more standard Monte Carlo methods (for example Metropolis-Hastings or Gibbs sampling) it can also provide estimates of the marginal likelihood by TI, which interpolates the expected value of the log likelihood between the prior and the posterior. To increase the stability of estimates obtained by TI, we give a novel generalization of PT, which allows interpolation of the value of the log likelihood between *any* choice of distribution and the posterior. A good choice may also improve sampling when the tempered posterior exhibits phase transition-like properties. This choice might be obtained by some deterministic approximation, and although not investigated in this paper, provides a springboard for combining deterministic and stochastic inference algorithms.

As a further example it is also shown how the "cluster" perturbation expansion can be applied to Gaussian Process classification models, where the evaluation of integrals for Bayesian marginalization are not analytically tractable.

The rest of the paper follows with a description of EC and EP in Section 2. Section 3 shows an example of corrections for a marginal distribution in a Gaussian Process classification model.

1. An information geometrical expansion for LBP is given by Ikeda et al. (2004), and for EP by Matsui and Tanaka (2008). LBP can also be improved with a message passing algorithm that corrects for the influence of loops (Mooij et al., 2007).

In Section 4 an inference algorithm is presented for mixture weights, that is, a mixture model with fixed component densities, while Appendix D treats the fully multivariate MoG. Section 5 contains short descriptions of PT with TI and a generalization suitable for statistical inference. Results are presented for real world examples in Section 6, and we conclude in Section 7.

2. Expectation Consistent Inference

The *expectation consistent* approximation provides a framework for finding a surrogate distribution $q(\theta)$ for $p(\theta|\mathcal{D})$ in Bayesian inference (Opper and Winther, 2005).[2] The message passing scheme of *expectation propagation* gives rise to an identical marginal likelihood approximation, and the following interpretation sheds light on both methods by looking at them as a set of self-consistent approximations to marginal or predictive distributions. The outline presented here allows for further *perturbation corrections* to be derived.

For the purpose of this paper the EC approximation rests on the observation that the predictive density $p(x|\mathcal{D})$ in (2) can be fairly precisely approximated without averaging over the actual posterior. The entire posterior can be replaced with a simpler distribution $q(\theta)$ *if* it produces the correct statistics for this average, that is,

$$p(x|\mathcal{D}) = \int d\theta\, p(x|\theta)p(\theta|\mathcal{D}) \approx \int d\theta\, p(x|\theta)q(\theta)\ .$$

It is sufficient for $q(\theta)$ to share some key properties, namely low order statistics, with $p(\theta|\mathcal{D})$. This is an ambitious demand that is generally not realizable, but we can transfer the principle of moment matching to the "cavity" posteriors $p(\theta|\mathcal{D}_{\setminus n})$, which correspond to reduced training sets $\mathcal{D}_{\setminus n}$ where the n^{th} example has been left out. By introducing a similar approximation to the "cavity" predictive distributions

$$p(x_n|\mathcal{D}_{\setminus n}) = \int d\theta\, p(x_n|\theta)p(\theta|\mathcal{D}_{\setminus n}) \approx \int d\theta\, p(x_n|\theta)q_{\setminus n}(\theta)$$

for each x_n in the training set, a computationally efficient approximation can be derived. We shall now *rather* require $q(\theta)$ to share key properties, namely lower order statistics, with *each* of the distributions $q_n(\theta) \propto p(x_n|\theta)q_{\setminus n}(\theta)$; this is explored in the next section.

2.1 EC and EP with Exponential Families

EC defines a tractable approximation $q(\theta)$ through expectation consistency with each $q_n(\theta)$. Our view of EC shall be narrowed to models factorizing in likelihood terms $p(x_n|\theta)$, and an exponential family prior

$$p(\theta) = \frac{1}{Z_0} \exp\left(\Lambda_0^T \phi(\theta)\right) h(\theta)\ ,$$

where Z_0 is the normalizing constant, $\phi(\theta)$ is a fixed vector of the corresponding sufficient statistics—for example for a univariate Gaussian we can choose $\phi(\theta) = \left(\theta, -\theta^2/2\right)$, Λ_0 is the associated parameter vector and the fixed function $h(\theta)$ encodes additional constraints (positivity, normalizations, etc.). The desired quality of approximation, and the possible convenience of obtaining tractable moments, typically guide the choice of $\phi(\theta)$.

2. A more general interpretation is possible, but for clarity we show the approximation for the generative model in (1).

The posterior will be approximated with a tractable density of the same exponential family as the prior,

$$q(\theta) = \frac{1}{Z(\Lambda,0)} \exp\left(\Lambda^T \phi(\theta)\right) p(\theta) \; . \tag{3}$$

By adding the condition $\Lambda = \sum_n \Lambda_n$, we allow each likelihood factor $p(x_n|\theta)$ of the posterior in (1) to correspond to simpler factor proportional to $\Lambda_{\setminus n} = \Lambda - \Lambda_n$ in (3): the Λ_n's therefore parameterize the likelihood term contributions to the approximation.[3] We here introduced a definition for normalization as

$$Z(\Lambda,a) = \int d\theta \prod_n \left[p(x_n|\theta)\right]^{a_n} \exp\left(\Lambda^T \phi(\theta)\right) p(\theta) \; ,$$

with a being a vector with elements a_n. The cavity posterior $p(\theta|\mathcal{D}_{\setminus n})$ should then be approximated by a member of the same exponential family

$$q_{\setminus n}(\theta) \propto \exp\left(\Lambda_{\setminus n}^T \phi(\theta)\right) p(\theta) \; ,$$

where $\Lambda_{\setminus n} = \Lambda - \Lambda_n$. This is obtained from (3) by removing a single likelihood approximation and renormalizing.

Let 1_n be a unit-vector in the n^{th} direction. We can now formalize our concluding remark: $q(\theta)$ is required to share lower order statistics with the *tilted* distributions

$$q_n(\theta) = \frac{1}{Z(\Lambda - \Lambda_n, 1_n)} p(x_n|\theta) \exp\left((\Lambda - \Lambda_n)^T \phi(\theta)\right) p(\theta) \; , \tag{4}$$

each of which are obtained from the posterior $p(\theta|\mathcal{D})$ by replacing the cavity posterior by its approximation. We therefore require consistency of the generalized moments, that is,

$$\left\langle \phi(\theta) \right\rangle_q = \left\langle \phi(\theta) \right\rangle_{q_n} \; , \qquad n = 1, \ldots, N \; .$$

One can also show that the corresponding marginal likelihood approximation is given by

$$Z_{\text{EC}} = Z(\Lambda,0) \prod_n \frac{Z(\Lambda - \Lambda_n, 1_n)}{Z(\Lambda,0)} \tag{5}$$

(Minka, 2005, Opper and Winther, 2005). In Appendix A we relate this approximation to variational bounds on the marginal likelihood.

2.1.1 EXPECTATION PROPAGATION

The final expression for the EC partition function in (5) depends upon the partition functions for two distributions q and q_n in (3) and (4), and consistency on the statistics $\phi(\theta)$ determines the Λ_n parameters. This moment consistency can be achieved via a message passing framework called EP, which appear, together with VB,[4] as special cases of a more generic message passing framework recently

3. In this context the likelihood terms (factors) are sometimes referred to as *sites*, and hence the Λ_n's as *site parameters* of *site functions* that are proportional to $\exp(\Lambda_n^T \phi(\theta))$ (Seeger, 2003).

4. VB finds its approximation $q(\theta)$ by lower-bounding the log marginal likelihood with Jensen's inequality (Jordan et al., 1999), giving $\log Z_{\text{VB}} \leq \log p(\mathcal{D})$. By writing

$$\log Z_{\text{VB}} = -\text{KL}(q(\theta) \| p(\theta|\mathcal{D})) + \log p(\mathcal{D})$$

the bound can be made as tight as possible by adjusting $q(\theta)$; this is achieved by minimizing the KL-divergence between $q(\theta)$ and $p(\theta|\mathcal{D})$.

Algorithm 1 EP message passing (Minka, 2001a)

```
1: initialize: Set all $\Lambda_n$ to zero, $\Lambda_n \leftarrow 0$, $n = 1, \ldots, N$. This choice corresponds to initializing in
   the prior, setting the sufficient statistics to $\mu \leftarrow \langle \phi(\theta) \rangle_{p(\theta)}$.
2: repeat
3:    Randomly choose example $n$, and make the following update steps:
4:    Update sufficient statistics
          $\mu \leftarrow \left\langle \phi(\theta) \right\rangle_{q_n(\theta;\Lambda_n)}$ .
5:    Determine $q(\theta;\Lambda)$ from $\mu$, that is, solve
          $\left\langle \phi(\theta) \right\rangle_{q(\theta;\Lambda')} = \mu$
      with respect to $\Lambda'$ and update
          $\Delta\Lambda \leftarrow \Lambda' - \Lambda$ followed by $\Lambda \leftarrow \Lambda + \Delta\Lambda$ .
      The EP updates can also be damped by $\gamma \in [0,1]$ through $\Delta\Lambda \leftarrow \gamma(\Lambda' - \Lambda)$.
6:    Update $q_n(\theta;\Lambda_n)$:
          $\Lambda_n \leftarrow \Lambda_n + \Delta\Lambda$ .
      This update ensures that $\Lambda = \sum_n \Lambda_n$; $q$ and $q_n$ are therefore in the forms of (3) and (4). We
      have no guarantee in this step that $q_n$ stays a proper distribution. A robust heuristic is to skip
      any update that makes $q_n$ improper.
7: until expectation consistency $\langle \phi(\theta) \rangle_{q_n(\theta;\Lambda_n)} = \langle \phi(\theta) \rangle_{q(\theta;\Lambda)} = \mu$ holds for $n = 1, \ldots, N$.
```

proposed by Minka (2005). EP defines a specific message algorithm which iteratively refines each Λ_n by minimising local Kullback-Leibler divergences $\mathrm{KL}(q_n(\theta)\|q(\theta))$; in other words it iteratively performs the required moment matching $\langle \phi(\theta) \rangle_q = \langle \phi(\theta) \rangle_{q_n}$. EP is presented in Algorithm 1 for our choice of q and q_n, and we shall henceforth use the terms EP and EC interchangeably.

If EP converges we will have expectation consistency $\langle \phi(\theta) \rangle_{q_n(\theta)} = \langle \phi(\theta) \rangle_{q(\theta)} = \mu$ because of the moment matching in lines 4 and 5 of Algorithm 1. Line 6 ensures that q and q_n follow the forms in (3) and (4). Solving for q in line 5 is analytical for most of the parameters as long as q is in the exponential family. (In the mixture of Gaussian examples in this paper, one has to solve two independent scalar non-linear equations for Dirichlet and Wishart densities. All other vector and matrix parameters can be found analytically.)

EP is not guaranteed to converge, in which case double-loop algorithms may be used. It has been observed by Heskes and Zoeter (2002) that when EP does not converge to a stable fixed point, even when considerable damping (choosing γ small in Algorithm 1) is used, the corresponding double-loop algorithm has a Hessian with a significantly negative eigenvalue(s). It has been suggested that the failure of convergence of canonical EP usually implies an inaccurate solution, with the choice of approximating family not being rich enough (Minka, 2001a).

2.2 Perturbation Corrections

The goal of this section is to derive formal expressions for the errors of the EC approximation to the marginal likelihood and the predictive distribution and to discuss ways of how this error can be computed using a formal perturbation expansion. In order to expand the EC approximation we use (4) to express each likelihood term by the approximating densities as

$$p(x_n|\theta) = \frac{Z(\Lambda - \Lambda_n, 1_n)}{Z(\Lambda, 0)} \frac{q_n(\theta)}{q(\theta)} \exp\left(\Lambda_n^T \phi(\theta)\right) ,$$

to find that

$$p(\theta)\prod_n p(x_n|\theta) = Z_{\mathrm{EC}}\, q(\theta) \prod_n \left(\frac{q_n(\theta)}{q(\theta)}\right) . \tag{6}$$

If we define

$$\varepsilon_n(\theta) = \frac{q_n(\theta) - q(\theta)}{q(\theta)}$$

such that $\frac{q_n(\theta)}{q(\theta)} = 1 + \varepsilon_n(\theta)$, we should expect $\varepsilon_n(\theta)$ to be on average small over a suitable measure when the EC approximation works well. Bearing this definition in mind, the exact posterior and the exact marginal likelihood can be written as

$$p(\theta|\mathcal{D}) = \frac{1}{R}\, q(\theta) \prod_n (1 + \varepsilon_n(\theta)) \qquad \text{and} \qquad Z = Z_{\mathrm{EC}}\, R , \tag{7}$$

with

$$R = \int d\theta\, q(\theta) \prod_n (1 + \varepsilon_n(\theta)) .$$

We expect that an expansion of posterior and Z in terms of $\varepsilon_n(\theta)$ truncated at low orders might give the dominant corrections to EC. Hence, we get the (2^N term finite) expansion

$$R = 1 + \sum_{n_1 < n_2} \left\langle \varepsilon_{n_1}(\theta)\varepsilon_{n_2}(\theta) \right\rangle_q + \sum_{n_1 < n_2 < n_3} \left\langle \varepsilon_{n_1}(\theta)\varepsilon_{n_2}(\theta)\varepsilon_{n_3}(\theta) \right\rangle_q + \ldots , \tag{8}$$

showing that EC is correct to the first order as the term $\sum_n \langle \varepsilon_n(\theta) \rangle_q = 0$ vanishes. The posterior in (7) can be similarly expanded with

$$p(\theta|\mathcal{D}) = \frac{q(\theta)\, \left(1 + \sum_n \varepsilon_n(\theta) + \sum_{n_1<n_2} \varepsilon_{n_1}(\theta)\varepsilon_{n_2}(\theta) + \ldots\right)}{1 + \sum_{n_1<n_2} \langle \varepsilon_{n_1}(\theta)\varepsilon_{n_2}(\theta) \rangle_q + \ldots} , \tag{9}$$

where we should keep as many terms in the numerator as in the denominator in order to keep the resulting density normalized to one.

The corresponding predictive distribution is

$$\begin{aligned} p(x|\mathcal{D}) &= \int d\theta\, p(x|\theta)\, p(\theta|\mathcal{D}) \\ &= \frac{\int d\theta\, q(\theta)\, p(x|\theta)\, \left(1 + \sum_n \varepsilon_n(\theta) + \sum_{n_1<n_2} \varepsilon_{n_1}(\theta)\varepsilon_{n_2}(\theta) + \ldots\right)}{1 + \sum_{n_1<n_2} \langle \varepsilon_{n_1}(\theta)\varepsilon_{n_2}(\theta) \rangle_q + \ldots} , \end{aligned} \tag{10}$$

where again as many terms in the numerator as in the denominator should be kept to ensure proper normalization.

If the expansions in (9) and (10) are truncated, the approximations are not guaranteed to be valid probability distributions, since as functional approximations they may be negative. Nevertheless, the quality of EC approximation is still improved, as is illustrated in Figures 1, 8, 12, and Table 1.

2.3 Tractability of Corrections

For the case where q_n is just a finite *mixture* of K simpler densities from the exponential family to which q belongs, then the number of mixture components in the j-th term of the expansion of R is just of the order $O(K^j)$ and an evaluation of low order terms is tractable and can be computed in $O((KN)^j)$ after q has been found.

In other cases, an exact computation of even the low order terms may be analytically intractable. If the dimensionality of necessary integrations is proportional to the order of the correction one may still resort to numerical quadratures. A different approach would be to re-expand each term ε_n in a different "measure of closeness" of densities which takes into account the moments $\phi(\theta)$ of the densities. This can be for example achieved in the case where $q(\theta)$ is Gaussian and the statistics $\phi(\theta)$ denote just the set of all first and a subset of second moments (or cumulants) of the random variable θ. Then we could resort to the use of characteristic functions $\chi(\kappa)$ and $\chi_n(\kappa)$ defined through

$$q(\theta) = \int d\kappa\, e^{i\kappa^T\theta}\chi(\kappa), \qquad q_n(\theta) = \int d\kappa\, e^{i\kappa^T\theta}\chi_n(\kappa)$$

for all n. The coefficients in a formal multivariate Taylor expansion of $\log\chi_n(\kappa)$ in powers of the vector κ define (up to a factor) the *cumulants* of q_n. Hence, the multivariate Taylor expansion of $r_n(\kappa) \equiv \log\chi_n(\kappa) - \log\chi(\kappa)$ in powers of κ contains only those cumulants in which q_n and q *differ*. Thus, we may write

$$\begin{aligned} q_n(\theta) - q(\theta) &= \int d\kappa\, e^{i\kappa^T\theta}\chi(\kappa)\left(1 - e^{\log\left(\frac{\chi_n(\kappa)}{\chi(\kappa)}\right)}\right) = \int d\kappa\, e^{i\kappa^T\theta}\chi(\kappa)\left(1 - e^{r_n(\kappa)}\right) \\ &= -\int d\kappa\, e^{i\kappa^T\theta}\chi(\kappa)\left(r_n(\kappa) + \frac{1}{2}r_n^2(\kappa) + \dots\right) . \end{aligned} \tag{11}$$

Hence, when the statistics $\phi(\theta)$ contain *all* first and *all* second moments of θ, the integral is expressed through cumulants of order 3 and higher. In this way the error of the EC approximation can be expressed in terms of higher order cumulants.

If we expand r_n in powers of κ, it is possible to express the integral (11) explicitly in a series containing derivatives of increasing order of the Gaussian $q(\theta) = \int d\kappa\, e^{i\kappa^T\theta}\chi(\kappa)$ with respect to θ. This is because each such derivative creates a factor κ in the Fourier integral via differentiations of the exponential $e^{i\kappa^T\theta}$. Finally, each term $\varepsilon_n(\theta) = \frac{q_n(\theta)-q(\theta)}{q(\theta)}$ can then be expressed by a series of Hermite polynomials in a standard way. This alternative expansion is introduced by Opper et al. (2008); its details and applications will be presented in a future paper.

2.4 First Order Correction

We have seen that in general, higher order correction terms require the computation of extra expectations. Remarkably, in contrast, the first order correction to the EC posterior (9) is obtained as simple sum of terms which where already computed in the EC approximation. Hence, it provides a simple and efficiently computable quantity to improve on EC/EP or judge its validity. A straightforward calculation gives

$$p(\theta|\mathcal{D}) \approx \sum_n q_n(\theta) - (N-1)q(\theta) . \tag{12}$$

The first order correction does *not* change the moments which are consistent in EC, but provides an approximation to nontrivial higher cumulants, which, for example, in the case of a Gaussian $q(\theta)$ would be *zero* in EC.

3. Gaussian Process Classification

The cluster expansion can be applied in a limited setting to non-parametric models with a Gaussian process prior. This section provides as an introductory case a correction to the marginal distribution, illustrating that a lower-order correction can be very accurate. For this family of models corrections to other quantities of interest, for example the log marginal likelihood and predictive distribution, have to rely on cumulant expansions (Opper et al., 2008), and will be treated in detail a companion paper.

A Gaussian process prior forms the cornerstone of many popular non-parametric Bayesian methods. It has been used to great effect on various regression and classification problems. A Gaussian prior is placed on an N-dimensional unobserved variable f, for example

$$p(f) = \mathcal{N}(f\,; 0, K) \ ,$$

where each f_n is associated with an input vector x_n, and K is a kernel matrix with entries $k(x_n, x_{n'})$ (Rasmussen and Williams, 2005). A binary classification task attaches a class label $y_n \in \{-1, +1\}$ to each input x_n, and a typical prediction would be the class of a new input x_* given the data $\mathcal{D} = \{x_n, y_n\}_{n=1}^{N}$. It is common to use the cumulative Normal distribution function $\Phi(\cdot)$ as a likelihood for correctly classifying a data point (Opper and Winther, 2000). The likelihood is dependent on the unobserved f_n associated with x_n, and hence

$$p(y_n|f_n) = \Phi(y_n f_n) \ .$$

The posterior distribution of f is therefore

$$p(f|\mathcal{D}) = \frac{1}{Z} \prod_{n=1}^{N} p(y_n|f_n)\, \mathcal{N}(f\,; 0, K) \ .$$

With this factorization the site functions are chosen to depend on only f_n such that the posterior is approximated by the same exponential family distribution (Gaussian) as the prior,

$$q(f) \propto \prod_{n=1}^{N} \exp\left(\tilde{\nu}_n f_n - \frac{1}{2}\tilde{s}_n f_n^2\right) \mathcal{N}(f\,; 0, K) \ .$$

The notation in this section is deliberately chosen to be consistent with that of Rasmussen and Williams (2005, chapter 3), and we refer the reader to the reference for an example EP algorithm. We assume that a fixed point of EP has been reached. Let $\tilde{S}$ be a diagonal matrix containing $\tilde{s}_n$, and $\tilde{\nu}$ be a vector containing $\tilde{\nu}_n$. The posterior approximation is therefore $q(f) = \mathcal{N}(f; \mu, \Sigma)$, with $\Sigma = (K^{-1} + \tilde{S})^{-1}$ and $\mu = \Sigma\tilde{\nu}$.

The cavity posterior approximations $q_{\setminus n}(f) = \mathcal{N}(f; \mu_{\setminus n}, \Sigma_{\setminus n})$ arise from setting $\tilde{\nu}_n = \tilde{s}_n = 0$ (giving diagonal matrix $\tilde{S}_{\setminus n}$ and vector $\tilde{\nu}_{\setminus n}$), where $\Sigma_{\setminus n}$ can be determined with a rank one update of Σ. The tilted distributions are therefore $q_n(f) \propto q_{\setminus n}(f)\,\Phi(y_n f_n)$.

The first order correction (12) can be applied to compute a correction to the marginal distribution of f_*, the latent function associated with a novel input x_*. Integrating $p(f_*|f)$ with (12) yields

$$p(f_*|\mathcal{D}) \approx \sum_{n=1}^{N} q_n(f_*) - (N-1) q(f_*) \ . \tag{13}$$

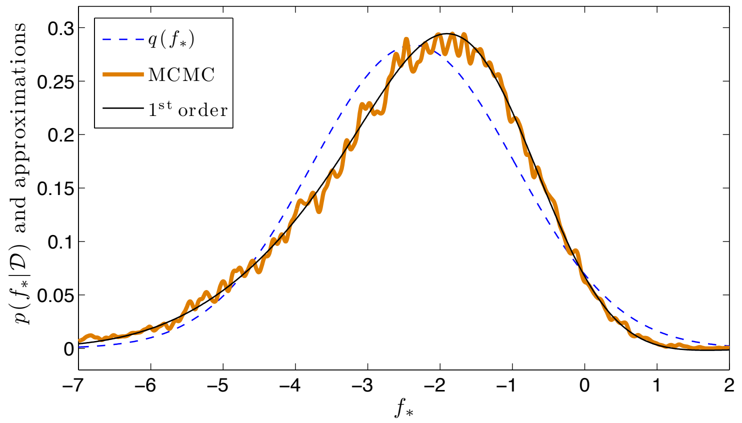


Figure 1: The first-order correction (13) is shown in black, with $q(f_*)$ in blue. Full details about the data set in question, as well as the individual terms in (13), are illustrated in Figure 13 in Appendix B. An MCMC estimate for the true marginal is overlayed in orange, and comes from averaging $p(f_*|f)$ over 20,000 MCMC samples from the posterior of $p(f|\mathcal{D})$. The "spikiness" is a result of the variance of $p(f_*|f)$ being very narrow: if the "noise-free" latent f is given, then f_* is highly correlated with f and well determined for this example. The first-order correction gives an excellent approximation.

Notice that corrections for the predictive distribution and log marginal likelihood cannot be expressed analytically in this way. Hence numerical quadrature or an expansion in terms of cumulants (Opper et al., 2008) is required. Higher-order terms of the above correction are also analytically intractable.

The detailed derivation of the correction is presented in Appendix B. Figure 1 provides a summary comparison of a first-order correction, $q(f_*)$, and a MCMC estimate of $p(f_*|\mathcal{D})$. The correction is very accurate and provides a much better fit than EC or EP at a negligible additional computational cost. Figure 13 in Appendix B gives further illustrations to accompany Figure 1. In Section 2.4 it was noted that the first order correction provides an approximation to nontrivial higher cumulants which would otherwise be *zero* in EC, even though the moments which are consistent in EC are not changed. Figure 2 illustrates this observation, showing an accurate approximation of the third cumulant for various distributions $p(f_*|\mathcal{D})$.

4. Mixture of Gaussians

We shall empirically examine the corrections to EP approximations through a multivariate mixture of Gaussians (MoG). Mixture models provide a more challenging testbed for EP than the Gaussian Process model illustrated in Section 3, as the posterior is multi-modal with many symmetries, and the site distributions are not log-concave. For clarity we relegate the MoG derivations to Appendix D, favouring a simpler but similar model here. As an outline to deriving an algorithm for a MoG we

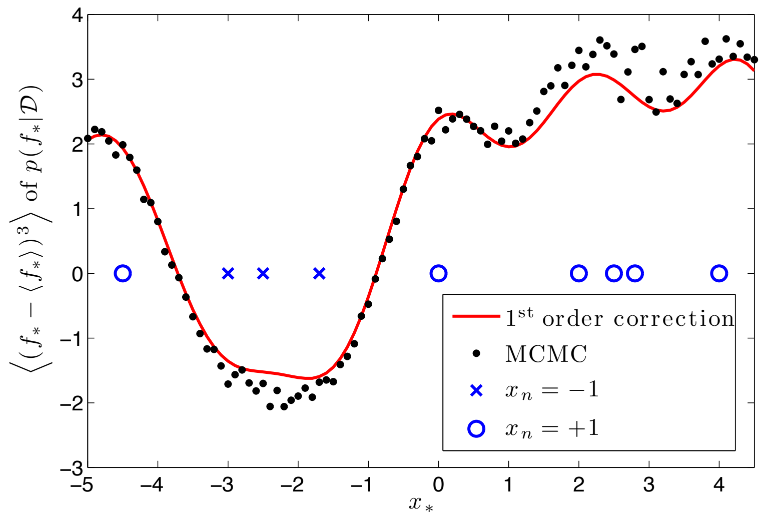


Figure 2: For different inputs x_*—and hence latent function f_*—the *third* cumulant of the first-order correction (13) is shown in red. It closely matches the true third cumulant of $p(f_*|f)$, which is plotted in black. The EC approximation $q(f_*)$'s higher cumulants are all zero. Figure 1 shows the particular approximations at $x_* = -2$, with further details appearing in Figure 13 and Appendix B.

consider the task of inferring the mixing proportions $\pi_k = p(k)$ in a model of the form

$$p(x|\theta) = \sum_k \pi_k p(x|k) \ ,$$

with $p(x|k)$ being fixed (Minka, 2001b). Since the mixing proportions should sum to one a Dirichlet prior for π is is a natural choice, and Appendix C gives a detailed description of all its properties needed in this context. We give the explicit EP message passing updates for the mixing proportions with fixed component densities in Algorithm 2 (this scheme is generalized to adaptive components in a straightforward way in Appendix D). Details for the required computations in Algorithm 2 are given below.

4.1 Variational and Predictive Distributions

The prior—and thus also the q-distribution in (3)—are Dirichlet,

$$q(\pi) = \mathcal{D}(\pi;\lambda) \ ,$$

with $\mathcal{D}(\pi;\lambda)$ given in Appendix C by (23). The parameters of q are $\lambda_k = \lambda_{k,0} + \sum_n \lambda_{k,n}$ (here λ_0 are the parameters of the prior, which we include into λ for simplicity).

We can also get the EC approximation to the predictive distribution both for new datum x, $p(x|\mathcal{D})$ and the cavity predictive distribution: $p(x_n|\mathcal{D}_{\setminus n})$. For the new datum x the approximation is straightforward using $q(\pi)$ as an approximate posterior:

$$p(x|\mathcal{D}) \approx \int d\pi\, p(x|\pi)\, q(\pi) = \sum_k \left\langle \pi_k \right\rangle_q p(x|k) \,, \tag{14}$$

with the mean value being

$$\left\langle \pi_k \right\rangle_q = \frac{\lambda_k}{\sum_{k'} \lambda_{k'}} \,.$$

For the "within data set" version we introduce the cavity distribution $q_{\setminus n}(\pi) = \mathcal{D}(\pi;\lambda_{\setminus n})$, using $\lambda_{k\setminus n} = \lambda_k - \lambda_{k,n}$, and derive a result that is very similar to the one above:

$$p(x_n|\mathcal{D}_{\setminus n}) \approx \sum_k \left\langle \pi_k \right\rangle_{q_{\setminus n}} p(x_n|k) \,. \tag{15}$$

For message passing we also need expectations of $q_n(\pi)$ from (4):

$$q_n(\pi) = \frac{1}{Z_n(\lambda_{\setminus n}, 1_n)} e^{\sum_{k'} (\lambda_{k'\setminus n} - 1) \log \pi_{k'}}\, \delta\left(\sum_{k'} \pi_{k'} - 1 \right) \sum_k \pi_k p(x_n|k) \,.$$

The above normalizer can easily be found by noting that

$$q_n(\pi) = \frac{Z(\lambda_{\setminus n}, 0)}{Z_n(\lambda_{\setminus n}, 1_n)} q_{\setminus n}(\pi) \sum_k \pi_k p(x_n|k) \,,$$

such that

$$Z_n(\lambda_{\setminus n}, 1_n) = Z(\lambda_{\setminus n}, 0) \sum_k \left\langle \pi_k \right\rangle_{q_{\setminus n}} p(x_n|k) \,.$$

In this simple case we have $Z(\lambda_{\setminus n}, 0) = Z_{\mathcal{D}}(\lambda_{\setminus n})$, with the normalization $Z_{\mathcal{D}}$ of the Dirichlet being given by (24) in Appendix C.

4.2 Expectations

When updating $\mu \leftarrow \langle \phi(\theta) \rangle_{q_n(\theta;\Lambda_n)}$ in Algorithm 2 the sufficient statistics can be computed using $\log Z_n(\lambda_{\setminus n})$ as a generating function:

$$\left\langle \log \pi_k \right\rangle_{q_n} = \frac{d \log Z_n(\lambda_{\setminus n})}{d\lambda_{k\setminus n}} = \left\langle \log \pi_k \right\rangle_{q_{\setminus n}} + \frac{r_{nk}}{\lambda_{k\setminus n}} - \frac{1}{\sum_{k'} \lambda_{k'\setminus n}} \,, \tag{16}$$

where the expression for $\langle \log \pi_k \rangle_{q_{\setminus n}}$ is given by (25) in Appendix C with $\lambda \to \lambda_{\setminus n}$, and the "responsibility" r_{nk} was introduced as

$$r_{nk} = \frac{\lambda_{k\setminus n} p(x_n|k)}{\sum_{k'} \lambda_{k'\setminus n} p(x_n|k')} \,.$$

Algorithm 2 Message Passing for Mixing Proportions

```
1: initialize: Set $\lambda_{k,n} \leftarrow 0$, for $n = 1,\dots,N$ and $k = 1,\dots,K$, initializing $q(\pi)$ to the prior. Set $\mu_k \leftarrow \langle \log \pi_k \rangle_{p(\pi)} = \psi(\pi_{k,0}) - \psi(\sum_k \pi_{k,0})$, where the digamma function $\psi(x)$ is defined as $\log \Gamma(x)/dx$.
2: repeat
3:    Randomly choose example $n$, and make the following update steps:
4:    Update the sufficient statistics
          $$\mu_k \leftarrow \left\langle \log \pi_k \right\rangle_{q_n(\pi;\lambda_n)} = \psi(\lambda_k - \lambda_{k,n}) - \psi\left(\sum_k (\lambda_k - \lambda_{k,n})\right) .$$
5:    Determine $q(\pi;\lambda')$ from $\mu$, that is, by solving $\langle \log \pi_k \rangle_{q(\pi;\lambda')} = \mu_k$ with respect to $\lambda'$. As shown in Appendix C, this involves solving for $\alpha \equiv \psi(\sum_k \psi^{-1}(\mu_k + \alpha))$, followed by with $\lambda'_k = \psi^{-1}(\mu_k + \alpha)$. Update
          $$\Delta\lambda_k \leftarrow \lambda'_k - \lambda_k \quad \text{and} \quad \lambda_k \leftarrow \lambda'_k .$$
6:    Update $q_n(\pi;\lambda_n)$ with
          $$\lambda_{k,n} \leftarrow \lambda_{k,n} + \Delta\lambda_k ,$$
      ensuring that $\lambda_k = \lambda_{k,0} + \sum_n \lambda_{k,n}$.
7: until expectation consistency $\langle \log \pi_k \rangle_{q_n(\pi;\lambda_n)} = \langle \log \pi_k \rangle_{q(\pi;\lambda)} = \mu$ holds $\forall n, k$.
8: Compute $\log Z_{\mathrm{EC}}$ from (5) with
          $$\log Z_{\mathrm{EC}} = \sum_n \log Z(\lambda - \lambda_0 - \lambda_n, 1_n) - (N-1) \log Z(\lambda - \lambda_0, 0)$$
          $$= \sum_n \log \left[ \frac{Z_{\mathcal{D}}(\lambda_{\setminus n})}{Z_{\mathcal{D}}(\lambda)} p(x_n | \mathcal{D}_{\setminus n}) \right] + \log Z_{\mathcal{D}}(\lambda_0) + \log Z_{\mathcal{D}}(\lambda) ,$$
   where $p(x_n|\mathcal{D}_{\setminus n})$ signifies the "cavity" predictive distribution from (15).
```

5. Parallel Tempering and Thermodynamic Integration

Having considered deterministic inference algorithms, the last bit of machinery that we shall need is a stochastic method to provide exact estimates in a large enough sample limit. Parallel tempering (PT) and thermodynamic integration (TI) are ideal for our purposes: PT is an efficient method of combining separate Monte Carlo simulations to sample across different modes of a target distribution and, as a by-product, TI can be used to estimate the normalizing constant or log marginal likelihood.

We conclude this section with a new practical generalization of PT and TI, which can in principle be used to combine stochastic and approximate methods. A further novel extension to the generalization is given in Appendix E.2.

5.1 Parallel Tempering (Replica Exchange)

A single MCMC simulation may run into difficulties if the target distribution is multimodal. The chain may get stuck in a local mode, and fail to fully explore other areas of the parameter space that have significant probability. A conceptual solution to this problem is to create a series of progressively flatter distributions through an inverse temperature parameter β, which ranges from zero to one. This gives a "tempered" posterior

$$p(\theta|\mathcal{D},\beta) = \frac{1}{Z(\beta)} p(\mathcal{D}|\theta)^{\beta} p(\theta) \ , \tag{17}$$

where the normalizing constant (partition function) is $Z(\beta) = \int d\theta \, p(\mathcal{D}|\theta)^{\beta} p(\theta)$. The prior is recaptured with $\beta = 0$, and the posterior with $\beta = 1$. We now simulate N_β replicas of (17) in parallel, each using a $\beta \in \{\beta_i\}_{i=1}^{N_\beta}$. Let the set $\{\beta_i\}$ be ordered as a ladder with $\beta_i < \beta_{i+1}$. The parameter space is replicated N_β times to $\{\theta_i\}_{i=1}^{N_\beta}$, and the full target distribution that is being sampled from is

$$p(\{\theta_i\}) = \prod_{i=1}^{N_\beta} \frac{1}{Z(\beta_i)} \exp\Big(\beta_i \log p(\mathcal{D}|\theta_i)\Big) p(\theta_i) \ .$$

We run the N_β chains independently to sample from distributions $p(\theta|\mathcal{D},\beta_i)$, and add an additional *replica-exchange* Metropolis-Hastings move to swap two β's, or equivalently two parameters, between chains. Let $\{\theta_i\}^{\text{new}}$ be a parameter set with θ_i and θ_j swapped. The acceptance probability of the move is $p(\text{accept}) = \min(1, p(\{\theta_i\}^{\text{new}})/p(\{\theta_i\}))$, and the acceptance ratio simplifies to

$$\frac{p(\{\theta_i\}^{\text{new}})}{p(\{\theta_i\})} = \exp\Big((\beta_i - \beta_j)\big(\log p(\mathcal{D}|\theta_j) - \log p(\mathcal{D}|\theta_i)\big)\Big) \ . \tag{18}$$

The temperatures of the two replica i and j have to be close to ensure non-negligible acceptance rates; neighboring pairs are typically taken as candidates. To fully satisfy detailed balance, pairs $\{i, i+1\}$ can be uniformly chosen, for example. With this formulation the states of the replicas are effectively propagated between chains, and the mixing of the Markov chain is facilitated by the fast relaxation at small β's.

From (18), the acceptance probability depends on the difference between $\log p(\mathcal{D}|\theta_i)$ and $\log p(\mathcal{D}|\theta_{i+1})$, and for some swaps to be accepted this difference should not be "too big"; there should be an *overlap* of some of the log likelihood evaluations of adjacent chains, as illustrated in Figure 3. For a simulation at inverse temperature β, define the mean evaluation of the log likelihood as

$$\big\langle \log p(\mathcal{D}|\theta) \big\rangle_\beta = \int d\theta \, \log p(\mathcal{D}|\theta) \, p(\theta|\mathcal{D},\beta) \ .$$

If we knew the variance in chain β, $\sigma_\beta^2 = \langle [\log p(\mathcal{D}|\theta)]^2 \rangle_\beta - \langle \log p(\mathcal{D}|\theta) \rangle_\beta^2$, then it can be shown that temperatures should be chosen according to the density $Q(\beta) \propto \sigma_\beta$ (Iba, 2001). This is obviously difficult, as σ_β^2 is not known in advance, and has to be estimated. Good results can be achieved under the assumption $\sigma_\beta^2 \propto 1/\beta^2$ (the equivalent of assuming a *constant* heat capacity in a physical system), giving a *geometric* progression, hence choosing β_i/β_{i+1} constant (Kofke, 2002).

5.2 Thermodynamic Integration

The samples from parallel tempering can be used for model comparison (Gregory, 2005, Skilling, 1998), as the marginal likelihood can be obtained from tempering. Firstly, notice that the integral

$$\int_0^1 d \log Z(\beta) = \int_0^1 d\beta \, \frac{d \log Z(\beta)}{d\beta} = \log Z(1) - \log Z(0) = \log Z(1) = \log p(\mathcal{D})$$

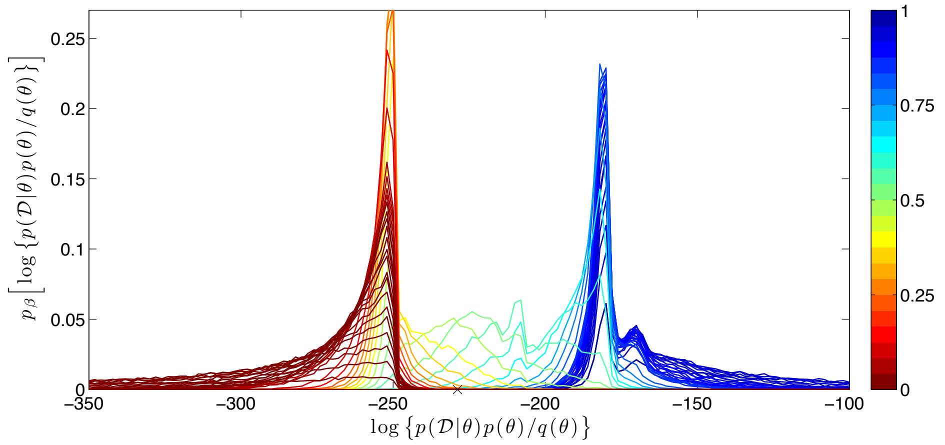


Figure 3: The density of $\log\{p(\mathcal{D}|\theta)p(\theta)/q(\theta)\}$ under replicas at different temperatures, $p(\theta|\mathcal{D},\beta)$, defined in (20). These densities correspond to "energy histograms," and following (18) there should be an overlap between adjacent replicas at different temperatures, so that acceptance of configuration or parameter swaps is allowed for. For interest, the log marginal likelihood $\log p(\mathcal{D})$ is indicated with a $\times$. The color bar indicates the inverse temperature β. This illustration comes from the **galaxy** data set with $K = 3$ components.

is equal to the log marginal likelihood, as $\beta = 0$ gives the prior, which integrates to one. We therefore have to determine the derivative $\frac{d}{d\beta}\log Z(\beta)$. By taking the derivative of the log normalizer (log partition function), we see that it evaluates as an average over the posterior

$$\frac{d\log Z(\beta)}{d\beta} = \frac{1}{Z(\beta)}\int d\theta \log p(\mathcal{D}|\theta) \times p(\mathcal{D}|\theta)^{\beta} p(\theta) = \left\langle \log p(\mathcal{D}|\theta)\right\rangle_{\beta} .$$

The log marginal likelihood equals

$$\log p(\mathcal{D}) = \int_0^1 d\beta \left\langle \log p(\mathcal{D}|\theta)\right\rangle_{\beta} \tag{19}$$

and can be numerically estimated from the Markov chain samples. If $\{\theta_i^{(t)}\}$ represents the samples for tempering parameter β_i, then the expectation is approximated with

$$\left\langle \log p(\mathcal{D}|\theta)\right\rangle_{\beta_i} \approx \frac{1}{T}\sum_{t=1}^{T} \log p(\mathcal{D}|\theta_i^{(t)}) .$$

We assume that a burn-in sample is discarded in the sum over t. As a set of chains are run in parallel at different inverse temperatures $0 = \beta_1 < \cdots < \beta_{N_\beta} = 1$, the integral can be evaluated numerically by interpolating the N_β expectations between zero and one (say with a piecewise cubic Hermite interpolation, available as part of `Matlab` and other standard software packages), and using for example the trapesium rule to obtain the desired result. Figure 4 illustrates how $\log p(\mathcal{D})$ is estimated.

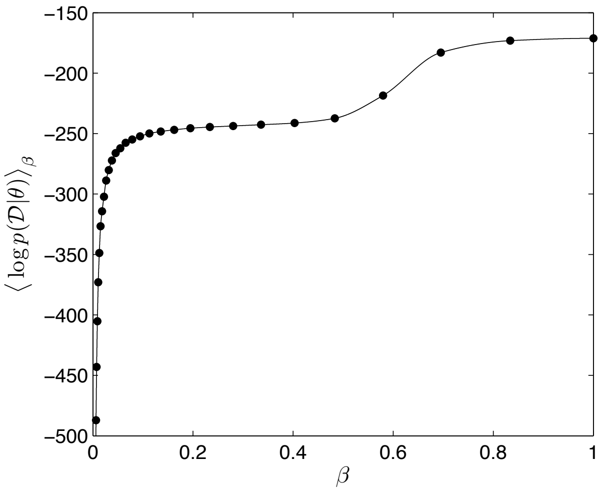


Figure 4: The log likelihood averages $\langle \log p(\mathcal{D}|\theta)\rangle_\beta$ are estimated from each of the MCMC simulations at temperatures $\{\beta_i\}$, and interpolated, so that Equation (19)'s integral can be evaluated numerically. This illustration comes from the **galaxy** data set with $K = 3$ components.

Parallel tempering can be done *complementary* to any Monte Carlo method at a single temperature. Appendix E presents Gibbs sampling to sample from $p(\theta|\mathcal{D},\beta)$ for the MoG problem.

5.3 A Practical Generalization of Parallel Tempering

The success of the interpolation obtaining $\langle \log p(\mathcal{D}|\theta)\rangle_\beta$, illustrated in Figure 4, is dependent on the slope

$$\frac{d\langle \log p(\mathcal{D}|\theta)\rangle_\beta}{d\beta} = \frac{d^2 \log Z(\beta)}{d\beta^2} = \sigma_\beta^2$$

at $\beta \approx 0$. Consider the following thought exercise: Imagine a non-informative (infinitely wide) prior at $\beta = 0$. Samples from this prior will strictly speaking have an infinite variance σ_0^2. With $\beta \approx 0$ we introduce the likelihood, practically infinitely decreasing the variance of our samples, causing $\langle \log p(\mathcal{D}|\theta)\rangle_\beta$ to asymptotically diverge at zero. As we narrow our prior the change in this mean should be less rapid, and this motivates a generalization of PT and TI such that we get a more stable interpolation.

We introduce a new distribution $q(\theta)$, which might be a narrower version of the prior, and modify (17) to

$$p(\theta|\mathcal{D},\beta) = \frac{1}{Z(\beta)} \left[p(\mathcal{D}|\theta) \frac{p(\theta)}{q(\theta)} \right]^\beta q(\theta) \; . \tag{20}$$

The log marginal likelihood can, as before, be determined with

$$\log p(\mathcal{D}) = \int_0^1 d\beta \left\langle \log p(\mathcal{D}|\theta) + \log \frac{p(\theta)}{q(\theta)} \right\rangle_\beta \; .$$

It is evident that setting $q(\theta) = p(\theta|\mathcal{D})$ gives an integral over a constant function, $\log p(\mathcal{D}) = \int_0^1 d\beta \, \langle \log p(\mathcal{D}) \rangle_\beta$. This suggests a wealth of possibilities of approximating $p(\theta|\mathcal{D})$ with $q(\theta)$ to effectively combine deterministic methods of inference with Markov chains. This comes with a cautionary note as VB, for example, may give a $q(\theta)$ that captures (lower-bounds) a mode of a possibly multimodal posterior, causing PT to lose its pleasing property of fast relaxation at high temperatures. In our results presented in Section 6, we have found it completely adequate to use a narrower version of the prior where necessary. Appendix E concludes with a short generalization to sample from (20) for the MoG problem.

6. Results

Low order corrections provide the tools to both improve inference accuracy, and to give an indication of the quality of approximate solutions. We illustrate and elaborate on these claims, with comparisons between various deterministic and stochastic methods, through this practical discussion. Data is viewed as being observed from a mixture model $p(x|\theta) = \sum_{k=1}^K \pi_k \, \mathcal{N}(x; \mu_k, \Gamma_k^{-1})$, as is discussed in Appendices D, E, and F. A Dirichlet prior is placed on π, and Normal-Wishart priors on μ_k and Γ_k; the approximating distribution $q(\theta) = q(\pi) \prod_k q(\mu_k, \Gamma_k)$ follows the same distribution as the prior.

6.1 Modes and Symmetries

Mixture models are invariant under component relabelling, with a $K!$ growth in the number of permutations also manifesting itself in symmetries in the posterior density. In aid of interpreting later results, we present some basic understanding of VB, EP, and low order corrections under this property. Our aim in this section is to use simple toy posteriors to facilitate discussion on the behavior of q under various scenarios and discuss how that might affect the estimation of the marginal likelihood and predictive distribution.

The labelling of hidden units of a two-layer neural network gives rise to symmetries similar to those observed in mixture models. For neural networks a statistical mechanics analysis shows that for small N the posterior is uni-modal and "star-like," as convex combinations of parameters with high posterior value which are equivalent under permutations will also have high density (Engel et al., 1992). The symmetry is broken into equivalent disconnected modes for large N.

For mixture models we can analyze the situation where q is restricted to approximate the posterior in one of the symmetric modes, as what will typically be the solution for both VB and EP/C when N is large. Minimizing the KL-divergence $\mathrm{KL}(q\|p)$ leads to a solution where q is proportional to p within the mode (and by construction zero otherwise). If there are $K!$ modes contributing equally to the normalizer and q is restricted to one of them, then q's normalizer is a factor of $K!$ smaller than p's and consequently $\mathrm{KL}(q\|p) = \log K!$ at the minimum (Bishop, 2006, page 484). However, *groups* of equivalent modes are often present. A simple example is a 3-component mixture with three "clusters" of data. If each component is associated with a cluster, there are 3! labelling symmetries. Another VB or EP fixed point may prune one mixture component (see MacKay 2001 for VB and Figure 11 for EP), leaving one component to cover two clusters of data, and one component the other; this solution has *yet another* 3! labelling symmetries, albeit possibly with a lesser contribution to the normalizer. In effect the correction is rather $O(\log K!)$, as illustrated in Figure 9. A useful approximation would be to correct the marginal likelihood estimate by a factor of $K!$ when N is large. The predictive distribution is invariant under the symmetry and will thus

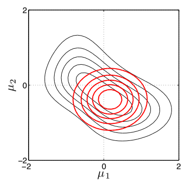
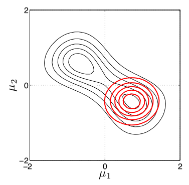
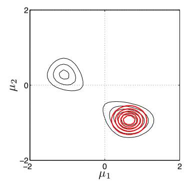
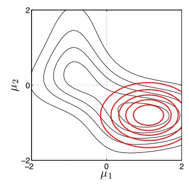
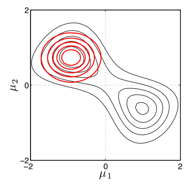
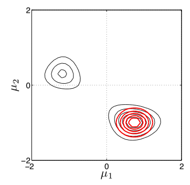
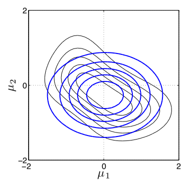
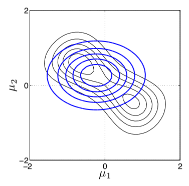
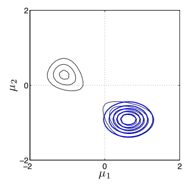
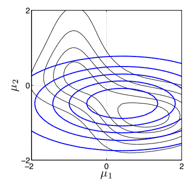
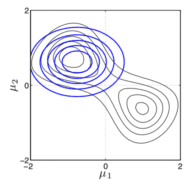
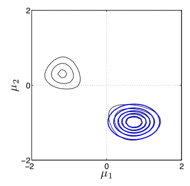
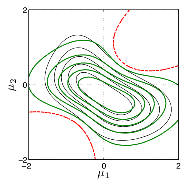
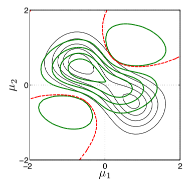
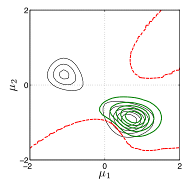
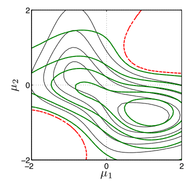
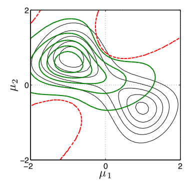
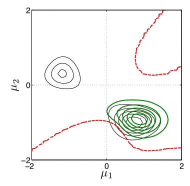
$N = 7$	$N = 14$	$N = 40$	$N = 7$	$N = 14$	$N = 40$
narrower prior: 			broader prior: 		
					
					

Table 1: A comparison between the VB (top row), and EC (middle row) approximations $q(\theta)$, and a first order correction to the EC approximation (bottom row). Data is assumed to come from a two-component mixture with *only* the means $\theta = \{\mu_1, \mu_2\}$ unknown. Under various priors and data set sizes we show the posterior $p(\theta|\mathcal{D})$ in thin black lines, with the VB, EC, and first-order corrected approximations overlaid in thicker lines. The first order correction integrates to one but is not guaranteed to be nonnegative (bottom row); dashed red lines are used to demarcate the regions of parameter space where the correction dips below zero.

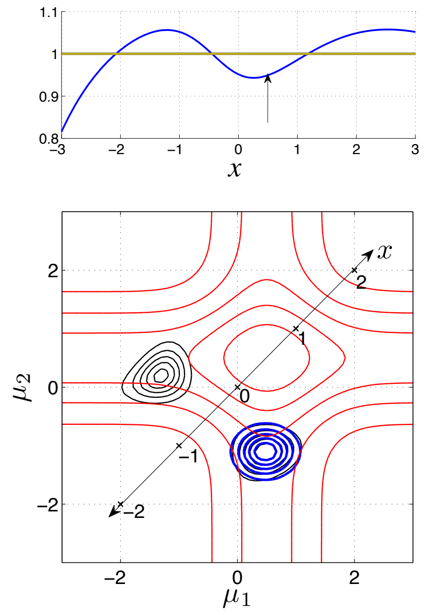
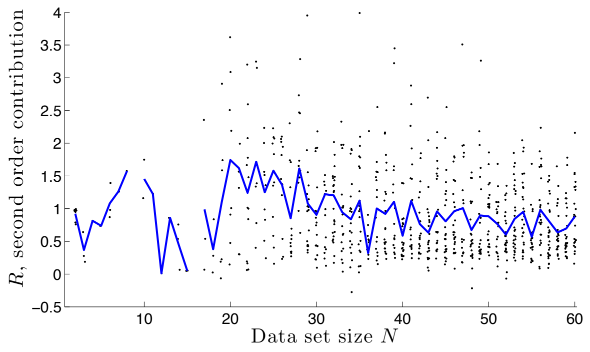


Figure 5: Symmetries and averages: The likelihood $p(x|\{\mu_1,\mu_2\}) = 0.4\mathcal{N}(x;\mu_1,1)+0.6\mathcal{N}(x;\mu_2,1)$ is plotted as red contours for the novel observation $x=\frac{1}{2}$ at the arrow in the top figure. Observing x centres the likelihood function at (x,x) along the μ_1–μ_2 axis in the bottom figure. The *predictive density* $p(x|\mathcal{D})$ is average of the likelihood over the *bi-modal* posterior (black contours), while the *approximate* predictive density is the average of the likelihood function over the *uni-modal* EC approximation (overlaid in blue). The near-symmetry of the posterior implies that each mode contributes approximately *half* its mass. When the EC approximation puts *all* its mass on one mode, and the modes are well separated, the two predictive densities are therefore similar. The top figure shows the ratio between the true and approximate $p(x|\mathcal{D})$; the discrepancy at negative x is due to the fact that the posterior is not perfectly symmetrical (e.g., when $x=-2$ its likelihood is centred at (-2,-2) and overlaps less with the EC mode).

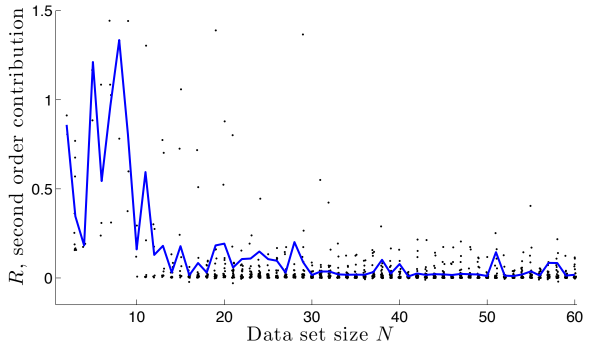
not be greatly affected by q approximating only one mode, as is shown in Figure 5. For small to intermediate values of N the situation is less clear, as the following example illustrates.

In Table 1 we illustrate a number of posterior distributions, with the VB and EC approximations overlaid. We also overlay the first order correction to $q(\theta)$, given from (12) by $p(\theta|\mathcal{D}) \approx \sum_n q_n(\theta) - (N-1)q(\theta)$. For Table 1 all parameters but the means were kept fixed, such that with $K=2$ the approximation $q(\theta)=q(\mu_1)q(\mu_2)$ is a factorised Gaussian. Both component variances were equal, and we used $(\pi_1,\pi_2)=(0.4,0.6)$. The modes are thus not completely symmetric but this set-up still illustrates the points made above well. We chose the component variances (set to one) such that the posterior modes overlap when N is small, with bimodality arising as N increases. We will see in the following sections that even though q is a rather crude approximation to the posterior the predictions for the predictive distribution are fairly precise.

The correction given by (12) integrates to one but is not guaranteed to be nonnegative, as it follows from discarding the higher order terms in (7). The first order correction to the predictive



(a) With overlapping mixture components, damped EP does not necessarily converge for small N (e.g., EP failed, in the sense that the 2nd order correction cannot be computed, on all 30 random data sets of size 8). The corrections are on average (blue line) large for broadly overlapping posterior modes, as EP does not necessarily lock onto one of them.



(b) With well separated clusters the 2nd order corrections indicate for which N, on average, EP prefers a modal solution. Note the better convergence of EP for larger N, with on average stabler fixed points than Figure 6(a). This is reflected in the corrections being close to zero.

Figure 6: The second order term of (8) on 3600 random data sets.

density, however, usually remains nonnegative because it is an *average* of $p(x|\theta)$ over (12). This underlines the fact that average properties will not be strongly affected by imprecision in approximating distributions.

Other local minima that we did not show in Table 1 was for N small, where $q(\mu_1)$ remains as broad as the prior and the component is effectively pruned, while $q(\mu_2)$, on the other hand, caters for both mixture components (MacKay, 2001).

6.2 Corrections

The illustrations in Table 1 suggest that the lowest nontrivial corrections can provide insight into the quality of approximation, as we expect corrections to be small for good approximations. To illustrate this claim, 30 random data sets $\mathcal{D}_N$ were drawn for each size $N = 1, \ldots, 60$ according to $\mathcal{D}_N \sim p(x)$, with $p(x)$ being a three-component mixture with $\pi = (0.2, 0.3, 0.5)$ and $\mu = (-2, 0, 2)$. Two cases were used for the variance: firstly, $\Gamma_k^{-1} = 0.5$ provides a model with overlapping mixture

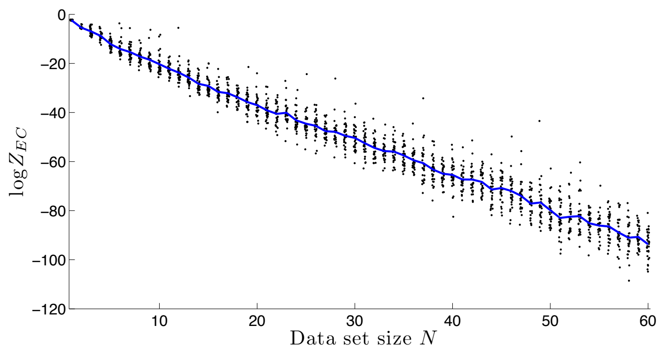


Figure 7: The growth of $\log Z_{\text{EC}}$ for the random data sets $\mathcal{D}_N$ used to obtain Figure 6(b).

components; secondly, $\Gamma_k^{-1} = 0.1$ gives a model with components that are further separated. EP was run with damping $\gamma = 0.5$, and the second order corrections to the log marginal likelihood approximations were computed where possible (i.e., EP converged, etc.); see Appendices D and F.

Figures 6(a) and 6(b) illustrate the "overlapping" and "separate" examples, showing that when N is small compared to $\log K!$, and the posterior is "starlike" and comparatively unimodal, EP often fails, and the nature of the problem is reflected in the large corrections. When N becomes large EP often converges (to one of $K!$ equivalent modes); small corrections immediately tell us that solution is close to exact, apart from here a $\log K!$ correction to the marginal likelihood.

We also observe that the corrections do not scale with N, whereas the free energy $\log Z_{\text{EC}}$ does, as shown in Figure 7. This is an important property, as it means that the quality of approximation does not deteriorate with increasing N.

When the observations $\mathcal{D}_N \sim p(x)$ are i.i.d. we expect that $\log Z_{\text{EC}}/N$, by its form as an empirical average over N terms, should converge to a non-random c_Z as $N \to \infty$. In fact a linear scaling $\log Z_{\text{EC}} \to c_Z N$ is observed in Figure 7. When and whether the expected correction $\langle \log R \rangle_{\mathcal{D}_N} = c_R(N) \to 0$ as $N \to \infty$ (and hence EP becomes exact) is an open question. This does not seem true for Figure 6(a): If the the posterior modes were well-separated then for large N, a change in one mean parameter in a factorized approximation will not greatly affect the other. If, in this case, the means are close compared to the standard deviations of the normal densities, the mean parameters will stay correlated also for large data sets, and the corrections will persistently stay bigger for large N.

6.3 Toy Example

To illustrate the difference between EC and VB, and show additional gains from perturbation corrections, we generated a small data set ($N = 7$) from a mixture of two Gaussians. The hyperparameters followed that of Section 6.4.

Under two model assumptions we show in Figure 8 that EC or EP (labeled "EC/P") gives a predictive density that is generally closer to the truth than that given by VB. Each example in the toy data set was duplicated (see Figure 8(b)) to show that this gain decreases under larger data sets; this decrease is due to the predictive density being an average of $p(x|\theta)$ over now more concentrated VB and EC posterior approximations. Secondly, meaningful improvements can be achieved through perturbation corrections. Figure 8(d) shows a second order correction to the log marginal likelihoods

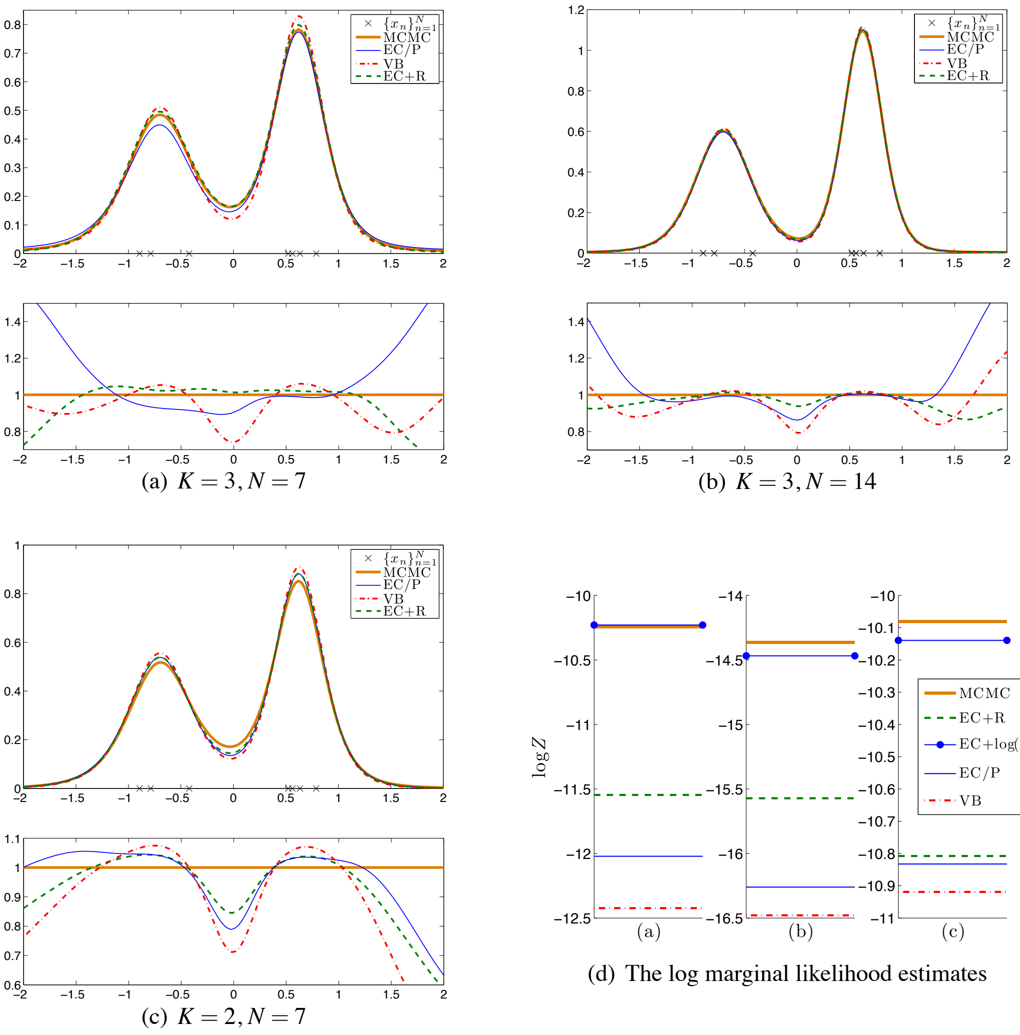


Figure 8: Predictive densities $p(x|\mathcal{D}, \mathcal{M}_K)$ given by VB, EC, and a perturbation correction (EC+R), with accompanying log marginal likelihood estimates and MCMC "truth" baselines. Note that if we "correct" with a factor $\log K!$ we get very close to the "truth" for VB and to a even higher degree for EC. EC+R overshoots in two cases but that might be because the perturbation corrected posterior is actually multi-modal. The lower figures in 8(a) to 8(c) show the *ratio* between each of the approximate predictive densities and the "truth."

of the examples in question, labeled "EC+R" (see Appendix F). A lower bound to the log marginal likelihood is provided by VB. The improvement is also visible when we are concerned with the predictive density, for which we show a first order correction.

6.4 A Practical Comparison

In this section we draw a comparison between the approximate log marginal likelihoods and predictive distributions given by VB, EP, and various corrections, and use estimations given by PT and TI as a benchmark. For interest we also include results from an implementation of $\alpha = \frac{1}{2}$ in Minka's general α-divergence message passing scheme for this problem, but refer the interested reader to Minka (2005) for further details. Finding a VB approximation follows directly from the expectation maximisation algorithm given by Attias (2000), with a slightly different parameterization of the Wishart distribution.

From the results that follow, we observe that the growth of $\log Z$, as a function of model size, gives a characteristic "Ockham hill" (defined in more detail later in this section), where the "peak" of the hill indicates the model with highest approximate $\log p(\mathcal{D})$. This graph can be used for model comparison or selection, as its form closely matches the MCMC evaluation of $\log p(\mathcal{D})$. We will also see that, following Section 6.1's discussion, the discrepancy between a $\log Z$ estimate and the true $\log p(\mathcal{D})$ grows as the model size is increased. Furthermore, the EC approximation gives a predictive distribution that is closer to the truth than VB, with the gain decreasing with increasing N. We will show that a principled algorithm initialization can circumvent many spurious local minima in the log marginal likelihood estimate. If completely arbitrary initialization schemes are implemented, one may note that the number of local solutions is influenced by the width of the prior distribution, with *more* local minima arising under broader prior distributions.

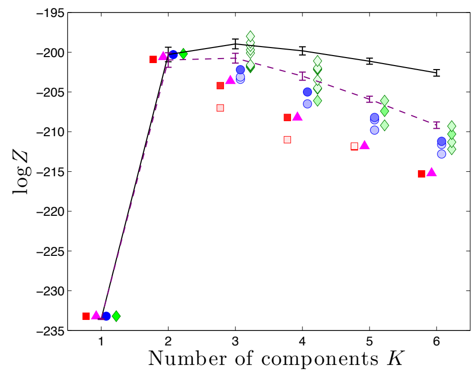
The data sets under investigation have been well studied, for example, by Richardson and Green (1997) for a reversible jump MCMC, and by Corduneanu and Bishop (2001) for variational Bayesian model selection: the **galaxy** data set contains the velocities (in 1000s of km/second) of 82 galaxies, diverging from our own, in the Corona Borealis region; the **acidity** data set contains the log measured acid neutralizing capacity indices for 155 lakes in North-central Wisconsin (USA); the **enzyme** data set contains enzymatic activity measurements, for an enzyme involved in the metabolism of carcinogenic substances, taken from 245 unrelated individuals; the **old faithful** data set contains 222 observation pairs consisting of eruption time and waiting time to the next eruption, from the Old Faithful Geyser in the Yellowstone National Park.

6.4.1 THE APPROXIMATE LOG MARGINAL LIKELIHOOD

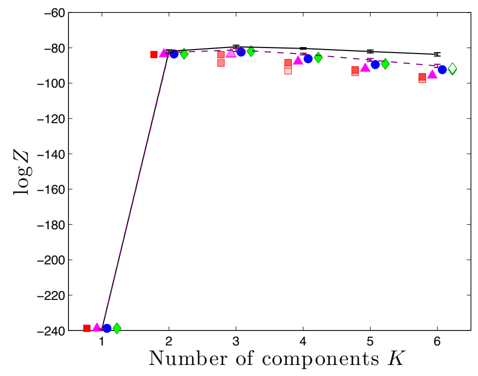
Ockham hills are useful for visualizing log marginal likelihood estimates for a set of plausible models with increasing explanatory power, for example, mixture models with increasing K. The largest estimates of $\log Z$ for the various models typically form a hill, peaking at the "optimal" model. As models become *less* complex, the hill falls steeply due to a poorer explanation of the data. For *more* complex models the plots show a slower downward trend, as an improvement in data fit is counterbalanced by a penalty from a larger parameter space in Bayesian marginalization. For mixture models this downward trend is even slower when the true log marginal is considered; this is mainly due to the number of modes in the true posterior increasing with the number of components, with an approximation possibly only capturing one of them.[5]

In the case of VB, the $\log Z$ approximation provides a lower bound to the marginal likelihood $p(\mathcal{D}|\mathcal{M})$, and this quantity is often used for model selection (Beal and Ghahramani, 2003, Bishop and Svensén, 2003, Corduneanu and Bishop, 2001). The model with the largest bound is typically kept, although the bound can also be used for model averaging. Regardless of our method of

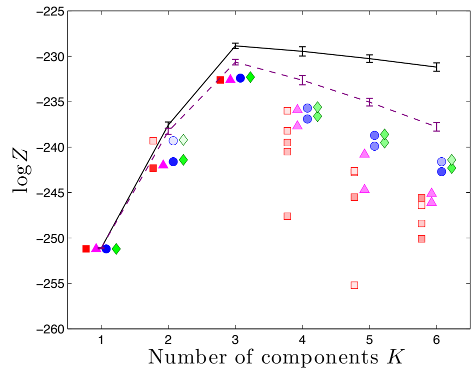
5. Rasmussen and Ghahramani (2001) present an account which includes "Ockham plateaus."



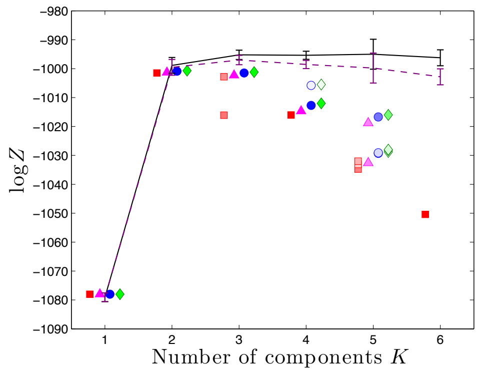
(a) **acidity** data set



(b) **enzyme** data set



(c) **galaxy** data set



(d) **old faithful** data set

Figure 9: Ockham hills for various data sets. VB is shown as red squares, $\alpha = \frac{1}{2}$ as magenta triangles, EC/P as blue circles, and EC+R as green diamonds. An estimate of $\log p(\mathcal{D}|\mathcal{M}_K)$ found by PT and TI is shown as a line. The effect of an $O(\log K!)$ correction on any of the approximate solutions can be seen by comparing them against the dashed-line plot of $\log p(\mathcal{D}|\mathcal{M}_K) - \log K!$. (For Figure 9(d)'s $K = 6$ the EP and $\alpha = \frac{1}{2}$ schemes did not converge.)

approximation, poor local minima in the objective function have to be avoided in order to obtain meaningful results.

Figure 9 shows such hills for the marginal likelihood approximations for different data sets for VB, $\alpha = \frac{1}{2}$ message passing, EP, and a second-order perturbation correction. The prior hyperparameters were $\lambda_{k,0} = 1$, $m_{k,0} = 0$, $\nu_{k,0} = 10^{-2}$, $a_{k,0} = 1$ and $B_{k,0} = 0.11$. For Figure 9(d) we took $B_{k,0} = [0.11, 0.01; 0.01, 0.11]$. For each of the models $\mathcal{M}_K$, with K mixture components, the figures show twenty approximations for each method, with the colour intensity of each plot corresponding to the frequency of reaching different approximations for $\log Z$. Each plot is complemented with estimates of $\log p(\mathcal{D}|\mathcal{M}_K)$. The estimates—shown as lines—were obtained form an average over ten PT and TI simulations, with two standard deviation error bars also being shown.

Finally, it is evident that the "true peak" in Figure 9(a) does not match the peak obtained by approximate inference. Without having to resort to MCMC and TI, the second order correction for $K = 3$ already confirms that the approximation might be inadequate.

6.4.2 THE EFFECT OF A GOOD INITIALIZATION

Finding the best VB/EP solution is strongly seed-dependent in the problem considered here. In this council of despair an educated guess may take us a long way: many inferior local minima in the VB/EP objective functions can be suppressed with a good algorithm initialization.

We base our factor initializations around a scaled version of the solution obtained by the VB expectation maximisation algorithm,

$$\exp(\Lambda_n^T \phi(\theta)) \propto \exp\left(\int dz_n \, q(z_n) \log p(x_n, z_n|\theta) \right) ,$$

which was seeded with a data clustering based on the k-means algorithm.[6] This is illustrated in Figure 9.

When using an "out of the box" EP scheme, starting with a slight asymmetric prior that is later corrected for, many lower minima are also found. The same behavior arises when the VB parameters are randomly initialized. Figure 10 shows more local minima than Figure 9(c), and the results in Bishop (2006, chapter 10), where the same principled initial guess for VB was used.[7]

The canonical EP scheme (and indeed $\alpha = \frac{1}{2}$) sometimes did not converge to a fixed point. This is evident in Figure 10 and has been observed in practice (Minka, 2001a): when EP does not converge, the reason can be traced back to the approximating family being a poor match to the exact posterior distribution.

6.4.3 THE PREDICTIVE DISTRIBUTION

Given a specific model $\mathcal{M}$, the predictive distribution can be approximated by using $p(x|\mathcal{D}) \approx \int d\theta \, p(x|\theta) q(\theta)$, as is shown for example in Figure 11. The final predictive distribution strongly depends on whether or not a global minimum in the objective function in (5) has been found, as is clear from Figure 11. To illustrate how much the approximate predictive distribution differs from the true predictive distribution, the figures show $p(x|\mathcal{D})$ obtained from an average over ten thousand $\beta = 1$ samples from a parallel tempered Markov chain. Figure 12 shows the gain achieved by EC/P over VB, and in turn the further improvement from a perturbation correction to the EC approximation (see Appendix F).

7. Conclusion and Outlook

In this paper we presented a method for computing systematic corrections to EC approximations in Bayesian inference. These corrections are useful not only in improving estimates like log marginal likelihood and predictive density approximations, but can also provide insight into the quality of an approximation in polynomial time. When the corrections are large the EC approximation may be

6. Similar to Appendix E, z indicates latent variables, with $p(\theta, z|\mathcal{D})$ approximated by $q(\theta)q(z)$. We point the interested reader to Attias (2000).
7. Markus Svensén, personal communication.

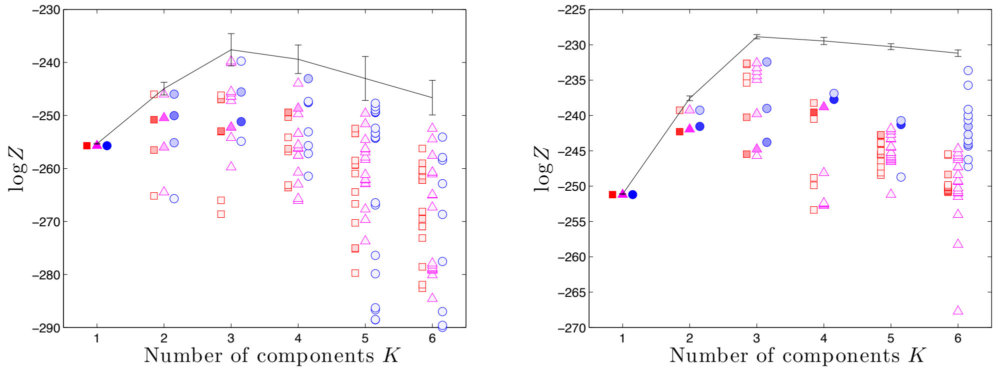


Figure 10: The effect of random algorithm initializations using the **galaxy** data set: For the *left* figure a broader prior with $\nu_{k,0} = 10^{-6}$, and for the *right* figure a much narrower prior with $\nu_{k,0} = 10^{-2}$, was used. Compared to Figure 9(c), note for example the additional local maxima at $K = 3$, and the greater number of local minima under a broader prior. (For $K = 6$ the EP scheme failed to converge without a sensible initialization.)

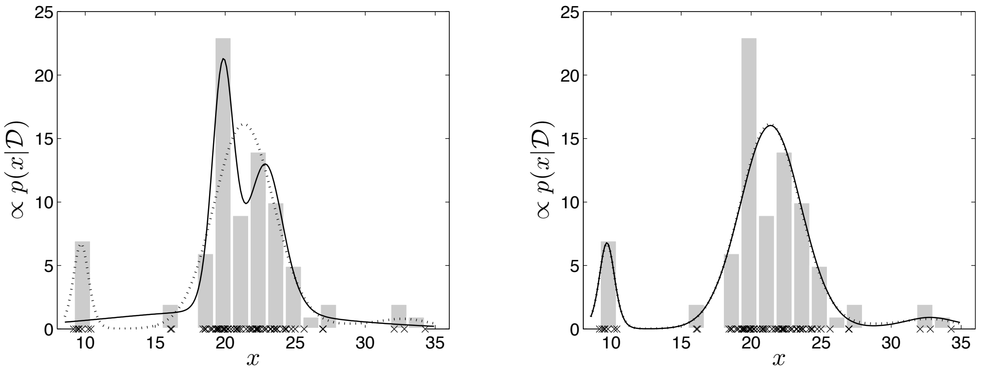


Figure 11: EP can have more than one stable fixed point: The predictive distribution $p(x|\mathcal{D}, \mathcal{M}_3)$, from two *different* approximations for the **galaxy** data set. For $K = 3$ under narrower prior in Figure 10, we see three local maxima of the EC objective function in (5): the predictive distribution shown on the *left* coincided with $\log Z_{\text{EC}} = -243.8$, whereas the approximation on the *right* coincided with a much higher $\log Z_{\text{EC}} = -232.4$. The true predictive distribution, obtained from an average over a PT MCMC sample, is shown with a dotted line.

questioned or discarded, and we hope to address the question of how it is done in practice in future work.

A juxtaposition of VB, EC and EP, PT with TI, and EC with corrections, was given in the context of Gaussian mixture models. We argued that EC can give improvements over VB, and can in turn be improved through a perturbation expansion. Throughout the paper our "gold standard" was given by PT, and we presented possible ways of improving it. We would like to include better

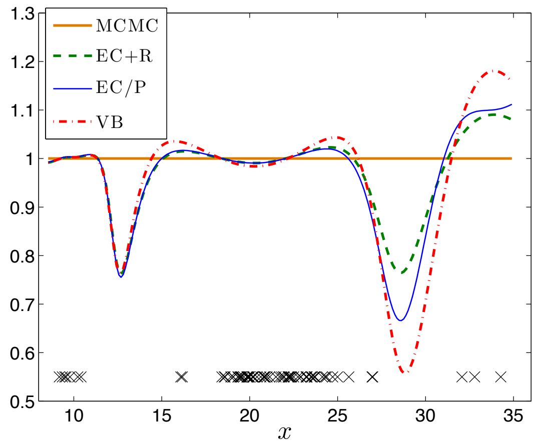


Figure 12: The ratio between each of the approximate predictive densities and the MCMC "truth" of $p(x|\mathcal{D},\mathcal{M}_3)$ for the **galaxy** data set. This figure corresponds to Figure 11 (right).

MCMC algorithms in this rich tapestry of methods: PT is not the best choice for near first-order phase transitions. In Figure 3 the high probability regions are very different above and below the transition at $\beta \approx 0.5$, suggesting multicanonical sampling as a viable alternative, since it aims at sampling from a distribution that is flat in the log likelihood and will therefore not have this "bottle-neck."

The choice of a unimodal $q(\theta)$ to capture the characteristics of a typically multimodal $p(\theta|\mathcal{D})$ also leads to various questions. When there are symmetries in the parameter space, with overlapping modes, we may ask whether or not we would achieve a better predictive density with EC, say, if the approximation is restricted to one mode. In the case where $p(\theta|\mathcal{D})$ is multimodal (large N) then fairly general arguments suggest that we should correct the marginal likelihood estimate by a factor of $K!$—higher order corrections may clarify for which N and under which conditions this transition will typically take place.

One way be improve the approximation is to generalize $q(\theta)$ for example by including a small fraction of the data points (similar to the proposed generalization of PT). However, that poses an additional problem an the matching of moments step in EP message passing gets much more complicated.

Finally, we focused on a MoG where the lower order terms in the correction R are tractable. For models where this is not the case, R can be expanded in terms of the higher order cumulants of $q_n(\theta)$ and $q(\theta)$. This approach will be presented in a companion paper.

Acknowledgments

We thank Zoubin Ghahramani for suggesting the generalization presented in Appendix E.2. We thank Tom Minka, Markus Svensén, and Neil Lawrence for helpful discussions and comments, and the anonymous reviewers for their comments.

Appendix A. Bounds on the Marginal Likelihood

It is interesting to compare the marginal likelihood approximation (5) with the one given by a variational approximation. Here one would use the fact that the relative entropy $\left\langle \log \frac{q(\theta)}{p(\theta|\mathcal{D})} \right\rangle_q \geq 0$ to approximate the free energy $-\log Z$ by the upper bound

$$-\log Z \leq \left\langle \log \left(\frac{q(\theta)}{p(\theta) \prod_n p(x_n|\theta)} \right) \right\rangle_q .$$

To compare with (5) we use the definition (3) to get

$$\frac{Z(\Lambda - \Lambda_n, 1_n)}{Z(\Lambda, 0)} = \left\langle p(x_n|\theta) \exp\left(-\Lambda_n^T \phi(\theta) \right) \right\rangle_q .$$

After inserting this expression into (5), taking logs and applying Jensen's inequality we arrive at

$$-\log Z_{\text{EC}} \leq \left\langle \log \left(\frac{\exp\left(\sum_n \Lambda_n^T \phi(\theta) \right)}{\prod_n p(x_n|\theta)} \right) \right\rangle_q - \log Z(\Lambda, 0) = \left\langle \log \left(\frac{q(\theta)}{p(\theta) \prod_n p(x_n|\theta)} \right) \right\rangle_q ,$$

where in the last step we have used $\sum_n \Lambda_n = \Lambda$. This shows that if one would use the distribution $q(\theta)$ derived from EC within the variational approximation, the EC approximation achieves a lower free energy. Since approximating densities in the variational approximation will usually differ from the EC result (by the way they are optimised) this does not imply that variational free energies are always higher than the EC counterpart. Also we cannot draw any conclusion about the relation to the true free energy.

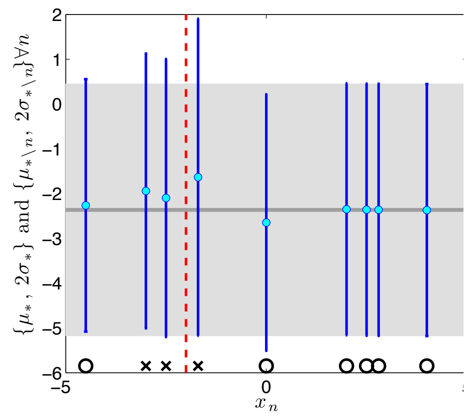
Appendix B. Gaussian Process Classification

This appendix provides the details of the derivation of first order correction to marginal distribution $p(f_*|\mathcal{D})$ for Gaussian process classification, as introduced in Section 3. Let k_* be the kernel vector with entries $k(x_*, x_n)$ for all n, and $\kappa_* = k(x_*, x_*)$. It is well-established that

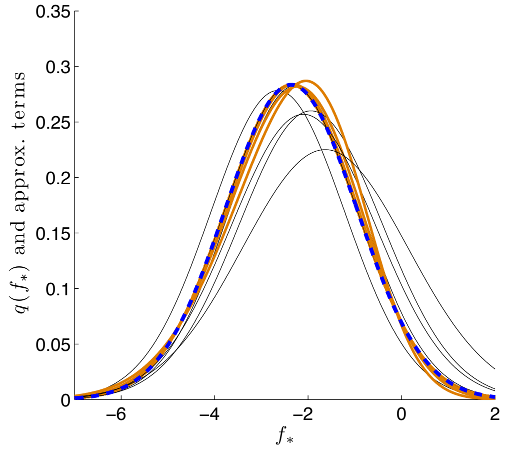
$$\begin{aligned} q(f_*) &= \mathcal{N}(f_*; \mu_*, \sigma_*^2), \\ \mu_* &= k_*^T K^{-1} \mu, \\ \sigma_*^2 &= \kappa_* - k_*^T (K + \tilde{S}^{-1})^{-1} k_* \end{aligned} \tag{21}$$

where $p(f_*|f) = \mathcal{N}(f_*; k_*^\top K^{-1} f, \kappa_* - k_*^\top K^{-1} k_*)$ was averaged over $q(f)$. To determine $q_n(f_*)$, we have to average $p(f_*|f)$ over $q_n(f)$: a lengthy derivation shows that the required integral $q_n(f_*) = \int df \, p(f_*|f) \, q_n(f)$ simplifies to

$$\begin{aligned} q_n(f_*) &= \Phi\left(\frac{y_n m_n(f_*)}{\sqrt{1 + V_n}} \right) \Big/ \Phi\left(\frac{y_n \mu_{\setminus n;n}}{\sqrt{1 + \Sigma_{\setminus n;n,n}}} \right) \times \mathcal{N}(f_*; \mu_{*\setminus n}, \sigma_{*\setminus n}^2), \\ \mu_{*\setminus n} &= k_*^T K^{-1} \mu_{\setminus n}, \\ \sigma_{*\setminus n}^2 &= \kappa_* - k_*^T (K + \tilde{S}_{\setminus n}^{-1})^{-1} k_* . \end{aligned} \tag{22}$$



(a) The mean μ_* and 2 standard deviations $2\sigma_*$ of the marginal $q(f_*)$ is shown as a gray error bar for a *novel* data point at $x_* = -2$ (dashed red line). Positive examples x_n are indicated with a $\circ$, and negative examples with a $\times$. At each x_n an error bar from $\mathcal{N}(f_*\,;\mu_{*\backslash n},\,\sigma^2_{*\backslash n})$ in (22) is shown for comparison. An exclusion of a negative example brings the density of the mean function f_* closer to zero; an exclusion of the the positive example at $x_n = 0$ increases the certainty about the class of x_*.



(b) The distributions $q(f_*)$ and each of the $q_n(f_*)$'s (which are not Gaussian) used in (13) are respectively shown in dashed blue and (thick solid) orange. These distributions are accumulated into the correction shown in Figure 1. To observe the influence of the non-linear "weight-function" of f_* in (22), the distributions $\mathcal{N}(f_*\,;\mu_{*\backslash n},\,\sigma^2_{*\backslash n})$ which are shown in Figure 13(a) are plotted as (thin) black lines.

Figure 13: An illustration of all the terms occurring in the first-order correction in (13) for an example data set.

This "tilted" predictive marginal in (22) has exactly the same form as $q(f_*)$ in (21), except[8] for its use of $\mu_{\backslash n}$ and $\tilde{S}_{\backslash n}$, and the nonlinear "weight" that is still a function of f_*, so that $q_n(f_*)$ is ultimately non-Gaussian. $\Sigma_{\backslash n;n,n}$ and $\mu_{\backslash n;n}$ denote elements (n,n) and n in $\Sigma_{\backslash n}$ and $\mu_{\backslash n}$.

In the ratio of cumulative Normals in (22) we have

$$V_n = \Sigma_{\backslash n;n,n} - \frac{\eta^2}{\sigma^2_{*\backslash n}},$$

$$m_n(f_*) = \mu_{\backslash n;n} + \frac{\eta(f_* - \mu_{*\backslash n})}{\sigma^2_{*\backslash n}},$$

where we define $\eta = k_*^T K^{-1} c_n$, with c_n being column n of $\Sigma_{\backslash n}$.

When comparing V_n and $\Sigma_{\backslash n;n,n}$ in the numerator and denominator in (22), we see that V_n is close to $\Sigma_{\backslash n;n,n}$ whenever η is small compared to $\sigma_{*\backslash n}$. Function $m_n(f_*)$ adjusts $\mu_{\backslash n;n}$ with a term *linear* in how far f_* differs from the Gaussian mean in (22), and is similarly close to $\mu_{\backslash n;n}$ when η is small compared to $\sigma^2_{*\backslash n}$.

Figure 13 provides an illustration of how the correction in (13) works. A squared exponential kernel $k(x_n, x_{n'}) = a\exp(-\frac{1}{2}\|x_n - x_{n'}\|^2/\ell^2)$ was used, with a being the (positive) amplitude, and ℓ

8. This representation is chosen for simplicity, although $\tilde{S}_{\backslash n}$ contains a zero on its diagonal.

the characteristic length-scale of the latent function. In Figure 13 the marginals $q(f_*)$ and $q_n(f_*)$ are shown, leading to the first-order correction originally shown in Figure 1.

Appendix C. Useful Distributions in the Exponential Family

In this paper we use the Dirichlet, Normal-Gamma and Normal-Wishart distributions for the MoG problem. For these distribution have to 1) compute their sufficient statistics, 2) for message passing solve the inverse problem: given the sufficient statistics we must solve for the parameters of the distribution and 3) for the predictions with the mixture models compute their predictive distribution.

C.1 Dirichlet

The Dirichlet distribution over the probability simplex π, $\sum_k \pi_k = 1$, is commonly used in two contexts: here as a prior over mixing proportions in the mixture model, and as a prior/posterior for the parameters of multinomial distribution. We denote the Dirichlet with parameters λ_k by

$$\mathcal{D}(\pi;\lambda) = \frac{1}{Z_{\mathcal{D}}(\lambda)} e^{\sum_k (\lambda_k - 1)\log \pi_k} \delta\left(\sum_k \pi_k - 1\right) , \tag{23}$$

$$Z_{\mathcal{D}}(\lambda) = \frac{\prod_k \Gamma(\lambda_k)}{\Gamma(\sum_k \lambda_k)} . \tag{24}$$

C.1.1 SUFFICIENT STATISTICS

The sufficient statistic of the Dirichlet is

$$\langle \log \pi_k \rangle = \frac{\partial \log Z_{\mathcal{D}}(\lambda)}{\partial \lambda_k} = \psi(\lambda_k) - \psi\left(\sum_k \lambda_k\right) , \tag{25}$$

where ψ is the digamma-function $d \log \Gamma(x)/dx$.

C.1.2 INVERSE

In line 5 of the Algorithms 1 and 2 we have to solve the inverse problem: given the statistics $m_k = \langle \log \pi_k \rangle$ find the parameters λ. This can be done effectively by first solving for

$$\alpha \equiv \psi\left(\sum_k \lambda_k\right) = \psi\left(\sum_k \psi^{-1}(m_k + \alpha)\right)$$

by Newton's method, and then setting $\lambda_k := \psi^{-1}(m_k + \alpha)$.

C.1.3 PREDICTIVE DISTRIBUTION

The Dirichlet can also be used as a prior for the parameters of the multinomial distribution. This distribution is used multi-category either counts or sequence data. For counts, $x = (x_1, \ldots, x_d)$ is a vector of counts for each of the possible d outcomes. For sequence data x is an indicator variable being one for the outcome and zero in all other entries. The multinomial distribution is:

$$p(x|\pi) = \frac{(\sum_k x_k)!}{x_1! \ldots x_d!} \prod_{k=1}^{d} \pi_k^{x_k} ,$$

where the combinatorial prefactor goes away in the sequence case. The predictive distribution for multinomial data and Dirichlet distributed posterior is straightforward to calculate using the result for the normalizer of the Dirichlet:

$$p(x|\lambda) = \int d\pi p(x|\pi) \mathcal{D}(\pi;\lambda) = \frac{(\sum_l x_l)!}{x_1! \dots x_d!} \frac{Z_{\mathcal{D}}(x+\lambda)}{Z_{\mathcal{D}}(\lambda)} \ .$$

C.2 Normal-Gamma

The Normal-Gamma (or Gauss-Gamma) model is use for a joint distribution of one dimensional mean and precision (inverse variance) variables:

$$\mathcal{NG}(\mu,\gamma;m,\nu,a,b) = \frac{1}{Z_{\mathcal{NG}}(m,\nu,a,b)} \exp\left[\begin{pmatrix} m\nu \\ -\frac{1}{2}\nu \\ -b-\frac{1}{2}\nu m^2 \\ a-\frac{1}{2} \end{pmatrix}^T \begin{pmatrix} \mu\gamma \\ \mu^2\gamma \\ \gamma \\ \log\gamma \end{pmatrix} \right] ,$$

$$Z_{\mathcal{NG}}(m,\nu,a,b) = \sqrt{\frac{2\pi}{\nu}} \frac{\Gamma(a)}{b^a} \ ,$$

where this distribution is obtained by multiplying the Normal and Gamma distributions:

$$\mathcal{N}(\mu;m,(\nu\gamma)^{-1}) = \sqrt{\frac{\nu\gamma}{2\pi}} \exp\left(-\frac{1}{2}(\mu-m)^2\nu\gamma\right) ,$$

$$\mathcal{G}(\gamma;a,b) = \frac{b^a}{\Gamma(a)} \exp((a-1)\log\gamma - b\gamma) \ ,$$

where a and b must be positive.

C.2.1 SUFFICIENT STATISTICS

The sufficient statistics are obtained by using $\log Z_{\mathcal{NG}}(m,\nu,a,b)$ as a generating function for the sufficient statistic. By taking derivatives of $\log Z_{\mathcal{NG}}$ with respect to the parameters $\{m,\nu,a,b\}$,

$$\begin{aligned} \nu\langle\mu\gamma\rangle - \nu m\langle\gamma\rangle &= 0, \\ m\langle\mu\gamma\rangle - \frac{1}{2}\langle\mu^2\gamma\rangle - m^2\langle\gamma\rangle &= -\frac{1}{2\nu}, \\ \langle\log\gamma\rangle &= \psi(a) - \log b, \\ -\langle\gamma\rangle &= -a/b \ , \end{aligned}$$

we can solve for $\langle\mu\gamma\rangle$, $\langle\mu^2\gamma\rangle$, $\langle\gamma\rangle$ and $\langle\log\gamma\rangle$.

C.2.2 INVERSE

We can use these expressions to solve for the parameters in terms of the sufficient statistics in the same fashion as above. We get closed form expressions for three of parameters

$$m = \frac{\langle\mu\gamma\rangle}{\langle\gamma\rangle} \ , \quad \nu = \frac{1}{\langle\mu^2\gamma\rangle - m^2\langle\gamma\rangle} \ , \quad b = \frac{a}{\langle\gamma\rangle} \ ,$$

and a should be found from

$$\psi(a) - \log a = \langle \log \gamma \rangle - \log \langle \gamma \rangle$$

by for example Newton's method.

C.2.3 PREDICTIVE DISTRIBUTION

The predictive distribution can be calculated straightforwardly from the normalizer and is a univariate Student-t distribution:

$$\begin{aligned} p(x|m,\nu,a,b) &= \frac{1}{\sqrt{2\pi}} \frac{Z_{\mathcal{NG}}\left(\frac{x+\nu m}{\nu+1}, \nu+1, a+\frac{1}{2}, b+\frac{\nu}{\nu+1}(x-m)^2\right)}{Z_{\mathcal{NG}}(m,\nu,a,b)} \\ &= \mathcal{T}\left(x;\ m,\ \frac{b}{a}\frac{\nu+1}{\nu},\ 2a\right) , \end{aligned}$$

where $\mathcal{T}\left(x;\mu,\sigma^2,d_{\mathrm{f}}\right)$ is a Student-t distribution with mean μ, variance σ^2 and d_{f} degrees of freedom:

$$\begin{aligned} \mathcal{T}\left(x;\mu,\sigma^2,d_{\mathrm{f}}\right) &= \frac{1}{Z_{\mathcal{T}}(\mu,\sigma^2,d_{\mathrm{f}})} \exp\left[-\frac{d_{\mathrm{f}}+1}{2}\log\left(1+\frac{1}{d_{\mathrm{f}}}\left(\frac{x-\mu}{\sigma}\right)^2\right)\right], \\ Z_{\mathcal{T}}(\mu,\sigma^2,d_{\mathrm{f}}) &= \sqrt{2\pi\sigma^2}\sqrt{\frac{d_{\mathrm{f}}}{2}} \frac{\Gamma\left(\frac{d_{\mathrm{f}}}{2}\right)}{\Gamma\left(\frac{d_{\mathrm{f}}+1}{2}\right)} . \end{aligned}$$

C.3 Normal-Wishart

The Normal-Wishart is the multidimensional generalization of the Normal-Gamma. We will write the Wishart distribution over positive definite symmetric matrices in the same form as the Gamma distribution:

$$\mathcal{W}(\Gamma;a,B) \propto \exp\left(\left(a-\frac{d+1}{2}\right)\log\det\Gamma - \mathrm{tr}B\Gamma\right) ,$$

where the degrees of freedom $2a$ should be greater than $d-1$ for the distribution to be normalizable. The Normal-Wishart is given by

$$\begin{aligned} \mathcal{NW}(\mu,\Gamma;m,\nu,a,B) &= \frac{1}{Z_{\mathcal{NW}}(m,\nu,a,B)} \\ &\quad \exp\left[\begin{pmatrix} \nu m \\ -\frac{1}{2}\nu \\ a-\frac{d}{2} \end{pmatrix}^T \begin{pmatrix} \Gamma\mu \\ \mu^T\Gamma\mu \\ \log\det\Gamma \end{pmatrix} - \mathrm{tr}(B+\frac{1}{2}\nu m m^T)\Gamma\right] , \\ Z_{\mathcal{NW}}(m,\nu,a,B) &= \pi^{d(d-1)/4}\left(\frac{2\pi}{\nu}\right)^{d/2}\prod_{l=1}^{d}\Gamma\left(a+\frac{1-l}{2}\right)e^{-a\log\det B} . \end{aligned}$$

C.3.1 SUFFICIENT STATISTICS

The sufficient statistics follow from a straightforward generalization of the results from the Normal-Gamma model:

$$\langle \Gamma \rangle = aB^{-1},$$

$$\begin{aligned}
\langle \Gamma\mu \rangle &= \langle \Gamma \rangle m, \\
\langle \mu^T \Gamma \mu \rangle &= \frac{d}{\nu} + m^T \langle \Gamma \rangle m, \\
\langle \log\det\Gamma \rangle &= \sum_{l=1}^{d} \psi\left(a + \frac{1-l}{2}\right) - \log\det B \ .
\end{aligned}$$

C.3.2 INVERSE

From the sufficient statistics we can get closed form expressions for the parameters

$$m = \langle \Gamma \rangle^{-1} \langle \Gamma\mu \rangle \ , \quad \nu = \frac{d}{\langle \mu^T \Gamma \mu \rangle - m^T \langle \Gamma \rangle m} \ , \quad B = a \langle \Gamma \rangle^{-1} \ ,$$

whereas a should be found from

$$\sum_{l=1}^{d} \psi\left(a + \frac{1-l}{2}\right) - d \log a = \langle \log\det\Gamma \rangle - \log\det\langle \Gamma \rangle \ .$$

C.3.3 PREDICTIVE DISTRIBUTION

A generalization of the result for the predictive distribution in one dimension to the multivariate case gives:

$$p(x|m,\nu,a,b) = \mathcal{T}\left(x;\ m,\ \frac{2B}{2a-d+1}\frac{\nu+1}{\nu},\ 2a-d+1\right) \ ,$$

where $\mathcal{T}(x;\mu,\Sigma,d_{\mathrm{f}})$ is the d-dimensional multivariate Student-t distribution with mean μ, covariance Σ and d_{f} degrees of freedom:

$$\begin{aligned}
\mathcal{T}(x;\mu,\Sigma,d_{\mathrm{f}}) &= \frac{1}{Z_{\mathcal{T}}(\mu,\Sigma,d_{\mathrm{f}})} \exp\left[-\frac{d_{\mathrm{f}}+d}{2} \log\left(1 + \frac{1}{d_{\mathrm{f}}}(x-\mu)^T \Sigma^{-1} (x-\mu)\right)\right], \\
Z_{\mathcal{T}}(\mu,\sigma^2,d_{\mathrm{f}}) &= \sqrt{\det(2\pi\Sigma)} \left(\frac{d_{\mathrm{f}}}{2}\right)^{d/2} \frac{\Gamma\left(\frac{d_{\mathrm{f}}}{2}\right)}{\Gamma\left(\frac{d_{\mathrm{f}}+d}{2}\right)} \ .
\end{aligned}$$

Appendix D. Inference for a Mixture of Gaussians

When we have unconstrained d-dimensional data we can model this with a Normal (or Gaussian) distribution,

$$p(x|\mu,\Gamma) = \mathcal{N}(x;\mu,\Gamma^{-1}) = \sqrt{\frac{\det\Gamma}{(2\pi)^d}} \exp\left(-\frac{1}{2}(x-\mu)^T \Gamma (x-\mu)\right) \ ,$$

where μ and Γ are respectively the mean vector and precision (or inverse covariance) matrix. The conjugate prior for the mean and precision is the Normal-Wishart distribution, which we describe in appendix C. In the following we choose Gaussians $p(x|\mu_k,\Gamma_k)$ as components densities $p(x|k)$ in the mixture.

D.1 Variational and Predictive Distributions

The q distribution follows the prior and is conveniently chosen to factorise over mixture components:

$$q(\theta) = q(\pi)\prod_k q(\mu_k, \Gamma_k)$$

with $q(\mu_k, \Gamma_k | m_k, \nu_k, a_k, B_k)$ being a Normal-Wishart distribution; see Appendix C. We can use the same machinery as Section 4 to derive the EC approximation to the predictive distribution and the statistics needed for message passing.

The predictive distribution $p(x|\mathcal{D})$ is again approximated by the form given in (14), with $p(x|k)$ replaced by

$$p(x|k) \equiv p(x|m_k, \nu_k, a_k, B_k) = \mathcal{T}\left(x;\ m_k,\ \frac{2B_k}{2a_k - d + 1}\frac{\nu_k + 1}{\nu_k},\ 2a_k - d + 1\right) . \tag{26}$$

Here $\mathcal{T}$ is a Student-t distribution, which we describe in greater detail in Appendix C.

Likewise, the within data set predictive distribution $p(x_n|\mathcal{D}_{\backslash n})$ follows (15); only now $p(x_n|k)$ is replaced with $p(x_n|k\backslash n)$, which takes the same form as (26) above, with Λ replaced by $\Lambda_{\backslash n}$. The Dirichlet cavity parameters are again $\lambda_{k\backslash n} = \lambda_{k,0} + \sum_{n' \neq n} \lambda_{k,n'}$. The other cavity parameters are similarly defined in terms of the appropriate parameter vector components, for instance $\nu_{k\backslash n} m_{k\backslash n} = \nu_{k,0} m_{k,0} + \sum_{n' \neq n} \nu_{k,n'} m_{k,n'}$.

We can again use the cavity parameters to write q_n in terms of $q_{\backslash n}$:

$$q_n(\theta) = \frac{Z(\Lambda_{\backslash n}, 0)}{Z_n(\Lambda_{\backslash n}, 1_n)} q_{\backslash n}(\theta) \sum_k \pi_k p(x_n | \mu_k, \Gamma_k) .$$

The normalizer is given by

$$Z_n(\Lambda_{\backslash n}, 1_n) = Z(\Lambda_{\backslash n}, 0) \sum_k \langle \pi_k \rangle_{q_{\backslash n}} p(x_n | k\backslash n) , \tag{27}$$

where the explicit form of $Z(\Lambda_{\backslash n}, 0)$ is

$$Z(\Lambda_{\backslash n}, 0) = Z_{\mathcal{D}}(\lambda_{\backslash n}) \prod_k Z_{\mathcal{NW}}(m_{k\backslash n}, \nu_{k\backslash n}, a_{k\backslash n}, B_{k\backslash n}) .$$

D.2 Expectations

The statistics of $q_n(\theta)$ for the mixture of Gaussians are computed by using $\log Z_n(\Lambda_{\backslash n})$ from (27) as a generating function. To simplify the derivative with respect to the predictive distribution $p(x_n|k\backslash n)$, we introduce another component-specific Normal-Wishart distribution

$$q_{k,n}(\mu_k, \Gamma_k) \propto p(x_n|\mu_k, \Gamma_k) q_{\backslash n}(\mu_k, \Gamma_k) ,$$

and write the predictive distribution as a ratio between the normalizers of $q_{k,n}$ and $q_{\backslash n}$:

$$p(x_n|k\backslash n) = \frac{(2\pi)^{-d/2} Z_{\mathcal{NW}}\left(\frac{\nu_{k\backslash n} m_{k\backslash n} + x_n}{\nu_{k\backslash n} + 1}, \nu_{k\backslash n} + 1, a_{k\backslash n} + \frac{1}{2}, B_{k\backslash n} + \frac{1}{2}\frac{\nu_{k\backslash n}}{\nu_{k\backslash n} + 1}(x_n - m_{k\backslash n})(x_n - m_{k\backslash n})^T\right)}{Z_{\mathcal{NW}}(m_{k\backslash n}, \nu_{k\backslash n}, a_{k\backslash n}, B_{k\backslash n})} .$$

For example, for $\langle \Gamma_k \mu_k \rangle$ we get

$$\left\langle \Gamma_k \mu_k \right\rangle_{q_n} = \frac{1}{\nu_{k\backslash n}} \frac{d \log Z_n(\Lambda_{\backslash n})}{d m_{k\backslash n}} = (1 - r_{nk}) \left\langle \Gamma_k \mu_k \right\rangle_{q_{\backslash n}} + r_{nk} \left\langle \Gamma_k \mu_k \right\rangle_{q_{k,n}} , \tag{28}$$

where the "responsibility" (the probability of example n being generated by the kth mixture component) is defined as

$$r_{nk} = \frac{\lambda_{k\backslash n} p(x_n | k\backslash n)}{\sum_{k'} \lambda_{k'\backslash n} p(x_n | k'\backslash n)} . \tag{29}$$

The expectation in (28) is expressed as a weighed sum of a "prior" expectation over the cavity distribution $q_{\backslash n}(\mu_k, \Gamma_k)$, and a "posterior" expectation over $q_{k,n}(\mu_k, \Gamma_k)$. The other moments $\langle \Gamma_k \rangle$, $\langle \mu_k^T \Gamma_k \mu_k \rangle$ and $\langle \log \det \Gamma_k \rangle$ can be expressed as weighed sums similar to (28):

$$\begin{aligned}
\left\langle \Gamma_k \right\rangle_{q_n} &= (1 - r_{nk}) \left\langle \Gamma_k \right\rangle_{q_{\backslash n}} + r_{nk} \left\langle \Gamma_k \right\rangle_{q_{k,n}}, \\
\left\langle \mu_k^T \Gamma_k \mu_k \right\rangle_{q_n} &= (1 - r_{nk}) \left\langle \mu_k^T \Gamma_k \mu_k \right\rangle_{q_{\backslash n}} + r_{nk} \left\langle \mu_k^T \Gamma_k \mu_k \right\rangle_{q_{k,n}}, \\
\left\langle \log \det \Gamma_k \right\rangle_{q_n} &= (1 - r_{nk}) \left\langle \log \det \Gamma_k \right\rangle_{q_{\backslash n}} + r_{nk} \left\langle \log \det \Gamma_k \right\rangle_{q_{k,n}} .
\end{aligned}$$

The explicit expressions for the $q_{k,n}$ are given below, whereas those for $q_{\backslash n}$ can be obtained from Appendix C:

$$\begin{aligned}
\left\langle \Gamma_k \right\rangle_{q_{k,n}} &= \left(a_{k\backslash n} + \frac{1}{2} \right) \left[B_{k\backslash n} + \frac{1}{2} \frac{\nu_{k\backslash n}}{\nu_{k\backslash n} + 1} (x_n - m_{k\backslash n})(x_n - m_{k\backslash n})^T \right]^{-1} , \\
\left\langle \Gamma_k \mu_k \right\rangle_{q_{k,n}} &= \left\langle \Gamma_k \right\rangle_{q_{k,n}} \frac{\nu_{k\backslash n} m_{k\backslash n} + x_n}{\nu_{k\backslash n} + 1}, \\
\left\langle \mu_k^T \Gamma_k \mu_k \right\rangle_{q_{k,n}} &= \frac{d}{\nu_{k\backslash n} + 1} + \left(\frac{\nu_{k\backslash n} m_{k\backslash n} + x_n}{\nu_{k\backslash n} + 1} \right)^T \left\langle \Gamma_k \right\rangle_{q_{k,n}} \left(\frac{\nu_{k\backslash n} m_{k\backslash n} + x_n}{\nu_{k\backslash n} + 1} \right), \\
\left\langle \log \det \Gamma_k \right\rangle_{q_{k,n}} &= \sum_{i=1}^{d} \psi \left(a_{k\backslash n} + \frac{1}{2} + \frac{1 - i}{2} \right) \\
&\qquad - \log \det \left(B_{k\backslash n} + \frac{1}{2} \frac{\nu_{k\backslash n}}{\nu_{k\backslash n} + 1} (x_n - m_{k\backslash n})(x_n - m_{k\backslash n})^T \right) .
\end{aligned}$$

We have already seen how to solve for $\langle \log \pi_k \rangle_{q_n}$, the only difference being r_{nk} in (16), which we now take from (29).

Appendix E. Gibbs Sampling for Parallel Tempering

Parallel tempering of a mixture of Gaussian distributions $p(x_n|\theta) = \sum_{k=1}^{K} \pi_k \mathcal{N}(x_n; \mu_k, \Gamma_k^{-1})$ requires a Monte Carlo simulation at inverse temperature β. We can either sample from $p(\theta|\mathcal{D}, \beta)$ using a Metropolis-Hastings (MH) method, or augment the parameter space with latent allocation variables z so that we can sample from $p(\theta, z|\mathcal{D}, \beta)$ with Gibbs sampling. We may also define $p(z|\mathcal{D}, \beta) = \int d\theta \, p(\theta, z|\mathcal{D}, \beta)$, and devise a MH scheme on this distribution by making random assignment changes to z.

The road of Gibbs sampling is pathed with tractable conditional distributions of $p(\theta, z|\mathcal{D}, \beta)$, and it is the one we choose. We extend the parameter space to include a binary latent allocation vector z_n

for each data point n to indicate which mixture component was responsible for generating it (Diebolt and Robert, 1994); consequently $z_{nk} \in \{0,1\}$, and $\sum_k z_{nk} = 1$. The complete joint distribution is therefore

$$p(\mathcal{D}, z|\theta)p(\theta) = \prod_n \prod_k \left[\pi_k \mathcal{N}(x_n; \mu_k, \Gamma_k^{-1})\right]^{z_{nk}} p(\theta) \ .$$

We can write the complete data likelihood as $p(\mathcal{D}, z|\theta) = p(\mathcal{D}|z,\theta)p(z|\theta)$, and in this form the likelihood, to the power β, multiplied by the prior over z and θ, is

$$p(\mathcal{D}|z,\theta)^\beta \times p(z,\theta) = \prod_n \prod_k \mathcal{N}(x_n; \mu_j, \Gamma_k^{-1})^{\beta z_{nk}} \times \prod_n \prod_k \pi_k^{z_{nk}} p(\theta) \ .$$

(Note that the introduction of z moves π to the prior.) With inverse temperature parameter β the tempered posterior distribution is $p(\theta, z|\mathcal{D}, \beta) \propto p(\mathcal{D}|z,\theta)^\beta p(z,\theta)$, and can be treated as any missing-value Gibbs sampling problem. The allocation variables are sampled with

$$z_{nk}|\pi, \mu, \Gamma \sim \frac{\pi_k \mathcal{N}(x_n; \mu_k, \Gamma_k^{-1})^\beta}{\sum_{k'} \pi_{k'} \mathcal{N}(x_n; \mu_{k'}, \Gamma_{k'}^{-1})^\beta} \ .$$

Given the allocation variables, we define

$$r_{nk} = \beta z_{nk}, \qquad \bar{x}_k = \frac{1}{N_k} \sum_n r_{nk} x_n,$$
$$N_k = \sum_n r_{nk} \qquad \Sigma_k, = \frac{1}{N_k} \sum_n r_{nk} (x_n - \bar{x}_k)(x_n - \bar{x}_k)^T \ ,$$

to give the conditional distributions needed for sampling the mixture parameters as

$$\begin{aligned} \pi|z &\sim \mathcal{D}\big(\pi; \lambda_{1,0} + \tfrac{1}{\beta}N_1, \ldots, \lambda_{K,0} + \tfrac{1}{\beta}N_K\big), \\ \mu_k, \Gamma_k|z &\sim \mathcal{NW}\big(\mu_k, \Gamma_k; m, \nu, a, B\big) \ , \end{aligned} \tag{30}$$

with

$$\begin{aligned} m &= \frac{\nu_{k,0} m_{k,0} + N_k \bar{x}_k}{\nu_{k,0} + N_k}, \\ \nu &= \nu_{k,0} + N_k, \\ a &= a_{k,0} + N_k/2, \\ B &= B_{k,0} + \frac{1}{2} N_k \Sigma_k + \frac{1}{2} \frac{N_k \nu_{k,0} (\bar{x}_k - m_{k,0})(\bar{x}_k - m_{k,0})^T}{\nu_{k,0} + N_k} \ . \end{aligned} \tag{31}$$

As $p(\mathcal{D}) = \int d\theta dz\, p(\mathcal{D}, \theta, z)$, we use the samples over θ and z to estimate the average log likelihood. If $\{\{\pi_{k,i}^{(t)}, \mu_{k,i}^{(t)}, \Gamma_{k,i}^{(t)}\}_{k=1}^K, \{z_{n,i}^{(t)}\}_{n=1}^N\}_{t=1}^T$ indicates the samples of chain i (after a burn-in period), then

$$\langle \log p(\mathcal{D}|\theta, z) \rangle_{\beta_i} \approx \frac{1}{T} \sum_t \sum_n \sum_k z_{nk,i}^{(t)} \log \mathcal{N}\big(x_n; \mu_{k,i}^{(t)}, \Gamma_{k,i}^{(t)\,-1}\big) \ . \tag{32}$$

Notice that the samples of the mixing weights $\pi_{k,i}^{(t)}$ are not used in estimating the log likelihood average over the posterior, but occur in the prior.

E.1 A Practical Generalization (I)

The generalization of PT and TI from Section 5.3 can be made by writing the tempered posterior distribution as

$$p(\theta, z|\mathcal{D}, \beta) \propto \prod_n \prod_k \mathcal{N}(x_n; \mu_k, \Gamma_k^{-1})^{\beta z_{nk}} \pi_k^{z_{nk}} p(\theta)^\beta q(\theta)^{1-\beta} ,$$

where $\prod_{n,k} \pi_k^{z_{nk}} = p(z|\theta)^\beta q(z|\theta)^{1-\beta}$ follows from $p(z|\theta) = q(z|\theta)$. To do Gibbs sampling as before, we have to determine the parameters of the "effective" prior in the above scenario. Here we let $q(\theta)$ be in the same family—for example a narrower version—of the prior. If superscripts p and q now differentiate between the parameters of $p(\theta)$ and $q(\theta)$, we use

$$\begin{aligned}
\lambda_0 &= \beta\lambda_0^p + (1-\beta)\lambda_0^q, \\
m_{k,0} &= \frac{\beta\nu_{k,0}^p m_{k,0}^p + (1-\beta)\nu_{k,0}^q m_{k,0}^q}{\beta\nu_{k,0}^p + (1-\beta)\nu_{k,0}^q}, \\
\nu_{k,0} &= \beta\nu_{k,0}^p + (1-\beta)\nu_{k,0}^q, \\
a_{k,0} &= \beta a_{k,0}^p + (1-\beta)a_{k,0}^q, \\
B_{k,0} &= \beta B_{k,0}^p + (1-\beta)B_{k,0}^q + \frac{1}{2}\frac{\beta\nu_{k,0}^p(1-\beta)\nu_{k,0}^q}{\beta\nu_{k,0}^p + (1-\beta)\nu_{k,0}^q}(m_{k,0}^p - m_{k,0}^q)(m_{k,0}^p - m_{k,0}^q)^T
\end{aligned}$$

as substitute for the usual prior in (30) and (31). The empirical expectation given in (32) should be generalized—simplifying the left hand side below with $p(z,\theta)/q(z,\theta) = p(\theta)/q(\theta)$—to

$$\left\langle \log p(\mathcal{D}|\theta, z) + \log\frac{p(\theta)}{q(\theta)} \right\rangle_{\beta_i} \approx \frac{1}{T}\sum_t \Bigg[\log p(\pi_i^{(t)}) - \log q(\pi_i^{(t)}) + \sum_k \Big[\log p(\mu_{k,i}^{(t)}, \Gamma_{k,i}^{(t)}) \\ - \log q(\mu_{k,i}^{(t)}, \Gamma_{k,i}^{(t)}) + \sum_n z_{nk,i}^{(t)} \log\mathcal{N}\big(x_n; \mu_{k,i}^{(t)}, {\Gamma_{k,i}^{(t)}}^{-1}\big)\Big]\Bigg] .$$

E.2 A Practical Generalization (II)

We implemented a further possible generalization, which arises from choosing $q(\theta) = p(\theta|\mathcal{D}')$, where $\mathcal{D}'$ contains a small subset of data points from $\mathcal{D}$. This sensibly restricts q to parameter space closer to the posterior, with the benefit that the normalizer of q needs to be calculated only once. With $|\mathcal{D}'|$ being small, q can be evaluated analytically without feeling the effect of the exponentially expanding number of terms. This brings an interesting tradeoff, as setting $\mathcal{D}' \leftarrow \mathcal{D}$ solves our original (difficult) problem. (Figure 3 used this choice of surrogate prior, with $\mathcal{D}'$ containing 3 out of a possible 82 data points.)

We can construct $q(\theta)$ as follows: Let $N' = |\mathcal{D}'|$ be the number of data points in $\mathcal{D}'$, so that $q(\theta)$ expands as a sum over $(N')^K$ terms. Allow $1, \ldots, K$ to be the digit set of a number system in base K. Make a list $\mathcal{S}$ of the first $(N')^K$ numbers in base K, such that each number s consists of N' digits, and each $x_{n'} \in \mathcal{D}'$ can be associated with a corresponding digit *position*. Each $s \in \mathcal{S}$ therefore defines a unique allocation for all $x_{n'} \in \mathcal{D}'$ to clusters $1, \ldots, K$ ($x_{n'}$'s digit *value*). We shall use the shorthand $\mathcal{D}'_s$ to indicate a data set with a data point to cluster allocation given by s.

The surrogate prior is a weighted sum of various posteriors

$$q(\theta) = p(\theta|\mathcal{D}') = \sum_{s\in\mathcal{S}} w_s p(\pi|\mathcal{D}'_s) \prod_k p(\mu_k, \Gamma_k|\mathcal{D}'_s) .$$

If $Z_s = \int d\theta\, p(\theta, \mathcal{D}'_s)$, the weights are determined with $w_s = Z_s / \sum_{s'} Z_{s'}$.

Instead of merely raising q to the power $1 - \beta$, we first turn q into a product amenable to Gibbs sampling by augmenting it with binary indicator variables $y = \{y_s\}_{s \in \mathcal{S}}$ that pick a particular component of q; $\sum_s y_s = 1$. With $q(y) = p(y) = \prod_s w_s^{y_s}$, the surrogate prior now takes the form $q(z|\theta)q(\theta|y)q(y)$; the prior becomes $p(z|\theta)p(\theta)p(y)$. The empirical expectation in (32) generalizes to

$$\left\langle \log p(\mathcal{D}|\theta, z) + \log \frac{p(\theta)}{q(\theta|y)} \right\rangle_{\beta_i} \approx \frac{1}{T} \sum_t \Bigg[\log p(\pi_i^{(t)}) - \sum_s y_{s,i}^{(t)} \log p(\pi_i^{(t)}|\mathcal{D}'_s) + \sum_k \Big[\log p(\mu_{k,i}^{(t)}, \Gamma_{k,i}^{(t)}) - \sum_s y_{s,i}^{(t)} \log p(\mu_{k,i}^{(t)}, \Gamma_{k,i}^{(t)}|\mathcal{D}'_s) + \sum_n z_{nk,i}^{(t)} \log \mathcal{N}\big(x_n; \mu_{k,i}^{(t)}, {\Gamma_{k,i}^{(t)}}^{-1}\big) \Big] \Bigg] .$$

Appendix F. Perturbation Corrections for Mixture of Gaussians

In this appendix we show how to compute the second-order terms of (8),

$$R = 1 + \sum_{n_1 < n_2} \Big\langle \varepsilon_{n_1}(\theta)\varepsilon_{n_2}(\theta) \Big\rangle_q + \sum_{n_1 < n_2 < n_3} \Big\langle \varepsilon_{n_1}(\theta)\varepsilon_{n_2}(\theta)\varepsilon_{n_3}(\theta) \Big\rangle_q + \dots , \tag{33}$$

and the first-order term in the numerator of (10),

$$p(x|\mathcal{D}) = \frac{\int d\theta\, q(\theta)\, p(x|\theta)\, \left(1 + \sum_n \varepsilon_n(\theta) + \sum_{n_1<n_2} \varepsilon_{n_1}(\theta)\varepsilon_{n_2}(\theta) + \dots\right)}{1 + \sum_{n_1<n_2} \langle \varepsilon_{n_1}(\theta)\varepsilon_{n_2}(\theta) \rangle_q + \dots} . \tag{34}$$

The sum in the second order term runs over all distinct pairs and the complexity thus grows as $O(N^2)$. However, one would expect that the largest contribution comes from nearby points, or more precisely points that belong to the same component, as indicated by a large responsibility for the same component. Although not done here, it is plausible to restrict the summation to only include these pairs without sacrificing much precision.

Let $\Lambda = \{\lambda, \{m_k, v_k, a_k, B_k\}_{k=1}^K\}$ be the parameters that solve the EC equations. We also have access to the parameters of each of the cavity distributions $\Lambda_{\setminus n}$.

For each n the parameters of $q_n(\theta)$ is given by the parameters of $p(x_n|\theta)q_{\setminus n}(\theta)$, which expands as a sum over the K mixture components. Each element k in the sum contains a product of a Dirichlet density and K Normal-Wishart densities. The Dirichlet parameter vector will have element k incremented by one, and as x_n is associated with component k, it will affect only the parameters of the kth Normal-Wishart. Therefore, apart from the cavity parameters $\Lambda_{\setminus n}$, we will also need for each $k = 1, \dots, K$:

$$\begin{aligned}
\lambda^*_{k\setminus n} &= \lambda_{k\setminus n} + 1 \quad \text{and} \quad \lambda^*_{k'\setminus n} = \lambda_{k'\setminus n} \text{ for } k' \neq k, \\
v^*_{k\setminus n} &= v_{k\setminus n} + 1, \\
m^*_{k\setminus n} &= \frac{v_{k\setminus n} m_{k\setminus n} + x_n}{v_{k\setminus n} + 1}, \\
a^*_{k\setminus n} &= a_{k\setminus n} + \tfrac{1}{2},
\end{aligned}$$

$$B^*_{k\setminus n} = B_{k\setminus n} + \frac{1}{2}\frac{v_{k\setminus n}}{v_{k\setminus n}+1}(m_{k\setminus n} - x_n)(m_{k\setminus n} - x_n)^T \; . \tag{35}$$

The normalizer of $q_n(\theta)$ follows from (27) to be $\int d\theta' p(x_n|\theta')q_{\setminus n}(\theta') = \sum_k \langle \pi_k \rangle_{q_{\setminus n}} p(x_n|k\setminus n) = Z_n$.

F.1 Corrections for the Marginal Likelihood

A single second-order term in (33) can be evaluated with

$$\begin{aligned}
\Big\langle \varepsilon_{n_1}(\theta)\varepsilon_{n_2}(\theta)\Big\rangle_q &= \int d\theta \, \frac{q_{n_1}(\theta)q_{n_2}(\theta)}{q(\theta)} - 1 \\
&= -1 + \frac{1}{Z_{n_1}}\frac{1}{Z_{n_2}}(2\pi)^{-d}\sum_{k=1}^{K}\sum_{l=1}^{K}\left\{ \frac{Z_{\mathcal{D}}(\lambda)\,Z_{\mathcal{D}}(\lambda^{(k,l)})}{Z_{\mathcal{D}}(\lambda_{\setminus n_1})\,Z_{\mathcal{D}}(\lambda_{\setminus n_2})} \right. \\
&\quad \cdots \times \left. \prod_{j=1}^{K} \frac{Z_{\mathcal{NW}}(m_j, v_j, a_j, B_j)\,Z_{\mathcal{NW}}(m_j^{(k,l)}, v_j^{(k,l)}, a_j^{(k,l)}, B_j^{(k,l)})}{Z_{\mathcal{NW}}(m_{j\setminus n_1}, v_{j\setminus n_1}, a_{j\setminus n_1}, B_{j\setminus n_1})\,Z_{\mathcal{NW}}(m_{j\setminus n_2}, v_{j\setminus n_2}, a_{j\setminus n_2}, B_{j\setminus n_2})} \right\} .
\end{aligned}$$

The above sum over k relates x_{n_1} to coming from a particular mixture component k, while the sum over l does the same for x_{n_2}. For a particular element in the sum over k and l we need parameters relating to each of the $j = 1, \ldots, K$ mixture components. For the Dirichlet normalizer the parameters $\lambda_j^{(k,l)}$ depend on whether $k = l$, implying that both x_{n_1} and x_{n_2} were generated from the same mixture component, or whether $k \neq l$, implying that x_{n_1} and x_{n_2} came from different mixture components. The elements of $\lambda^{(k,l)}$ are:

$$\lambda_j^{(k,l)} = \lambda_{j\setminus n_1} + \lambda_{j\setminus n_2} - \lambda_j \qquad \text{for } j \neq k, l \; .$$

When $k \neq l$ two indices j remain to be defined; if $k = l$ we will have one remaining index to take care of:

$$\begin{aligned}
\lambda_j^{(k,l)} &= \lambda^*_{j\setminus n_1} + \lambda_{j\setminus n_2} - \lambda_j && \text{for } j = k \text{ and } k \neq l, \\
\lambda_j^{(k,l)} &= \lambda_{j\setminus n_1} + \lambda^*_{j\setminus n_2} - \lambda_j && \text{for } j = l \text{ and } k \neq l, \\
\lambda_j^{(k,l)} &= \lambda^*_{j\setminus n_1} + \lambda^*_{j\setminus n_2} - \lambda_j && \text{for } j = k = l \; .
\end{aligned}$$

For each element in the sum over k and l the Normal-Wishart parameters are similarly defined. When $j \neq k, l$ we have:

$$\begin{aligned}
v_j^{(k,l)} &= v_{j\setminus n_1} + v_{j\setminus n_2} - v_j, \\
m_j^{(k,l)} &= \frac{v_{j\setminus n_1} m_{j\setminus n_1} + v_{j\setminus n_2} m_{j\setminus n_2} - v_j m_j}{v_{j\setminus n_1} + v_{j\setminus n_2} - v_j}, \\
a_j^{(k,l)} &= a_{j\setminus n_1} + a_{j\setminus n_2} - a_j, \\
B_j^{(k,l)} &= B_{j\setminus n_1} + \frac{1}{2} v_{j\setminus n_1} m_{j\setminus n_1} m^T_{j\setminus n_1} + B_{j\setminus n_2} + \frac{1}{2} v_{j\setminus n_2} m_{j\setminus n_2} m^T_{j\setminus n_2} \\
&\quad \cdots - B_j - \frac{1}{2} v_j m_j m_j^T - \frac{1}{2} v_j^{(k,l)} m_j^{(k,l)} m_j^{(k,l)T} \; .
\end{aligned}$$

As was seen for the mixture weights, we will need further definitions: when $j = k$ and $k \neq l$ we shall use $v_j^{(k,l)} = v^*_{j\backslash n_1} + v_{j\backslash n_2} - v_j$; a similar definition follows when $j = l$. Finally, when $j = k = l$ we find that $v_j^{(k,l)} = v^*_{j\backslash n_1} + v^*_{j\backslash n_2} - v_j$. The other Normal-Wishart parameters follow the same route. The correction evaluates in $O(N^2K^2)$ complexity.

F.2 Corrections for the Predictive Distribution

From (34) we can compute a first-order correction to the predictive distribution with $p(x|\mathcal{D}) \approx \int d\theta\, q(\theta)p(x|\theta)(1 + \sum_n \varepsilon_n(\theta))$, which we rewrite as

$$p(x|\mathcal{D}) \approx \sum_n \int d\theta\, p(x|\theta)q_n(\theta) - (N-1)\int d\theta\, p(x|\theta)q(\theta)\ .$$

Each predictive density in the above equation simplifies as

$$\int d\theta\, p(x|\theta)q_n(\theta) = \frac{1}{Z_n}\sum_k \sum_l \begin{cases} \dfrac{\lambda_{k\backslash n}\lambda_{l\backslash n}}{(\sum_{k'}\lambda_{k'\backslash n}+1)\sum_{k'}\lambda_{k'\backslash n}} p(x_n|k\backslash n)\, p(x|l\backslash n) & \text{if } k \neq l \\[2ex] \dfrac{(\lambda_{k\backslash n}+1)\lambda_{k\backslash n}}{(\sum_{k'}\lambda_{k'\backslash n}+1)\sum_{k'}\lambda_{k'\backslash n}} p(x_n|k\backslash n)\, p(x|x_n, k\backslash n) & \text{if } k = l. \end{cases}$$

We have seen how to compute $p(x_n|k\backslash n)$ in (26) and the discussion that followed it; we similarly define $p(x|l\backslash n)$. Density $p(x|x_n, k\backslash n)$ is again the Student-t distribution of (26), but now Λ is replaced with $\Lambda^*_{\backslash n}$ from (35). The correction evaluates in $O(NK^2)$ complexity.

References

H. Attias. A variational Bayesian framework for graphical models. In T. Leen et al., editor, *Advances in Neural Information Processing Systems 12*. MIT Press, Cambridge, 2000.

M. J. Beal and Z. Ghahramani. The variational Bayesian EM algorithm for incomplete data: with application to scoring graphical model structures. *Bayesian Statistics*, 7:453–464, 2003.

C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

C. M. Bishop and M. Svensén. Bayesian hierarchical mixtures of experts. In U. Kjaerulff and C. Meek, editors, *Proceedings of the Nineteenth Conference on Uncertainty in Artificial Intelligence*, pages 57–64. Morgan Kaufmann, 2003.

M. Chertkov and V. Y. Chernyak. Loop series for discrete statistical models on graphs. *Journal of Statistical Mechanics: Theory and Experiment*, 2006:P06009, 2006.

A. Corduneanu and C. M. Bishop. Variational Bayesian model selection for mixture distributions. In T. Richardson and T. Jaakkola, editors, *Proceedings Eighth International Conference on Artificial Intelligence and Statistics*, pages 27–34. Morgan Kaufmann, 2001.

J. Diebolt and C. P. Robert. Estimation of finite mixture distributions through Bayesian sampling. *Journal of the Royal Statistical Society B*, 56(2):363–375, 1994.

A. Engel, H. M. Köhler, F. Tschepke, H. Vollmayr, and A. Zippelius. Storage capacity and learning algorithms for two-layer neural networks. *Phys. Rev. A*, 45(10):7590–7609, May 1992. doi: 10.1103/PhysRevA.45.7590.

V. Gómez, J. M. Mooij, and H. J. Kappen. Truncating the loop series expansion for belief propagation. *Journal of Machine Learning Research*, 8:1987–2016, 2007.

P. Gregory. *Bayesian Logical Data Analysis for the Physical Sciences*. Cambridge University Press, 2005.

T. Heskes and O. Zoeter. Expectation propagation for approximate inference in dynamic Bayesian networks. In A. Darwiche and N. Friedman, editors, *Proceedings UAI-2002*, pages 216–233, 2002.

Y. Iba. Extended ensemble Monte Carlo. *International Journal of Modern Physics C*, 12(5):623–656, 2001.

S . Ikeda, T. Tanaka, and S. Amari. Information geometry of turbo and low-density parity-check codes. *IEEE Transactions on Information Theory*, 50(6):1097, 2004.

M. I. Jordan, Z. Ghahramani, T. S. Jaakkola, and L. K. Saul. An introduction to variational methods for graphical models. *Machine Learning*, 37(2):183–233, 1999.

D. A. Kofke. On the acceptance probability of replica-exchange Monte Carlo trials. *Journal of Chemical Physics*, 117(15):6911–6914, 2002.

D. J. C. MacKay. A problem with variational free energy minimization. Technical report, Department of Physics, University of Cambridge, 2001.

H. Matsui and T. Tanaka. Analysis on equilibrium point of expectation propagation using information geometry. In *International Conference on Neural Information Processing (ICONIP)*, 2008.

T. P. Minka. Expectation propagation for approximate Bayesian inference. In *UAI 2001*, pages 362–369, 2001a.

T. P. Minka. *A family of algorithms for approximate Bayesian inference*. PhD thesis, MIT Media Lab, 2001b.

T. P. Minka. Divergence measures and message passing. Technical Report MSR-TR-2005-173, Microsoft Research, Cambridge, UK, 2005.

J. M. Mooij, B. Wemmenhove, H. J. Kappen, and T. Rizzo. Loop corrected belief propagation. In *Proceedings of the Eleventh International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2007.

M. Opper and O. Winther. Gaussian processes for classification: Mean field algorithms. *Neural Computation*, 12:2655–2684, 2000.

M. Opper and O. Winther. Expectation consistent approximate inference. *Journal of Machine Learning Research*, 6:2177–2204, 2005.

M. Opper, U. Paquet, and O. Winther. Improving on expectation propagation. In *Neural Information Processing Systems*, 2008.

C. E. Rasmussen and Z. Ghahramani. Occam's razor. In T. Leen, T. Dietterich, and V. Tresp, editors, *Advances in Neural Information Processing Systems 16*. MIT Press, Cambridge, MA, 2001.

C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning*. The MIT Press, 2005.

S. Richardson and P. J. Green. On Bayesian analysis of mixtures with an unknown number of components. *Journal of the Royal Statistical Society (B)*, 59(4):731–792, 1997.

M. Seeger. *Bayesian Gaussian Process Models: PAC-Bayesian Generalisation Error Bounds and Sparse Approximations*. PhD thesis, University of Edinburgh, 2003.

J. Skilling. Probabilistic data analysis: an introductory guide. *Journal of Microscopy*, 190(1):28–36, 1998.

E. Sudderth, M. Wainwright, and A. Willsky. Loop series and Bethe variational bounds in attractive graphical models. In J.C. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems 20*, pages 1425–1432. MIT Press, Cambridge, MA, 2008.

Journal of Machine Learning Research 10 (2009) 1305-1340 Submitted 3/08; Revised 9/08; Published 6/09

Robust Process Discovery with Artificial Negative Events

Stijn Goedertier STIJN.GOEDERTIER@ECON.KULEUVEN.BE
David Martens[*] DAVID.MARTENS@ECON.KULEUVEN.BE
Jan Vanthienen JAN.VANTHIENEN@ECON.KULEUVEN.BE
Bart Baesens[†] BART.BAESENS@ECON.KULEUVEN.BE
Faculty of Business and Economics
Katholieke Universiteit Leuven
Leuven, Naamesestraat 69, Belgium

Editor: Paolo Frasconi, Kristian Kersting, Hannu Toivonen and Koji Tsuda

Abstract

Process discovery is the automated construction of structured process models from information system event logs. Such event logs often contain positive examples only. Without negative examples, it is a challenge to strike the right balance between recall and specificity, and to deal with problems such as expressiveness, noise, incomplete event logs, or the inclusion of prior knowledge. In this paper, we present a configurable technique that deals with these challenges by representing process discovery as a multi-relational classification problem on event logs supplemented with Artificially Generated Negative Events (AGNEs). This problem formulation allows using learning algorithms and evaluation techniques that are well-know in the machine learning community. Moreover, it allows users to have a declarative control over the inductive bias and language bias.

Keywords: graph pattern discovery, inductive logic programming, Petri net, process discovery, positive data only

1. Introduction

Learning descriptive or predictive models from sequence data is an important data mining task with applications in Web usage mining, fraud detection, bio-informatics, and process discovery. The learning task can be formulated as follows: given a sequence database that contains a finite number of sequences, find a useful generative model that describes or predicts its spatio-temporal properties. Depending on the application domain, a variety of sequence models and corresponding learning algorithms are available. For instance, probabilistic generative models such as (hidden) Markov models have been successfully applied in speech analysis, and bio-informatics (Durbin et al., 1998), whereas deterministic models such as partial orders have been applied in domains such as Web usage mining (Mannila and Meek, 2000; Pei et al., 2006). In this paper, we focus on the problem of process discovery, the discovery of business process models from event-based data generated by information systems. Process models typically contain structures such as sequences, or-splits, and-splits (parallel threads), or-joins, and-joins, loops (iterations), non-local, non-free, history-dependent or-splits, and duplicate activities. Because Petri nets can represent these con-

∗. David Martens is also at the Department of Business Administration and Public Management, University College Ghent, Association Ghent University, Belgium.
†. Bart Baesens is also at the School of Management, University of Southampton, United Kingdom.

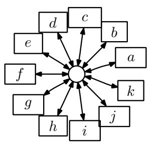
structs, they are known to be a convenient process modeling language (van der Aalst, 1998; Alves de Medeiros, 2006) provides a detailed overview of how these structures can be represented with Petri nets.

The motivation for process discovery is the abundant availability of information system event logs. The analysis of such event logs can provide insight into how processes *actually* take place, and to what extent actual processes deviate from a normative process model. Information system events keep track of, among others, the completion of an activity of a particular activity type. For example, an event can report that a particular activity of type ‘apply for license’ has completed. The goal of process discovery is to construct a useful process model that *describes* the event sequences recorded in the event log. Example 1, illustrates the learning problem for a fictitious driver’s license application process. Given the event sequence database in Example 1, a useful process models is to be conceived.

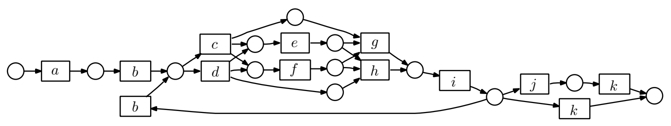
Example 1 *DriversLicenseI—discovery of a driver’s license application process with loop. The transitions correspond to activity types that have the following meaning: a start, b apply for license, c attend classes cars, d attend classes motor bikes, e obtain insurance, f theoretical exam, g practical exam cars, h practical exam motor bikes, i get result, j receive license, and k end.*

σ_1	*abcefgijk*
σ_2	*abdfehijk*
σ_3	*abdefhijbdfehijk*
σ_4	*abcfegik*
…	…

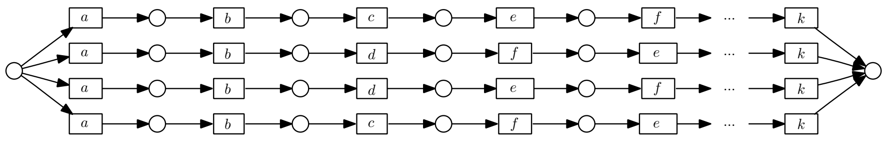
(a) sequences



(b) general



(c) specific and general



(d) over-specific

The construction of useful process models from an event log is subjected to many challenging problems. An inherent difficulty is that process discovery is limited to a setting of unsupervised learning. Event logs rarely contain negative information about state transitions that were prevented from taking place. Such negative information can be useful to discover the discriminating properties of the underlying process. In the absence of negative information, it is necessary to provide a learner with a particular *inductive bias*, to accurately strike the right balance between generality and specificity. The absence of negative information, makes the learning task aim at accurately summarizing an event log such that the discovered process model allows the observed behavior (recall) but does not include unintended, random behavior that is not present in the event log (specificity). In addition to accuracy, the learning problem faces challenges such as expressiveness, noise, incomplete event logs, and the inclusion of prior knowledge:

- **accuracy**: accuracy refers to the extent to which the induced model fits the behavior in the event log. Accuracy necessitates a tradeoff between specificity and recall. The Petri net in

Example 1, called a flower model, is capable of parsing every sequence in the event log. However, it can be considered to be overly general as it allows any activity to occur in any order. In contrast, the Petri net in Example 1 is overly specific, as it provides a mere enumeration of the different sequences in the event log. The Petri net in Example 1 is likely to be the more suitable process model. It is well-structured, and strikes a reasonable balance between specificity and generality, allowing for instance an unseen sequence *abcefgik*, but disallowing random behavior. An additional difficulty is that in the absence of negative information, the specificity of a learned process model is difficult to quantify.

- **expressiveness**: expressiveness relates to the ability to comprehensively summarize an event log using a rich palette of structures such as sequences, or-splits, and-splits (parallel threads), or-joins, and-joins, loops (iterations), history-dependent or-splits, and duplicate activities.
- **noise**: human-centric processes are prone to exceptions and logging errors. This causes additional low-frequent behavior to be present in the event log that is unwanted in the process model to be learned. Process discovery algorithms face the challenge of not overfitting this noise.
- **incomplete logs**: incomplete event logs do not contain the complete set of sequences that occur according to the underlying, real-life process. This is particularly the case for processes that portray a large amount of concurrent and recurrent behavior. In this case, process discovery algorithms must be capable of generalizing beyond the observed behavior.
- **prior knowledge**: the problem of consolidating the knowledge extracted from the data with the knowledge representing the experience of domain experts, is called the knowledge fusion problem (Dybowski et al., 2003). Prior knowledge constrains the hypothesis space of a sequence mining algorithm. In the context of process discovery, prior knowledge might refer to knowledge about concurrency (parallelism), locality, or exclusivity of activities. When a learner produces a process model that is not in line with the prior knowledge of a domain expert, the expert might refuse using the discovered process model. For instance, a domain expert might refuse a process model in which a pair of activities cannot take place concurrently, whereas in reality such parallelism is actually allowed.

In this paper, these challenges addressed by representing process discovery as an ILP classification learning problem on event logs supplemented with artificially generated negative events (AGNEs). The AGNEs technique is capable of constructing Petri net models from event logs and has been implemented as a mining plugin in the ProM framework. A benchmark experiment with 34 artificial event logs and comparison to four state-of-the-art process discovery algorithms indicate that the technique is expressive, robust to noise, and capable of dealing with incomplete event logs. A second contribution of the paper is the definition of a new metric for quantifying the specificity of an induced process model, based on these artificially generated negative events.

The remainder of this paper is structured as follows. Section 2 introduces some preliminaries and notations about inductive logic programming, event logs, and Petri nets. Section 3 explains the rationale for generating artificial negative events and provides a detailed description of the used algorithms. Section 4 discusses the process discovery technique. Section 5 introduces the new specificity metric that is based on negative events. Sections 6 and 7 provide both an experimental and practical evaluation of the process discovery technique. Finally, Section 8 provides an overview of the work in the area of process discovery and outlines the contributions made.

2. Preliminaries

This section introduces the most important concepts and notations that are used in the remainder of this paper.

2.1 Inductive Logic Programming

Inductive Logic Programming (ILP) (Muggleton, 1990; Džeroski and Lavrač, 1994, 2001; Džeroski, 2003) is a research domain in machine learning involving learners that use logic programming to represent data, background knowledge, the hypothesis space, and the learned hypothesis. ILP learners are called multi-relational learners and extend classical, uni-relational learners in the sense that they can not only learn patterns that occur within single tuples (within rows), but can also find patterns that may range over different tuples of different relations (between multiple rows of a single or multiple tables). For process discovery, this multi-relational property is much desired, as it allows discovering patterns that relate the occurrence of an event to the occurrence of any other event in the event log.

Within ILP, *concept learning* is an important learning task. An ILP classification learner will search for a hypothesis H in a hypothesis space $\mathcal{S}$ that best discriminates between the positive P and negative examples N ($E = P \cup N$) in combination with some given background knowledge B. A particularly salient feature of such learners is that they have a highly configurable language bias. The language bias $\mathcal{L}$ specifies the hypothesis space $\mathcal{S}$ of logic programs H that can be considered. In addition, users of ILP learners can specify background knowledge B as a logic program. Such background knowledge is a more parsimonious encoding of knowledge that is true about every example, than is the case for the attribute-value encoding of propositional learners. In addition to their multi-relational capabilities, the power of ILP concept learners lies with the configurability of their language bias $\mathcal{L}$ and background knowledge B. The effectiveness by which an ILP learner can be applied to a learning task depends on the choices that are made in representing the examples E, the background knowledge B and the language bias $\mathcal{L}$.

In this text, we make use of TILDE (Blockeel and De Raedt, 1998), a first-order decision tree learner available in the ACE-ilProlog data mining system (Blockeel et al., 2002). Blockeel (1998) formalizes the learning task of TILDE as follows:
given:

- a set of classes C
- a set of classified examples E, each example $(e, c) \in E$ is an independent logic program for which the predicate $Class(c)$ denotes that e is classified into class c.
- a logic program B that represents the background knowledge
- a language bias $\mathcal{L}$ that specifies a hypothesis space $\mathcal{S}$ of logic programs.

find: a hypothesis $H \in \mathcal{S}$ (a logic program) such that for all labeled examples $(e, c) \in E$,

- $\forall e \in E : H \wedge e \wedge B \vDash Class(c)$
- $\forall e \in E, \forall c' \in C \backslash \{c\} : H \wedge e \wedge B \nvDash Class(c')$.

TILDE is a first-order generalization of the well-known C4.5 algorithm for decision tree induction (Quinlan, 1993). Like C4.5, TILDE (Blockeel and De Raedt, 1998; Blockeel et al., 2002) obtains classification rules by recursively partitioning the data set according to logical conditions that can be represented as nodes in a tree. This top-down induction of logical decision trees (TILDE) is driven by refining the node criteria according to the provided language bias $\mathcal{L}$. Unlike C4.5, TILDE is capable of inducing first-order logical decision trees (FOLDT). A FOLDT is a tree that holds logical formula containing variables instead of propositions. Blockeel and De Raedt (1998) show how each FOLDT can be converted into a logic program.

2.2 Event Logs

In process discovery, an event log is a database of event sequences. Each event reports an instantaneous state change of an activity of a particular activity type. Activities and events that pertain to the same process instance are identified by a so-called case identifier. In process discovery, the MXML format for event logs (van Dongen and van der Aalst, 2005a) is the commonly accepted format for event logs. To use event logs with TILDE event logs have to be represented as a logical program. Let X be a set of event identifiers, P a set of case identifiers, the alphabet A a set of activity types, and $E = \{completed, completeRejected\}$ a set of event types. An event predicate is a quintuple $Event(x, p, a, e, t)$, where $x \in X$ is the event identifier, $p \in P$ is the case identifier, $a \in A$ the activity type, $e \in E$ the event type, and $t \in \mathbb{N}$ the time of occurrence of the event. The function $Case \in X \cup L \rightarrow P$ denotes the case identifier of an event or a sequence. The function $AT \in X \rightarrow A$ denotes the activity type of an event. The function $ET \in X \rightarrow E$ denotes the event type of an event. The function $Time \in X \rightarrow \mathbb{N}$ denotes the time of occurrence of an event. The set X of identifiers has a complete ordering, such that $\forall x, y \in X : x < y \vee y < x$ and $\forall x, y \in L : x < y \Rightarrow Time(x) \leq Time(y)$. The event types $E = \{completed, completeRejected\}$ respectively indicate the completion of a particular activity or that the completion of a particular activity could not take place, a negative event.

Let an event log L be a set of sequences σ. Let $\sigma \in L$ be an event sequence, an ordered set of event identifiers $x \in X$ of events pertaining to the same process instance as denoted by the case id; $\sigma = \{x \mid x \in X \wedge Case(x) = Case(\sigma)\}$. The function $Position \in X \times L \rightarrow \mathbb{N}_0$ denotes the position of an event with identifier $x \in X$ in the sequence $\sigma \in L$. Two subsequent event identifiers within a sequence σ can be represented as a sequence $x.y \subseteq \sigma$. We define the .-predicate as follows

$$x.y \Leftrightarrow \exists x, y \in \sigma : x < y \wedge \nexists z \in \sigma : x < z < y.$$

In the text, this predicate is used within the context of a single sequence σ which is therefore left implicit. Given that $AT(x) = a, AT(y) = b$ the information in the sequence can be further abbreviated as *ab*, because the order of the activity types in a sequence is the most important information for the purpose of process discovery. This notation is used to represent the event log in Example 1. Each row σ_i in the event log represents a different execution sequence that corresponds to a particular process instance.

2.3 Petri Nets

Example 1 is a Petri net representation of a simplified driver's license application process. Petri nets represent a graphical language with a formal semantics that has been used to represent, analyze, verify, and simulate dynamic behavior (Murata, 1989). Petri nets consist of places, tokens, and arcs. *Places* (drawn as circles) can contain *tokens* and are a distributed representation of state.

Each different distribution of tokens over the places of a Petri net indicate a different state. Such a state is called a *marking*. *Transitions* (drawn as rectangles) can consume and produce tokens and represent a state change. *Arcs* (drawn as arrows) connect places and transitions and represent a flow relationship. More formally, a marked Petri net is a pair $((P,T,F),s)$ where,

- P is a finite set of places,
- T is a finite set of transitions such that $P \cap T = \emptyset$, and
- $F \subseteq (P \times T) \cup (T \times P)$ is a finite set of direct arcs, and
- $s \in P \to \mathbb{N}$ is a bag over P denoting the marking of the net (van der Aalst, 1997, 1998).

Petri nets are bipartite directed graphs, that means that each arc must connect a transition to a place or a place to a transition. The transitions in a Petri net can be labeled or not. Transitions that are not labeled are called *silent transitions*. Different transitions that bear the same label are called *duplicate transitions*.

The behavior of a Petri net is determined by the *firing rule*. A transition t is *enabled* iff each input place p of t contains at least one token. When a transition is enabled it can *fire*. When a transition fires, it brings about a state change in the Petri net. In essence, it consumes one token from each input place p and produces one token in each output place p of t. To evaluate the extent to which a Petri net is capable of parsing an event sequence, transitions might be forced to fire. A transition that is not enabled, can be *forced to fire*. When a transition is forced to fire, it consumes one token from each input place that contains a token—if any—and produces one token in each output place. Petri nets are capable of representing structures such as sequences, or-splits, and-splits (parallel threads), or-joins, and-joins, loops (iterations), history-dependent or-splits, and duplicate activities that are typical for organizational processes.

3. Artificial Negative Events

Event logs generally contain sequences of positive events only. To make a tradeoff between overly general or overly specific process models, learners make additional assumptions about the given event sequences. Such assumptions are part of the inductive bias of a learner. Process discovery algorithms generally include the assumption that event logs portray the complete behavior of the underlying process and implicitly use this completeness assumption to make a tradeoff between overly general and overly specific process models. Our technique makes this completeness assumption explicit by inducing artificial negative information from the event log in a configurable way.

For processes that contain a lot of recurrent and concurrent behavior, the completeness assumption can become problematic. For example, a process containing five parallel activities (ten parallel pairs) that are placed in a loop, has $\sum_{i=1}^{n}(5!)^i$ different possible execution sequences (n being the maximum number of allowed loops).

The more recurrent behavior a process has, the more different kinds of event sequences a process can produce. This makes a given event log less likely to contain all possible behavior. The problem of recurrent behavior is addressed by restricting the window size (parameter: *windowSize*). Window size is the number of events in the subsequence one hopes to detect at least once in the sequence database. The larger the window size, the less probable that a similar subsequence is contained by the other sequences in the event log. A limited window size can be advantageous in the presence of loops (recurrent behavior) in the underlying process. Limiting the window size to a smaller subsequence of the event log, makes the completeness assumption less strict. An unlimited window size ($windowSize = -1$) results in the most strict completeness assumption.

The problem of concurrent behavior is addressed by exploiting some available parallelism information, discovered by induction or provided as prior knowledge by a domain expert. Given a subsequence and parallelism information, all parallel variants of the subsequence can be calculated. Taking into account the parallel variants of a subsequence makes the completeness assumption less strict.The function $AllParallelVariants(\tau)$ returns the set of all parallel variants that can be obtained by permuting the activities in each sub-sequence of τ while preserving potential dependency relationships among non-parallel activities.

Negative events record that at a particular position in an event sequence, a particular event cannot occur. At each position k in each event sequence τ_i, it is examined which negative events can be induced for this position. Algorithm 1 gives an overview of the negative event induction and is discussed in the next paragraphs. In a first step, the event log is made more compact, by grouping process instances $\sigma \in L$ that have identical sequences into grouped process instances $\tau \in M$ (line 1). By grouping similar process instances, searching for similar behavior in the event log can be performed more efficiently.

In the next step, all negative events are induced for each grouped process instance (lines 2–12). Making a completeness assumption about an event log boils down to assuming that behavior that does not occur in the event log, should not occur in the process model to be learned. Negative examples can be introduced in grouped process instances τ_i by checking at any given positive event $x_k \in \tau_i$ at position $k = Position(x_k, \tau_i)$ whether another event of interest z_k of activity type $b \in A \backslash \{AT(x_k)\}$ also could occur. For each event $x_k \in \tau_i$, it is tested whether there exists a similar sequence $\tau_j^{\|} \in AllParallelVariants(\tau_j) : \tau_j \neq \tau_i$ in the event log in which at that point a state transition y_k has taken place that is similar to z_k (line 6). If such a state transition does not occur in any other sequence, such behavior is not present in the event log L. This means under the completeness assumption that the state transition cannot occur. Consequently, z_k can be added as a negative event at this point k in the event sequence τ_i (lines 7–8). On the other hand, if a similar sequence is found with this behavior, no negative event is generated.

Finally, the negative events in the grouped process instances are used to induce negative events into the similar non-grouped sequences. If a grouped sequence τ contains negative events at position k, then the ungrouped sequence σ contains corresponding negative events at position k. At each position, a large number of negative events can generally be generated. To avoid an imbalance in the proportion of negative versus positive events the addition of negative events can be manipulated with a negative event injection probability π (line 13). π is a parameter that influences the probability that a corresponding negative event is recorded in an ungrouped trace σ. The smaller π, the less negative events are generated at each position in the ungrouped event sequences. A value of $\pi = 1.0$ means that every induced negative event for a grouped sequence is included in every similar, corresponding, non-grouped sequence. A value of $\pi = 0$ will result in no negative events being induced for any of the corresponding, non-grouped sequences.

Example 2 illustrates how in an event log of two (grouped) sequences τ_1 and τ_2 artificial negative events can be generated. The event sequences originate from a simplified driver's license process, depicted at the bottom. Given the parallelism information, $Parallel(e, f)$, the event sequences each have two parallel variants. When generating negative events into event sequence τ_1, it is examined whether instead of the first event b the events $c, d, e, f, g, h,$ or i could also have occurred at the

Algorithm 1 Generating artificial negative events

```
1: Group similar sequences $\sigma \in L$ into $\tau \in M$
2: for all $\tau_i \in M$ do
3:   for all $x_k \in \tau_i$ do
4:     $k = Position(x_k, \tau_i)$
5:     for all $b \in A \backslash \{AT(x_k)\}$ do
6:       if $\nexists \tau_j^{\|} \in AllParallelVariants(\tau_j) : \forall \tau_j \in M \wedge \tau_j \neq \tau_i \wedge$
         $\forall y_l \in \tau_j^{\|}, Position(y_l, \tau_j^{\|}) = l = Position(x_l, \tau_j), k - windowSize < l < k : AT(y_l) = AT(x_l) \wedge$
         $y_k \in \tau_j^{\|}, Position(y_k, \tau_j^{\|}) = k, AT(y_k) = b$ then
7:         $z_k$ = event with $AT(z_k) = b, ET(z_k) = completeRejected$
8:         recordNegativeEvent($z_k, k, \tau_i$)
9:       end if
10:    end for
11:  end for
12: end for
13: Induce negative events in the non-grouped sequences $\sigma \in L$ according to an injection frequency $\pi$
```

first position. Because there is no similar sequence $\tau_j^{\|}$ in which c, d, e, f, g, h, or i occur at this position, it can be concluded that they are negative events. Consequently $\neg c, \neg d, \neg e, \neg f, \neg g, \neg h$, and $\neg i$ are added as negative events at this position. Other artificial negative events are generated in a similar fashion. Notice that history-dependent processes generally will require a larger window size to correctly detect all non-local dependencies. In Example 2, an unlimited window size is used. Should the window size be limited to 1, for instance, then it would no longer be possible to take into account the non-local dependency between the activity pairs *c*–*g* and *d*–*h*. In the experiments at the end of this paper, an unlimited window size has been used (parameter value -1).

4. Process Discovery

Having an algorithm to artificially generate negative events, it becomes possible to represent process discovery as a binary classification problem, that learns the conditions that discriminate between the occurrence of an activity (a positive event), or the non-occurrence of an activity (a negative event). Algorithm 2 outlines four major steps in the AGNEs process discovery procedure. These four steps are addressed this section.

4.1 Step 1: Induce Parallelism and Locality Information

The starting point is the gathering of information about local dependency ($Local(a,b)$), and parallelism relationships ($Parallel(a,b)$, or $Serial(a,b)$) that exist between pairs of activities $a, b \in A$ in an event log L. Locality information is used in the language bias of the classification learner, to constrain the hypothesis space to locally discriminating preconditions when required. Without locality information, the classification learner is likely also to come up with non-local dependencies

Example 2 *(Continued from example 1.) DriversLicense—Generating artificial negative events for a simplified event log with two abbreviated event sequences.*

τ_1	b	c	e	f	g	i
τ_2	b	d	f	e	h	i

Parallel(e,f).

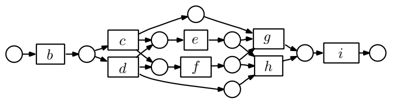
(a) given event log and background knowledge

$\tau_1^{\parallel}$	b	c	f	e	g	i
$\tau_1^{\parallel}$	b	c	e	f	g	i
$\tau_2^{\parallel}$	b	d	f	e	h	i
$\tau_2^{\parallel}$	b	d	e	f	h	i

(b) the derived parallel variants

τ_1	b	c	e	f	g	i
	$\neg c$	$\neg b$	$\neg b$	$\neg b$	$\neg b$	$\neg b$
	$\neg d$	$\neg e$	$\neg c$	$\neg c$	$\neg c$	$\neg c$
	$\neg e$	$\neg f$	$\neg d$	$\neg d$	$\neg d$	$\neg d$
	$\neg f$	$\neg g$	$\neg g$	$\neg g$	$\neg e$	$\neg e$
	$\neg g$	$\neg h$	$\neg h$	$\neg h$	$\neg f$	$\neg f$
	$\neg h$	$\neg i$	$\neg i$	$\neg i$	$\neg h$	$\neg g$
	$\neg i$				$\neg i$	$\neg h$
τ_2	b	d	f	e	h	i
	$\neg c$	$\neg b$	$\neg b$	$\neg b$	$\neg b$	$\neg b$
	$\neg d$	$\neg e$	$\neg c$	$\neg c$	$\neg c$	$\neg c$
	$\neg e$	$\neg f$	$\neg d$	$\neg d$	$\neg d$	$\neg d$
	$\neg f$	$\neg g$	$\neg g$	$\neg g$	$\neg e$	$\neg e$
	$\neg g$	$\neg h$	$\neg h$	$\neg h$	$\neg f$	$\neg f$
	$\neg h$	$\neg i$	$\neg i$	$\neg i$	$\neg g$	$\neg g$
	$\neg i$				$\neg i$	$\neg h$

(c) the induced negative events



(d) the underlying Petri net

```
Algorithm 2 Process discovery by AGNEs: overview
1: step 1: induce parallelism and locality from frequent temporal constraints
2: step 2: generate artificial negative events
3: step 3: learn the preconditions of each activity type
4: step 4: transform the preconditions into a Petri net
```

that cannot easily be transformed into a graphical model. Parallelism information is used to prevent the construction of sequential models, where in reality concurrency is expected.

This information is either gathered by the analysis of frequent temporal constraints that hold in the event log or by means of user-specified prior knowledge. Frequent temporal constraints are temporal constraints that hold in a sufficient number of sequences σ within an event log L. Goedertier (2008), defines a number of temporal constraints and shows how local dependency and parallelism information can be derived from it.

Additionally, the end-user can also specify locality and parallelism information in terms of prior knowledge. In particular, the following predicates can be used: *PriorParallel*(a,b), *PriorSerial*(a,b), *PriorLocal*(a,b), and *PriorNonLocal*(a,b). Information specified using these predicates defeats any inference about locality or parallelism made on the basis of the information gathered from frequent temporal constraints.

4.2 Step 2: Generate Artificial Negative Events

A second step in the process discovery technique is the induction of artificial negative events, as described in Section 3. The inferred information about parallel activity pairs can be used for parallel variant calculation. Furthermore, the induction of negative events is dependent on the window size *windowSize*, and negative event injection probability π parameters.

4.3 Step 3: Learn the Preconditions of each Activity Type

It is now possible to represent process discovery as a multiple, first-order classification learning problem that learns the preconditions that discriminate between the occurrence of either a positive or a negative completion event for each activity type. In our experiments, we make use of TILDE, an existing multi-relational classification learner, to perform the actual classification learning.

The motivation for representing process discovery as a classification problem is that it allows using classification learning and evaluation techniques that are well-known in the machine learning community. Furthermore, classification learners have the potential to deal with so-called **duplicate tasks**. Duplicate tasks refer to the reoccurrence of identically labeled transitions, homonyms, in different contexts within the event logs. Classification learning can detect the different execution contexts for these transitions and derive different preconditions for them. Transforming these preconditions into graphs will eventually result in duplicate, homonymic activities in the graph model that correspond to the different usage contexts. Techniques for process discovery, such as heuristics miner and genetic miner, which have causal matrices as internal representation (Weijters et al., 2006; Alves de Medeiros et al., 2007), are unable to discover duplicate activities. Alves de Medeiros (2006), however, presents a non-trivial extension of the genetic miner that includes duplicate tasks.

The motivation for using a multi-relational, first-order representation is that first-order learning allows relating the occurrence of an event to the occurrence of any other event. This enables the detection of patterns that involve non-local, historic, dependencies between events. Alternatively, the history of each event could in part be represented as extra propositions, for instance by including all preceding positive events as extra columns in the event log. This propositional representation would have many difficulties.

Being able to detect non-local, historic patterns in an event log can also produce counter-intuitive results. A non-local relationship might have more discriminating power than a local one and can therefore be preferred by a learner. Unfortunately, an excess of non-local patterns makes it more difficult to generate a graphical model, a Petri net, containing local connections. Because TILDE allows specifying a language bias with dynamic refinement (Blockeel and De Raedt, 1998), it becomes possible to constrain the hypothesis space to locally discriminating patterns whenever necessary. One technique is the dynamic generation of constants, that can be used to constrain the combinations of activity type constants that are to be considered for a particular language bias construct. Additionally, it is also possible to constrain the occurrence of particular literals in a hypothesis $\mathcal{H}$, given the presence or absence of other literals that are already part of the hypothesis.

The classification task of TILDE is to predict whether for a given activity type $a \in A$, at a given time point $t \in \mathbb{N}$ in a given sequence $\sigma \in L$, a positive or a negative event takes place. In the case of a positive event, the target predicate evaluates to $Class(a, \sigma, t, completed)$. In the case of a negative event, the target predicate evaluates to $Class(a, \sigma, t, completeRejected)$. In the language bias, the target activity, indicated by a, will be used for dynamically constraining the combinations of activity type constants generated.

The primary objective of AGNEs being the construction of a graphical model from an event log, the language bias consists of a logical predicate that can represent the conditions under which a Petri net place contains a token. This predicate is called the “no-sequel” predicate, $NS(a_1, a, \sigma, t)$. Let

$a_1, a \in A$ be activity types, $\sigma \in L$ the sequence of a process instance, and t the time of observation. The *NS* predicate is defined as follows:

$$\forall a_1, a \in A, t \in \mathbb{N} : \exists x \in \sigma : AT(x) = a_1 \ \wedge\ Time(x) < t \\ \wedge\ \nexists y \in \sigma : AT(y) = a \ \wedge\ Time(x) < Time(y) < t \quad \Rightarrow NS(a_1, a, \sigma, t).$$

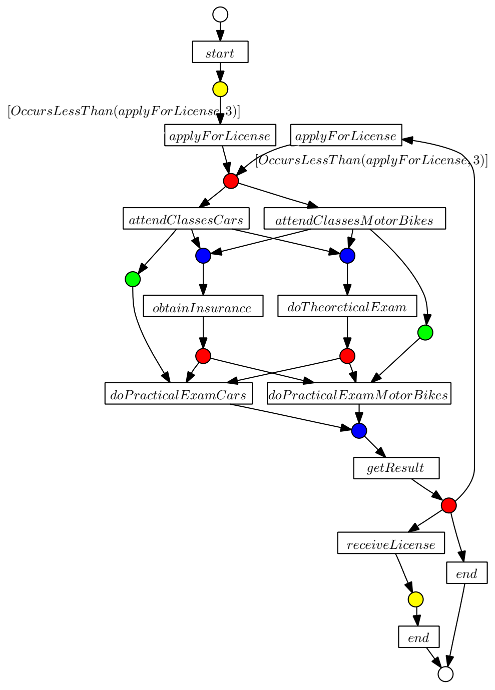
In the remainder of this text, the arguments σ and t will be implicitly assumed and therefore left out. The predicate $NS(a_1, a)$ evaluates to true when at the time of observation, an activity a_1 has completed, but has not (yet) been followed by an activity a. In combination with conjunction ($\wedge$), disjunction ($\vee$) and negation-as-failure ($\sim$), the $NS(a_1, a)$ predicate makes it possible to learn fragments of Petri nets using a multi-relational classification learner.

Example 3 shows how the preconditions in the Petri net can be represented as conjunctions and disjunctions of *NS* atoms. This representation accounts for the different contexts in which the activities *applyForLicense* and *end* can take place and derives different preconditions for these activities. In addition, it can represent the non-local, non-free, history-dependent choice construct between the activities *attendClassesCars–doPracticalExamCars* and *attendClassesMotorBikes–doPracticalExamMotorBikes* and the maximum recurrence of the activity *applyForLicense*. The Petri net fragments included in the language bias of AGNEs are enumerated in Figure 1 and are briefly discussed in the remainder of this section.

Example 3 *(Continued from example 2) DriversLicenseI—A Petri net in terms of NS preconditions*

	activity	*precondition*
a	*start*	*true*
b	*applyForLicense*	$NS(a,b)$
b	*applyForLicense*	$(NS(i,b) \wedge NS(i,j) \wedge NS(i,k))$ $\wedge\ OccursLessThan(b,3)$
c	*attendClassesCars*	$NS(b,c) \wedge NS(b,d)$
d	*attendClassesMotorBikes*	$NS(b,c) \wedge NS(b,d)$
e	*obtainInsurance*	$NS(c,e) \vee NS(d,e)$
f	*doTheoreticalExam*	$NS(c,f) \vee NS(d,f)$
g	*doPracticalExamCars*	$(NS(f,g) \wedge NS(f,h)) \wedge$ $(NS(e,g) \wedge NS(e,h)) \wedge NS(c,g)$
h	*doPracticalExamMtrBikes*	$(NS(f,g) \wedge NS(f,h)) \wedge$ $(NS(e,g) \wedge NS(e,h)) \wedge NS(d,h)$
i	*getResult*	$NS(g,i) \vee NS(h,i)$
j	*receiveLicense*	$NS(i,b) \wedge NS(i,j) \wedge NS(i,k)$
k	*end*	$NS(j,k)$
k	*end*	$NS(i,b) \wedge NS(i,j) \wedge NS(i,k)$

(a) DriversLicenseI—activity preconditions



(b) DriversLicenseI—Petri net

Figure 1(a) shows a graphical representation of the local sequence predicate that is part of the language bias. A local sequence of two transitions labeled $a_1, a \in A$ in a Petri net can be represented by $NS(a_1, a)$. In the language bias, the following constraints apply:

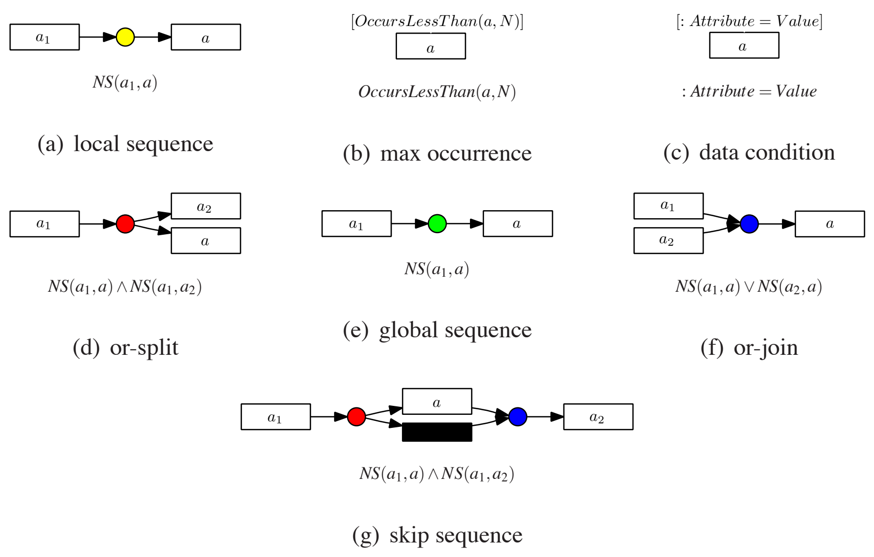


Figure 1: Petri net patterns in the language bias

sequence	$NS(a_1,a)$	
constraints:	$Local(a_1,a)$.	(1)
	$\nexists l \in \mathcal{H} : l = [NS(\_,a_1)]$.	(2)

The conversion from *NS*-based preconditions into Petri nets requires the conditions to refer to local, immediately preceding events as much as possible. Therefore, constraint (1) requires to restrict the generation of constants a_1 to constants that are local to a. In constraint (1) we do not require a_1 and a to be different activity types. This is sufficient for the discovery of length-one loops. Graphically, a conjunction of *NS* constructs can be considered to be the logical counterpart of a single Petri net place. Besides the case of length-one loops—that can be expressed with a single *NS* construct—there is no reason for a Petri net place to contain both an input and an output arc that is connected to the same transition. Constraint (2) has been put in place to keep a multi-relational learner from constructing such useless hypotheses. The constraint stipulates that the construct $NS(a_1,a)$ can only be added to the current hypothesis $\mathcal{H}$, if $\mathcal{H}$ does not already contain a logical condition that boils down to an output arc towards a_1.

The same $NS(a_1,a)$ construct, with different constraints, can be used to keep track of a global sequence (see Figure 1(e)). Global sequences are used to represent non-local, non-free choice constructs. We require TILDE only to consider global sequence between activity types for which both a precedence and a response relationship has been detected from the event log, this is expressed as a language bias constraint. Because we assume that any transition must be locally connected, that is connected to other transitions to which it is local in the event log, we require as an additional constraint that the global sequence construct must not be added first to any hypothesis. The other Petri net based language constructs depicted in Figure 1 have similar definitions and are described in detail by Goedertier (2008).

The language bias of TILDE is limited to conjunctions and disjunctions of *NS* constructs of length two and three. Or-splits and or-joins that involve more activity types are obtained by grouping conjunctions and disjunctions of *NS* constructs into larger conjunctions and disjunctions in step 4. However, this limitation in length sometimes leads to TILDE make inadequate refinements. Solving this language bias issue, requires constructing a proprietary ILP classification algorithm that during each refinement step allows considering conjunctions of *NS* constructs of variable lengths. We leave this improvements to future work.

4.4 Step 4: Transform the Preconditions into a Petri Net

In the previous step, TILDE has learned preconditions for each activity type independently. In a Petri net, the preconditions for each activity type are nonetheless interrelated. Therefore, the logic programs of activity preconditions are submitted to several rule-level pruning steps, to make sure that they do not produce redundant duplicate places in the Petri net to be constructed. These pruning steps occur among the conditions within a single precondition, within a set of preconditions to the same activity type and among preconditions of activity types that pertain to the same or-split. Given a pruned rule set of preconditions for each activity type, the construction of a Petri net is fairly straightforward. Each induced precondition corresponds to a different transition in a Petri net to be constructed. Because AGNEs may produce several preconditions for an activity type, the constructed Petri net may contain duplicate transitions. Algorithm 3 provides an overview of these procedures. In the remainder of this section, these different pruning steps will be discussed and illustrated for a sample of rules from the DriversLicenseI example.

The logic programs constructed by TILDE contain rules that classify the occurrence of either a positive or a negative event. By construction, the language bias of AGNEs is such that TILDE will never consider a negation of an *NS* construct to be explanatory for the occurrence of a positive event. Therefore, TILDE will never construct a tree in which a right leaf predicts a positive event. The latter entails that, in equivalent the logic program, the rules with a $class(a,\sigma,t,comleted)$ rule head can be taken from the logic programs without loss of information (line 1). Example 4 depicts the preconditions that are induced by TILDE for the apply for license and end activities in the DriversLicenseI example.

In a second step, a number of intra-rule pruning operations take place (lines 2–6). The top-down refinement of hypotheses, can result in the derivation of logically redundant conditions. These logical redundancies are removed for each rule (line 3). Each rule in the logic program consists of conjunctions of groups of *NS* constructs. A conjunction of a pair of or-split constructs that originate from the same activity a_1 can be combined into a larger or-split (line 4). Likewise, a conjunction of a pair of or-joins can be combined into a larger or-join (line 5). Example 4 illustrates intra-rule pruning for the DriversLicenseI example.

In a third step, a number of inter-rule pruning operations are performed for the preconditions that pertain to each activity type. In particular, a precondition that subsumes another precondition for the same activity type, is redundant and thus removed from consideration (lines 10–12). Furthermore, it is examined whether a more specific or-join can be constructed out of the groups of *NS* constructs within the different preconditions of the same rule (lines 13–15). Finally, all rules are examined to find the most specific or-split condition from the preconditions extracted by TILDE (lines 18–26). Example 4 illustrates this pruning for the DriversLicenseI example.

Example 4 *(Continued from example 3.) DriversLicensel—pruning*

rules induced by TILDE		:
b apply for license	$(NS(i,b) \vee NS(a,b)) \wedge (NS(i,b) \wedge NS(i,j))$.	
b apply for license	$(NS(i,b) \vee NS(a,b))$.	
k end	$(NS(j,k))$.	
k end	$(NS(i,b) \wedge NS(i,k))$.	
intra-rule pruning:		
b apply for license	$(NS(i,b) \wedge NS(i,j))$. *(line 3)*	
b apply for license	$(NS(i,b) \vee NS(a,b))$.	
k end	$(NS(j,k))$.	
k end	$(NS(i,b) \wedge NS(i,k))$.	
inter-rule pruning, most specific or split:		
b apply for license	$(NS(i,b) \wedge NS(i,j) \wedge NS(i,k))$. *(lines 18–26)*	
b apply for license	$(NS(a,b))$. *(lines 10–12)*	
k end	$(NS(j,k))$.	
k end	$(NS(i,b) \wedge NS(i,j) \wedge NS(i,k))$. *(lines 18–26)*	

5. Evaluation Metrics

Discovered process models preferably allow the behavior in the event log (recall) but no other, unobserved, random behavior (specificity). Having formulated process discovery as a binary classification problem on event logs supplemented with artificial negative events, it becomes possible to use the true positive and true negative rate from classification learning theory to quantify the recall and specificity of a process model:

- **true positive rate** TP_{rate} or **recall**: the percentage of correctly classified positive events in the event log. This probability can be estimated as follows: $TP_{rate} = \frac{TP}{TP+FN}$, where TP is the amount of correctly classified positive events and FN is the amount of incorrectly classified positive events.

- **true negative rate** TN_{rate} or **specificity**: the percentage of correctly classified negative events in the event log. This probability can be estimated as follows: $TN_{rate} = \frac{TN}{TN+FP}$, where TN is the amount of correctly classified negative events and FP is the amount of incorrectly classified negative events.

Accuracy is the sum of the true positive and true negative rate, weighted by the respective class distributions. The fact that accuracy is relative to the underlying class distributions can lead to unintuitive interpretations. Moreover, in process discovery, these class distributions can be quite different and have no particular meaning. In order to make abstraction of the proportion of negative and positive events, we propose, from a practical viewpoint to attach equal importance to both recall and specificity: $acc = \frac{1}{2}recall + \frac{1}{2}specificity$. According to this definition, the accuracy of a majority-class predictor is $\frac{1}{2}$. Flower models, such as the one in Example 1(b), are an example of such a majority-class predictors. Because a flower model represents random behavior, it has a perfect recall of the all behavior in the event log but it also has much *additional* behavior compared to the event log. Because of the latter fact, the flower model has zero specificity, and an accuracy of $\frac{1}{2}$. Any useful process model should have an accuracy higher than $\frac{1}{2}$.

Algorithm 3 Rule-level pruning

```
1:  $R^+ = \{r \in R \mid r$ has rule head $class(a,\sigma,t,comleted)$ $\}$.
    // intra-rule pruning:
2:  for all $r \in R^+$ do
3:      reduce $r$ according to $(NS_1 \vee NS_2) \wedge NS_1 \equiv NS_1$.
4:      group or-splits: $NS(a_1,a) \wedge NS(a_1,a_2) \in r$ and $NS(a_1,a) \wedge NS(a_1,a_3) \in r$ into $NS(a_1,a) \wedge$
        $NS(a_1,a_2) \wedge NS(a_1,a_3)$.
5:      group or-joins: $NS(a_1,a) \vee NS(a_2,a) \in r$ and $NS(a_3,a) \vee NS(a_4,a) \in r$ into $NS(a_1,a) \vee$
        $NS(a_2,a) \vee NS(a_3,a) \vee NS(a_4,a)$.
6:  end for
    //inter-rule pruning:
7:  for all $a \in A$ do
8:      $R_a^+ = \{r \in R^+ \mid r$ is a precondition of $a\}$
9:      for all $r \in R_a^+$ do
10:         if $\exists s \in R_a^+$ : $s$ is more specific than $r$ then
11:             remove $r$.
12:         end if
13:         if $\exists s \in R_a^+$ : $s$ combined with $r$ lead to a more specific or-join than $r$ then
14:             replace the or-join in $r$ with the more specific or-join.
15:         end if
16:     end for
17: end for
    //keep the most specific or-split:
18: for all $a \in A$ do
19:     $R_a^{split} = \{r \in R^+ \mid r$ contains an or-split going out $a$ $\}$.
20:     $s$ = the most specific or-split by combining the or-splits going out $a$ in $R_a^{split}$.
21:     for all $r \in R_a^{split}$ do
22:         if $s$ is more specific than $r$ then
23:             replace the or-split in $r$ with the or-split in $s$.
24:         end if
25:     end for
26: end for
```

The metrics that are introduced in this section will be used to evaluate the performance of AGNEs in the following sections. They have been defined and implemented for Petri nets. However, they can also be applied to other generative models.

5.1 Existing Metrics

Weijters et al. (2006) define a metric that has a somewhat similar interpretation as TP_{rate}: the parsing measure PM. The measure is defined as follows:

- **parsing measure** PM: the number of sequences in the event log that are correctly parsed by the process model, divided by the total number of sequences in the event log. For efficiency, the similar sequences in the event log are grouped. Let k represent the number of grouped sequences, n_i the number of process instances in a grouped sequence i, c_i a variable that is equal to 1 if grouped sequence i can be parsed correctly, and 0 if grouped sequence i cannot be parsed correctly. The parsing measure can be defined as follows Weijters et al. (2006):

$$PM = \frac{\sum_{i=1}^{k} n_i c_i}{\sum_{i=1}^{k} n_i}.$$

PM is a coarse-grained metric. A single missing arc in a Petri net can result in parsing failure for all sequences. A process model with a single point of failure is generally better than a process model with more points of failure. This is not quantified by the parsing measure PM.

Rozinat and van der Aalst (2008) define two metrics that have a somewhat similar interpretation as TP_{rate} and TN_{rate}: the fitness metric f and the advanced behavioral appropriateness metric $a_B^{'}$:

- **fitness** f: Fitness is a metric that is obtained by trying whether each (grouped) sequence in the event log can be reproduced by the generative model. This procedure is called *sequence replay*. The metric assumes the generative model to be a Petri net. At the start of the sequence replay, f has an initial value of one. During replay, the transitions in the Petri net will produce and consume tokens to reflect the state transitions. However, the proportion of tokens that must additionally be created in the marked Petri net, so as to *force* a transition to fire, is subtracted from this initial value. Likewise, the fitness measure f punishes for extra behavior by subtracting the proportion of remaining tokens relative to the total number of produced tokens from this initial value. Let k represent the number of grouped sequences, n_i the number of process instances, c_i the number of tokens consumed, m_i the number of missing tokens, p_i the number of produced tokens, and r_i the number of remaining tokens for each grouped sequence i $(1 \leq i \leq k)$. The fitness metric can be defined as follows (Rozinat and van der Aalst, 2008):

$$f = \frac{1}{2}\left(1 - \frac{\sum_{i=1}^{k} n_i m_i}{\sum_{i=1}^{k} n_i c_i}\right) + \frac{1}{2}\left(1 - \frac{\sum_{i=1}^{k} n_i r_i}{\sum_{i=1}^{k} n_i p_i}\right).$$

- **behavioral appropriateness** $a_B^{'}$: Behavioral appropriateness is a metric that is obtained by an exploration of the state space of a Petri net and by comparing the different types of *follows* and *precedes* relationships that occur in the state space with the different types of *follows* and *precedes* relationships that occur in the event log. The metric is defined as the proportion of number of *follows* and *precedes* relationships that the Petri net has in common with the event log vis-à-vis the number of relationships allowed by the Petri net. Let S_F^m be the S_F relation and S_P^m be the S_P relation for the process model, and S_F^l the S_F relation and S_P^l the S_P relation for the event log. The advanced behavioral appropriateness metric $a_B^{'}$ is defined as follows (Rozinat and van der Aalst, 2008):

$$a_B^{'} = \left(\frac{|S_F^l \cap S_F^m|}{2.|S_F^m|} + \frac{|S_P^l \cap S_P^m|}{2.|S_P^m|}\right).$$

Rozinat and van der Aalst (2008) also report a solution to two non-trivial problems that are encountered when replaying Petri nets with silent steps and duplicate activities. In the presence of silent steps (or invisible tasks) it is non-trivial to determine whether there exist a suitable firing sequence of invisible tasks such that the right activities become enabled for the Petri net to optimally replay a given sequence of events. Likewise, in the presence of multiple enabled duplicate activities, it is a non-trivial problem of determining the optimal firing, as the firing of one duplicate activity affects the ability of the Petri net to replay the remaining events in a given sequence of events. Rozinat and van der Aalst (2008) present two local approaches that are based on heuristics involving the next activity event in the given sequence of events.

Fitness f and behavioral appropriateness $a_B^{'}$ are particularly useful measures to evaluate the performance of a discovered process model. Moreover, these metrics have been implemented in the ProM framework. However, the interpretation of the fitness measure requires some attention: although it accounts for recall as it punishes for the number of missing tokens that had to be created, it also punishes for the number of tokens that remain in a Petri net after log replay. The latter can be considered *extra* behavior. Therefore, the fitness metric f also has a specificity semantics attached to it. Furthermore, it is to be noted that the behavioral appropriateness $a_B^{'}$ metric is not guaranteed to account for all non-local behavior in the event log (for instance, a non-local, non-free, history-dependent choice construct that is part of a loop will not be detected by the measure). In addition, the $a_B^{'}$ metric requires an analysis of the state space of the process model or a more or less exhaustive simulation to consider all allowable sequences by the model.

5.2 New Metrics

The availability of an event log supplemented with artificial negative events, allows for the definition of a new specificity metric that does not require a state space analysis. Instead, specificity can be calculated by parsing the (grouped) sequences, supplemented with negative events. We therefore define:

- **behavioral recall r_B^p**: The behavioral recall r_B^p metric is obtained by parsing each grouped event sequence. The values for *TP* and *FN* are initially zero. Starting from the initial marking, each sequence is parsed. Whenever an enabled transitions fires, the value for *TP* is increased by one. Whenever a transition is not enabled, but must be forced to fire the value for *FN* is increased. As an optimization, identical sequences are only replayed once. Let k represent the number of grouped sequences, n_i the number of process instances, TP_i number of events that are correctly parsed, and FN_i the number events for which a transition was forced to fire for each grouped sequence i $(1 \leq i \leq k)$. At the end of the sequence replay, r_B^p is obtained as follows:

$$r_B^p = \left(\frac{\sum_{i=1}^{k} n_i TP_i}{\sum_{i=1}^{k} n_i TP_i + \sum_{i=1}^{k} n_i FN_i} \right).$$

  In the case of multiple enabled duplicate transitions, sequence replay fires the transition of which the succeeding transition is the next positive event (or makes a random choice). In the case of multiple enabled silent transitions, log replay fires the transition of which the succeeding transition is the next positive event (or makes a random choice). Unlike the fitness metric f, r_B^p does not punish for remaining tokens. Whenever after replaying a sequence tokens remaining tokens cause *additional behavior* by enabling particular transitions, this is punished by our behavioral specificity metric s_B^n.

- **behavioral specificity s_B^n**: The behavioral specificity s_B^n metric can be obtained during the same replay as r_B^p. The values for *TN* and *TP* are initially zero. Whenever during replay, a negative event is encountered for which no transitions are enabled, the value for *TN* is increased by one. In contrast, whenever a negative event is encountered during sequence replay for which there is a corresponding transition enabled in the Petri net, the value for *FP* is increased by one. As an optimization, identical sequences only are to be replayed once. Let k represent the number of grouped sequences, n_i the number of process instances, TN_i number of negative events for which no transition was enabled, and FP_i the number negative events for which a transition was enabled during the replay of each grouped sequence i $(1 \leq i \leq k)$. At the end of the sequence replay, s_B^n is obtained as follows:

$$s_B^n = \left(\frac{\sum_{i=1}^{k} n_i TN_i}{\sum_{i=1}^{k} n_i TN_i + \sum_{i=1}^{k} n_i FP_i} \right).$$

Because the behavioral specificity metric s_B^n checks whether the Petri net recalls negative event, it is inherently dependent on the way in which these negative events are generated. For the moment, the negative event generation procedure is configurable by the negative event injection probability π, and whether or not it must account for the parallel variants of the given sequences of positive events. For the purpose of uniformity, negative events are generated in the test sets with π equal to 1.0 and account for parallel variants = true.

6. Experimental Evaluation

AGNEs has been implemented in Prolog. In particular, the frequent temporal constraint induction, the artificial negative event generation, the language bias constraints, and the pruning and graph construction algorithms all have been written in Prolog. As mentioned before, AGNEs makes use of TILDE, an existing multi-relational classifier (Blockeel and De Raedt, 1998), available in the ACE-ilProlog data mining system (Blockeel et al., 2002). To be able to benefit from the facilities of the ProM framework, a plugin was written that makes AGNEs accessible in ProM.[1] Figure 3 depicts a screen shot of AGNEs in ProM.

In this section the results of an experimental evaluation of AGNEs are presented. First we will discuss the properties of the event logs and the parameter settings that have been used. Then, in Section 6.1, we analyse the expressiveness of AGNEs and benchmark the ability of AGNEs to generalize from incomplete event logs. In Section 6.2, we analyze the results of a number of noise experiments with different types and levels of noise that have been carried out to test how the learning algorithm behaves in the presence of noise.

In order to evaluate and compare the performance of AGNEs, a benchmark experiment with 34 event logs has been set up. These event logs have previously been used by Alves de Medeiros (2006) and Alves de Medeiros et al. (2007) to evaluate the genetic miner algorithm. Table 1 describes the properties of the underlying artificial process models of the event logs. The number of different process instance sequences (column "$\neq$ process inst.") gives an indication of the amount of different behavior that is present in the event log. This number is to be compared with the total number of process instances in the event logs. In general, the presence of loops and parallelism exponentially increases the amount of different behavior that can be produced by a process. Therefore, the number of activity types that are pairwise parallel and the number and type of loops have been reported in Table 1. In correspondence with the naming conventions used by Alves de Medeiros, nfc stands for

1. The AGNEs plugin is available from `http://www.processintelligence.be/AGNEs.php`.

non-free-choice, l1l and l2l stands for the presence of a length-one and length-two loop respectively, and st and unst stands for structured and unstructured loops. Furthermore, the presences of special structures such as skip activities and (parallel or serial) duplicate activities have been indicated. For most of the event logs in the experiment, a reference model was available that can be assumed to represent the behavior in the event log. Columns "r_B^p reference model" and "s_B^n reference model" indicate the behavioral recall and specificity of the reference models with respect to the original event logs.

In the experiments, the performance of AGNEs is compared to the performance of four state-of-the-art process discovery algorithms: α^+ (van der Aalst et al., 2004; Alves de Medeiros et al., 2004), α^{++} (Wen et al., 2007), genetic miner (Alves de Medeiros et al., 2007) and heuristics miner (Weijters et al., 2006). Being the first large-scale, comparative benchmark study in the literature of process discovery, we have chosen to include algorithms that already have appeared as journal publications (α^+, α^{++}, and genetic miner) or that are much referenced in the literature (heuristics miner).

During all experiments, the algorithms were run with the same, standard parameter settings of their ProM 4.2 implementation, as reported in Table 2. These parameter settings coincide with the ones used to run similar experiments by the authors of the algorithms. To enable a comparison on the same terms, AGNEs was not provided with prior knowledge regarding parallelism or locality of activity types. In particular, the thresholds used to induce frequent temporal patterns have been given the following values ($t_{absence} = 0.9$, $t_{chain} = 0.08$, $t_{succ} = 0.8$, $t_{ordering} = 0.08$, $t_{triple} = 0.10$). In practice, a good threshold depends on the amount of low-frequent behavior (noise) one is willing to accept within the discovered process model. The negative event injection probability π influences the proportion of artificially generated negative events in an event log. A strong imbalance of this proportion may bias a classification learner towards a majority class prediction, without deriving any useful preconditions for a particular activity type. As a rule of thumb, it is a good idea to set this parameter value as low as possible, without the learner making a majority class prediction. In the experiments π has been given a default value of 0.08. Ex-post, AGNEs warns the user when too low a value for π has led to a majority-class prediction. TILDE's C4.5 gain metric was used as a heuristic for selecting the best branching criterion. In addition, TILDE's C4.5 post pruning method was used with a standard confidence level of 0.25. Furthermore, TILDE is forced to stop node splitting when the number of process instances in a tree node drops below 5. The same parameter settings have been used on all 34 data sets. Empirical validation has shown the parameter settings to work well across all data sets.

The AGNEs technique, has run times in between 20 seconds and 2 hours for the data sets in the experiments on a Pentium 4, 2.4 Ghz machine with 1GB internal memory. These processing times are well in excess of the processing times of α^+, α^{++} and heuristics miner. In comparison to the run times of the genetic miner algorithm, processing times are considerably shorter. Most of the time is required by TILDE to learn the preconditions for each activity type. The generation of negative events also can take up some time. As process discovery generally is not a real-time data mining application, less attention has been given to computation times.

6.1 Zero-noise, Cross Validation Experiment

To evaluate AGNEs' expressiveness and ability to generalize, a 10-fold cross-validation experiment has been set up. In the literature on process discovery, cross-validation has only been considered by

	activity types	≠ process inst.	process inst.	r_B^p reference model	s_B^n reference model	$\|\|$ activity types	loops	skip	nfc	duplicate
a10skip	12	6	300	1.000	1.000	1		1		
a12	14	5	300	1.000	1.000	2				
a5	7	13	300	1.000	1.000	1	1 l1l			
a6nfc	8	3	300	1.000	1.000	1			1	
a7	9	14	300	1.000	1.000	4				
a8	10	4	300	1.000	1.000	1				
al1	9	98	300	1.000	0.996	n.a.	1 unst			
al2	13	92	300	1.000	0.992	n.a.	2 unst			
betaSimplified	13	4	300	1.000	1.000	0		1	1	2
bn1	42	4	300	1.000	1.000	0				
bn2	42	25	300	1.000	1.000	0	1 st			
bn3	42	150	300	1.000	0.999	0	2 st			
choice	12	16	300	1.000	1.000	0				
DriversLicense	9	2	300	1.000	1.000	0				
DriversLincensel	11	87	350	1.000	0.986	1	1 st	1	1	1
herbstFig3p4	12	32	300	1.000	0.999	3	1 st			
herbstFig5p19	8	6	300	1.000	1.000	1				1
herbstFig6p18	7	153	300	1.000	0.977	0	1 l1l, 1 l2l			
herbstFig6p19	5	136	300	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.
herbstFig6p31	9	4	300	1.000	1.000	0				1
herbstFig6p33	10	4	300	1.000	1.000	0				1
herbstFig6p36	12	2	300	1.000	1.000	0			1	
herbstFig6p37	16	135	300	1.000	0.996	36				
herbstFig6p38	7	5	300	1.000	1.000	3				1 par.
herbstFig6p39	7	12	300	1.000	1.000	1				
herbstFig6p41	16	12	300	1.000	1.000	4				
herbstFig6p45	8	12	300	1.000	1.000	5				
l1l	6	69	300	1.000	0.988	1	2 l1l			
l1lSkip	6	269	300	1.000	0.732	0	2 l1l			
l2l	6	10	300	1.000	1.000	0	1 l2l			
l2lOptional	6	9	300	1.000	1.000	0	1 l2l			
l2lSkip	6	8	300	1.000	0.999	0	1 l2l			
parallel5	10	109	300	1.000	1.000	10				
repair2	8	48	1000	0.998	0.995	2	1 unst			

Table 1: Event log properties

Goedertier et al. (2008) and Rozinat et al. (2007). The reason for the absence of cross-validation experiments, is that process discovery is an inherently *descriptive* learning task rather than a *predictive* one. The primary intent of process discovery is to produce a model that accurately describes the event log at hand. Nonetheless, it is interesting to test the *predictive* ability of process discovery algorithms in an experimental setting. To apply cross-validation, a randomization routine has been

Algorithm (Ref.)	**Parameter settings**	
	derive succession from partial order information = true	
α^{+}	enforce causal dependencies within events of the same activity = false	
(Alves de Medeiros et al., 2004)	enforce parallelism by overlapping events = false	
α^{++}	(no settings)	
(Wen et al., 2007)		
	relative to best threshold = 0.05	
	positive observations = 10	
	dependency threshold = 0.9	
	length-one-loops threshold = 0.9	
	length-two-loops threshold = 0.9	
heuristics miner	long-distance threshold = 0.9	
Weijters et al. (2006)	dependency divisor = 1	
	and threshold = 0.1	
	use all-activities-connected heuristic = true	
	use long-distance dependency heuristic = false	
	population size = 100	
	max number generations = 1000	
	initial population type = possible duplicates	
	power value = 1	
genetic miner	elitism rate = 0.2	
(Alves de Medeiros et al., 2007)	selection type = tournament 5	
	extra behavior punishment with $\kappa = 0.025$	
	enhanced crossover type with crossover probability = 0.8	
	enhanced mutation type with mutation probability = 0.2	
	prior knowledge	none
	temporal constraints	$t_{absence} = 0.9, t_{chain} = 0.08, t_{succ} = 0.8,$
		$t_{ordering} = 0.08, t_{triple} = 0.1$
	negative event generation	injection probability $\pi = 0.08$
		calculate parallel variants = true
		include global sequences = true
AGNEs	language bias:	include occurrence count = false
		data conditions = none
	TILDE	splitting heuristic: gain
		minimal cases in tree nodes = 5
		C4.5 pruning with confidence level = 0.25
	graph construction	$t_{connect} = 0.4$

Table 2: Parameter settings

written in SWI-Prolog that groups similar sequences, randomly partitions the grouped event log in $n = 10$ uniform subgroups, and produces n pairs of training and test event logs. Training event logs are used for the purpose of process discovery. Test event logs are used for evaluation, this is for calculating the specificity and recall metrics. For the purpose of this experiment, no noise was added to the event logs.

Specificity metrics must be calculated based on the combination of training and test event logs, the entire event log. Although this might seem unintuitive, specificity and specificity metrics make a completeness assumption as well, as they account for the amount of *extra* behavior in a process model vis-à-vis the event log (Rozinat and van der Aalst, 2008). To correctly evaluate the proposed learning technique, it is important that the negative events in the test set accurately indicate the state transitions that are not present in the event log. For this reason, the negative events in the test log are created with information from the entire event log. Should the negative event generation be based on training set instances only, it is possible that additional, erroneous negative events are injected because it is possible that some behavior is not present in the test set. In short, the experiment applies the above-described partitioning, after having generated negative events for each grouped process instance. Intended to be used by the evaluation metric, the negative events have been generated with an injection probability π equal to 1, an infinite window size, and by considering parallel variants. Evidently, the thus generated negative events were not retained in the training set. For training purposes, negative events have been calculated based on the information in the training set only. For the same reasons, the behavioral appropriateness metric a'_B has also been calculated based on the whole of training and test set data.

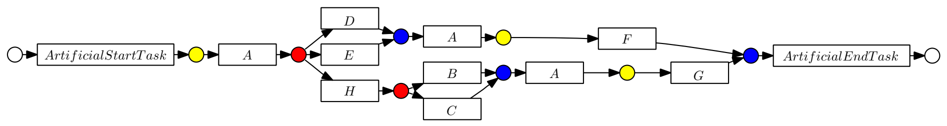
In the experiment, only 19 out of the 34 event logs were retained, as the other event logs have less than 10 different sequences. Table 3 shows the aggregated, average results of the 10-fold cross validation experiment over 190 event logs. The **best** average performance over the 34 event logs is underlined and denoted in bold face for each metric. We then use a paired t-test to test the significance of the performance differences (Van Gestel et al., 2004). Performances that are **not significantly different at the 5% level** from the top-ranking performance with respect to a one-tailed paired t-test are tabulated in bold face. Statistically *significant underperformances at the 1% level* are emphasized in italics. Performances significantly different at the 5% level but not at the 1% level are reported in normal font. For the *PM* measure, no paired t-tests could be performed, because the metric could not be calculated on some of the process models discovered by α^+ and α^{++}. The latter is the case when the discovered process models have disconnected elements.

From the results for the parsing measure *PM*, the fitness measure f, and behavioral recall measures r_B^p, it can be concluded that genetic miner scores slightly better on the recall requirement. Moreover, the behavioral specificity metric s_B^n shows genetic miner and heuristics miner to produce slightly more specific models.

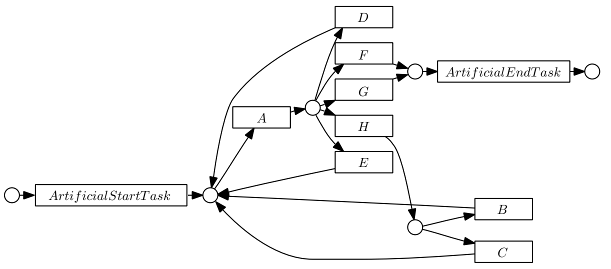
		PM	f	a'_B	acc	r_B^p	s_B^n	acc_B^{pn}
zero_noise	α^+	0.72	*0.96*	**0.96**	**0.96**	*0.97*	*0.83*	*0.90*
	α^{++}	0.77	*0.97*	*0.81*	*0.88*	0.97	*0.90*	0.93
	AGNEs	0.80	0.98	*0.81*	*0.89*	**0.98**	*0.91*	0.94
	genetic	0.83	**0.99**	*0.84*	*0.91*	**0.98**	**0.93**	**0.95**
	heuristics	0.79	*0.97*	*0.85*	*0.91*	0.97	**0.93**	**0.95**

Table 3: 10-fold cross validation experiment - aggregated results

From the cross-validation experiment, we conclude that AGNEs portrays similar generalization behavior to other process discovery algorithms. The reason that it is not sensitive to incomplete event logs can be attributed to the following. Given an incomplete event log, AGNEs is likely to generate a proportion of incorrect negative events. However, this proportion of negative events is relatively small, as the negative event injection parameter π is not required to be excessively large. More importantly, the coarse-grained language bias that combines *NS* constructs into larger structures, prevents TILDE from overfitting the incomplete event log and allows it to generalize, to some extent, beyond the observed behavior. The additional incorporation of process knowledge expressed by a domain expert would only add to this benefit. Finally, the negative event injection procedure takes into account parallelism and window size. Concurrent and recurrent behavior are the root causes of incomplete event logs. The ability to include information about parallel variants and window size, gives our learning technique a configurable inductive bias, with different strategies to account for incompleteness.



(a) AGNEs result



(b) heuristics miner result

Figure 2: herbstFig6p33: AGNEs detects the duplicate activity *A*

The metrics r_B^p and s_B^n do not indicate a large or significant difference for the performance of α^{++}, AGNEs, genetic miner, and heuristics miner. Only by looking at the individual process models, the expressiveness of AGNEs with respect to the detection of non-local, non-free choice constructs or the discovery of duplicate tasks becomes apparent. Example 3, discussed in Section 4.3, shows how AGNEs is capable of detecting non-local, non-free choice constructs, even within the loop of the DriversLicensel reference problem. AGNEs is also particularly suited for the detection of duplicate activities. In the herbstFig6p33 event log, the activity *A* occurs in three different contexts and AGNEs draws three different, identically labeled transitions correspondingly. Figure 2 compares the results of heuristics miner and AGNEs on this event log.

The goal of process discovery is to give an idea of how the processes recorded in the event log *actually* have taken place. This goal makes process discovery an inherently *descriptive* learning task. To evaluate the accuracy of the discovered process model, it is therefore justified to compare the learned process models on the same sequence the process models are learned from. In the process discovery literature, this training-log-based evaluation has been the dominant evaluation paradigm (Alves de Medeiros et al., 2007; Weijters et al., 2006). Consequently, the remaining experiments of this paper use training-log-based evaluation.

6.2 Training-log-based Noise Experiment

In these experiments we have stirred up the 34 event logs with artificial noise. In the literature, six artificial noise types have been described (Maruster, 2003; Alves de Medeiros et al., 2007): (1) *missing head*: the removal of the head of a sequence, (2) *missing body*: the removal of the mid section of a sequence, (3) *missing tail*: the removal of the end section of a sequence, (4) *swap tasks*: the interchange of two random events in a sequence, (5) *remove task*: the removal of a random event in a sequence, and (6) *mix all*: a combination of all of the above. The noise has been added with the AddNoiseLogFilter event log filter available in the ProM framework. This filter has been applied after ungrouping the 34 event logs. To keep the size of the experiment under control, we have limited the noise types used in our experiments to *mix all* and *swap tasks*. For both noise types, the used noise levels of 5%, 10%, 20% and 50% are applied.

Table 4 reports the average results of the discovered process models over the all 34 corresponding zero noise event logs. As is known from the literature, heuristics miner is resilient to noise, whereas the formal approaches of α^+ and α^{++} and the genetic miner are known to overfit the noise in event logs. On 11 event logs from the bn1, bn2, and bn3 processes, the α^{++} implementation was incapable of producing an outcome. These missing values resulted in a score of 0 for each measure. Furthermore, the state space analysis required to calculate the behavioral appropriateness measure $a_B^{'}$ produces invalid outcomes that occur 27 times for the results of the genetic miner algorithm out of a total of 272 (34 x 8) experiments, the reported results for the genetic miner are less suitable for comparison. For this reason, we only indicate the significance of the r_B^p and s_B^n outcomes with respect to the top ranking performance. For the PM, f, and $a_B^{'}$ metrics, the algorithm that has obtained the <u>best</u> average score, is underlined. For every noise level, AGNEs obtains accuracy results that are robust and not significantly different from the results obtained by heuristics miner. This is a remarkable result, as AGNEs is a more expressive algorithm than heuristics miner, also capable of detecting more complex structures such as non-local dependencies and duplicate activities.

To calibrate the metrics, we also report their evaluation of the so-called flower model for the zero-noise case. Because the flower model parses every possible sequence, it has a perfect recall but zero specificity. These properties are to some extent reflected in the metrics in Table 4. The fitness measure f and the behavioral recall measure r_B^p are both 1.0, whereas the behavioral specificity metric s_B^n amounts to 0. The behavioral appropriateness measure $a_B^{'}$ does not really seem to quantify the lack of specificity of the flower model.

The reasons why AGNEs is robust to noise can be put down to the following. First of all, the generation of negative events is not invalidated by the presence of noise. Noise is *additional* low-frequent behavior that will result in less negative events being generated by AGNEs. However, the presence of noisy positive events does not lead to the generation of noisy negative ones. Another property that adds to robustness, is that the constraints in AGNEs ’ language bias allows it to come up with structured patterns and to some extent prevents the construction of arbitrary connections between transitions, while remaining expressive with regard to short loops, duplicate tasks and non-local behavior. Finally, the formulation of process discovery as a classification allows for the application of an already robust classification algorithm (TILDE). Like many classification learners, TILDE takes into account the frequency of an anomaly, when constructing the preconditions for each activity type. Moreover, TILDE applies the same tree-level pruning method as C4.5 (Quinlan, 1993).

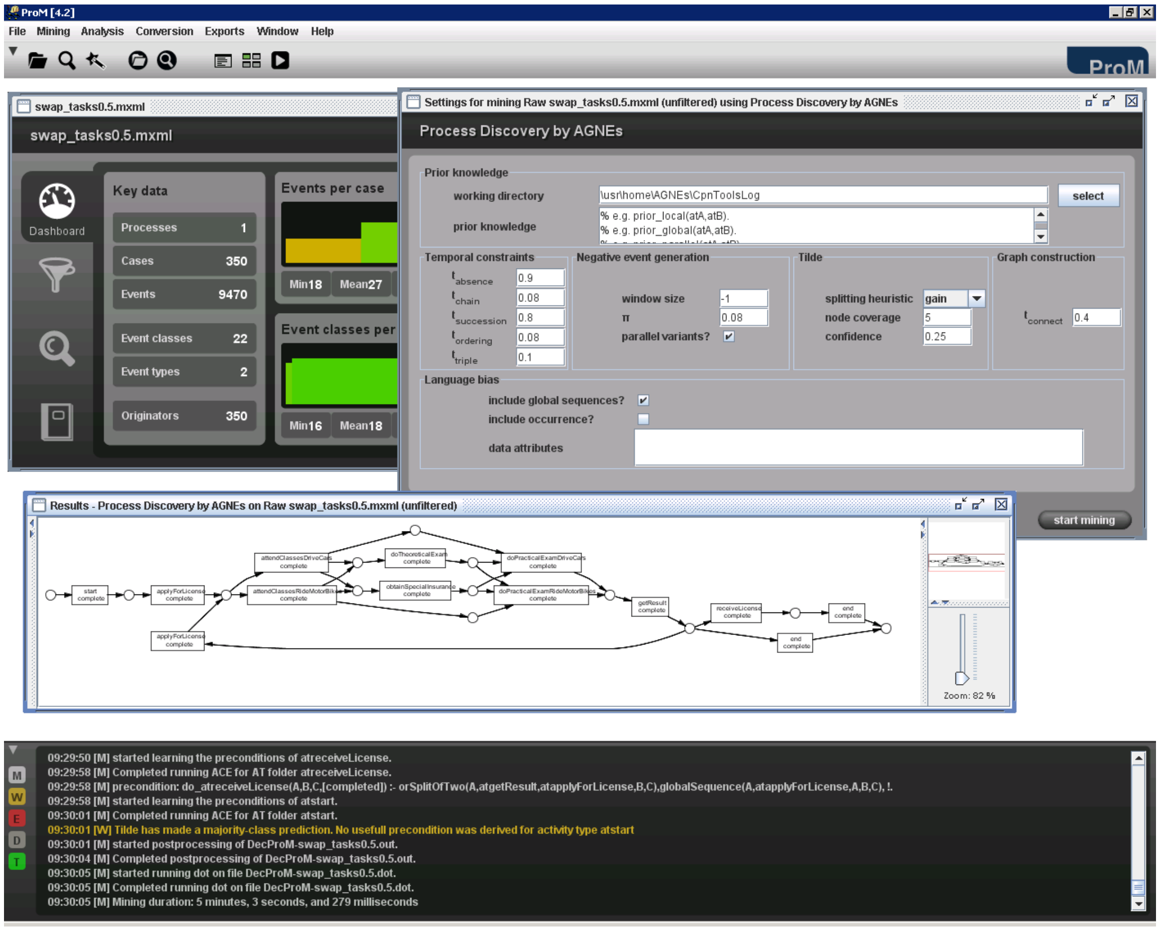
7. A Case Study

This section shows the result of the AGNEs process discovery algorithm applied to an event log of customer-initiated processes, recorded by a European telecom provider. The goal of the case study was to investigate whether AGNEs can be usefully applied to map the routing choices that are made between queues of a workflow management system (WfMS). With 18721 process instances and 127 queues, the obtained log file has a size of over 130 megabytes in the form of a comma-separated text file. The case study gives an idea of the scalability of the algorithm towards large event logs and the usefulness of the AGNEs process discovery algorithm on realistic, real-life processes.

		PM	f	$a_B^{'}$	acc	r_B^p	s_B^n	acc_B^{pn}
zero-noise	α^+	0.72	*0.96*	**0.92**	**0.94**	*0.96*	*0.85*	*0.91*
	α^{++}	0.82	**0.98**	**0.87**	**0.92**	**0.98**	**0.93**	**0.96**
	AGNEs	0.90	**0.99**	**0.87**	**0.93**	**0.99**	**0.96**	**0.98**
	genetic	0.91	**1.00**	*0.83*	**0.91**	**0.99**	**0.95**	**0.97**
	heuristics	0.88	**0.99**	**0.86**	**0.92**	**0.99**	**0.95**	**0.97**
	flower	1.00	1.00	0.23	0.61	1.00	0.00	0.50
mix_all0.05	α^+	0.11	0.83	0.85	0.84	*0.87*	*0.62*	*0.74*
	α^{++}	0.00	0.79	0.65	0.72	*0.75*	*0.63*	*0.69*
	AGNEs	0.89	0.99	0.87	0.93	**0.99**	**0.95**	**0.97**
	genetic	0.74	0.99	0.63	0.81	**0.98**	**0.91**	**0.95**
	heuristics	0.88	0.98	0.87	0.93	**0.98**	**0.94**	**0.96**
mix_all0.1	α^+	0.08	0.80	0.84	0.82	*0.84*	*0.59*	*0.72*
	α^{++}	0.00	0.73	0.80	0.76	*0.64*	*0.64*	*0.64*
	AGNEs	0.83	0.99	0.89	0.94	**0.99**	**0.96**	**0.97**
	genetic	0.51	0.97	0.59	0.78	*0.94*	*0.78*	*0.86*
	heuristics	0.88	0.99	0.86	0.92	**0.99**	**0.95**	**0.97**
mix_all0.2	α^+	0.00	0.77	0.91	0.84	*0.82*	*0.51*	*0.67*
	α^{++}	0.00	0.65	0.65	0.65	*0.49*	*0.63*	*0.55*
	AGNEs	0.79	0.97	0.87	0.92	**0.97**	**0.94**	**0.96**
	genetic	0.47	0.96	0.53	0.74	*0.93*	*0.73*	*0.83*
	heuristics	0.86	0.98	0.85	0.92	**0.98**	**0.94**	**0.96**
mix_all0.5	α^+	0.00	0.63	0.75	0.69	*0.67*	*0.46*	*0.56*
	α^{++}	0.00	0.51	0.61	0.58	*0.26*	*0.70*	*0.48*
	AGNEs	0.54	0.96	0.77	0.87	**0.97**	**0.90**	**0.93**
	genetic	0.20	0.95	0.43	0.69	*0.86*	*0.53*	*0.69*
	heuristics	0.66	0.97	0.74	0.85	**0.96**	**0.88**	**0.92**
swap_tasks0.05	α^+	0.00	0.65	0.85	0.75	*0.76*	*0.45*	*0.60*
	α^{++}	0.00	0.59	0.67	0.63	*0.52*	*0.61*	*0.56*
	AGNEs	0.90	0.99	0.87	0.93	**0.99**	**0.96**	**0.97**
	genetic	0.44	0.95	0.61	0.78	*0.90*	*0.74*	*0.82*
	heuristics	0.88	0.99	0.85	0.92	**0.99**	**0.95**	**0.97**
swap_tasks0.1	α^+	0.00	0.58	0.86	0.72	*0.69*	*0.48*	*0.58*
	α^{++}	0.00	0.53	0.66	0.59	*0.38*	*0.61*	*0.49*
	AGNEs	0.78	0.98	0.87	0.93	**0.98**	**0.94**	**0.96**
	genetic	0.38	0.94	0.53	0.74	*0.89*	*0.65*	*0.77*
	heuristics	0.80	0.97	0.86	0.92	**0.98**	**0.94**	**0.96**
swap_tasks0.2	α^+	0.00	0.54	0.77	0.66	*0.59*	*0.52*	*0.55*
	α^{++}	0.00	0.45	0.65	0.55	*0.27*	*0.67*	*0.47*
	AGNEs	0.73	0.97	0.86	0.92	**0.98**	**0.93**	**0.95**
	genetic	0.19	0.93	0.62	0.77	*0.84*	*0.52*	*0.68*
	heuristics	0.69	0.96	0.87	0.92	**0.96**	**0.88**	**0.92**
swap_tasks0.5	α^+	0.00	0.41	0.61	0.51	*0.40*	*0.63*	*0.51*
	α^{++}	0.00	0.36	0.61	0.48	*0.16*	**0.77**	*0.46*
	AGNEs	0.32	0.91	0.72	0.81	**0.95**	**0.82**	**0.89**
	genetic	0.07	0.93	0.77	0.85	*0.79*	*0.40*	*0.59*
	heuristics	0.45	0.94	0.66	0.80	**0.94**	**0.83**	**0.89**

Table 4: noise experiments - average, zero-noise training-log-based results

The event log consists of events about customer-initiated processes that are handled at three different locations by the employees of the telecom provider. The handling of cases is organized in a first line and a second line. First-line operators are junior operators that deal with frequent customer requests for which standardized procedures have been put in place. When a first-line operator cannot process a case, it is routed to a queue of the second line. Second-line case handling is operated by senior experts who have the authority to make decisions to solve the more involved cases. The


ProM [4.2]
File Mining Analysis Conversion Exports Window Help
swap_tasks0.5.mxml
Dashboard
Events per case
Min 18 Mean 27
Event classes per
Min 16 Mean 18
Results - Process Discovery by AGNEs on Raw swap_tasks0.5.mxml (unfiltered)
start complete
applyForLicense complete
attendClassesDriveCars complete
attendClassesRideMotorBikes complete
doTheoreticalExam complete
obtainSpecialInsurance complete
doPracticalExamDriveCars complete
doPracticalExamRideMotorBikes complete
getResult complete
receiveLicense complete
end complete
Zoom: 82 %
start mining

Key data

Processes	1
Cases	350
Events	9470
Event classes	22
Event types	2
Originators	350

Settings for mining Raw swap_tasks0.5.mxml (unfiltered) using Process Discovery by AGNEs

Process Discovery by AGNEs

Prior knowledge

working directory	\usr\home\AGNEs\CpnToolsLog
prior knowledge	% e.g. prior_local(atA,atB). % e.g. prior_global(atA,atB).

Temporal constraints

$t_{absence}$	0.9
t_{chain}	0.08
$t_{succession}$	0.8
$t_{ordering}$	0.08
t_{triple}	0.1

Negative event generation

window size	-1
π	0.08
parallel variants?	✓

Tilde

splitting heuristic	gain
node coverage	5
confidence	0.25

Graph construction

$t_{connect}$	0.4

Language bias

include global sequences?	✓
include occurrence?	
data attributes	

```
09:29:50 [M] started learning the preconditions of atreceiveLicense.
09:29:58 [M] Completed running ACE for AT folder atreceiveLicense.
09:29:58 [M] precondition: do_atreceiveLicense(A,B,C,[completed]) :- orSplitOfTwo(A,atgetResult,atapplyForLicense,B,C),globalSequence(A,atapplyForLicense,A,B,C), !.
09:29:58 [M] started learning the preconditions of atstart.
09:30:01 [M] Completed running ACE for AT folder atstart.
09:30:01 [W] Tilde has made a majority-class prediction. No usefull precondition was derived for activity type atstart
09:30:01 [M] started postprocessing of DecProM-swap_tasks0.5.out.
09:30:04 [M] Completed postprocessing of DecProM-swap_tasks0.5.out.
09:30:05 [M] started running dot on file DecProM-swap_tasks0.5.dot.
09:30:05 [M] Completed running dot on file DecProM-swap_tasks0.5.dot.
09:30:05 [M] Mining duration: 5 minutes, 3 seconds, and 279 milliseconds
```

Figure 3: AGNEs in ProM 4.2

second-line processes are coordinated and supported by means of a workflow management system (WfMS). The obtained event log consists of these second-line case handling events. The second-line WfMS is organized as a system of 127 logical queues. Each queue corresponds to a number of similar types of activities that are to be carried out. At any given moment each active case resides in exactly one queue. Employees can process a case by taking it out of the queue into their personal work bin. Every evolution of a case is documented by adding notes. Moreover, employees can classify the nature of the case according to a number of data fields. In addition, a worker or dispatcher has the ability to reroute cases to different queues whenever this is necessary. The system imposes no restrictions with regard to the routing of cases. Queues represent a work distribution system and are akin to roles in WfMS. For the purpose of this analysis, queues are considered to be activity types. The 40 most frequently occurring queues were retained for further analysis. Nine process instances that did not involve at least one of these 40 queues were retained from the event log.

We compare the mining results of AGNEs, genetic miner, and heuristics miner. Some parameter settings that are different from the experimental evaluation in the previous section. In particular, AGNEs was provided with the prior knowledge that no activity can occur concurrently: $\forall a, b \in A$:

	PM	f	$a_B^{'}$	acc	r_B^p	s_B^n	acc_B^{pn}
AGNEs	0.06	0.94	0.67	0.80	0.93	0.67	0.80
genetic	0.03	x.xx	x.xx	x.xx	0.83	0.62	0.73
heuristics	$\underline{0.80}$	$\underline{0.97}$	$\underline{0.72}$	$\underline{0.85}$	$\underline{0.96}$	$\underline{0.88}$	$\underline{0.92}$
flower model	0.00	1.00	0.76	0.88	1.00	0.00	0.50

Table 5: telecom—training-log-based results

$PriorSerial(a,b)$. This prior knowledge is justifiable, as no case can be routed to or reside in several queues at the same time. Genetic miner has been running for 5000 generations, with a population size of 10. These parameter settings correspond to the parameter settings in the case study described by Alves de Medeiros et al. (2007). To account for the prior knowledge that no concurrent behavior is contained in the event log, heuristics miner needs to have an infinite AND threshold. In general AGNEs allows to provide a-priori locality and parallelism information for individual pairs of activity types. This fine-grained a-priori knowledge cannot be provided by fine-tuning the AND threshold with heuristics miner. Currently, it is not possible to constrain the search space of genetic miner with this a-priori knowledge.

The results of applying these process discovery algorithms on the filtered event log are displayed in Figure 4. Table 5 gives an overview of the metrics that compare the discovered process models to the original event log. As the purpose of the case study is to provide the most accurate description of the event log, the use of training data for evaluation is justified. To calibrate the metrics, we also report their evaluation of the so-called flower model. Because the flower model represents random behavior, it has a perfect recall of the all behavior in the event log but it also has much *additional* behavior compared to the event log. Because of the latter fact, the flower model has zero specificity. These properties are to some extent reflected in the metrics in Table 5. The fitness measure f and the behavioral recall measure r_B^p are both 1.0, whereas the behavioral specificity metric s_B^n amounts to 0. The table also indicates the usefulness of the metrics proposed in this paper. The parsing measure PM does not reflect the recall of the flower model. Furthermore, the behavioral appropriateness measure $a_B^{'}$ does not really seem to quantify the lack of specificity of the flower model. For the genetic miner mining result, the ProM implementations of f, and $a_B^{'}$ did not produce an outcome. To calculate these metrics, a conversion of the heuristics nets into Petri nets is required. The resulting Petri nets, which have many invisible transitions, are seemingly too complex to calculate the metric. These results are an indication of the usefulness of the new specificity metric proposed in this paper.

Comparing the available r_B^p and s_B^n outcomes, it can be observed that AGNEs performs better than genetic miner, but worse than heuristics miner on the obtained event log. The discovered process model by AGNEs has an accuracy of 80%, whereas the models discovered by genetic miner and heuristics miner have an accuracy of 73% and 92% respectively. This case study brings forward that human-centric processes contained in the event log can take place in a less structured fashion than often is assumed by process discovery algorithms. In this particular case, for instance, it seems that OR-splits and OR-joins can involve a rather high number of outgoing or incoming activities. In the current implementation, the language bias of TILDE is limited to conjunctions and disjunctions of *NS* constructs of length two and three. This imposes limitations on the hypotheses that can be learned by AGNEs. As indicated in Section 4.3, solving this language bias issue, requires construct-

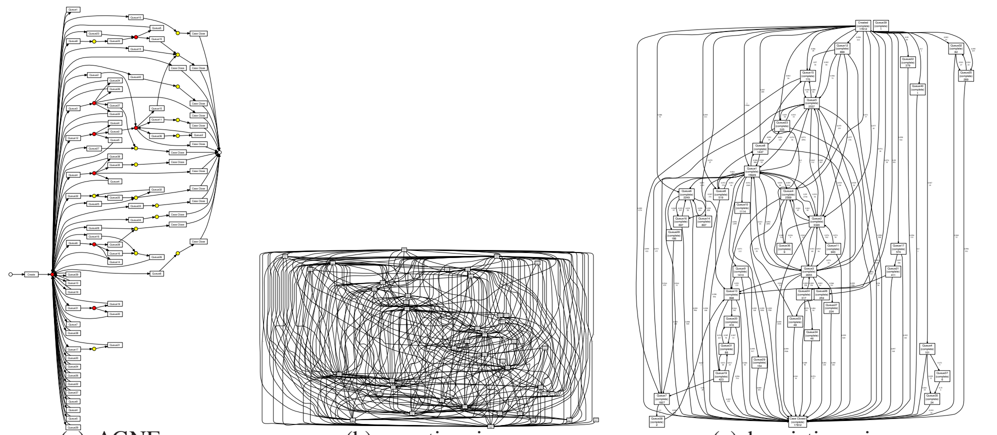

(a) AGNEs (b) genetic miner (c) heuristics miner

Figure 4: telecom—mining results

ing a proprietary ILP classification algorithm that during each refinement step allows considering conjunctions of *NS* constructs of variable lengths. In this regard, the language bias of Heuristics Miner seems to be less limiting, although Heuristics Miner does not have many of the declarative properties of AGNEs. Another outcome of the case study, is that the proposed measures for behavioral recall r_B^p and behavioral specificity s_B^n in practice turn out to be valuable metrics for assessing the accuracy of a discovered process model.

8. Related Work

Process discovery can be seen as an application of the machine learning of grammars from positive data, of which Angluin and Smith (1983) provide an overview. Gold (1967) has shown that important classes of recursively enumerable languages cannot be *identified in the limit* from a finite number of positive examples only. Instead, both positive and negative examples are required for grammar learning to distinguish the right hypothesis among an infinite number of grammars that fit the positive examples. Whereas Gold's negative learnability result applies to the learning of grammars with perfect accuracy, process discovery is more concerned with the ability to discover process models that have *only* a good recall and specificity. Learning grammars from only positive examples requires a tradeoff between overly general and overly specific hypotheses. Muggleton (1996) shows that in a Bayesian framework, logic programs are learnable with arbitrarily low error from positive examples only. Bayes' theorem allows to formulate this tradeoff as a tradeoff between size and generality of the hypotheses and learning can be considered to maximization the posterior probability over all hypotheses. In this paper, a new approach for making the tradeoff between generality and specificity is proposed, by inducing artificial negative events using a (highly configurable) assumption about the completeness of the behavior displayed by the positive examples in the event log. Another difference with grammar learning is that in grammar learning, the hypothesis space is often expressed as production rules, automata or regular expressions, whereas process discovery uses formalisms that can represent the concurrency and synchronization concerns of processes more elegantly.

Process models are in general deterministic. In the literature, there are many formalisms to represent and learn the probability distributions of *stochastic*, generative grammars over sequences of observed characters and unobserved state variables. Historically, techniques like Markov models, hidden Markov models, factorial hidden Markov models, and dynamic Bayesian networks have been first applied to speech recognition, and bio-informatics (Durbin et al., 1998). Each representation has its own particular modeling features that makes it more or less suited for representing the human-centric behavior of business processes. Factorial hidden Markov models, for instance, have a distributed state representation that allows for the modeling of concurrent behavior (Ghahramani and Jordan, 1997). Hidden Markov models have furthermore been provided with a first-order, extension that allows for the representation of sequences of logical atoms rather than alphabets of flat characters (Kersting et al., 2006). Other authors describe learning mixture models to identify meaningful clusters of (hidden) Markov models (Smyth, 1997; Cadez et al., 2003). Whereas stochastic models provide useful information, their probabilistic nature tends to compromise the comprehensibility of discovered process models. Business processes have well-defined start, end, split, and synchronization nodes. The network structure of stochastic models does not visualize this. For instance, although hidden states could be useful in representing duplicate activities—the same activity label is logged in different contexts—a hidden Markov model is unlikely to be capable of comprehensively representing its different usage contexts.

In contrast, Mannila and Meek (2000) describe a technique to learn two-component mixture models of global partial orders that provide an understandable, global view of the sequential data. The authors assume the presence of one dominant, global partial order and consider a generic partial order with random behavior to deal with low-frequent variations (noise) from the former model. Silva et al. (2005) describe a probabilistic model and algorithm for process discovery that discovers so-called and/or graphs in polynomial time. These and/or graphs are comprehensible, directed acyclic graphs that have the advantage over global partial order representations that they can differentiate between parallel and serial split and join points. Pei et al. (2006) describe a scalable technique for discovering the complete set of frequent, closed partial orders from sequential data. The three aforementioned techniques assume each item to occur only once within a sequence, and do not consider recurrent behavior (cycles), nor duplicate activities.

The term process discovery was coined by Cook and Wolf (1998), who apply it in the field of software engineering. Their Markov algorithm can only discover sequential patterns as Markov chains cannot elegantly represent concurrent behavior. The idea of applying process discovery in the context of workflow management systems stems from Agrawal et al. (1998) and Lyytinen et al. (1998). The value of process discovery for the general purpose of process mining (van der Aalst et al., 2007) is well illustrated by the plugins within the ProM framework. In analogy with the WEKA toolset for data mining (Witten and Frank, 2000), the ProM Framework consists of a large number of plugins for the analysis of event logs (Process Mining Group, TU/Eindhoven, 2008). The *Conformance Checker* plugin (Rozinat and van der Aalst, 2008), for instance, allows identifying the discrepancies between an idealized process model and an event log. Moreover, with a model that accurately describes the event log, it becomes possible to use the time-information in an event log for the purpose of performance analysis, using, for instance, the *Performance Analysis with Petri nets* plugin.

Table 6 provides a chronological overview of process discovery algorithms that have been applied to the context of workflow management systems. The α algorithm can be considered to be a theoretical learner for which van der Aalst et al. (2004) prove that it can learn an important class of workflow nets, called structured workflow nets, from complete event logs. The α algorithm assumes event logs to be complete with respect to all allowable binary sequences and assumes that the event log does not contain any noise. Therefore, the α algorithm is sensitive to noise and incompleteness of event logs. Moreover, the original α algorithm was incapable of discovering short loops or non-local, non-free choice constructs. Alves de Medeiros et al. (2004) have extended the α algorithm to mine short loops and called it α^+. Wen et al. (2007) made an extension for detecting implicit dependencies, for detecting non-local, non-free choice constructs. None of the algorithms can detect duplicate activities. The main reason why the α algorithms are sensitive to noise, is that they does not take into account the frequency of binary sequences that occur in event logs. Weijters and van der Aalst (2003) and Weijters et al. (2006) have developed a robust, heuristic-based method for process discovery, called heuristics miner, that is known to be noise resilient. Heuristics miner can discover short loops, and non-local, non-free choice as it can consider non-local dependencies within an event log. However, heuristics miner cannot detect duplicate activities.

Algorithm (Ref.)	**Summary**
global partial orders (Mannila and Meek, 2000)	Learns a two-component mixture model of a dominant series-parallel partial order and a trivial partial order by searching for the dominant partial order that yields the highest probability for generating the observed sequence database.
little thumb, **heuristics miner** (Weijters and van der Aalst, 2003) (Weijters et al., 2006)	Extends the α algorithm by taking into account the frequency of the follows relationship, to calculate dependency/frequency tables from the event log and uses heuristics to convert this information into a heuristics net.
α, α^+ (van der Aalst et al., 2004) (Alves de Medeiros et al., 2004)	Derives a Petri net from local, binary ordering relations detected within an event log.
splitpar—InWoLvE (Herbst and Karagiannis, 2004)	Derives a so-called stochastic activity graph and converts it into a structured process model in the Adonis Modeling language.
multi-phase miner (van Dongen and van der Aalst, 2005b)	Constructs a process model for every sequence in the log and aggregates the model into an event-driven process chain.
α^{++} (Wen et al., 2007)	Extends the α algorithm to discover non-local, non-free choice constructs.
– (Silva et al., 2005)	A probabilistic approach to process discovery.
frecpo (Pei et al., 2006)	A scalable technique for discovering the complete set of frequent, closed partial orders from sequential data.
FSM/Petrify miner (van der Aalst et al., 2006)	Derives a highly configurable finite state machine from the event log and folds the finite state machine into regions using the theory of regions.
– (Ferreira and Ferreira, 2006)	Learns the case data preconditions and effects of activities with ILP classification techniques and user-supplied negative events.
genetic miner (Alves de Medeiros et al., 2007)	A genetic algorithm that selects the more complete and precise heuristics nets over generations of nets.
DecMiner (Lamma et al., 2007)	A classification technique that learns the preconditions of activities with the ICL ILP learner from event logs with user-supplied negative sequences.
fuzzy miner (Günther and van der Aalst, 2007)	An adaptive simplification and visualization technique based on significance and correlation measures to visualize unstructured processes.

Table 6: Chronological overview of process discovery algorithms

van Dongen and van der Aalst (2005b) present a multi-phase approach to process mining that starts from the individual process sequences, constructs so-called instance graphs for each sequence that account for parallelism, and then aggregates these instance graphs according to previously detected binary relationships between activity types. Interestingly, the aggregation ensures that every discovered process model has a perfect recall, but generally scores less on specificity. Herbst and Karagiannis (2004) describe the working of the splitpar algorithm that is part of the InWoLvE framework for process analysis. This algorithm derives a so-called stochastic activity graph and converts it into a structured process model. The splitpar algorithm is capable of detecting duplicate activities, but it is not able to discover non-local dependencies.

Alves de Medeiros et al. (2007) describe a genetic algorithm for process discovery. The fitness function of this genetic algorithm incorporates both a recall and a specificity measure that drives the genetic algorithm towards suitable models. The genetic miner is capable of detecting non-local patterns in the event log and is described to be fairly robust to noise. In her PhD, Alves de Medeiros (2006) describes an extension to this algorithm for the discovery of duplicate tasks.

van der Aalst et al. (2006) present a two-phase approach to process discovery that allows to configure when states or state transitions are considered to be similar. The ability to manipulate similarity is a good approach to deal with incomplete event logs. In particular, several criteria can be considered for defining similarity of behavior and states: the inclusion of future or past events, the maximum horizon, the activities that determine state, whether ordering matters, the activities that visibly can bring about state changes, etcetera. Using these criteria, a configurable finite state machine can be constructed from the event log. In a second phase, the finite state machine is folded into regions using the existing theory of regions (Cortadella et al., 1998). For the moment, the second phase of the algorithm still poses difficulties with respect to constructing suitable process models. The approach presented in this paper considers window size (maximum horizon) and parallel variants as similarity criteria when generating artificial negative events.

Günther and van der Aalst (2007) present an adaptive simplification and visualization technique based on significance and correlation measures to visualize the behavior in event logs at various levels of abstraction. The contribution of this approach is that it can also be applied to less structured, or unstructured processes of which the event logs cannot easily be summarized in concise, structured process models.

Several authors have used classification techniques for the purpose of process discovery. Maruster et al. (2006), for instance, were among the first to investigate the use of rule-induction to predict dependency relationships between activities from a corpus of reference logs that portray various levels of noise and imbalance. To this end, the authors use a propositional rule induction technique, the uni-relational classification learner RIPPER (Cohen, 1995), on a table of direct metrics for each process task in relation to each other process task, which is generated in a pre-processing step.

Ferreira and Ferreira (2006) apply a combination of ILP learning and partial-order planning techniques to process mining. Rather than generating artificial negative events, negative examples are collected from the users who indicate whether a proposed execution plan is feasible or not. By iteratively combining planning and learning, a process model is discovered that is represented in terms of the case data preconditions and effects of its activities. In addition to this new process mining technique, the contribution of this work is in the truly integrated BPM life cycle of process generation, execution, re-planning and learning. Lamma et al. (2007) also describe the use of ILP to process mining. The authors assume the presence of negative sequences to guide the search algorithm. Unlike the approach of Ferreira and Ferreira, who use partial-order planning to present the user with an execution plan to accept or reject (a negative example), this approach does not provide an immediate answer to the origin of the negative events. Contrary to our approach, the latter two approaches are not concerned with the construction of a graphical, control-flow based process model and do not consider the generation of artificial negative events.

9. Conclusion

Process discovery aims at accurately summarizing an event log in a structured process model. So far, the discovery of structured processes by supplementing event logs with *artificial* negative events has not been considered in the literature. The advantage is that it allows representing process discovery as a multi-relational classification problem to which existing classification learners can be applied. In this paper, the generation of artificial negative events gives rise to a new process discovery algorithm and to new metrics for quantifying the recall and specificity of a process model vis-à-vis an event log.

Process discovery algorithms must deal with challenges such as expressiveness, noise, incomplete event logs and the inclusion of prior knowledge. Dealing with one challenge sometimes leads to poor performance with respect to another. The technique presented in this paper, simultaneously addresses many of these challenges. This can be concluded from the results of a large benchmark study applied to AGNEs and four state-of-the art process discovery algorithms. A comparative benchmark study of this scale is the first in the field of process discovery. The benchmark experiments indicate that our technique can discover complex structures such as short loops, duplicate activities, and non-free choice constructs, while remaining robust to noise. In addition, our technique has a new, declarative way of dealing with incomplete event logs that diminishes the effects of concurrent and recurrent behavior on the generation of artificial negative events. Finally, our technique is capable of having prior knowledge constrain the hypothesis space during process discovery. These declarative aspects—the inclusion of prior knowledge, the configurability of the negative event generation procedure and the language bias—potentially make it very useful in practical applications. Another outcome of the benchmark study is the usefulness of the new specificity metric, which in contrast to existing metrics can always be calculated and produces intuitive results.

Acknowledgments

We extend our gratitude to the guest editors, the anonymous reviewers, and the production editor, as their many constructive and detailed remarks certainly contributed much to the quality of this paper. Further, we would like to thank the Flemish Research Council for financial support (FWO postdoctoral research grant, Odysseus grant B.0915.09).

References

Rakesh Agrawal, Dimitrios Gunopulos, and Frank Leymann. Mining process models from workflow logs. In *Proceedings of the 6th International Conference on Extending Database Technology (EDBT'98)*, volume 1377 of *Lecture Notes in Computer Science*, pages 469–483. Springer, 1998.

Ana Karla Alves de Medeiros. *Genetic Process Mining*. PhD thesis, Technische Universiteit Eindhoven, 2006.

Ana Karla Alves de Medeiros, Boudewijn F. van Dongen, Wil M. P. van der Aalst, and Anton J. M. M. Weijters. Process mining: Extending the alpha-algorithm to mine short loops. BETA working paper series 113, Eindhoven University of Technology, 2004.

Ana Karla Alves de Medeiros, Anton J. M. M. Weijters, and Wil M. P. van der Aalst. Genetic process mining: an experimental evaluation. *Data Mining and Knowledge Discovery*, 14(2): 245–304, 2007.

Dana Angluin and Carl H. Smith. Inductive inference: Theory and methods. *ACM Computing Surveys*, 15(3):237–269, 1983.

Hendrik Blockeel. *Top-Down Induction of First-Order Logical Decision Trees*. PhD thesis, Katholieke Universiteit Leuven, 1998.

Hendrik Blockeel and Luc De Raedt. Top-down induction of first-order logical decision trees. *Artificial Intelligence*, 101(1-2):285–297, 1998.

Hendrik Blockeel, Luc Dehaspe, Bart Demoen, Gerda Janssens, Jan Ramon, and Henk Vandecasteele. Improving the efficiency of inductive logic programming through the use of query packs. *Journal of Artificial Intelligence Research*, 16:135–166, 2002.

Igor Cadez, David Heckerman, Christopher Meek, Padhraic Smyth, and Steven White. Model-based clustering and visualization of navigation patterns on a web site. *Data Mining and Knowledge Discovery*, 7(4):399–424, 2003. ISSN 1384-5810.

William W. Cohen. Fast effective rule induction. In *Proceedings of the 12th International Conference on Machine Learning*, pages 115–123, Tahoe City, CA, 1995. Morgan Kaufmann Publishers.

Jonathan E. Cook and Alexander L. Wolf. Discovering models of software processes from event-based data. *ACM Transactions on Software Engineering and Methodology*, 7(3):215–249, 1998.

Jordi Cortadella, Michael Kishinevsky, Luciano Lavagno, and Alexandre Yakovlev. Deriving petri nets from finite transition systems. *IEEE Transactions on Computers*, 47(8):859–882, 1998.

Richard Durbin, Sean R. Eddy, Anders Krogh, and Graeme Mitchison. *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids*. Cambridge university press, 1998.

Sašo Džeroski. Multi-relational data mining: an introduction. *SIGKDD Explorations*, 5(1):1–16, 2003.

Sašo Džeroski and Nada Lavrač. *Inductive Logic Programming: Techniques and Applications*. Ellis Horwood, New York, 1994.

Sašo Džeroski and Nada Lavrač, editors. *Relational Data Mining*. Springer-Verlag, Berlin, 2001.

Richard Dybowski, Kathryn B. Laskey, James W. Myers, and Simon Parsons. Introduction to the special issue on the fusion of domain knowledge with data for decision support. *Journal of Machine Learning Research*, 4:293–294, 2003.

Hugo Ferreira and Diogo R. Ferreira. An integrated life cycle for workflow management based on learning and planning. *International Journal of Cooperative Information Systems*, 15(4):485–505, 2006.

Zoubin Ghahramani and Michael I. Jordan. Factorial hidden markov models. *Machine Learning*, 29(2-3):245–273, 1997.

Stijn Goedertier. *Declarative Techniques for Modeling and Mining Business Processes*. Phd thesis, Katholieke Universiteit Leuven, Faculty of Business and Economics, Leuven, September 2008.

Stijn Goedertier, David Martens, Bart Baesens, Raf Haesen, and Jan Vanthienen. Process mining as first-order classification learning on logs with negative events. In *Proceedings of the 3rd Workshop on Business Processes Intelligence (BPI'07)*, volume 4928 of *Lecture Notes in Computer Science*. Springer, 2008.

E. Mark Gold. Language identification in the limit. *Information and Control*, 10(5):447–474, 1967.

Christian W. Günther and Wil M. P. van der Aalst. Fuzzy mining - adaptive process simplification based on multi-perspective metrics. In *Proceedings of the 5th International Conference on Business Process Management, BPM 2007,* volume 4714 of *Lecture Notes in Computer Science*, pages 328–343. Springer, 2007.

Joachim Herbst and Dimitris Karagiannis. Workflow mining with InWoLvE. *Computers in Industry*, 53(3):245–264, 2004.

Kristian Kersting, Luc De Raedt, and Tapani Raiko. Logical hidden markov models. *Journal of Artificial Intelligence Research*, 25:425–456, 2006.

Evelina Lamma, Paola Mello, Marco Montali, Fabrizio Riguzzi, and Sergio Storari. Inducing declarative logic-based models from labeled traces. In *Proceedings of the 5th International Conference on Business Process Management, BPM 2007,* volume 4714 of *Lecture Notes in Computer Science*, pages 344–359. Springer, 2007.

Kalle Lyytinen, Lars Mathiassen, Janne Ropponen, and Anindya Datta. Automating the discovery of as-is business process models: Probabilistic and algorithmic approaches. *Information Systems Research*, 9(3):275–301, 1998.

Heikki Mannila and Christopher Meek. Global partial orders from sequential data. In *Proceedings of the 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '00)*, pages 161–168, New York, NY, USA, 2000. ACM.

Laura Maruster. *A Machine Learning Approach to Understand Business Processes*. PhD thesis, Eindhoven University of Technology, Eindhoven, 2003.

Laura Maruster, Anton J. M. M. Weijters, Wil M. P. van der Aalst, and Antal van den Bosch. A rule-based approach for process discovery: Dealing with noise and imbalance in process logs. *Data Mining and Knowledge Discovery*, 13(1):67–87, 2006.

Stephen Muggleton. Inductive logic programming. In *Proceedings of the 1st International Conference on Algorithmic Learning Theory*, pages 42–62, 1990.

Stephen Muggleton. Learning from positive data. In Stephen Muggleton, editor, *Inductive Logic Programming Workshop*, volume 1314 of *Lecture Notes in Artificial Intelligence*, pages 358–376. Springer, 1996.

Tadao Murata. Petri nets: Properties, analysis and applications. *Proceedings of the IEEE*, 77(4): 541–580, 1989.

Jian Pei, Haixun Wang, Jian Liu, Ke Wang, Jianyong Wang, and Philip S. Yu. Discovering frequent closed partial orders from strings. *IEEE Transactions on Knowledge and Data Engineering*, 18 (11):1467–1481, 2006.

Process Mining Group, TU/Eindhoven. Process mining web page: research, tools and application. `http://processmining.org`, 2008.

J. Ross Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1993.

Anne Rozinat and Wil M. P. van der Aalst. Conformance checking of processes based on monitoring real behavior. *Information Systems*, 33(1):64–95, 2008.

Anne Rozinat, Ana Karla Alves De Medeiros, Christian W. Günther, Anton J. M. M. Weijters, and Wil M. P. van der Aalst. Towards an evaluation framework for process mining algorithms. BETA working paper series 224, Eindhoven University of Technology, 2007.

Ricardo Silva, Jiji Zhang, and James G. Shanahan. Probabilistic workflow mining. In *KDD '05: Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining*, pages 275–284, New York, NY, USA, 2005. ACM.

Padhraic Smyth. Clustering sequences with hidden markov models. In *Advances in Neural Information Processing Systems*, volume 9, pages 648–654. MIT Press, 1997.

Wil M. P. van der Aalst. Verification of workflow nets. In *Proceedings of the 18th International Conference on the Application and Theory of Petri Nets 1997 (ICATPN '97)*, volume 1248 of *Lecture Notes in Computer Science*, pages 407–426. Springer, 1997.

Wil M. P. van der Aalst. The application of petri nets to workflow management. *Journal of Circuits, Systems, and Computers*, 8(1):21–66, 1998.

Wil M. P. van der Aalst, Anton J. M. M. Weijters, and Laura Maruster. Workflow mining: Discovering process models from event logs. *IEEE Transactions on Knowledge and Data Engineering*, 16(9):1128–1142, 2004.

Wil M. P. van der Aalst, Vladimir Rubin, Boudewijn F. van Dongen, Ekkart Kindler, , and Christian W. Günther. Process mining: A two-step approach using transition systems and regions. BPM-06-30, BPM Center Report, 2006.

Wil M. P. van der Aalst, Hajo A. Reijers, Anton J. M. M. Weijters, Boudewijn F. van Dongen, Ana Karla Alves de Medeiros, Minseok Song, and H. M. W. (Eric) Verbeek. Business process mining: An industrial application. *Information Systems*, 32(5):713–732, 2007.

Boudewijn F. van Dongen and Wil M. P. van der Aalst. A meta model for process mining data. In *EMOI-INTEROP*, volume 160 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2005a.

Boudewijn F. van Dongen and Wil M. P. van der Aalst. Multi-phase process mining: Aggregating instance graphs into EPCs and Petri nets. In *Proceedings of the 2nd International Workshop on Applications of Petri Nets to Coordination, Workflow and Business Process Management (PNCWB)*, 2005b.

Tony Van Gestel, Johan A.K. Suykens, Bart Baesens, Stijn Viaene, Jan Vanthienen, Guido Dedene, B. De Moor, and J. Vandewalle. Benchmarking least squares support vector machine classifiers. *Machine Learning*, 54(1):5–32, 2004.

Anton J. M. M. Weijters and Wil M. P. van der Aalst. Rediscovering workflow models from event-based data using little thumb. *Integrated Computer-Aided Engineering*, 10(2):151–162, 2003.

Anton J. M. M. Weijters, Wil M. P. van der Aalst, and Ana Karla Alves de Medeiros. Process mining with the heuristics miner-algorithm. BETA working paper series 166, Eindhoven University of Technology, 2006.

Lijie Wen, Wil M. P. van der Aalst, Jianmin Wang, and Jiaguang Sun. Mining process models with non-free-choice constructs. *Data Mining and Knowledge Discovery*, 15(2):145–180, 2007.

Ian H. Witten and Eibe Frank. *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2000.

Journal of Machine Learning Research 10 (2009) 1341-1366 Submitted 5/08; Revised 3/09; Published 7/09

Feature Selection with Ensembles, Artificial Variables, and Redundancy Elimination

Eugene Tuv EUGENE.TUV@INTEL.COM
Intel, Logic Technology Development
5000 W Chandler Blvd, CH5295
Chandler, AZ, 85226, USA

Alexander Borisov ALEXANDER.BORISOV@INTEL.COM
Intel, Logic Technology Development
Lopatina st. 3-199
N.Novgorod
Russia, 603163

George Runger RUNGER@ASU.EDU
Industrial Engineering Department
Arizona State University
Tempe, AZ, 85287, USA

Kari Torkkola KARITO@AMAZON.COM
Amazon.com
701 5th Avenue, Suite 1800
Seattle, WA, 98104, USA

Editor: Isabelle Guyon and Amir Reza Saffari

Abstract

Predictive models benefit from a compact, non-redundant subset of features that improves interpretability and generalization. Modern data sets are wide, dirty, mixed with both numerical and categorical predictors, and may contain interactive effects that require complex models. This is a challenge for filters, wrappers, and embedded feature selection methods. We describe details of an algorithm using tree-based ensembles to generate a compact subset of non-redundant features. Parallel and serial ensembles of trees are combined into a mixed method that can uncover masking and detect features of secondary effect. Simulated and actual examples illustrate the effectiveness of the approach.

Keywords: trees, resampling, importance, masking, residuals

1. Introduction

Large data sets are becoming the norm and traditional methods designed for data sets with a modest number of features will struggle in the new environment. This problem area was described by Guyon and Elisseeff (2003) along with other publications in the same issue, and it has increased in importance since then. Additional comments and examples have been provided by Liu and Yu (2005) in a recent survey article.

1.1 Feature Selection

There are three major categories of feature selection methods. Filter methods score variables, typically individually, and eliminate some before a model is constructed. The filter needs to be generated carefully to relate well to the requirements of the modeling task. In particular, the filter may not consider the value of one variable in the presence of others. For example, the widely-used value difference metric (VDM) (Stanfill and Waltz, 1986) and its modified version (MVDM) (Cost and Salzberg, 1993) consider the conditional probability distribution of the response at a predictor value. Such a measure is not sensitive to the effects of some predictors in a model with others present even though interactions among predictors might be critical for an effective subset. A sequential, subset search is sometimes implemented to improve the performance when interactions are important, although a greedy search also has disadvantages in the presence of interactions. Several common filter methods such as ReliefF (Robnik-Sikonja and Kononenko, 2003), CFS (Hall, 2000), and FOCUS (Almuallin and Dietterich, 1994) were modified with sequential search and compared by Yu and Liu (2004).

Wrapper methods form a second group of feature selection methods. The prediction accuracy (or the change in accuracy) of a model directly measures the value of a feature set. Although effective, the exponential number of possible subsets places computational limits for the wide data sets that are the focus of this work.

Embedded methods form a third group for feature selection. These methods use all the variables to generate a model and then analyze the model to infer the importance of the variables. Consequently, they directly link variable importance to the learner used to model the relationship.

1.2 Subset Feature Selection

Fundamentally, the goal of feature selection is to model a target response (or output) variable y, with a subset of the (important) predictor variables (inputs). This is a general goal and several more specific objectives can be identified. Each can lead to different strategies and algorithms. In *filtering* the interest is to remove irrelevant variables. Another objective is *variable ranking* where the interest is in obtaining relative relevance for all input variables with respect to the target. Finally, we might be interested in a compact, yet effective model, where the goal is to identify the smallest subset of independent features with the most predictive power, although a few alternative groups might be reasonable. An important concept here is *the masking relationships* among the predictor variables. Masking occurs when one variable can effectively represent others in a model. Along with the related issue of masking, this paper focuses on the subset selection.

1.3 Contributions of this Paper

Existing tree ensembles such as random forest (Breiman, 2001) or gradient boosting trees (Friedman, 1999) were developed primarily for predictive modeling. In addition, they can provide an importance ranking of the features, but this information has been considered an ad hoc benefit. Random forest (RF) is a random subspace method, and is capable of efficiently ranking features for large data sets. We exploit this property of RF, augment the original data with *artificial contrast variables* constructed independently from the target, and use their ranking for removal of irrelevant variables from the original set. The tree construction method is also modified to produce a more reliable variable ranking in the presence of high cardinality variables. A variable masking measure is then introduced that incorporates surrogate variable scores from ensembles of trees. This forms the basis for *redundancy elimination*. Residual effects are calculated to enable the method to detect variables of secondary importance. These elements are integrated into an efficient algorithm for subset selection called ACE (artificial contrasts with ensembles).

The structure of this paper is as follows. In Section 2 we describe previous work and outline directions taken in this paper. Section 3 describes variable importance measures defined through tree ensembles and explains how they could be used to remove irrelevant features using random, artificial features. Next, we introduce a masking measure and use it for redundancy elimination. Section 4 describes the details of the ACE algorithm to generate the selected subset, and compares ACE with its closest competitors in detail. Section 5 provides results from experiments. Section 6 provides conclusions.

2. Background

This section defines the problem of finding the best susbset of features, discusses previous approaches, and outlines our solution.

2.1 Markov Boundaries

Let F be a full set of features. A feature selection solution can be described in terms of a Markov blanket (Koller and Sahami, 1996). Given a target feature Y, let $M \subset F$ and $Y \notin M$. M is said to be a Markov blanket for Y if $Y \perp (F - M) | M$. That is, Y is conditionally independent of other features given M. A minimal Markov blanket is referred to as Markov boundary (MB) and such a subset might be considered a feature selection solution. However, an important issue is that a MB need not be unique. Redundant features can replace others in a feature subset. Usually feature redundancy is defined in terms of feature correlation (Hall, 2000). For example, two features are redundant to each other if their values are completely correlated. In reality, it is not so straightforward to determine feature redundancy if a feature is partially correlated to a set of features.

Our goal is to focus on the important case with redundant features and obtain at least one MB. In most real-life problems exactly determining the MB or measuring feature relevance is very difficult because of a limited sample size, high time complexity, and noise in the data. Furthermore, evaluation of the distribution of the input variables and the response always relies on some model (linear, support vector machine, frequency tables, trees, etc.). In practice, most algorithms just try to remove irrelevant features and then apply some heuristics that remove “possibly” redundant variables.

2.2 Existing Approaches in Feature Selection

The nature of real life data sets provides strong restrictions for model fitting and feature selection methods. First, data sets may be very large both in terms of the number of predictors and in the number of samples (tens of thousands $\times$ tens of millions). Second, the predictors and the response can be of mixed type (both numeric and categoric), and can contain missing values. Lastly and also very importantly, dependency of the response on predictors can be highly non-linear, noisy and multivariate.

This leaves most existing methods out of scope for such problems. For example, wrapper methods (forward selection or backward elimination) are simply computationally unfeasible when dealing with thousands of predictors. Filter methods are also useless for the minimal subset selection problem, as they do not deal with the notion of redundancy and most of them are inherently univariate. However, there are filters that use a“local” feature importance measure (like RELIEF) that can be considered multivariate (Kira and Rendell, 1992), but still they do not deal with redundancy giving just a ranked list of features instead of a selected minimal subset.

Subset evaluation filter methods such as CFS (Hall, 2000) are neither suitable because they do not deal explicitly with redundancy, and have to iterate over many feature subsets incurring a high time complexity. For example, the time complexity of the CFS is at least quadratic in the number of features and linear in number of samples. Also CFS is highly sensitive to outliers as it uses correlations between features.

Many embedded methods that use a built-in feature relevance measurement, such as SVM-RFE (Guyon et al., 2002) and linear regression with backward feature elimination are heavily dependent on the model (linear or SVM), that can fail to fit the data well. These methods have at least quadratic complexity in the number of samples for fitting an SVM and at least cubic complexity in the number of features ($O(nm^2+m^3)$, where m is the number of features, and n is number of samples) for fitting a regression model. Data sets with tens of thousands of features or samples become very time consuming and impractical to handle. For example, SVM-RFE involves retraining the SVM after features with smallest relevance are removed, thus incurring at least cubic complexity in number of samples ($O(max(m,n)n^2)$).

An issue that discourages using regression methods and methods that rely on some kind of distance measure between observations (linear regression, SVM, Kernel-based methods, RELIEF) is the difficulty of dealing with outliers in the input (predictor) space. Also, selection of important model parameters (kernel width and type, feature relevance thresholds, etc) is non-obvious, and the results of feature selection depend heavily on them.

Most methods return just a ranked list of features instead of an optimal subset. These methods include RELIEF, Koller's Markov blanket based backward elimination (referred to here as MBBE) (Koller and Sahami, 1996), and SVM-RFE. Some methods such as FCBS use a relevance threshold that is not clear how to adjust (Yu and Liu, 2004). In reality, the user also obtains a number of feature subsets corresponding to different values of parameters without a hint of how to choose the best subset.

Many methods work with frequency tables. They can thus deal well with categorical inputs only. For numerical inputs, they require discretization. Such methods are not always able to deal with interacting variables and have great difficulties with multivariate dependencies on numerical inputs. Examples of such methods are FCBS and MBBE. These two algorithms need discretization because they use an entropy measure computed on frequency tables. If the number of categories is large, or if we use frequency tables with more than two inputs, the tables can be sparse and may not represent the data distribution well. Another issue for MBBE is computational complexity. Considering all feature pairs incurs a quadratic complexity on the number of features.

Hence we see that most methods at hand are either not applicable at all to the best subset selection problem, or have some major problems. The most useful methods in such a setting (that appeared to be applicable to the examples of large “real-life” data in the challenge data sets discussed in Sec. 5.3) are methods based on backward feature elimination using an approximate Markov blanket concept (Koller and Sahami, 1996; Yu and Liu, 2004). Our method approximates the optimal Markov blanket redundancy elimination procedure, but without most of the drawbacks of previous methods.

2.3 Towards Efficient and Approximately Optimal Feature Selection

We propose a method that uses an idea similar to those proposed by Koller and Sahami (1996) and Yu and Liu (2004) that tries to overcome their limitations. It does not have quadratic time complexity in the number of features, can deal with thousands of predictors, uses a model (ensembles of trees) that can be applied to mixed variable types, thus eliminating need for discretization of numeric inputs, does not require imputation of missing values, captures local information (like RELIEF), is invariant to a monotone transformation of inputs, thus not very sensitive to noise and outliers, and deals well with multivariate dependencies.

It is well known that trees and especially ensembles of trees can provide robust and accurate models in "real-life" data settings. They handle mixed and noisy data, and are scale insensitive. Ensembles of trees have high predictive power and are resistant to over-fitting (Breiman, 2001). Our approach relies heavily on ensembles of trees.

First, we find irrelevant features that are conditionally independent of the response given the rest of the features. It is accomplished by comparing the relevance of the original variables with the relevance of random, artificial features (appended to the original data) constructed from the same distribution, but independently from the response. These features are referred to as artificial contrasts. We measure feature relevance as variable importance in random forests with a modified robust splitting criteria. We assume that if an original variable had a relevance score not statistically higher than that of an artificial probe (independent from the target by construction) then it is also independent from the target, irrelevant, and should be removed. Note that we try to remove irrelevant features by directly assessing conditional independence without searching for a MB, the existence of which is a much stronger requirement. Although the idea of artificial contrasts was already used by other researchers in simple filter methods with success (Stoppiglia et al., 2003), its application to tree ensembles is novel and promising. Actually, our approach can be considered as non-parametric because all parameters in our algorithm can be assigned reasonable default values that work well for wide range of problems.

Then the redundant feature elimination step is performed. Redundancy between features is measured using surrogate scores. The variable with the largest impurity reduction score at a node is the primary splitter. If surrogate variables (ones that partition the node in same way as the primary variable) are present, these surrogate variables are considered as "masked". Masking scores between all pairs of important variables are computed and evaluated using a statistical test, and variables masked by more important variables ("approximately redundant") are removed iteratively.

Finally, after a set of non-redundant relevant features has been found, our method removes the influence of the found subset with an ensemble and proceeds. Because redundancy elimination is approximate in nature this iterative approach is another advantage of our method. It allows one to recover variables with small importance and to reduce the chance to lose important variables during redundancy elimination.

3. Tree Ensembles for Feature Selection

For our embedded method, we focus on ensembles of decision trees for the following reasons. Trees can be applied in ubiquitous scenarios so that they provide a good entry point for feature selection for interdisciplinary, wide data sets. They apply to either a numerical or a categorical response. They are nonlinear, simple and fast learners that handle also both numerical and categorical predictors well. They are scale invariant and robust to missing values. A simple decision tree also provides an embedded measure of variable importance that can be obtained from the number and the quality of splits that are generated from a predictor variable. However, a single tree is produced by a greedy algorithm that generates an unstable model. A small change to the data can result in a very different model. Consequently, ensemble methods have been used to counteract the instability of a single tree.

Supervised ensemble methods construct a set of simple models, called base learners, and use their weighted outcome (or vote) to predict new data. That is, ensemble methods combine outputs from multiple base learners to form a committee with improved performance. Numerous empirical studies confirm that ensemble methods often outperform any single base learner (Freund and Schapire, 1996; Bauer and Kohavi, 1999; Dietterich, 2000a). The improvement can be dramatic when a base algorithm is unstable. More recently, a series of theoretical developments (Bousquet and Elisseeff, 2001; Poggio et al., 2002; Mukherjee et al., 2006; Poggio et al., 2004) also confirmed the fundamental role of stability for the generalization of a learning algorithm. More comprehensive overviews of ensemble methods were presented by Dietterich (2000b) and Valentini and Masulli (2002). There are two primary approaches to ensemble construction: parallel and serial.

A parallel ensemble combines independently constructed and diverse base learners. That is, different base learners should make different errors on new data. An ensemble of such base learners can outperform any single one of its components since diverse errors cancel out (Hansen and Salamon, 1990; Amit and Geman, 1997). Parallel ensembles are variance-reduction techniques, and in most cases, they are applied to unstable, high-variance algorithms (such as trees). Also, Valentini and Dietterich (2003) showed that ensembles of low-bias support vector machines (SVMs) often outperformed a single, best-tuned, canonical SVM (Boser et al., 1992).

Random forest (RF) is an exemplar for parallel ensembles (Breiman, 2001). It is an improved bagging method (Breiman, 1996) that extends the "random subspace" method (Ho, 1998). It grows a forest of random decision trees on bagged samples showing excellent results comparable with the best known classifiers. A RF can be summarized as follows: (1) Grow each tree on a bootstrap sample of the training set to maximum depth, (2) Given M predictors, select at random $m < M$ variables at each node, and (3) Use the best split selected from the possible splits on these m variables. Note that for every tree grown in RF, about one-third of the cases are out-of-bag (out of the bootstrap sample). The out-of-bag (OOB) samples can serve as a test set for the tree grown on the non-OOB data. We discuss later how OOB samples can be used for feature selection.

In serial ensembles, every new learner relies on previously built learners so that the weighted combination forms an accurate model. A serial ensemble algorithm is often more complex. It is targeted to reduce both bias and variance. A serial ensemble results in an additive model built by a forward-stagewise algorithm. The *adaboost* algorithm was introduced by Freund and Schapire (1996). At every step of ensemble construction the boosting scheme adds a new base learner that is forced (by iteratively reweighting the training data) to concentrate on the training observations that are misclassified by the previous sequence. Boosting showed dramatic improvement in accuracy even with very weak base learners (like decision stumps, single split trees). Breiman (1998) and Friedman et al. (2000) showed that the adaboost algorithm is a form of gradient optimization in functional space, and is equivalent to a forward-stagewise, additive algorithm with the exponential loss function $\Psi(y, F(\mathbf{x})) = \exp(-yF(\mathbf{x}))$ referred to as a gradient boosted tree (GBT).

3.1 Relative Variable Importance Metrics

A single decision tree partitions the input space into a set of disjoint regions, and assigns a response value to each corresponding region. It uses a greedy, top-down recursive partitioning strategy. At every step an exhaustive search is used to test all variables and split points to achieve the maximum reduction in impurity. Therefore, the tree constructing process itself can be considered as a type of variable selection (a kind of forward selection, embedded algorithm), and the impurity reduction due to a split on a specific variable indicates the relative importance of that variable to the tree model (Breiman et al., 1984). For ensembles, the metric is averaged over the collection of base learners. Note, that this relative importance automatically incorporates variable interaction effects thus being very different from the relevance measured by a univariate filter method.

For a single decision tree the measure of variable importance is

$$VI(X_i,T) = \sum_{t \in T} \Delta I(X_i,t), \tag{1}$$

where $\Delta I(X_i,t)$ is the decrease in impurity due to an actual (or potential) split on variable X_i at a node t of the optimally pruned tree T (Breiman et al., 1984). Node impurity $I(t)$ for regression is defined as $\sum_{i\in t}(y_i - \bar{y})^2/N(t)$ where the sum and mean are taken over all observations i in node t, and $N(t)$ is the number of observations in node t. For classification $I(t) = Gini(t)$ where $Gini(t)$ is the Gini index of node t defined as

$$Gini(t) = \sum_{i \neq j} p_i^t p_j^t,$$

and p_i^t is the proportion of observations in t whose response label equals i ($y = i$) and i, j run through all response class numbers. The Gini index is in the same family of functions as *cross-entropy* $= -\sum_i p_i^t log(p_i^t)$, and measures node impurity. It is zero when t has observations only from one class, and is maximum when classes are perfectly mixed. The decrease $\Delta I(X_i,t)$ computes the impurity at the node t and the weighted average of impurities at each child node of t. The weights are proportional to the number of observations that are assigned to each child from the split at node t so that $\Delta I(X_i,t) = I(t) - p_L I(t_L) - p_R I(t_R)$.

For an ensemble of M trees this importance measure is easily generalized. It is simply averaged over the trees

$$E(X_i) = \frac{1}{M}\sum_{m=1}^{M} VI(X_i,T_m). \tag{2}$$

The averaging makes this measure more reliable.

This split weight measure $\Delta I(X_i,t)$ in Equation (1) can be improved if OOB samples are used. The split value for a variable is calculated using the training data as usual. However, the variable selected as the primary splitter uses only the OOB samples. Also, the variable importance measure is calculated from only the OOB samples. This provides a more accurate and unbiased estimate of variable importance in each tree and improves the filtering of noise variables.

Breiman (2001) also proposed a *sensitivity* based measure of variable relevance evaluated by a RF. For a classification problem it is summarized as follows: (1) Classify the OOB cases and count the number of votes cast for the correct class in every tree grown in the forest, (2) randomly permute the values of variable m in the OOB cases and classify these cases down the tree, (3) Subtract the number of votes for the correct class in the variable-m-permuted OOB data from the original OOB data, and (4) Average this number over all trees in the forest to obtain the raw importance score for variable m. Similar ideas were presented by Parmanto et al. (1996) and a similar resampling strategy was successfully used in a more traditional model by Wisnowski et al. (2003). The sensitivity measure is computationally expensive. Furthermore, it does not account for masking, nor does it consider an iterative process with residuals (that we describe in Sec. 4.2). Experiments by Tuv (2006) demonstrated that weaker but independent predictors can rank higher than stronger, but related predictors. Also, related predictors can all be identified as important. Neither of these results are desirable for a best subset model and a more effective algorithm is described in Sec. 4.

With the importance measure (2) we can thus merely rank the variables. The following two subsections discuss how to amend the ranking so that irrelevant variables can be reliably detected, and how the redundancies among the remaining relevant variables can then be handled.

3.2 Removing Irrelevant Features by Artificial Contrasts

Although an ensemble can be used to calculate a relative feature ranking from the variable importance score in (2) the metric does not separate relevant features from irrelevant. Only a list of importance values is produced without a clear indication which variables to include, and which to discard. Also, trees tend to split on variables with more distinct values. This effect is more pronounced for categorical predictors with many levels. It often makes a less relevant (or completely irrelevant) input variable more “attractive” for a split only because it has high cardinality.

The variable importance score in (2) is based on the relevance of an input variable to the target. Consequently, any stable feature ranking method should favor a relevant input X_i over an artificially generated variable with the same distribution as X_i but generated to be irrelevant to the target. That is, a higher variable importance score is expected from a true relevant variable than from an artificially generated contrast variable. With sufficient replicates in an analysis one can select important variables from those that have statistically significantly higher variable importance scores than the contrast variables (Tuv et al., 2006). Here, these contrast variables are integrated into a subset algorithm. We discuss this in detail in Section 4.

Also, artificial contrasts can be applied to masking discussed in the next subsection. Given a selected subset of relevant variables, one computes the masking scores of all variables by elements of this subset, and the masking of contrast variables by this subset. Masking scores statistically higher than the contrast variables are considered to be real masking. Variables that are masked are dropped from the relevant subset list over a sequence of iterations of the algorithm.

3.3 Masking Measures

An important issue for variable importance in tree-based models is how to evaluate or rank variables that were masked by others with slightly higher splitting scores, but could provide as accurate a model if used instead. One early approach in the CART methodology used surrogate splits (Breiman et al., 1984). The predictive association of a surrogate variable X^s for the best splitter X^* at a tree node T is defined through the probability that X^s predicts the action of X^* correctly and this is estimated as

$$p(X^s, X^*) = p_L(X^s, X^*) + p_R(X^s, X^*),$$

where $p_L(X^s, X^*)$ and $p_R(X^s, X^*)$ define the estimated probabilities that both X^s and X^* send a case in T left (right). The predictive measure of association $\lambda(X^*|X^s)$ between split X^s and primary split X^* is defined as

$$\lambda(X^*|X^s) = \frac{\min(\pi_L, \pi_R) - [1 - p(X^s, X^*)]}{\min(\pi_L, \pi_R)},$$

where π_L, π_R are the proportions of cases sent to the left(or right) by X^*. It measures the relative reduction in error $(1 - p(X^s, X^*))$ due to using X^s to predict X^* as compared with the "naive" rule that matches the action with $\max(\pi_L, \pi_R)$ (with error $\min(\pi_L, \pi_R)$). If $\lambda(X^*|X^s) < 0$ then X^s is disregarded as a surrogate for X^*. Sometimes a small, nonnegative threshold is used instead. The variable importance sum in Equation (1) is taken over all internal tree nodes where X_i is a primary splitter or a surrogate variable ($\lambda(X^*|X_i) > 0$ for a primary splitter X^*). Often a variable that does not appear as a primary splitter in a tree is still ranked high on the variable importance list constructed using surrogate variables.

We extend the surrogate concept to define a masking score as follows. Variable i is said to mask variable j in a tree, if there is a split in variable i in a tree with a surrogate on variable j. We define the masking measure for a pair of variables i, j in tree T as

$$M_{ij}(T) = \sum_{\{t \in T | \text{split on } X_i\}} w(X_i, t) \lambda(X_i | X_j),$$

where $w(X_i, t) = \Delta I(X_i, t)$ is the decrease in impurity from the primary split on variable X_i, and summation is done over the nodes where primary split was made on variable X_i. Here we take into account both the similarity between variables X_i, X_j at the node, and the contribution of the actual split of variable X_i to the model. For an ensemble the masking measure is simply averaged over the trees. Note that in general the measure is not symmetric in the variables. One variable may mask several others, but for a single selected masked variable the reverse may not be true.

4. Algorithm: Ensemble-Based Feature Selection with Artificial Variables and Redundancy Elimination

We now integrate the previously described concepts and metrics into a subset selection algorithm. The fundamental steps outlined in Section 2.3 consist of using the advantages of a parallel ensemble to detect important variables among potentially a very large feature set, using the advantages of a serial ensemble to de-mask the important variables, and calculating residuals and repeating in order to recover variables of secondary importance.

Within the algorithm, artificial contrast variables are re-generated a number of times. Then the significance from a paired t-test over the replicates is used to identify important variables and masked variables. Essentially the t-test is used to define thresholds for selection and masking. These thresholds could also be set as tunable parameters. An advantage of the statistical test is that the significance of selected variables relative to noise can be quantified.

4.1 Algorithm Details

1. **Identify Important Variables:** Artificially generated noise variables are used to determine a threshold to test for statistically significant variable importance scores. The test is used to remove irrelevant variables. Details are presented in the displayed algorithms and further described as follows.

   In each replicate $r, r = 1, 2, \ldots, R$ artificial variables are constructed as follows. For every real variable X_j $j = 1, 2, \ldots, M$ a corresponding artificial variable Z_j is generated from a random permutation. Then in each replicate a small RF is trained and variable importance scores are computed for real and artificial variables. The scores from each replicate r are compiled into the rth row of a matrix $\mathbf{V}$ $R \times 2M$. Furthermore, the $1-\alpha$ percentile of the importance scores in replicate r is calculated from only the artificial variables. This is denoted as v_r and the vector of percentiles over the R replicates is $\mathbf{v}$ $R \times 1$. For each real variable X_j a paired t-test compares importance scores for X_j (obtained from the jth column of $\mathbf{V}$) to the vector of scores $\mathbf{v}$. A test that results in statistical significance identifies an important variable.

Significance is evaluated through a suitably small p-value. The use of a p-value requires a feature to consistently score higher than the artificial variables over multiple replicates. Furthermore, this statistical testing framework also allows any method to control false selections to be applied. We routinely use the Bonferroni adjustment, but a false discovery rate approach is also reasonable. Each replicate uses a RF with $L =$ 20-50 trees to score the importance of the original and artificial noise variables. Also, the split weight calculation for variable importance in (2) only uses OOB samples as described previously.

2. **Calculate Masking Scores:** A masking matrix is computed from independent replicates in order to evaluate the statistical significance of masking results. Suppose there are m important variables from step 1. For similar reasons as in the previous step, replicates and noise variables are used to detect masking among the relevant variables. These are currently the same replicates that are used for variable importance. A set of R independent GBT models are generated each with $L =$ 10-50 trees. Note that all variables are tested in each node in each tree in a serial ensemble. Therefore, richer, more effective masking information is obtained from a serial ensemble than from a random subspace method like RF. In these calculations, the surrogate scores and the split weights are calculated from the OOB samples as in the previous step. Let $M_{i,j}^r$ denote the masking score for variables X_i and X_j from the ensemble in replicate r, for $r = 1, 2, \ldots, R$. Also, let $M_{i,\alpha}^r$ denote the $(1-\alpha)$-percentile of the masking score in replicate r from the distribution of scores between variable X_i and the noise variables. That is, $M_{i,\alpha}^r$ denotes the $(1-\alpha)$-percentile of $M_{i,j}^r$ for $j = m+1, \ldots, 2m$. Similar to the check for variable importance, a paired t-test compares the masking score between variables (X_i, X_j) with masking score $M_{i,\alpha}^r$ computed from the noise variables. There is a significant masking between variables (X_i, X_j) if the paired t-test is significant. Variable X_j is masked by variable X_i if the test is significant.

3. **Eliminate Masked Variables:** Masked variables are removed from the list of important variables as follows. Given a list of important variables upon entry to this step, the variables are sorted by the importance score calculated in step 2. The most important variable is added to an exit list, and dropped from the entry list. Assume this is variable X_i. All variables that are masked by X_i are dropped from the entry list. This is repeated until the entry list is empty. The exit list represents the unmasked important variables.

4. **Generate Residuals for Incremental Adjustment:** An iteration is used to enhance the ability of the algorithm to detect variables that are important, but possibly weaker than a primary set. Given a current subset of important variables, only this subset is used to predict the target. Residuals are calculated and form a new target. For a numerical target the residuals are simply the actual minus the predicted values. For a classification problem residuals are calculated from a multiclass logistic regression procedure (Friedman et al., 2000). We predict the log-odds of class probabilities for each class (typically GBT is used), and then take

pseudo residuals as summarized in the following multi-class logistic regression algorithm. The algorithms are described using the notation in Table 1.

The iterations are similar to those used in forward selection. See, for example, Stoppiglia et al. (2003). The Gram-Schmidt procedure first selects the variable with highest correlation with the target. To remove the information from this variable the remaining predictors and the target are othogonalized with respect to the selected variable. This provides residuals from the fit of the target to the first selected variable. In the feature selection method here we do not require orthogonal predictors, but we adjust the target for the variables already selected through residuals. We also can select more than a single variable in each iteration. The method also uses a conservative selection criterion (Bonferroni adjustment) and the residuals allow a variable to enter on another iteration. There are similar procedures used elsewhere in regression model building. Least angle regression (Efron et al., 2004) and projection pursuit methods (Friedman et al., 1981) are well known examples that use residuals in forward-stagewise modeling.

The algorithm returns to step 1 and continues until no variables with statistically significant importance scores remain. The current subset of important variables is used for the prediction model. Whenever step 1 is calculated, all variables are used to build the ensembles—not only the currently important ones. This approach allows the algorithm to recover partially masked variables that still contribute predictive power to the model. This can occur after the effect of a masking variable is completely removed, and the partial masking is eliminated. The algorithms for numerical (regression) and categorical (classification) targets are presented as algorithms 1 and 2. A separate algorithm 3 describes the variable masking calculations.

Algorithm 1: Ensemble-Based Feature Selection, Regression

```
1.  Set $\Phi \leftarrow \{\}$; set $F \leftarrow \{X_1,\ldots,X_M\}$; set $W = 0$ $(|W| = M)$.
2.  for $r = 1,\ldots,R$ do
3.      $\{Z_1,\ldots,Z_M\} \leftarrow$ permute$\{X_1,\ldots,X_M\}$
4.      set $F_P \leftarrow F \cup \{Z_1,\ldots,Z_M\}$
5.      $r^{th}$ row of $\mathbf{V} = \mathbf{V}_{r.} = g_I(F_P, Y)$;
    endfor
6.  $R \times 1$ vector (element wise) $\mathbf{v} = Percentile_{1-\alpha}(\mathbf{V}[\cdot, M+1,\ldots,2M])$
7.  Set $\hat{\Phi}$ to those $\{X_j\}$ for which element wise $\mathbf{V}_{.j} > \mathbf{v}$
    with specified paired t-test significance (0.05)
8.  Set $\hat{\Phi} = RemoveMasked(\hat{\Phi}, W + g_I(F_P, Y))$
9.  If $\hat{\Phi}$ is empty, then quit.
10. $\Phi \leftarrow \Phi \cup \hat{\Phi}$;
11. $Y = Y - g_Y(\hat{\Phi}, Y)$
12. $W(\hat{\Phi}) = W(\hat{\Phi}) + g_I(\hat{\Phi}, Y)$
13. Go to 2.
```

Algorithm 2: Ensemble-Based Feature Selection, Classification

```
1.   set $\Phi \leftarrow \{\}$; $G_k(F) = 0, W_k = 0$
2.   for $k = 1, \ldots, K$ do
3.     set $V = 0$.
4.     for $r = 1, \ldots, R$ do
         $\{Z_1, \ldots, Z_M\} \leftarrow$ permute$\{X_1, \ldots, X_M\}$
         set $F \leftarrow X \cup \{Z_1, \ldots, Z_M\}$
         Compute class proportion $p_k(x) = exp(G_k(x)) / \sum_{l=1}^{K} exp(G_l(x))$
         Compute pseudo-residuals $Y_i^k = I(Y_i = k) - p_k(x_i)$
         $\mathbf{V}_{r.} = \mathbf{V}_{r.} + g_I(F, Y^k)$;
       endfor
5.     Element wise $\mathbf{v} = Percentile_{1-\alpha}(\mathbf{V}[\cdot, M+1, \ldots, 2M])$
6.     Set $\hat{\Phi}_k$ to those $\{X_k\}$ for which $\mathbf{V}_{.k} > \mathbf{v}$
       with specified paired t-test significance (0.05)
7.     Set $\hat{\Phi}_k = RemoveMasked(\hat{\Phi}_k, W_k + g_I(F, Y^k))$
8.     $\Phi \leftarrow \Phi \cup \hat{\Phi}_k$;
       for $k = 1, \ldots, K$ do
9.       $G_k(F) = G_k(F) + g_Y(\hat{\Phi}_k, Y^k)$
10.      $W_k(\hat{\Phi}_k) = W_k(\hat{\Phi}_k) + g_I(\hat{\Phi}_k, Y^k)$
       endfor
     endfor
11.  If $\hat{\Phi}_k$ for all $k = 1, \ldots, K$ is empty, then quit.
12.  Go to 2.
```

4.2 Comparison to Previous Work

Two earlier methods are closely related to ACE, FCBS (Yu and Liu, 2004) and MBBE (Koller and Sahami, 1996). We compare our method in detail to these two methods. Because we use a multivariate model (tree) instead of frequency tables, our method fits in the category of embedded methods. This is unlike FCBS and MBBE that can be considered as correlation filters, although Koller works with frequency tables of 2-5 variables.

FCBS first sorts features by correlation with the response using a symmetric uncertainty, optionally removing the bottom of the list by a user-specified threshold, then

1. The feature most correlated to the response is selected.

2. All features that have correlation with the selected feature higher than it's correlation with response are considered redundant and removed. The feature is added to the minimal subset (and this is an approximate heuristic for Markov blanket filtering).

3. Return to 1).

FCBS is similar in structure to our method, with the following important differences.

Algorithm 3: RemoveMasked(F,W)

```
1    Let $m = |F|$.
2.   for $r = 1, \dots, R$ do
3.       $\{Z_1, \dots, Z_m\} \leftarrow$ permute$\{X_1, \dots, X_m\}$
4.       set $F_P \leftarrow F \cup \{Z_1, \dots, Z_m\}$
5.       Build GBT model $G_r = GBT(F_P)$.
6.       Calculate masking matrix $M^r = M(G_r)$ ($2m \times 2m$ matrix).
     endfor
7.   Set $M^r_{i,\alpha_m} = Percentile_{1-\alpha_m}(M^r[i, m+1, \dots, 2m]),\ r = 1, \dots, R$
8.   Set $M^*_{ij} = 1$ for those $i, j = 1 \dots m$ for which $M^r_{ij} > M^r_{i,\alpha_m},\ r = 1, \dots, R$
     with specified paired t-test significance (0.05), otherwise set $M^*_{ij} = 0$
9.   Set $L = F, L^* = \{\}$.
10.  Move $X_i \in L$ with $i = \operatorname{argmax}_i W_i$ to $L^*$.
11.  Remove all $X_j \in L$ from $L$, for which $M^*_{ij} = 1$.
12.  Return to step 10 if $L \neq \{\}$.
```

1. We use tree importance instead of univariate correlation with the response. This makes ACE much more robust and accurate.

2. We use a surrogate masking measure instead of correlation. This takes the response into account, not only the correlations between inputs. No arbitrary thresholds for correlation are used.

3. We compute residuals to find smaller effects reducing the chance to drop a non-redundant feature.

Koller's MBBE works as follows:

1. For each feature X_i, find the set M_i of K features ($K = 1-4$) that are most correlated to it. (That is, which provide little information on the response when added to the selected feature in frequency table models.) Additional information is measured as KL-distance (Kullback and Liebler, 1951) $D(P(y|X_i, X_j), P(y|X_i))$. The set M_i is called the approximate Markov blanket for feature X_i. The authors state that $K = 1-2$ gives the best results.

2. For each feature compute the relevance score $\delta_i = D(P(y|M_i, X_i), P(y|M_i))$. This represents the additional information it brings when added to its approximate Markov blanket, and remove features that have the smallest relevance scores (i.e., most redundant).

3. Repeat (1,2) until all features are ranked in the order they are deleted. This method returns a ranked list of features rather than one subset.

Our ACE algorithm works more like FCBS as it uses only one feature as an approximate MB for each feature (as does the MBBE algorithm with $K = 1$). Furthermore, it filters features by relevance before computing redundancy between the features, and reports a final minimum feature subset.

K	Number of classes (if classification problem)
X	set of original variables
Y	target variable
M	Number of variables
R	Number of replicates for t-test
α	quantile used for variable importance estimation
α_m	quantile used for variable masking estimation
Z	permuted versions of X
W	cumulative variable importance vector.
W_k	cumulative variable importance vector for k-th class in classification.
F	current working set of variables
Φ	set of important variables
$\mathbf{V}$	variable importance matrix $(R \times 2M)$
$\mathbf{V}_{r.}$	rth row of variable importance matrix $\mathbf{V}$, $r = 1 \dots R$
$\mathbf{V}_{.j}$	jth column of matrix $\mathbf{V}$
$g_I(F,Y)$	function that trains an ensemble of L trees based on variables F and target Y, and returns a row vector of importance for each variable in F
$g_Y(F,Y)$	function that trains an ensemble based on variables F and target Y, and returns a prediction of Y
$G_k(F)$	current predictions for log-odds of k-th class
$GBT(F)$	GBT model built on variable set F
$M(G)$	Masking measure matrix calculated from model G
M^k	Masking matrix for k-th GBT ensemble G_t.
M^*	Masking flags matrix

Table 1: Notation in Algorithms 1-3

However, the major difference is that our redundancy measure approximates KL-distance taking the response into account and uses local information. Thus, it can deal with multivariate dependencies. MBBE for $K > 1$ will incur three (or more) dimensional frequency tables that are hard to deal with if number of categories is large.

The learner $g(.,.)$ in the ACE algorithms is an ensemble of trees. Any classifier/regressor function can be used, from which the variable importance from all variable interactions can be derived. To our knowledge, only ensembles of trees can provide this conveniently.

The computational complexity of the algorithm is of the same order as the maximal complexity of a RF on the whole feature set and a GBT model on the selected important feature subset. A GBT model is usually more complex, because all surrogate splits at every tree node are computed. However, a smaller tree depth setting for the GBT model reduces the calculations in this part of the algorithm. The complexity is proportional to

$$(Fsel + Fimpvar) * N * logN * Ntrees * Nensembles * Niter + Niter * Fimpvar^2,$$

where the variables are defined as follows: *Niter* is the number of iterations of the ACE algorithm (for example, for the challenge discussed in Sec. 5.3 this was always less than 10 and usually 3-4); *Nensembles* is the number of replicates for t-tests (equal to 20 in the challenge); *Ntrees* is the number of trees in the RF or ensemble (equal to 20-100 in the challenge); N is the number of samples; *Fsel* is the number of selected variables per tree split in RF (equal to the square root of the total number features or less); *Fimpvar* is the number of selected important variables (for example, for the challenge data set NOVA discussed in Section 5.3 this was approximately 400-800 depending on parameters). The algorithm is very fast with approximately a minute for one feature selection iteration on the challenge NOVA data set with 16K variables with 20 replicates with 70 trees on a Windows XP-based four-processor Xeon (2 x HT) 3.4GHz workstation.

5. Experiments

In order to evaluate the goodness of feature selection algorithms, two options have been used in the literature. The first is not to evaluate the actual feature selection performance at all, but the performance of a subsequent learner in some task. This facilitates the use of any data set in the "evaluation" but does not give much useful information at all in characterizing the actual feature selection. The second option is to directly evaluate the feature selection performance without using a subsequent proxy task. The latter dictates the need to know the ground truth behind the data, which typically means that the data must be artificially generated, either completely, or by adding some redundant and/or irrelevant features to some known data.

As the topic of the paper at hand is a method for the subset feature selection, the first evaluation method is affected by the choice of the classifier. The effects of feature selection are mixed in with how well the learner is able to handle redundant or irrelevant features. The results would thus depend on the choice of learners and on the choice of data sets. Therefore we will mainly describe experiments with two types of simulated data with known ground truth.

Experiments on data with linear relationships are presented first. Then a nonlinear data generator is used to study the sensitivity to multiple variable interactions with nonlinear relations. Further results are from the 2007 International Joint Conference on Neural Networks (IJCNN), "Agnostic learning vs. prior knowledge challenge & data representation discovery workshop". The algorithm described here had the second best performance in the agnostic track. Here we demonstrate the effect of the subset to predictor performance as compared to the full set of features. Also, an actual manufacturing data set as well as a comparison to a previous analysis of the well known Hepatitis data are also presented in terms of predictive power of the resulting feature set.

5.1 Generated Data with Linear Relationships

The data in this experiment has an additive structure with one numeric response variable and 203 input variables. Inputs $x_1,\ldots,x_{100}$ are highly correlated with one another, and they are all reasonably predictive of the response (regression $R^2 \sim 0.5$). But a,b, and c are independent variables that are much weaker predictors (regression $R^2 \sim 0.1$). Further $u_1,\ldots,u_{100}$ are i.i.d. $N(0,1)$ variables that are unrelated to the target. The target variable was generated as an additive model with additional noise using $y = x_1 + a + b + c + \varepsilon$, where $\varepsilon \sim N(0,1)$. This structure is chosen because it is well known that linear (oblique) relationships are not optimal for a tree representation. However, they are ideal for correlation-based methods. Thus we have here the worst possible case for ACE and the best possible case for CFS. The methods were evaluated on 50 data sets of size 400 samples.

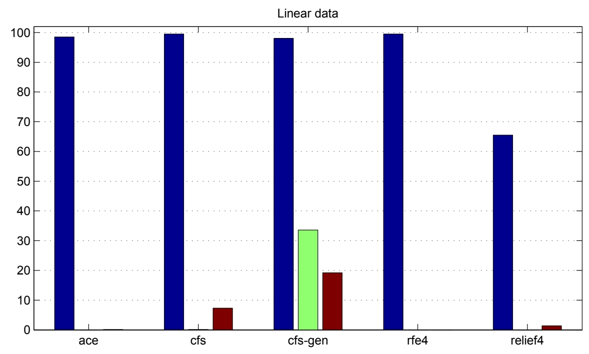


Figure 1: Artificial data with linear relationships. Subset discovery methods (ACE, CFS, CFS-gen) and methods finding a subset of predefined size four (RFE4, Relief4) are compared. The results for each method consist of three bars. The first is the percentage of relevant variables detected (out of four), the second is the percentage of redundant variables detected (out of 100), and the third is the percentage of irrelevant variables detected (out of 100). The results are averages over 50 data sets.

Figure 1 depicts the performance of ACE against methods that also discover the subsets (CFS with best-first search, CFS with genetic search), as well as against some subset ranking methods (RFE, Relief).

RFE and Relief are ranking methods. In this experiment they were given the advantage of knowing the number of relevant features beforehand, that is, their task was to “find the best possible four variable subset” (RFE4, Relief4), whereas ACE and CFS had to also find the number themselves. A further advantage was given to RFE by matching the underlying support vector regressor to the problem with a linear kernel (using the standard RBF kernel produced inferior results). This experiment demonstrates one aspect of the advantages of ACE. In a task ideal for correlation-based methods but hard for trees, we show equal performance.

5.2 Generated Nonlinear Data

Next, experiments were conducted using a well-known data generator (Friedman, 1999), which produces data sets with multiple non-linear interactions between input variables. The true model can be designed with relevant, redundant, and noise inputs. We selected 10 relevant inputs plus random, uniform (0, 1) noise. Also, 20 redundant inputs were used. Each was a random linear combination of three inputs plus random, uniform noise. Finally, 40 noise inputs were added, so that 70 features were available to the full model. The target function was generated as a weighted sum of 10 multidimensional Gaussians, each Gaussian at a time involving about four input variables randomly drawn from the relevant 10 variables. Thus all of the relevant 10 input variables are involved in the target, to a varying degree. The Gaussian functions also have a random mean vector and a random covariance matrix as described by Friedman (1999). Weights for the Gaussians were randomly drawn from $U[-1,1]$.

The data generator produces continuous-valued variables. Thus the data sets can be used as such for regression problems. Data sets of two different sizes were generated, 1000 and 4000 samples. In order to generate classification problems, the target variable was discretized to two levels. Mixed-type data was generated by randomly discretizing half of the variables, each to a random number of levels drawn from $U[2,32]$. There are thus eight different experiments altogether. For each experiment, 50 data sets were generated with different seeds. Figure 2 presents the results for each case as average percentages of features selected in each group (relevant, redundant, or noise) over the 50 generated data sets.

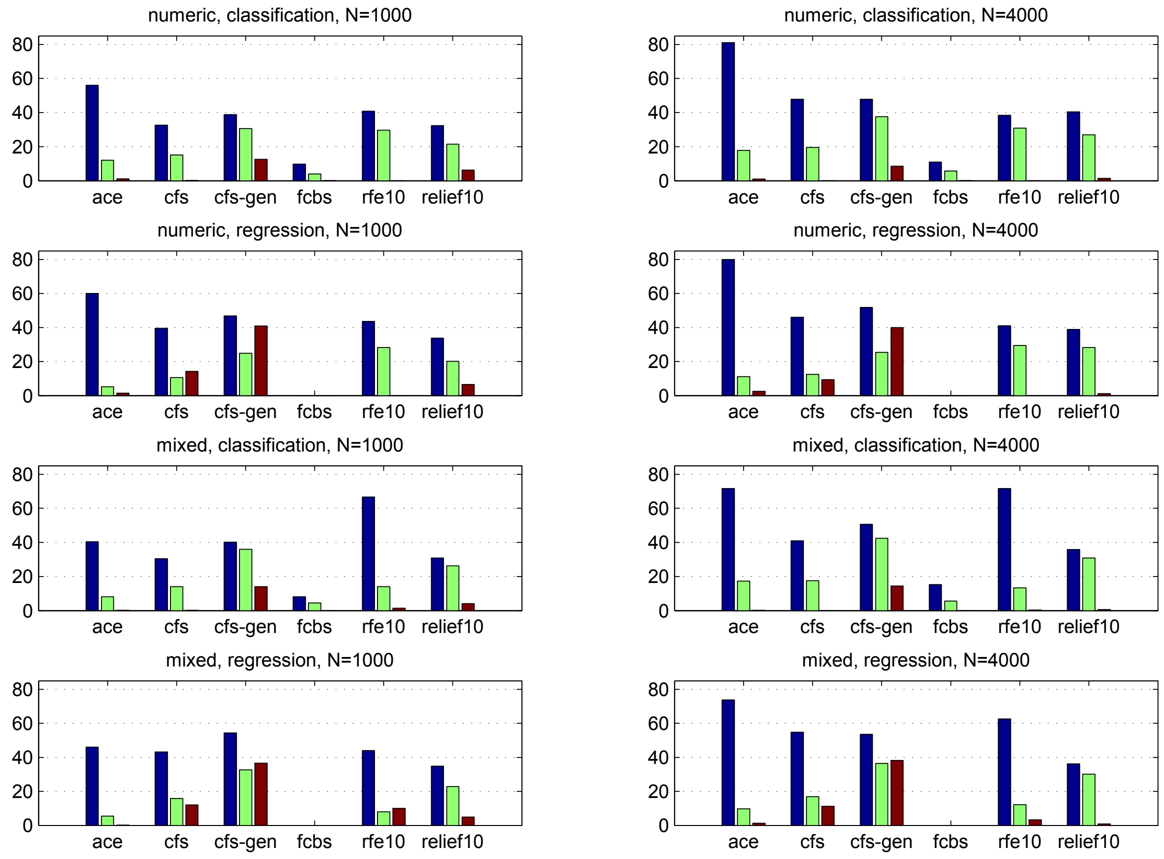


Figure 2: Artificial data with nonlinear relationships. Subset discovery methods (ACE, CFS, CFS-gen, FCBS) and methods finding a subset of predefined size 10 (RFE10, Relief10) are compared. FCBS works only in classification problems. The results for each method consist of three bars. The first is the percentage of relevant variables detected (out of 10), the second is the percentage of redundant variables detected (out of 20), and the third is the percentage of irrelevant variables detected (out of 40). The results are averages over 50 data sets.

RFE and Relief were again given the advantage of knowing the number of relevant features beforehand, that is, their task was to “find the best possible ten-variable subset”, whereas ACE, CFS, and FCBS had to also find the number by themselves. A further advantage was given to RFE by matching the underlying support vector classifier to the problem with an RBF kernel. Using a linear kernel produced inferior results.

The notable failure of FCBS on this data can be explained as follows. Most numerical important variables are dropped at the discretization step of FCBS, because MDL discretization works as a filter method, and it cannot deal with the multivariate dependency from Friedmans's generator. It works well with discrete variables only when the number of categories is small and the response is categorical with a small number of categories.

This experiment demonstrates another aspect of the universality of ACE. The only case where another method (RFE10) showed a superior result was a classification problem with a smaller sample size and mixed type inputs. Again RFE10 was given the advantage of knowing the number of relevant features and an appropriate kernel beforehand.

5.3 IJCNN 2007 Agnostic Learning vs. Prior Knowledge Challenge

In this experiment we show the effect of the selected subset within various classification tasks. The ACE feature selection algorithm was applied to the data sets in the Agnostic Learning Challenge. The number of training/validation/testing instances and the number of features are shown in the following list:

- ADA, Marketing, 4147/415/41471, 48 features
- GINA, Handwriting recognition, 3153/315/31532, 970 features
- HIVA, Drug discovery, 3845/384/38449, 1617 features
- NOVA, Text, 1754/175/17537, 16969 features
- SYLVA, Ecology, 13086/1309/130857, 216 features

For feature selection with ACE, the number of trees, importance and masking quantiles were parameters that were optimized. Next GBT with embedded feature selection (to prevent overfitting) (Borisov et al., 2006) was built on the subset. The following parameters of GBT were optimized: number of trees, tree depth, shrinkage, number of selected features per tree node, and the importance adjustment rate for embedded feature selection, stratified sampling for 0/1 class proportions, and priors. The optimization strategy (manual) was to set reasonable parameter values, and then try to adjust each parameter sequentially, so that the test error decreased. The model was trained on 60% of the training data during parameter optimization. Several passes over all the GBT parameters were used, and one for the feature selection parameters. Priors were selected using cross validation. Feature selection and GBT were used on K partitions of the data and then optimal priors were selected on the remaining part.

Table 2 shows the results before and after subset selection for the five challenge data sets. The CV-error was either preserved or reduced through a good subset. The overall results were the second best in the agnostic learning challenge. Redundancy elimination was applied on ADA, HIVA, SYLVA, and feature selection without redundancy elimination was used on NOVA and GINA.

5.4 TIED Data Set

A data set with multiple Markov boundaries was generated by Statnikov and Aliferis (2009). The data was obtained from a discrete Bayesian network with 1000 variables and a target variable with four classes. A training set was constructed with 750 instances simulated from the network.

Original	Features	CV-error from all features	Best subset size	CV-error from selected subset
Ada	47	0.1909	16	0.1855
Gina	970	0.0527	75	0.0506
Hiva	1617	0.2847	221	0.2559
Nova	12993	0.0591	400	0.0518
Sylva	212	0.0133	69	0.0129

Table 2: IJCNN 2007 Agnostic Learning vs. Prior Knowledge Challenge results.

Variable	p-value	Importance Score
3	0	100.0%
2	0	98.4%
10	0	96.4%
1	1.E-10	96.4%
11	3.E-07	83.3%
12	2.E-07	83.3%
13	5.E-07	79.1%
18	3.E-09	67.5%
19	2.E-07	67.5%
15	2.E-07	41.4%
20	2.E-06	39.5%
29	2.E-06	29.8%
8	3.E-06	26.2%
14	1.E-08	11.6%
4	8.E-06	9.5%
9	6.E-06	8.3%

Table 3: Feature selection scores for the TIED data set. Variables in any Markov boundary are recovered as significant with three false alarms.

The network contained 72 Markov boundaries. Each boundary contained five variables (one from each of the following subsets):(1)$\{X_9\}$, (2) $\{X_4, X_8\}$, (3)$\{X_{11}, X_{12}, X_{13}\}$, (4) $\{X_{18}, X_{19}, X_{20}\}$, and (5) $\{X_1, X_2, X_3, X_{10}\}$.

The ACE feature selection method described here was used to remove irrelevant features. After three iterations of the residual calculations described previously the algorithm stopped with the important variables (and p-values from the artificial contrasts) shown in Table 3. The list of statistically significant variables reproduces all the variables in any of the Markov boundaries listed above, with false alarms from variables X_{14}, X_{15}, and X_{29}.

Although ACE recovered the variables in the Markov boundaries, there are limitations with the masking methods for a multi-class target. The GBT ensembles model each class (versus the rest) with a binary logistic function and averages variable masking scores over the binary models. Consequently, some attenuation of the importance scores are expected. Redundancy elimination did not effectively eliminate masking in the TIED data. However, we used the TIED network and TIED data for binary problems with each class versus the rest. For example, for class 1 versus the rest the TIED network generates the same collection of 72 Markov boundaries. The results from ACE without redundancy elimination for the binary target are shown in Table 4. The list of statistically significant variables reproduces all the variables in any of the Markov boundaries, with no false alarms.

Variable	Importance Score
4	100.0%
8	100.0%
19	88.1%
18	88.1%
20	88.1%
9	64.8%
13	39.5%
12	39.5%
11	39.5%
10	21.9%
2	21.9%
3	21.9%
1	21.9%
6	0.0%

Table 4: Variable importance for TIED data modified for a binary target (class 1 versus the rest). All variables in the true Markov boundaries are identified with no false alarms.

As the importance scores are arranged in decreasing order in Table 4, groups of variables with similar scores become noticeable and these groups correspond to the subsets (equivalence classes) in the cross-product that defines the Markov boundaries. That is, the most important variables in Table 4 are those in the subset $\{X_4, X_8\}$ in the Markov boundaries and the last group matches the subset $\{X_1, X_2, X_3, X_{10}\}$. The equivalent groups are clear from their importance scores in this case.

The analysis with redundancy elimination generated the list of significantly significant variables in Table 5. One equivalent variable from the subset $\{X_1, X_2, X_3, X_{10}\}$ was missed in the recovery of a Markov boundary. The contribution from this subset was, however, small. The predictive performance of a tree ensemble on the recovered variables is nearly identical to a model on a true Markov boundary. In addition, the three variables $\{X_{18}, X_{20}, X_4\}$ are identified in Table 5 as important, but they are redundant in the true network. Although these comprise false alarms, the magnitudes of the importance scores indicate that the last three variables are much less important than the others. Similar results (not shown here) were obtained for the binary target class 2 (versus the rest). Results without any errors were obtained for classes 0 and 3 (each versus the rest). Specifically, for class 0 the Markov boundaries from the TIED network consist of one element from $\{X_1, X_2, X_3, X_{10}\}$. In this case the ACE analysis without redundancy elimination recovered these four variables without false alarms. The analysis with redundancy elimination correctly recovered a single variable from this set. Similarly for class 3, without redundancy elimination all variables in the Markov boundaries $\{X_{12}, X_{13}, X_{14}\}$ were recovered, and only one variable from this set was recovered with redundancy elimination.

Variable	Importance Score
8	100.0%
9	61.3%
19	43.8%
1	10.2%
18	2.6%
20	0.9%
4	0.3%

Table 5: Variable importance for TIED data modified for a binary target (class 1 versus the rest) with redundancy elimination.

5.5 Manufacturing Data

In multiple real world applications collecting unnecessary variables is a cost issue and finding a suitable subset is critical in terms of cost-efficiency. As an example we present manufacturing data from a process that contained approximately 10K rows and consisted of 35 predictors that were all numerical, continuous measurements. The target was a binary response and approximately 20% of the data belonged in the rare class. Because the data is actual manufacturing data, the specific variable names are not provided. The data was analyzed extensively with traditional regression methods (the response was coded as 0 and 1) and models obtained were complex and not accurate. A list of the results from our algorithm is shown in Table 6. It is not unusual for manufacturing data to consist of related predictors. Without redundancy elimination, 20 variables were identified as related to the target. However, after masking scores were used to remove redundant predictors the final subset model consisted of only five predictors.

The predictive accuracy for the binary target was nearly identical using a GBT model with these 5 predictors to the full set of 35 predictors. Table 6 also compares other subset selection algorithms to ACE in terms of their predictive accuracy and the size of the selected feature set.

A previous analysis of this data by Berrado and Runger (2007) used association rules applied after the predictors were discretized with simple equal-frequency discretization. Only rules with consequent equal to the rare target class were considered. A total of 25 rules were detected that met the minimum support threshold. These rules contained 14 variables and 13 out of 14 are listed in the Table 6. Although the objectives of the association analysis were different, the relatively high proportion of important variables is consistent with the results in Table 6.

5.6 Hepatitis Data

The hepatitis data available from the UC-Irvine repository has been widely analyzed. There are 155 patients and 19 predictors and the response is a binary survival result. Breiman (2001) considered this data and cited a previous analysis from the Stanford Medical School and another analysis by Diaconis and Efron (1983). The analysis from the medical school concluded that the important variables were 6, 12, 14, 19. But Breiman (2001) concluded after a set of analyses that number 12 or 17 provided predictive power nearly equivalent to the full set of variables, and that these masked each other. A notable difficulty is the small sample size in this example.

Variables	ACE without redundancy elim.	ACE with redundancy elim.	CFS	CFS-gen	FCBS
V11	100.0%	72.4%	1	1	
V4	96.1%	100.0%	1	1	
V5	49.8%	49.4%	1	1	1
V12	48.6%		1	1	
V14	46.6%		1	1	
V10	43.5%		1	1	
V2	43.3%	36.4%	1	1	
V13	38.7%	21.6%			
V8	30.3%		1		
V1	27.9%			1	
V9	23.7%				
V3	23.6%			1	
V19	21.8%				
V7	21.5%				
V20	20.4%				
V26				1	
V27				1	
Errors		0.145	0.144	0.145	0.190

Table 6: Manufacturing data with a binary target with redundancy elimination excludes many variables. Only a smaller subset of the relevant predictors remain. We compare the extracted variables to other subset selection algorithms (selected variables are marked as '1' in the table). The error rate for the full set of variables was 0.146.

We confirmed the strong masking between variables 12 and 17 (and vice versa) from our masking matrix. We also obtained a subset model that consists of variables 6, 17, 14, 19, and 11, similar to medical school. Variable 11 was also identified in unpublished lecture notes by Breiman. The subset selected by our algorithm has the lowest cross-validation error using logistic regression.

6. Conclusions

We have presented an efficient method for feature subset selection that builds upon the known strengths of the tree ensembles and is designed explicitly to discover a non-redundant, effective subset of features in large, dirty, and complex data sets.

Our method attempts to eliminate irrelevant variables using statistical comparisons with artificial contrasts to obtain a threshold for importance estimated from the parallel ensembles of trees capable of scoring very large number of variables.

It uses serial ensembles to discover significant masking effects for redundancy elimination. Furthermore we have showed that the redundancy elimination based on feature masking approximates the Markov blanket redundancy filtering. It also uses an iterative strategy to allow for weaker predictors to be identified after stronger contributors.

Variables	ACE	CFS	CFS-gen	FCBS
malaise-6	1	1	1	
albumin-17	1			
bilirubin-14	1	1	1	
histology-19	1	1	1	
spiders-11	1	1	1	1
age-1		1	1	
sex-2		1	1	1
ascites-12		1	1	1
varices-13		1	1	
Errors	0.142	0.155	0.155	0.194

Table 7: Hepatitis data. Features selected from ACE compared to other subset selection algorithms (selected variables are marked as '1' in the table). The baseline error rate for the full set of variables was 0.148.

The superior performance of the algorithm is illustrated with a number of experiments on both artificial and real data as well as by its success in the agnostic learning challenge.

Acknowledgments

This material is partly based upon work supported by the National Science Foundation under Grant No. 0743160.

References

H. Almuallin and T. G. Dietterich. Learning boolean concepts in the presence of many irrelevant features. *Artificial Intelligence*, 69(1-2):279–305, 1994.

Y. Amit and D. Geman. Shape quantization and recognition with randomized trees. *Neural Computation*, 9(7):1545–88, 1997.

E. Bauer and R. Kohavi. An empirical comparison of voting classification algorithms: Bagging, boosting and variants. *Machine Learning*, 36(1/2):525–536, 1999.

A. Berrado and G.C. Runger. Using metarules to organize and group discovered association rules. *Data Mining and Knowledge Discovery*, 14(3):409–431, 2007.

A. Borisov, V. Eruhimov, and E. Tuv. Tree-based ensembles with dynamic soft feature selection. In I. Guyon, S. Gunn, M. Nikravesh, and L. Zadeh, editors, *Feature Extraction Foundations and Applications: Studies in Fuzziness and Soft Computing*. Springer, 2006.

B. Boser, I. Guyon, and V. Vapnik. A training algorithm for optimal margin classifiers. In D. Haussler, editor, *5th Annual ACM Workshop on COLT, Pittsburgh, PA*, pages 144–152. ACM Press, 1992.

O. Bousquet and A. Elisseeff. Algorithmic stability and generalization performance. In *Advances in Neural Information Processing Systems*, volume 13, pages 196–202. MIT Press, 2001.

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

L. Breiman. Arcing classifiers. *The Annals of Statistics*, 26(3):801–849, 1998.

L. Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

L. Breiman, J. Friedman, R. Olshen, and C. Stone. *Classification and Regression Trees*. Wadsworth, Belmont, MA, 1984.

S. Cost and S. Salzberg. A weighted nearest neighbor algorithm for learning with symbolic features. *Machine Learning*, 10(1):57–78, 1993.

P. Diaconis and B. Efron. Computer intensive methods in statistics. *Scientific American*, (248): 116–131, 1983.

T. G. Dietterich. An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. *Machine Learning*, 40(2):139–157, 2000a.

T. G. Dietterich. Ensemble methods in machine learning. In *First International Workshop on Multiple Classifier Systems 2000, Cagliari, Italy*, volume 1857 of *Lecture Notes in Computer Science*, pages 1–15. Springer, 2000b.

B. Efron, T. Hastie, L. Johnstone, and R. Tibshirani. Least angle regression. *Annals of Statistics*, 32:407–499, 2004.

Y. Freund and R. E. Schapire. Experiments with a new boosting algorithm. In *The 13th International Conference on Machine Learning*, pages 148–156. Morgan Kaufman, 1996.

J. Friedman. Greedy function approximation: a gradient boosting machine. *Technical report, Dept. of Statistics, Stanford University*, 1999.

J. Friedman, T. Hastie, and R. Tibshirani. Additive logistic regression: A statistical view of boosting. *Annals of Statistics*, 28:832–844, 2000.

J. H Friedman, M. Jacobson, and W. Stuetzle. Projection pursuit regression. *Journal of the American Statistical Association*, 76:817–823, 1981.

I. Guyon and A. Elisseeff. An introduction to variable and feature selection. *Journal of Machine Learning Research*, 3:1157–1182, Mar 2003.

I. Guyon, J. Weston, S. Barnhill, and V. Vapnik. Gene selection for cancer classification using support vector machines. *Machine Learning*, 46(1-3):389–422, 2002.

M. A. Hall. Correlation-based feature selection for discrete and numeric class machine learning. In *Proceedings of the 17th International Conference on Machine Learning*, pages 359–366, 2000.

L. K. Hansen and P. Salamon. Neural network ensembles. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 12(10):993–1001, 1990.

T. K. Ho. The random subspace method for constructing decision forests. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 20(8):832–844, 1998.

K. Kira and L. A. Rendell. A practical approach to feature selection. In *ML92: Proceedings of the ninth international workshop on Machine learning*, pages 249–256, San Francisco, CA, USA, 1992. Morgan Kaufmann Publishers Inc. ISBN 1-5586-247-X.

D. Koller and M. Sahami. Toward optimal feature selection. In *Proceedings of ICML-96, 13th International Conference on Machine Learning*, pages 284–292, Bari, Italy, 1996. URL `citeseer.nj.nec.com/koller96toward.html`.

S. Kullback and R.A. Liebler. On information and sufficiency. *Annals of Mathematical Statistics*, 22:76–86, 1951.

H. Liu and L. Yu. Toward integrating feature selection algorithms for classification and clustering. *IEEE Trans. Knowledge and Data Eng.*, 17(4):491–502, 2005.

S. Mukherjee, P. Niyogi, T. Poggio, and R. Rifkin. Learning theory: Stability is sufficient for generalization and necessary and sufficient for consistency of empirical risk minimization. *Advances in Computational Mathematics*, 25:161–193, 2006.

B. Parmanto, P. Munro, and H. Doyle. Improving committee diagnosis with resampling techniques. In D. S. Touretzky, M. C. Mozer, and M. Hesselmo, editors, *Advances in Neural Information Processing Systems 8*, pages 882–888. Cambridge, MA: MIT Press, 1996.

T. Poggio, R. Rifkin, S. Mukherjee, and A. Rakhlin. Bagging regularizes. In *CBCL Paper 214/AI Memo 2002-003*. MIT, Cambridge, MA, 2002.

T. Poggio, R. Rifkin, S. Mukherjee, and P. Niyogi. General conditions for predictivity in learning theory. *Nature*, 428:419–422, 2004.

M. Robnik-Sikonja and I. Kononenko. Theoretical and empirical analysis of relief and relieff. *Machine Learning*, 53:23–69, 2003.

C. Stanfill and D. Waltz. Toward memory-based reasoning. *Communications of the ACM*, 29: 1213–1228, December 1986.

A. Statnikov and C.F. Aliferis. Tied: An artificially simulated dataset with multiple Markov boundaries. *Journal of Machine Learning Research Workshop Conference & Proceedings*, 2009. to appear.

H. Stoppiglia, G. Dreyfus, R. Dubois, and Y. Oussar. Ranking a random feature for variable and feature selection. *Journal of Machine Learning Research*, 3:1399–1414, March 2003.

E. Tuv. Ensemble learning and feature selection. In I. Guyon, S. Gunn, M. Nikravesh, and L. Zadeh, editors, *Feature Extraction, Foundations and Applications*. Springer, 2006.

E. Tuv, A. Borisov, and K. Torkkola. Feature selection using ensemble based ranking against artificial contrasts. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN)*, 2006.

G. Valentini and T. Dietterich. Low bias bagged support vector machines. In *ICML 2003*, pages 752–759, 2003.

G. Valentini and F. Masulli. Ensembles of learning machines. In M. Marinaro and R. Tagliaferri, editors, *Neural Nets WIRN Vietri-02*, Lecture Notes in Computer Science. Springer-Verlag, 2002.

J.W. Wisnowski, J.R. Simpson, D.C. Montgomery, and G.C. Runger. Resampling methods for variable selection in robust regression. *Computational Statistics and Data Analysis*, 43(3):341–355, 2003.

L. Yu and H. Liu. Efficient feature selection via analysis of relevance and redundancy. *J. of Machine Learning Research*, 5:1205–1224, 2004.

Journal of Machine Learning Research 10 (2009) 1367-1385 Submitted 1/09; Revised 4/09; Published 7/09

A Parameter-Free Classification Method for Large Scale Learning

Marc Boullé MARC.BOULLE@ORANGE-FTGROUP.COM
Orange Labs
2, avenue Pierre Marzin
22300 Lannion, France

Editor: Soeren Sonnenburg

Abstract

With the rapid growth of computer storage capacities, available data and demand for scoring models both follow an increasing trend, sharper than that of the processing power. However, the main limitation to a wide spread of data mining solutions is the non-increasing availability of skilled data analysts, which play a key role in data preparation and model selection.

In this paper, we present a parameter-free scalable classification method, which is a step towards fully automatic data mining. The method is based on Bayes optimal univariate conditional density estimators, naive Bayes classification enhanced with a Bayesian variable selection scheme, and averaging of models using a logarithmic smoothing of the posterior distribution. We focus on the complexity of the algorithms and show how they can cope with data sets that are far larger than the available central memory. We finally report results on the Large Scale Learning challenge, where our method obtains state of the art performance within practicable computation time.

Keywords: large scale learning, naive Bayes, Bayesianism, model selection, model averaging

1. Introduction

Data mining is "the non-trivial process of identifying valid, novel , potentially useful, and ultimately understandable patterns in data" (Fayyad et al., 1996). Several industrial partners have proposed to formalize this process using a methodological guide named CRISP-DM, for CRoss Industry Standard Process for Data Mining (Chapman et al., 2000). The CRISP-DM model provides an overview of the life cycle of a data mining project, which consists in the following phases: business understanding, data understanding, data preparation, modeling, evaluation and deployment. Practical data mining projects involve a large variety of constraints, like scalable training algorithms, understandable models, easy and fast deployment. In a large telecommunication companies like France Telecom, data mining applies to many domains: marketing, text mining, web mining, traffic classification, sociology, ergonomics. The available data is heterogeneous, with numerical and categorical variables, target variables with multiple classes, missing values, noisy unbalanced distributions, and with numbers of instances or variables which can vary over several orders of magnitude. The most limiting factor which slows down the spread of data mining solutions is the data preparation phase, which consumes 80% of the process (Pyle, 1999; Mamdouh, 2006) and requires skilled data analysts. In this paper, we present a method[1] (Boullé, 2007) which aims at automatizing the data preparation and modeling phases of a data mining project, and which performs well on a large variety of problems. The interest of the paper, which extends the workshop abstract (Boullé, 2008), is

1. The method is available as a shareware, downloadable at `http://perso.rd.francetelecom.fr/boulle/`.

to introduce the algorithmic extensions of the method to large scale learning, when data does not fit into memory, and to evaluate them in an international challenge.

The paper is organized as follows. Section 2 summarizes the principles of our method and Section 3 focuses on its computational complexity, with a detailed description of the extensions introduced for large scale learning. Section 4 reports the results obtained on the Large Scale Learning challenge (Sonnenburg et al., 2008). Finally, Section 5 gives a summary and discusses future work.

2. A Parameter-Free Classifier

Our method, introduced in Boullé (2007), extends the naive Bayes classifier owing to an optimal estimation of the class conditional probabilities, a Bayesian variable selection and a compression-based model averaging.

2.1 Optimal Discretization

The naive Bayes classifier has proved to be very effective on many real data applications (Langley et al., 1992; Hand and Yu, 2001). It is based on the assumption that the variables are independent within each output class, and solely relies on the estimation of univariate conditional probabilities. The evaluation of these probabilities for numerical variables has already been discussed in the literature (Dougherty et al., 1995; Liu et al., 2002). Experiments demonstrate that even a simple equal width discretization brings superior performance compared to the assumption using a Gaussian distribution. In the MODL approach (Boullé, 2006), the discretization is turned into a model selection problem and solved in a Bayesian way. First, a space of discretization models is defined. The parameters of a specific discretization are the number of intervals, the bounds of the intervals and the output frequencies in each interval. Then, a prior distribution is proposed on this model space. This prior exploits the hierarchy of the parameters: the number of intervals is first chosen, then the bounds of the intervals and finally the output frequencies. The choice is uniform at each stage of the hierarchy. Finally, the multinomial distributions of the output values in each interval are assumed to be independent from each other. A Bayesian approach is applied to select the best discretization model, which is found by maximizing the probability $p(Model|Data)$ of the model given the data. Owing to the definition of the model space and its prior distribution, the Bayes formula is applicable to derive an exact analytical criterion to evaluate the posterior probability of a discretization model. Efficient search heuristics allow to find the most probable discretization given the data sample. Extensive comparative experiments report high performance.

The case of categorical variables is treated with the same approach in Boullé (2005), using a family of conditional density estimators which partition the input values into groups of values.

2.2 Bayesian Approach for Variable Selection

The naive independence assumption can harm the performance when violated. In order to better deal with highly correlated variables, the selective naive Bayes approach (Langley and Sage, 1994) exploits a wrapper approach (Kohavi and John, 1997) to select the subset of variables which optimizes the classification accuracy. 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 or text mining. The problem comes both from the search algorithm, whose complexity is quadratic in the number of variables, and from the selection process which is prone to overfitting. In Boullé (2007), the overfitting problem is tackled by relying on a Bayesian approach, where the best model is found by maximizing the probability of the model given the data. The parameters of a variable selection model are the number of selected variables and the subset of variables. A hierarchic prior is considered, by first choosing the number of selected variables and second choosing the subset of selected variables. The conditional likelihood of the models exploits the naive Bayes assumption, which directly provides the conditional probability of each label. This allows an exact calculation of the posterior probability of the models. Efficient search heuristic with super-linear computation time are proposed, on the basis of greedy forward addition and backward elimination of variables.

2.3 Compression-Based Model Averaging

Model averaging has been successfully exploited in Bagging (Breiman, 1996) using multiple classifiers trained from re-sampled data sets. In this approach, the averaged classifier uses a voting rule to classify new instances. Unlike this approach, where each classifier has the same weight, the Bayesian Model Averaging (BMA) approach (Hoeting et al., 1999) weights the classifiers according to their posterior probability. In the case of the selective naive Bayes classifier, an inspection of the optimized models reveals that their posterior distribution is so sharply peaked that averaging them according to the BMA approach almost reduces to the maximum a posteriori (MAP) model. In this situation, averaging is useless. In order to find a trade-off between equal weights as in bagging and extremely unbalanced weights as in the BMA approach, a logarithmic smoothing of the posterior distribution, called compression-based model averaging (CMA), is introduced in Boullé (2007). Extensive experiments demonstrate that the resulting compression-based model averaging scheme clearly outperforms the Bayesian model averaging scheme.

3. Complexity Analysis

In this section, we first recall the algorithmic complexity of the algorithms detailed in Boullé (2005, 2006, 2007) in the case where all the data fits in central memory, then introduce the extension of the algorithms when data exceeds the central memory.

3.1 When Data Fits into Central Memory

The algorithm consists in three phases: data preprocessing using discretization or value grouping, variable selection and model averaging.

3.1.1 PREPROCESSING

In the case of numerical variables, the discretization Algorithm 1 exploits a greedy-heuristic. It first sorts the input values, then starts with initial single value intervals and searches for the best merge between adjacent intervals. This merge is performed if the cost (evaluation) of the discretization decreases after the merge and the process is reiterated until no further merge can decrease the discretization cost. With a straightforward implementation of the algorithm, the method runs in $O(N^3)$ time, where N is the number of instances. However, the method can be optimized in $O(N \log N)$ time owing to the additivity of the discretization cost with respect to the intervals and using a maintained sorted list of the evaluated merges (such as an AVL binary search tree for example).

Algorithm 1 Discretization

```
Input: $X_k$    // Numerical input variable
Output: $D$    // Best discretization
 1: // Initialization
 2: Sort the input values
 3: $D$ = Elementary discretization with one interval per value value
 4: // Optimization
 5: $cost_{best} = cost(D), m_{best} = \emptyset$
 6: repeat
 7:   for all merge $m$ between adjacent intervals do
 8:     if $(cost(D \cup m) < cost_{best})$ then
 9:       $cost_{best} = cost(D \cup m), m_{best} = m$
10:     end if
11:   end for
12:   if $(cost_{best} = cost(D))$ then
13:     $D = D \cup m_{best}$
14:   end if
15: until no improvement
```

The case of categorical variables is handled using a similar greedy heuristic. Overall, the preprocessing phase is super-linear in time and requires $O(KN\log N)$ time, where K is the number of variables and N the number of instances.

3.1.2 VARIABLE SELECTION

In the variable selection Algorithm 2, the method alternates fast forward and backward variable selection steps based on randomized reorderings of the variables, and repeats the process several times in order to better explore the search space and reduce the variance caused by the dependence over the order of the variables.

Evaluating one variable selection for the naive Bayes classifier requires $O(KN)$ time, since the conditional probabilities, which are linear in the number of variables, require $O(K)$ time per instance. Updating the evaluation of a variable subset by adding or dropping one single variable requires only $O(1)$ time per instance and thus $O(N)$ time for all the instances. Therefore, one loop of fast forward or backward variable selection, which involves $O(K)$ adds or drops of variables, can be evaluated in $O(KN)$ time.

In order to bound the time complexity, each succession of fast forward and backward variable selection is repeated only twice ($MaxIter = 2$ in algorithm 2), which is sufficient in practice to be close to convergence within a small computation time.

The number of repeats of the outer loop is fixed to $\log_2 N + \log_2 K$, so that the overall time complexity of the variable selection algorithm is $O(KN(\log K + \log N))$, which is comparable to that of the preprocessing phase.

Algorithm 2 Variable Selection

```
Input: $X = (X_1, X_2, \ldots X_K)$ // Set of input variables
Output: $B$ // Best subset of variables
 1: $B = \emptyset$ // Start with an empty subset of variables
 2: for Step=1 to $\log_2 KN$ do
 3:   // Fast forward backward selection
 4:   $S = \emptyset$ // Initialize an empty subset of variables
 5:   $Iter = 0$
 6:   repeat
 7:     $Iter = Iter + 1$
 8:     $X' = \text{Shuffle}(X)$ // Randomly reorder the variables to add
 9:     // Fast forward selection
10:     for $X_k \in X'$ do
11:       if $(cost(S \cup \{X_k\}) < cost(S))$ then
12:         $S = S \cup \{X_k\}$
13:       end if
14:     end for
15:     $X' = \text{Shuffle}(X)$ // Randomly reorder the variables to remove
16:     // Fast backward selection
17:     for $X_k \in X'$ do
18:       if $(cost(S - \{X_k\}) < cost(S))$ then
19:         $S = S - \{X_k\}$
20:       end if
21:     end for
22:   until no improvement or $Iter \geq MaxIter$
23:   // Update best subset of variables
24:   if $(cost(S) < cost(B))$ then
25:     $B = S$
26:   end if
27: end for
```

3.1.3 MODEL AVERAGING

The model averaging algorithm consists in collecting all the models evaluated in the variable selection phase and averaging then according to a logarithmic smoothing of their posterior probability, with no overhead on the time complexity.

Overall, the preprocessing, variable selection and model averaging have a time complexity of $O(KN(\log K + \log N))$ and a space complexity of $O(KN)$.

3.2 When Data Exceeds Central Memory

Whereas standard scalable approaches, such as data sampling or online learning, cannot process all the data simultaneously and have to consider simpler models, our objective is to enhance the scalability of our algorithm without sacrificing the quality of the results.

With the $O(KN)$ space complexity, large data sets cannot fit into central memory and training time becomes impracticable as soon as memory pages have to be swapped between disk and central memory.[2] To avoid this strong limitation, we enhance our algorithm with a specially designed chunking strategy.

3.2.1 CHUNKING STRATEGY

First of all, let us consider the access time from central *m*emory (t_m) and from sequential *r*ead (t_r) or random disk *a*ccess (t_a). In modern personal computers (year 2008), t_m is in the order of 10 nanoseconds. Sequential read from disk is so fast (based on 100 Mb/s transfer rates) that t_r is in the same order as t_m: CPU is sometimes a limiting factor, when parsing operations are involved. Random disk access t_a is in the order of 10 milliseconds, one million times slower than t_m or t_r. Therefore, the only way to manage very large amounts of memory space is to exploit disk in a sequential manner.

Let S=$O(KN)$ be the size of the data set and M the size of the central memory. Our chunking strategy consists in minimizing the number of exchanges between disk and central memory. The overhead of the scalable version of the algorithms will be expressed in terms of sequential disk read time t_r.

3.2.2 PREPROCESSING

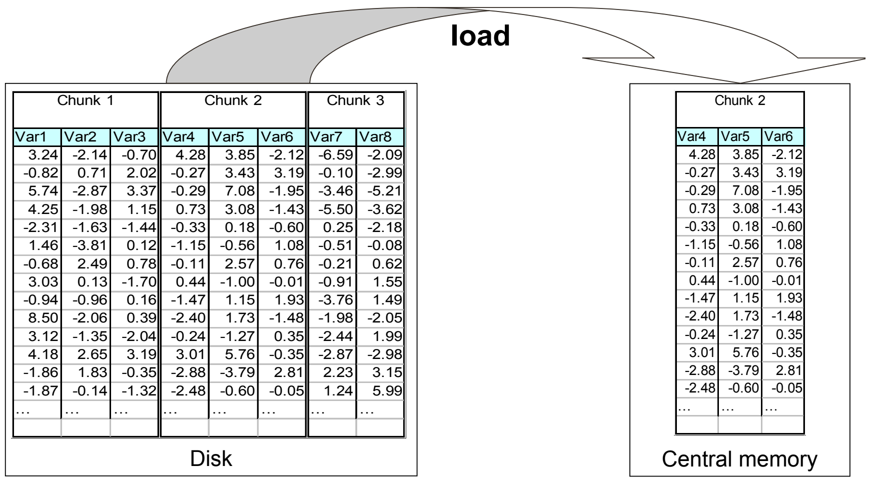
For the preprocessing phase of our method, each variable is analyzed once after being loaded into central memory. The discretization Algorithm 1 extensively accesses the values of the analyzed variable, both in the initialization step (values are sorted) and in the optimization step. We partition the set of input variables into $C = \lceil S/M \rceil$ chunks of $K_1, \ldots, K_C$ variables, such that each chunk can be loaded into memory ($K_c N < M, 1 \leq c \leq C$). The preprocessing phase loops on the C subsets, and at each step of the loop, reads the data set, parses and loads the chunk variables only, as shown in Figure 1. Each memory chunk is preprocessed as in Algorithm 1, in order to induce conditional probability tables (CPT) for the chunk variables.

After the preprocessing, the chunk variables are recoded by replacing the input values by their index in the related CPT. The recoded chunk of variables are stored in C files of integer indexes, as shown in Figure 2. These recoded chunks are very fast to load, since the chunk variables only need to be read, without any parsing (integer indexes are directly transfered from disk to central memory).

The scalable version of the preprocessing algorithm is summarized in Algorithm 3. Each input variable is read C times, but parsed and loaded only once as in the standard case. The overhead in terms of disk read time is $O((C-1)KNt_r)$ time for the load steps. In the recoding step, each variable is written once on disk, which involves an overhead of $O(KNt_r)$ (disk sequential write time is similar to disk sequential read time). Overall, the overhead is $O(CKNt_r)$ time.

This could be improved by first splitting the input data file into C initial disk chunks (without parsing), with an overall overhead of only $O(3KNt_r)$ time (read and write all the input data before preprocessing, and write recoded chunks after preprocessing). However, this involves extra disk space and the benefit is small in practice, since even C read steps are not time expensive compared to the rest of the algorithm.

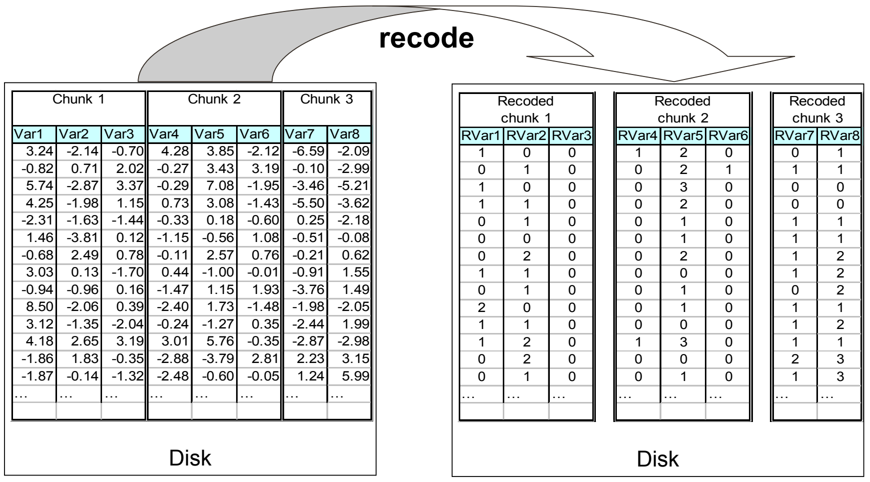
2. On 32 bits CPU, central memory is physically limited to 4 Go, or even to 2 Go on Windows PCs.



Chunk 1			Chunk 2			Chunk 3	
Var1	Var2	Var3	Var4	Var5	Var6	Var7	Var8
3.24	-2.14	-0.70	4.28	3.85	-2.12	-6.59	-2.09
-0.82	0.71	2.02	-0.27	3.43	3.19	-0.10	-2.99
5.74	-2.87	3.37	-0.29	7.08	-1.95	-3.46	-5.21
4.25	-1.98	1.15	0.73	3.08	-1.43	-5.50	-3.62
-2.31	-1.63	-1.44	-0.33	0.18	-0.60	0.25	-2.18
1.46	-3.81	0.12	-1.15	-0.56	1.08	-0.51	-0.08
-0.68	2.49	0.78	-0.11	2.57	0.76	-0.21	0.62
3.03	0.13	-1.70	0.44	-1.00	-0.01	-0.91	1.55
-0.94	-0.96	0.16	-1.47	1.15	1.93	-3.76	1.49
8.50	-2.06	0.39	-2.40	1.73	-1.48	-1.98	-2.05
3.12	-1.35	-2.04	-0.24	-1.27	0.35	-2.44	1.99
4.18	2.65	3.19	3.01	5.76	-0.35	-2.87	-2.98
-1.86	1.83	-0.35	-2.88	-3.79	2.81	2.23	3.15
-1.87	-0.14	-1.32	-2.48	-0.60	-0.05	1.24	5.99
…	…	…	…	…	…	…	…

Chunk 2		
Var4	Var5	Var6
4.28	3.85	-2.12
-0.27	3.43	3.19
-0.29	7.08	-1.95
0.73	3.08	-1.43
-0.33	0.18	-0.60
-1.15	-0.56	1.08
-0.11	2.57	0.76
0.44	-1.00	-0.01
-1.47	1.15	1.93
-2.40	1.73	-1.48
-0.24	-1.27	0.35
3.01	5.76	-0.35
-2.88	-3.79	2.81
-2.48	-0.60	-0.05
…	…	…

Figure 1: Preprocessing: the input file on disk is read several times, with one chunk of variables parsed and loaded into central memory one at a time.



Chunk 1			Chunk 2			Chunk 3	
Var1	Var2	Var3	Var4	Var5	Var6	Var7	Var8
3.24	-2.14	-0.70	4.28	3.85	-2.12	-6.59	-2.09
-0.82	0.71	2.02	-0.27	3.43	3.19	-0.10	-2.99
5.74	-2.87	3.37	-0.29	7.08	-1.95	-3.46	-5.21
4.25	-1.98	1.15	0.73	3.08	-1.43	-5.50	-3.62
-2.31	-1.63	-1.44	-0.33	0.18	-0.60	0.25	-2.18
1.46	-3.81	0.12	-1.15	-0.56	1.08	-0.51	-0.08
-0.68	2.49	0.78	-0.11	2.57	0.76	-0.21	0.62
3.03	0.13	-1.70	0.44	-1.00	-0.01	-0.91	1.55
-0.94	-0.96	0.16	-1.47	1.15	1.93	-3.76	1.49
8.50	-2.06	0.39	-2.40	1.73	-1.48	-1.98	-2.05
3.12	-1.35	-2.04	-0.24	-1.27	0.35	-2.44	1.99
4.18	2.65	3.19	3.01	5.76	-0.35	-2.87	-2.98
-1.86	1.83	-0.35	-2.88	-3.79	2.81	2.23	3.15
-1.87	-0.14	-1.32	-2.48	-0.60	-0.05	1.24	5.99
…	…	…	…	…	…	…	…

Recoded chunk 1			Recoded chunk 2			Recoded chunk 3	
RVar1	RVar2	RVar3	RVar4	RVar5	RVar6	RVar7	RVar8
1	0	0	1	2	0	0	1
0	1	0	0	2	1	1	1
1	0	0	0	3	0	0	0
1	1	0	0	2	0	0	0
0	1	0	0	1	0	1	1
0	0	0	0	1	0	1	1
0	2	0	0	2	0	1	2
1	1	0	0	0	0	1	2
0	1	0	0	1	0	0	2
2	0	0	0	1	0	1	1
1	1	0	0	0	0	1	2
1	2	0	1	3	0	1	1
0	2	0	0	0	0	2	3
0	1	0	0	1	0	1	3
…	…	…	…	…	…	…	…

Figure 2: Recoding: the input file on disk is read and recoded once, and several recoded chunks of variables are written on disk.

Algorithm 3 Scalable preprocessing

```
Input: input data on disk
Output: recoded data on several recoded disk chunks, conditional probability tables
 1: // Compute chunking parameters
 2: estimate input data size $S$ and available central memory size $M$
 3: compute number of chunks: $C = \lceil S/M \rceil$
 4: partition the set of variables into $C$ balanced chunks of variables
 5: // Main loop
 6: for all chunk do
 7:     // Load chunk into central memory
 8:     for all variable do
 9:         read input data
10:         if (variable in current chunk) then
11:             parse and load variable
12:         end if
13:     end for
14:     // Preprocessing
15:     for all variable in memory chunk do
16:         call the discretization or value grouping algorithm
17:         keep the conditional probability table (CPT) into memory
18:     end for
19:     // Recoding
20:     for all variable in memory chunk do
21:         write a recoded chunk by replacing each input value by its index in the CPT
22:     end for
23: end for
```

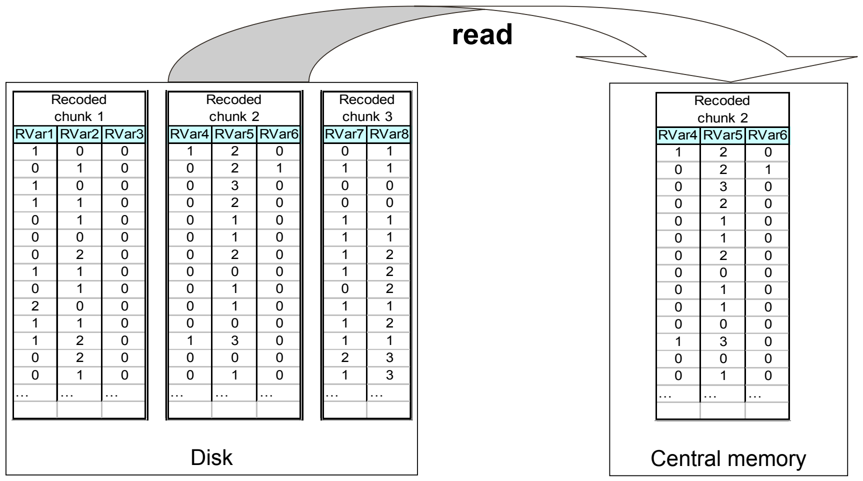
3.2.3 VARIABLE SELECTION

In the variable selection phase, the algorithm performs $\mathrm{O}(\log K + \log N)$ steps of fast forward backward variable selection, based on random reordering of the variables. Each step evaluates the add or drop of one variable at a time, so that the whole input data has to be read.

In the scalable version of the variable selection phase, we exploit the recoded chunks of variables as shown in Figure 3: one single chunk is loaded in memory at a time. Instead of reordering randomly all the variables, we first reorder randomly the list of recoded chunks, then reorder randomly the subset of variables in the recoded chunk currently loaded in memory. This "double stage" reordering is the only place where the scalable algorithm does not exactly mirror the behavior of the standard algorithm.

Each step of fast forward backward variable selection involves reading each recoded chunk $\mathrm{O}(1)$ times (exactly $2 * MaxIter$ times with $MaxIter = 2$), thus implies an overhead of $\mathrm{O}(KNt_r)$ read time.

Overall, the algorithmic overhead consists of additional steps for reading the input data: $C \approx S/M$ steps for the preprocessing phase and $(\log K + \log N)$ read for variable selection phase, with an overhead of $\mathrm{O}(KN(S/M + \log K + \log N)t_r)$ time. Since sequential disk read time t_s is comparable to memory access time, we expect that the scalable version of the algorithm has a practicable computation time.



Recoded chunk 1		
RVar1	RVar2	RVar3
1	0	0
0	1	0
1	0	0
1	1	0
0	1	0
0	0	0
0	2	0
1	1	0
0	1	0
2	0	0
1	1	0
1	2	0
0	2	0
0	1	0
…	…	…

Recoded chunk 2		
RVar4	RVar5	RVar6
1	2	0
0	2	1
0	3	0
0	2	0
0	1	0
0	1	0
0	2	0
0	0	0
0	1	0
0	1	0
0	0	0
1	3	0
0	0	0
0	1	0
…	…	…

Recoded chunk 3	
RVar7	RVar8
0	1
1	1
0	0
0	0
1	1
1	1
1	2
1	2
0	2
1	1
1	2
1	1
2	3
1	3
…	…

Recoded chunk 2		
RVar4	RVar5	RVar6
1	2	0
0	2	1
0	3	0
0	2	0
0	1	0
0	1	0
0	2	0
0	0	0
0	1	0
0	1	0
0	0	0
1	3	0
0	0	0
0	1	0
…	…	…

Figure 3: Read: the recoded chunks of variables are read and loaded (no parsing) into central memory one at a time.

4. Results on the Large Scale Learning Challenge

In this section, we summarize the Pascal Large Scale Learning Challenge (Sonnenburg et al., 2008) and describe our submission. We then analyze the advantages and limitations of our non-parametric modeling approach on large scale learning and evaluate our chunking strategy. We finaly report the challenge results.

4.1 Challenge Data Sets and Protocol

The challenge is designed to enable a direct comparison of learning methods given limited resource. The data sets, summarized in Table 1 represent a variety of domains and contain up to millions of instances, thousands of variables and tens of gigabytes of disk space.

The challenge data sets are a convenient way to evaluate the scalability of our offline method, when data set size is larger than the central memory by one order of magnitude (on computers with 32 bit CPU). In the Wild Competition Track of the challenge, the participants are "free to do anything that leads to more efficient, more accurate methods, for example, perform feature selection, find effective data representations, use efficient program-code, tune the core algorithm etc". The method have to be trained on subsets of the train data set of increasing size, with sizes $10^2, 10^3, 10^4, 10^5, \ldots$ up to the maximum number of train examples.

For all the training sessions, the accuracy is estimated using the area over the precision recall curve (aoPRC) criterion. The training time is estimated by the organizer in the final evaluation by a re-run of the algorithm on a single CPU Linux machine, 32 Gb RAM. It is noteworthy that the evaluated training time is the computation time for the optimal parameter settings, excluding data loading times and any kind of data preparation or model selection time.

Data set	Domain	Training	Validation	Dimensions	Best aoPRC	Our aoPRC
alpha	artificial	500000	100000	500	0.1345	0.2479
beta	artificial	500000	100000	500	0.4627	**0.4627**
gamma	artificial	500000	100000	500	0.0114	**0.0114**
delta	artificial	500000	100000	500	0.0798	**0.0798**
epsilon	artificial	500000	100000	2000	0.0343	0.0625
zeta	artificial	500000	100000	2000	0.0118	0.0305
fd	face detection	5469800	532400	900	0.1829	**0.1829**
ocr	character recognition	3500000	670000	1156	0.1584	0.1637
dna	dna split point detection	50000000	1000000	200	0.8045	0.8645
webspam	webspam detection	350000	50000	variable	0.0004	0.0031

Table 1: Challenge data sets and final best vs our results for the aoPRC criterion.

The overall results are summarized based on performance figures: displaying training time vs. test error, data set size vs. test error and data set size vs. training time. A final ranking is computed by aggregating a set of performance indicators computed from these figures.

4.2 Our Submission

We made one fully automatic submission, using the raw representation of the data sets, which is numerical for all the data sets except for dna where the variables are categorical.

For the two image-based data sets (fd and ocr), we also studied the impact of initial representation using centered reduced rows, and finally chose the representation leading to the best results: raw for ocr and centered-reduced for fd.

The webspam data set is represented using a tri-gram representation, which implies 16 millions of potential variables ($K = 256^3 \approx 16 * 10^6$). Although this is a sparse data set, our algorithm exploits a dense representation and cannot manage such a large number of variables. We applied a simple dimensionality reduction method inspired from the random projection technique (Vempala, 2004). We choose to build a compact representation, using a set of $K' = 10007$ (first prime number beyond 10000) constructed variables: each constructed feature is the sum of the initial variables having the same index modulo K':

$$\forall k', 0 \leq k' < K', Var_{k'} = \sum_{k, k \bmod K' = k'} Var_k.$$

As our method is available as a scoring tool for Windows end users, it is implemented for 32 bits CPUs and limited to 2 Gb RAM, thus does not benefit from the 32 Gb RAM allowed in the challenge. The algorithmic extensions presented in Section 3.2 are thus extensively exploited for the largest training sessions.

4.3 Non-Parametric Modeling with Large Data Sets

In Boullé (2007), we have analyzed the accuracy performance of our method using extensive experiments. In the section, we give new insights on our non-parametric approach and its impact on large scale learning. Actually, the MODL method considers a space of discretization models which grows with the size of the training data set. The number of intervals is taken between 1 to N intervals, where N is the number of training instances, the boundaries come from the input data and the class conditional parameters per interval are confined to a finite number of frequencies (rather than continuous distributions in [0, 1]), on the basis on instance counts in the data. Both the space of discretization models and the prior distribution on the model parameters are data-dependent. More precisely, they depends on the input data, and the best model, with the maximum a posteriori probability, is selected using the output data.

To illustrate this, we chose the delta data set and exploit the 500000 labeled instances for our experiment. We use training sets of increasing sizes 100, 1000, 10000, 100000 and 400000, and keep the last 100000 instances as a test set. The best variable according to the MODL criterion is Var150 for all data set sizes above 1000. In Figure 4, we draw the discretization (with boundaries and conditional probability of the positive class) of Var150, using the MODL method and the unsupervised equal frequency method with 10 intervals, as a baseline. For data set size 100, the empirical class distribution is noisy, and the MODL method selects a discretization having one single interval, leading to the elimination of the variable. When the data set size increases, the MODL discretizations contain more and more intervals, with up to 12 intervals in the case of data set size 400000. Compared to the fixed size equal frequency discretization, the MODL method builds discretizations which are both more accurate and robust and obtain a good approximation of the underlying class distribution.

This also demonstrates how the univariate MODL preprocessing method behaves as a variable ranking method with a clear threshold: any input variable discretized into one single interval does not contain any predictive information and can be eliminated. In Figure 5, we report the number of selected variables per data set size, which goes from 6 for data set size 100 to about 350 for data set size 400000. Figure 5 also report the test AUC (area under the ROC curve (Fawcett, 2003)), which significantly improves with the data set size and exhibits a clear correlation with the number of selected variables.

One drawback of this approach is that all the training data have to be exploited simultaneously to unable a deep optimization of the model parameters and fully benefit from their expressiveness. Although learning with very large data sets is possible, as shown in this paper, the approach is limited to offline learning and does not apply to the one-pass learning paradigm or to learning on stream.

Overall, our non-parametric approach build models of increasing complexity with the size of the training data set, in order to better approximate the underlying class distribution. Small training data sets lead to simple models, which are fast to deploy (few selected variables with simple discretization), whereas large training data sets allow more accurate models at the expense of more training time and deployment time.

4.4 Impact of the Chunking Strategy

In Section 3.2, we have proposed a chunking strategy that allows to learn with training data sets which size is far larger than central memory. The only difference with the standard algorithm is the “double stage” random reordering of the variables, which is performed prior to each step of fast forward or backward variable selection in Algorithm 2. In this section, we evaluate our chunking strategy and its impact on training time and test AUC.

We exploit the delta data set with trainings sets of increasing sizes 100, 1000, 10000, 10000, 400000, and a test set of 100000 instances. The largest training size fits into central memory on our

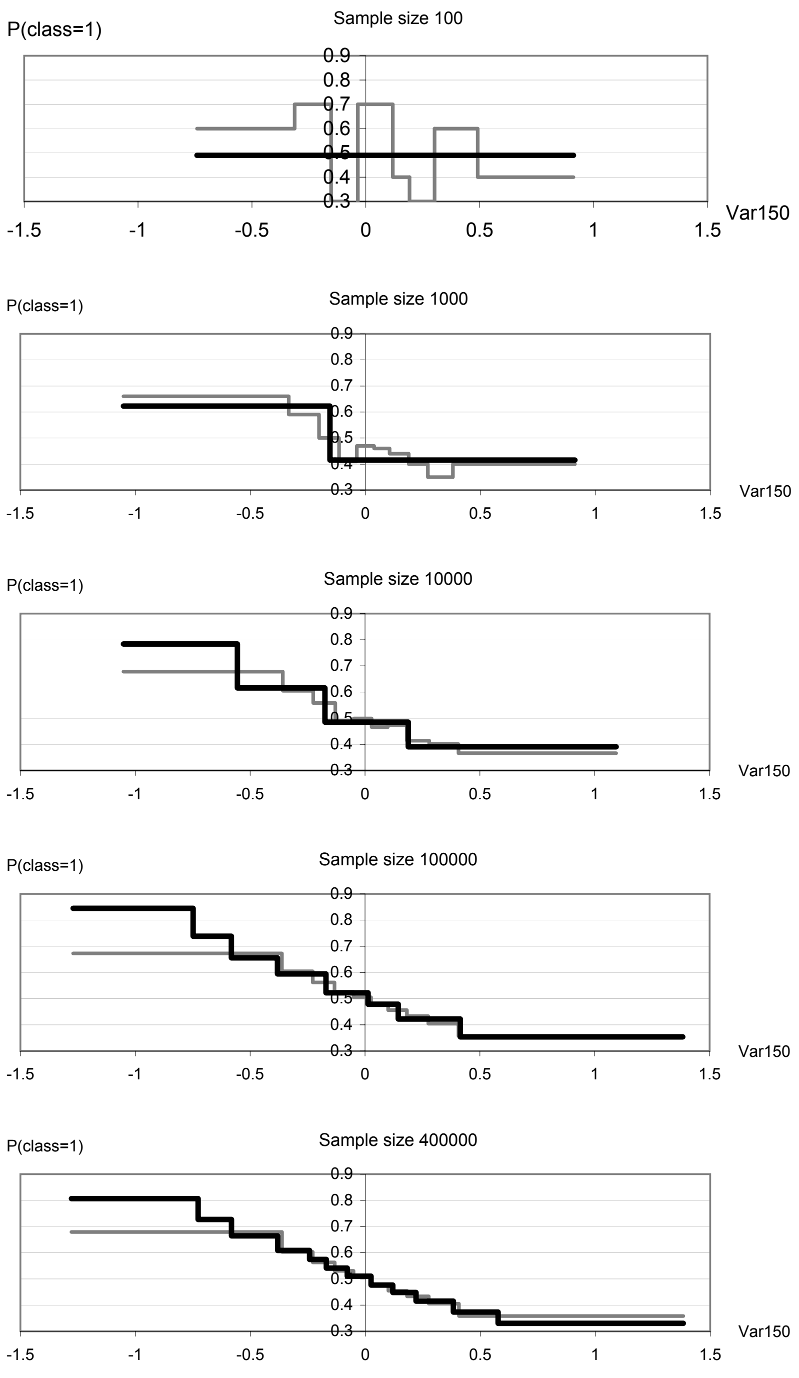
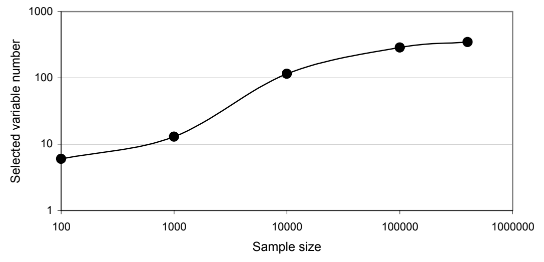


Figure 4: Discretization of Var150 for the delta data set, using the MODL method (in black) and the 10 equal frequency method (in gray).



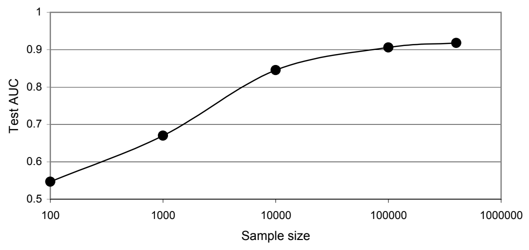
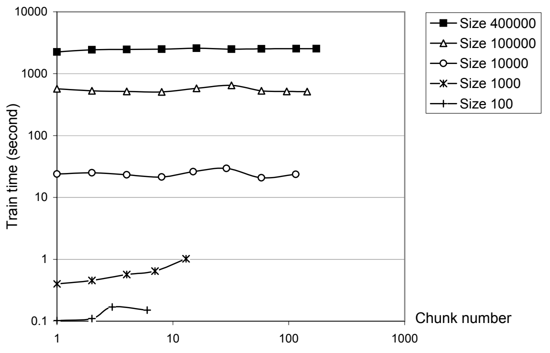


Figure 5: Data set size versus selected variable number and test AUC for the delta data set.

machine (Pentium 4, 3.2 Ghz, 2 Go RAM, Windows XP), which allows to compare the standard algorithm with the chunking strategy. In Figure 6, we report the wall-clock variable selection training time and the test AUC for the standard algorithm (chunk number = 1) and the chunking strategy, for various numbers of chunks.



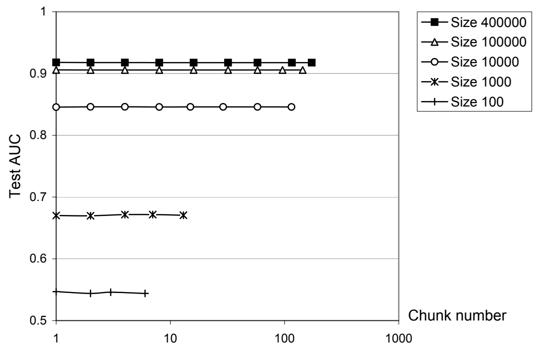


Figure 6: Impact on the chunking strategy on the wall-clock variable selection training time and the test AUC for the delta data set.

The results show that the impact on test AUC is insignificant: the largest differences go from 0.003 for data set size 100 to 0.0004 for data set size 400000. The variable selection training time is almost constant with the number of chunks. Compared to the standard algorithm, the time overhead is only 10% (for data set size above 10000), whatever be the number of chunks. Actually, as analyzed in Section 3.2, the recoded chunks of variables are extremely quick to load into memory, resulting in just a slight impact on training time.

We perform another experiment using the fd data set, with up to 5.5 millions of instances and 900 variables, which is one order of magnitude larger than our central memory resources. Figure 7 reports the wall-clock full training time, including load time, preprocessing and variable selection. When the size of the data sets if far beyond the size of the central memory, the overhead in wall-clock time is by only a factor two. This overhead mainly comes from the preprocessing phase of the algorithm (see Algorithm 3), which involves several reads of the initial training data set. Overall, the wall-clock training time is about one hour per analyzed gigabyte.

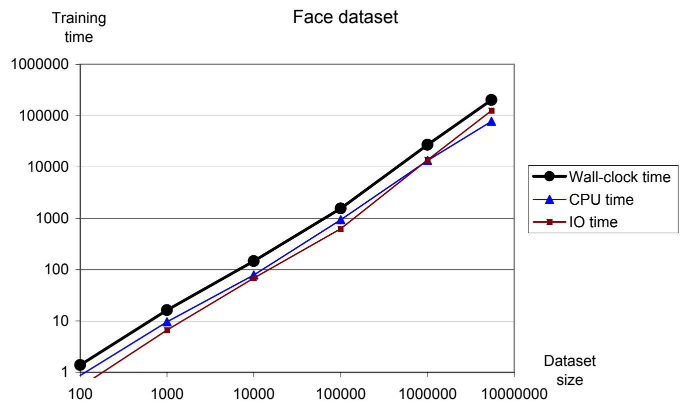


Figure 7: Training time for the fd data set.

Rank	Score	Submitter	Title
1	2.6	chap - Olivier Chapelle	Newton SVM
2	2.7	antoine - Antoine Bordes	SgdQn
3	3.2	yahoo - Olivier Keerthi	SDM SVM L2
4	3.5	yahoo - Olivier Keerthi	SDM SVM L1
5	5	MB - Marc Boulle	Averaging of Selective Naive Bayes Classifiers final
6	5.8	beaker - Gavin Cawley	LR
7	6.8	kristian - Kristian Woodsend	IPM SVM 2
8	7.3	ker2 - Porter Chang	CTJ LSVM01
9	7.7	antoine - Antoine Bordes	LaRankConverged
10	8	rofu - Rofu yu	liblinear

Table 2: Challenge final ranking.

4.5 Challenge Results

The challenge attracted 49 participants who registered, for a total of 44 submissions. In the final evaluation, our submission is ranked 5^{th} using the the aggregated performance indicator of the organizers, as shown in Table 2. To go beyond this aggregated indicator and analyze the intrinsic multi-criterion nature of the results, we now report and comment separately the training time and accuracy performance of our method.

4.5.1 TRAINING TIME

Most of the other competitors exploit linear SVM, using for example Newton based optimization in the primal (Chapelle, 2007), Sequential Dual Method optimization (Hsieh et al., 2008) or stochastic gradient optimization (Bordes and Bottou, 2008). These method were applied using a specific data preparation per data set, such as 0-1, L1 or L2 normalization, and the model parameters were tuned using most of the available training data, even for the smallest training sessions. They exploit incremental algorithms, similar to online learning methods: they are very fast and need only a few passes on the data to reach convergence.

We report in Figure 8 the training time vs data set size performance curves (in black for our results).[3] As far as the computation time only is considered, our offline training time is much longer than that of the online methods of the other competitors, by about two orders of magnitude. However, when the overall process is accounted for, with data preparation, model selection and data loading time, our training time becomes competitive, with about one hour of training time per analyzed gigabyte.

4.5.2 ACCURACY

Table 1 reports our accuracy results in the challenge for the area over the precision recall curve (aoPRC) criterion in the final evaluation of the challenge. The detailed results are presented in Figure 9, using the data set size vs accuracy performance curves.

On the webspam data set, the linear SVM submissions of Keerthi obtain excellent performance, which confirms the effectiveness of SVM for text categorization (Joachims, 1998). On the alpha data set, our method (based on the naive Bayes assumption) fails to exploit the quadratic nature of this artificial data set and obtains poor results. Except for this data set, our method always ranks among the first competitors and obtains the 1^{st} place with a large margin on four data sets: beta, gamma, delta and fd. This is a remarkable performance for a fully automatic method, which exploits the limited naive Bayes assumption.

Overall, our method is highly scalable and obtains competitive performance fully automatically, without tuning any parameter.

5. Conclusion

We have presented a parameter-free classification method that exploits the naive Bayes assumption. It estimates the univariate conditional probabilities using the MODL method, with Bayes optimal discretizations and value groupings for numerical and categorical variables. It searches for a subset of variables consistent with the naive Bayes assumption, using an evaluation based on a Bayesian model selection approach and efficient add-drop greedy heuristics. Finally, it combines all the evaluated models using a compression-based averaging schema. In this paper, we have introduced a carefully designed chunking strategy, such that our method is no longer limited to data sets that fit into central memory.

Our classifier is aimed at automatically producing competitive predictions in a large variety of data mining contexts. Our results in the Large Scale Learning Challenge demonstrates that our method is highly scalable and automatically builds state-of-the art classifiers. When the data set size is larger than the available central memory by one order of magnitude, our method exploits an efficient chunking strategy, with a time overhead of only a factor two.

Compared to alternative methods, our method requires all the data simultaneously to fully exploit the potential of our non-parametric preprocessing: it is thus dedicated to offline learning, not to online learning. Another limitation is our selective naive Bayes assumption. Although this is often leveraged in practice when large number of features can be constructed using domain knowledge, alternative methods might be more effective when their bias fit the data. Overall, our approach is

3. For readability reasons in this paper, we have represented our results in black and the others in gray, which allows to present our performance relatively to the distribution of all the results. On the challenge website, Figures 8 and 9 come in color, which allows to discriminate the performance of each method.

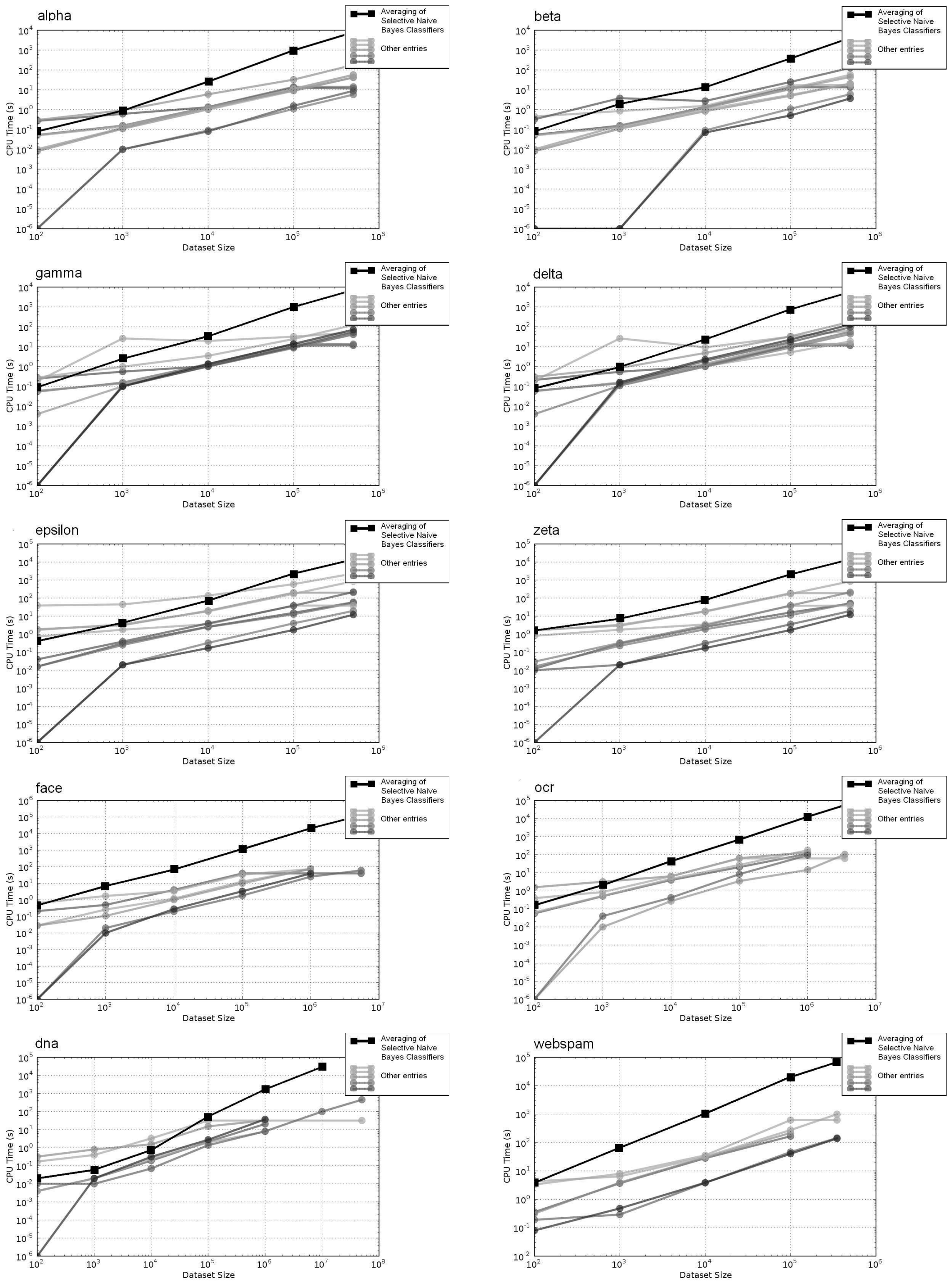


Figure 8: Final training time results: data set size versus training time (excluding data loading).

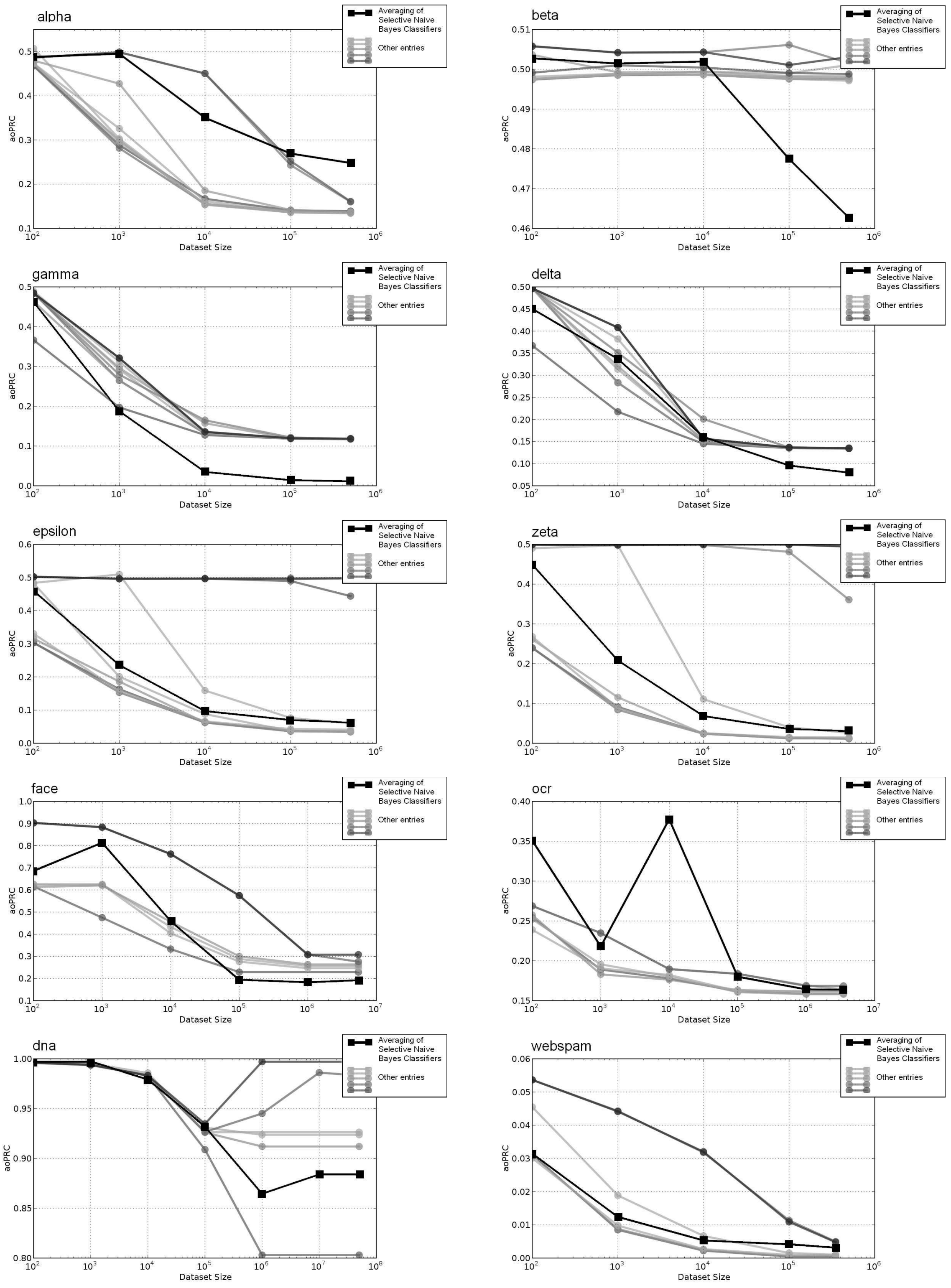


Figure 9: Final accuracy results: data set size versus test error.

of great interest for automatizing the data preparation and modeling phases of data mining, and exploits as much as possible all the available training data in its initial representation.

In future work, we plan to further improve the method and extend it to classification with large number of class values and to regression.

Acknowledgments

I am grateful to Bruno Guerraz, who got the initial Windows code to work under Linux. I would also like to thank the organizers of the Large Scale Learning Challenge for their valuable initiative.

References

A. Bordes and L. Bottou. Sgd-qn, larank: Fast optimizers for linear svms. In *ICML 2008 Workshop for PASCAL Large Scale Learning Challenge*, 2008. http://largescale.first.fraunhofer.de/workshop/.

M. Boullé. A Bayes optimal approach for partitioning the values of categorical attributes. *Journal of Machine Learning Research*, 6:1431–1452, 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*, 8:1659–1685, 2007.

M. Boullé. An efficient parameter-free method for large scale offline learning. In *ICML 2008 Workshop for PASCAL Large Scale Learning Challenge*, 2008. http://largescale.first.fraunhofer.de/workshop/.

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

O. Chapelle. Training a support vector machine in the primal. *Neural Computation*, 19:1155–1178, 2007.

P. Chapman, J. Clinton, R. Kerber, T. Khabaza, T. Reinartz, C. Shearer, and R. Wirth. *CRISP-DM 1.0 : Step-by-step Data Mining Guide*, 2000.

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.

U.M. Fayyad, G. Piatetsky-Shapiro, and P. Smyth. Knowledge discovery and data mining: towards a unifying framework. In *KDD*, pages 82–88, 1996.

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.

C-J. Hsieh, K-W. Chang, C-J. Lin, S. Keerthi, and S. Sundararajan. A dual coordinate descent method for large-scale linear svm. In *ICML '08: Proceedings of the 25th international conference on Machine learning*, pages 408–415, New York, NY, USA, 2008. ACM.

T. Joachims. Text categorization with support vector machines: Learning with many relevant features. In *European Conference on Machine Learning (ECML)*, pages 137–142, Berlin, 1998. Springer.

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.

R. Mamdouh. *Data Preparation for Data Mining Using SAS*. Morgan Kaufmann Publishers, 2006.

D. Pyle. *Data Preparation for Data Mining*. Morgan Kaufmann Publishers, Inc. San Francisco, USA, 1999.

S. Sonnenburg, V. Franc, E. Yom-Tov, and M. Sebag. Pascal large scale learning challenge, 2008. http://largescale.first.fraunhofer.de/about/.

S. Vempala. *The Random Projection Method*, volume 65 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*. American Mathematical Society, 2004.

Journal of Machine Learning Research 10 (2009) 1387-1390 Submitted 8/08; Revised 2/09; Published 7/09

Model Monitor (M^2): Evaluating, Comparing, and Monitoring Models

Troy Raeder TRAEDER@CSE.ND.EDU
Nitesh V. Chawla NCHAWLA@CSE.ND.EDU
Department of Computer Science and Engineering
University of Notre Dame
Notre Dame, IN 46556, USA

Editor: Soeren Sonenberg

Abstract

This paper presents Model Monitor (M^2), a Java toolkit for robustly evaluating machine learning algorithms in the presence of changing data distributions. M^2 provides a simple and intuitive framework in which users can evaluate classifiers under hypothesized shifts in distribution and therefore determine the best model (or models) for their data under a number of potential scenarios. Additionally, M^2 is fully integrated with the WEKA machine learning environment, so that a variety of commodity classifiers can be used if desired.

Keywords: machine learning, open-source software, distribution shift, scenario analysis

1. Introduction

Most work in machine learning implicitly assumes a *stationary distribution*, meaning that the population from which our data is drawn does not change over time. However, a number of real-world applications present the challenge of a drift in distribution between training and testing (Quiñonero et al., 2008). At the same time, the prevalent evaluation paradigm does not take into account the effect of changing distributions on the performance of the models being compared. We posit that it is important to effectively evaluate the generalization capacity of models or classifiers on a range of distribution divergences. We propose M^2, a Java toolkit that is designed to help researchers and practitioners grapple with potential shifts in data distribution. The fundamental issues that we address in this toolkit are: a) which of several competing models is most robust to changing distributions? b) when, why, and how may the learned model fail? Specifically, it carries the following fundamental capabilities.

1. Detection of distribution shift: M^2 contains methods for isolating features whose distribution has changed significantly between training and testing sets. This allows for following: a) sensitivity/fragility analysis on features with respect to distributional shifts; b) if a testing set is available, determination of features that change significantly between the training and testing sets; and c) if an effective model is beginning to degrade in performance, problematic features can be isolated and the model can be updated before its performance gets worse.

2. Exploration of hypothetical scenarios: If a user is building a model on new data and expects the distribution to change over time, he or she can inject these changes into the test data and

determine which classifiers[1] (trained on current data) perform best on both the current and hypothetical future distributions.

3. Empirical comparison of classifiers: M^2 can compare classifiers on performance across several data sets and distribution shifts. This capability allows researchers to objectively evaluate the robustness of the classifiers under distribution shift. This can guide the selection of a classifier that is more robust to distributional drifts. Various cross-validation schemes are available, and we implement statistical tests, so that the user knows whether the observed differences are statistically significant.

M^2 is, to our knowledge, the only publicly-available software to tackle the important problem of learning under changing distributions. At present, it is designed mainly for two-class problems. Most of the currently-implemented performance measures will compare the performance of the positive class (i.e., class 1) against the performance on all other classes.

The current version is available on `mloss.org`, and its distribution package contains a user's guide, developer's guide, JavaDoc documentation, and examples. To afford flexibility to the user M^2 fully integrates with WEKA (Witten et al., 1999), but also provides support for interfacing with arbitrary custom classifiers. Input data can either be in the WEKA format or the comma-delimited format of traditional UCI repository data sets (often called C4.5 style).

2. Implementation

M^2 is implemented in Java and is therefore fully cross-platform. It provides a Swing graphical user interface, and implements several different distribution shifts, methods of evaluation such as 10-fold, 5x2 CV, etc., performance measures, and statistical measures for the comparison of classifiers. Each of these critical components has been validated by a series of automated unit tests to insure correct operation and protect against regression.

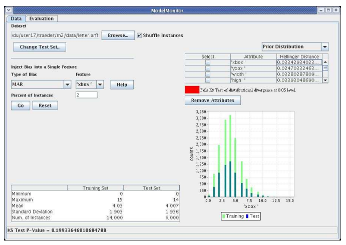
2.1 Performance Measures

In addition to the standard performance measures (*Accuracy*, *AUROC*, *Precision*, *Recall*, F_1*-Measure*), we have implemented other performance and loss measures commonly used for binary classification tasks, including *Brier Score* and *Negative Cross Entropy*. For any of the tasks described above, M^2 supports all of the standard validation measures including *cross-validation* (5-by-2 and 10-fold from the GUI, arbitrary configurations otherwise), splitting a data set into training and testing, and specifying an entirely separate file of test data.

2.2 Distribution Shifts

M^2 can introduce a number of different changes in distribution, including *missing at random*, *missing not at random*, *random noise*, *mean shift*, *variance change*, and a *prior probability shift*. Specific details on how each of the distribution shifts are implemented is available in the user's guide.

1. Model and classifier will be used inter-changeably in this paper.



Dataset: edu/user17/traeder/m2/data/letter.arff

[✓ Shuffle Instances]

Inject Bias into a Single Feature

Type of Bias: MAR

Feature: 'xbox '

Percent of Instances: 2

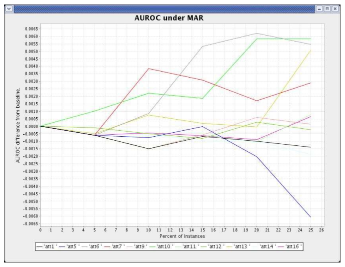
Select	Attribute	Hellinger Distance
☐	'xbox '	0.03342934023...
☐	'ybox '	0.02470332463...
☐	'width '	0.03280287809...
☐	'high '	0.03393048690...

🟥 Fails KS Test of distributional divergence at 0.05 level.

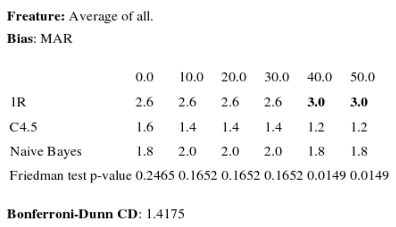
	Training Set	Test Set
Minimum	0	0
Maximum	15	14
Mean	4.03	4.007
Standard Deviation	1.903	1.936
Num. of Instances	14,000	6,000

KS Test P-Value = 0.19933646010684788

(a) Detecting distribution shift



(b) Plot of classifier sensitivity to MAR bias.



Freature: Average of all.

Bias: MAR

	0.0	10.0	20.0	30.0	40.0	50.0
1R	2.6	2.6	2.6	2.6	**3.0**	**3.0**
C4.5	1.6	1.4	1.4	1.4	1.2	1.2
Naive Bayes	1.8	2.0	2.0	2.0	1.8	1.8
Friedman test p-value	0.2465	0.1652	0.1652	0.1652	0.0149	0.0149

Bonferroni-Dunn CD: 1.4175

(c) Comparison of classifiers across multiple data sets.

Figure 1: Types of data presented to the user.

2.3 Statistical Measures

The bulk of the implementation has to do with methods for detecting and evaluating the effect of shifts in distribution. For detecting shifts in distribution, we provide both *Hellinger distance* and the *Kolmogorov-Smirnov* (KS) test. Recent work by Cieslak and Chawla (2007) has shown that systematic shifts in distribution tend to cause significant changes in Hellinger distance. The KS test helps to distinguish between changes caused by random fluctuation and changes caused by systematic bias.

For comparing the performance of multiple classifiers, we have implemented the *Friedman test* and the *Bonferroni-Dunn test*. The average performance of several classifiers, across multiple data sets is compared with the Friedman test, as suggested by Demšar (2006), and if a significant difference in performance is found, the Bonferroni-Dunn test determines the classifiers or sets of classifiers between which the significant difference exists.

3. Functionality

Here we provide a quick overview of the toolkit's major functionality. Each feature is described in greater detail in the user's guide.

3.1 Detecting Distribution Shift

When a user starts M^2, he or she can load any data set in either C4.5 names/data format or the WEKA ARFF format. The user then specifies a test set (either by selecting hold-out or cross-validation or by choosing a separate test file).

At this point, the user can begin evaluating the training and test data sets for shifts in distribution. Each attribute in the data set is automatically displayed, and selecting it will produce a histogram of both distributions so that any differences can be visually inspected. Additionally M^2 automatically calculates the Hellinger distance for each feature between the training and test sets. If a feature's distribution fails the KS test (at the $\alpha = 0.05$ confidence level), the Hellinger distance for that feature is highlighted. In this way, users can quickly locate features whose distribution may have changed. A simple example appears in Figure 1(a).

3.2 Exploration of Hypothetical Scenarios

After loading a data set, the user has the opportunity to introduce any of a number of distribution shifts into the test data. Upon doing so, the histograms displayed on the screen will be updated, and new Hellinger distances and KS-test p-values are calculated. The user can then quickly evaluate any classifier on both the original and new ("biased") test sets to compare the results. WEKA classifiers can be chosen from a GUI chooser, whereas for general classifiers, a commandline must be specified.

For more sophisticated sensitivity analysis, the user may specify a range of distribution shifts. In this case, he or she provides a set of increasingly severe biases, and then the tool evaluates the classifier's performance under each one. The results are presented to the user both graphically (plotting classifier preformance against severity of shift) and in tabular form, to facilitate further offline processing.

3.3 Empirical Comparison of Classifiers

Finally, users may test any one classifier against several others across multiple data sets and distribution shifts. For each specified distribution shift, the software will evaluate each classifier on all the data sets. It ranks the classifiers on each data set, calculates the average rank for each classifier, and then performs the Friedman test for analysis of variance on the resulting table of ranks. If the Friedman test determines (again at the 0.05 level) that there is a significant difference between the classifiers, we run the Bonferroni-Dunn test to determine exactly which classifiers are different from the specified reference classifier.

In addition to producing an annotated table of ranks containing the statistical test information, this step produces all the same output that the other portions of the program produce. Taken together, these results provide a complete picture of the classifiers' relative performance in the relevant scenarios.

Acknowledgments

The authors would like to acknowledge Ryan Lichtenwalter, Dave Cieslak, and Karsten Steinhaeuser for helpful discussions on both implementation and user interface issues.

References

D.A. Cieslak and N.V. Chawla. Detecting Fractures in Classifier Performance. *ICDM 2007*, pages 123–132, 2007.

J. Demšar. Statistical Comparisons of Classifiers over Multiple Data Sets. *JMLR*, 7:1–30, 2006.

J. Quiñonero, M. Sugiama, A. Schwaighofer, and N. D. Lawrence, editors. *Dataset Shift in Machine Learning*. MIT Press, 2008.

I.H. Witten, E. Frank, L. Trigg, M. Hall, G. Holmes, and S.J. Cunningham. Weka: Practical Machine Learning Tools and Techniques with Java Implementations. *ICONIP/ANZIIS/ANNES*, 99:192–196, 1999.

Journal of Machine Learning Research 10 (2009) 1391-1445 Submitted 3/08; Revised 4/09; Published 7/09

A Least-squares Approach to Direct Importance Estimation[*]

Takafumi Kanamori KANAMORI@IS.NAGOYA-U.AC.JP
Department of Computer Science and Mathematical Informatics
Nagoya University
Furocho, Chikusaku, Nagoya 464-8603, Japan

Shohei Hido HIDO@JP.IBM.COM
IBM Research
Tokyo Research Laboratory
1623-14 Shimotsuruma, Yamato-shi, Kanagawa 242-8502, Japan

Masashi Sugiyama SUGI@CS.TITECH.AC.JP
Department of Computer Science
Tokyo Institute of Technology
2-12-1 O-okayama, Meguro-ku, Tokyo 152-8552, Japan

Editor: Bianca Zadrozny

Abstract

We address the problem of estimating the ratio of two probability density functions, which is often referred to as the *importance*. The importance values can be used for various succeeding tasks such as *covariate shift adaptation* or *outlier detection*. In this paper, we propose a new importance estimation method that has a closed-form solution; the leave-one-out cross-validation score can also be computed analytically. Therefore, the proposed method is computationally highly efficient and simple to implement. We also elucidate theoretical properties of the proposed method such as the convergence rate and approximation error bounds. Numerical experiments show that the proposed method is comparable to the best existing method in accuracy, while it is computationally more efficient than competing approaches.

Keywords: importance sampling, covariate shift adaptation, novelty detection, regularization path, leave-one-out cross validation

1. Introduction

In the context of *importance sampling* (Fishman, 1996), the ratio of two probability density functions is called the *importance*. The problem of estimating the importance is attracting a great deal of attention these days since the importance can be used for various succeeding tasks such as *covariate shift adaptation* or *outlier detection*.

> **Covariate Shift Adaptation:** Covariate shift is a situation in supervised learning where the distributions of inputs change between the training and test phases but the conditional distribution of outputs given inputs remains unchanged (Shimodaira, 2000; Quiñonero-Candela et al., 2008). Covariate shift is conceivable in many real-world

$*$. A MATLAB® or R implementation of the proposed importance estimation algorithm, *unconstrained Least-Squares Importance Fitting* (uLSIF), is available from `http://sugiyama-www.cs.titech.ac.jp/∼sugi/software/uLSIF/`.

applications such as bioinformatics (Baldi and Brunak, 1998; Borgwardt et al., 2006), brain-computer interfaces (Wolpaw et al., 2002; Sugiyama et al., 2007), robot control (Sutton and Barto, 1998; Hachiya et al., 2008), spam filtering (Bickel and Scheffer, 2007), and econometrics (Heckman, 1979). Under covariate shift, standard learning techniques such as maximum likelihood estimation or cross-validation are biased and therefore unreliable—the bias caused by covariate shift can be compensated by weighting the loss function according to the importance (Shimodaira, 2000; Zadrozny, 2004; Sugiyama and Müller, 2005; Sugiyama et al., 2007; Huang et al., 2007; Bickel et al., 2007).

Outlier Detection: The outlier detection task addressed here is to identify irregular samples in a validation data set based on a model data set that only contains regular samples (Schölkopf et al., 2001; Tax and Duin, 2004; Hodge and Austin, 2004; Hido et al., 2008). The values of the importance for regular samples are close to one, while those for outliers tend to be significantly deviated from one. Thus the values of the importance could be used as an index of the degree of outlyingness.

Below, we refer to the two sets of samples as the *training* set and the *test* set.

A naive approach to estimating the importance is to first estimate the training and test density functions from the sets of training and test samples separately, and then take the ratio of the estimated densities. However, density estimation is known to be a hard problem particularly in high-dimensional cases if we do not have simple and good parametric density models (Vapnik, 1998; Härdle et al., 2004). In practice, such an appropriate parametric model may not be available and therefore this naive approach is not so effective.

To cope with this problem, direct importance estimation methods which do not involve density estimation have been developed recently. The *kernel mean matching* (KMM) method (Huang et al., 2007) directly gives estimates of the importance at the training inputs by matching the two distributions efficiently based on a special property of *universal reproducing kernel Hilbert spaces* (Steinwart, 2001). The optimization problem involved in KMM is a convex quadratic program, so the unique global optimal solution can be obtained using a standard optimization software. However, the performance of KMM depends on the choice of tuning parameters such as the kernel parameter and the regularization parameter. For the kernel parameter, a popular heuristic of using the median distance between samples as the Gaussian width could be useful in some cases (Schölkopf and Smola, 2002; Song et al., 2007). However, there seems no strong justification for this heuristic and the choice of other tuning parameters is still open.

A probabilistic classifier that separates training samples from test samples can be used for directly estimating the importance, for example, a *logistic regression* (LogReg) classifier (Qin, 1998; Cheng and Chu, 2004; Bickel et al., 2007). Maximum likelihood estimation of LogReg models can be formulated as a convex optimization problem, so the unique global optimal solution can be obtained. Furthermore, since the LogReg-based method only involves a standard supervised classification problem, the tuning parameters such as the kernel width and the regularization parameter can be optimized based on the standard cross-validation procedure. This is a very useful property in practice.

The *Kullback-Leibler importance estimation procedure* (KLIEP) (Sugiyama et al., 2008b; Nguyen et al., 2008) also directly gives an estimate of the importance function by matching the two distributions in terms of the Kullback-Leibler divergence (Kullback and Leibler, 1951). The optimization problem involved in KLIEP is convex, so the unique global optimal solution—which tends to be sparse—can be obtained, when linear importance models are used. In addition, the tuning parameters in KLIEP can be optimized based on a variant of cross-validation.

As reviewed above, LogReg and KLIEP are more advantageous than KMM since the tuning parameters can be objectively optimized based on cross-validation. However, optimization procedures of LogReg and KLIEP are less efficient in computation than KMM due to high non-linearity of the objective functions to be optimized—more specifically, exponential functions induced by the LogReg model or the log function induced by the Kullback-Leibler divergence. The purpose of this paper is to develop a new importance estimation method that is equipped with a build-in model selection procedure as LogReg and KLIEP and is computationally more efficient than LogReg and KLIEP.

Our basic idea is to formulate the direct importance estimation problem as a least-squares function fitting problem. This formulation allows us to cast the optimization problem as a convex quadratic program, which can be efficiently solved using a standard quadratic program solver. Cross-validation can be used for optimizing the tuning parameters such as the kernel width or the regularization parameter. We call the proposed method *least-squares importance fitting* (LSIF). We further show that the solutions of LSIF is piecewise linear with respect to the ℓ_1-regularization parameter and the entire regularization path (that is, all solutions for different regularization parameter values) can be computed efficiently based on the *parametric optimization technique* (Best, 1982; Efron et al., 2004; Hastie et al., 2004). Thanks to this regularization path tracking algorithm, LSIF is computationally efficient in model selection scenarios. Note that in the regularization path tracking algorithm, we can trace the solution path without a quadratic program solver—we just need to compute matrix inverses.

LSIF is shown to be efficient in computation, but it tends to share a common weakness of regularization path tracking algorithms, that is, *accumulation of numerical errors* (Scheinberg, 2006). The numerical problem tends to be severe if there are many change points in the regularization path. To cope with this problem, we develop an approximation algorithm in the same least-squares framework. The approximation version of LSIF, which we call *unconstrained LSIF* (uLSIF), allows us to obtain the closed-form solution that can be computed just by solving a system of linear equations. Thus uLSIF is numerically stable when regularized properly. Moreover, the leave-one-out cross-validation score for uLSIF can also be computed analytically (cf. Wahba, 1990; Cawley and Talbot, 2004), which significantly improves the computational efficiency in model selection scenarios. We experimentally show that the accuracy of uLSIF is comparable to the best existing method while its computation is faster than other methods in covariate shift adaptation and outlier detection scenarios.

Our contributions in this paper are summarized as follows. A proposed density-ratio estimation method, LSIF, is equipped with cross-validation (which is an advantage over KMM) and is computationally efficient thanks to regularization path tracking (which is an advantage over KLIEP and LogReg). Furthermore, uLSIF is computationally even more efficient since its solution and leave-one-out cross-validation score can be computed analytically in a stable manner. The proposed methods, LSIF and uLSIF, are similar in spirit to KLIEP, but the loss functions are different: KLIEP uses the log loss while LSIF and uLSIF use the squared loss. The difference of the log functions allows us to improve computational efficiency significantly.

The rest of this paper is organized as follows. In Section 2, we propose a new importance estimation procedure based on least-squares fitting (LSIF) and show its theoretical properties. In

Section 3, we develop an approximation algorithm (uLSIF) which can be computed efficiently. In Section 4, we illustrate how the proposed methods behave using a toy data set. In Section 5, we discuss the characteristics of existing approaches in comparison with the proposed methods and show that uLSIF could be a useful alternative to the existing methods. In Section 6, we experimentally compare the performance of uLSIF and existing methods. Finally in Section 7, we summarize our contributions and outline future prospects. Those who are interested in practical implementation may skip the theoretical analyses in Sections 2.3, 3.2, and 3.3.

2. Direct Importance Estimation

In this section, we propose a new method of direct importance estimation.

2.1 Formulation and Notation

Let $\mathcal{D} \subset (\mathbb{R}^d)$ be the data domain and suppose we are given independent and identically distributed (i.i.d.) training samples $\{x_i^{\mathrm{tr}}\}_{i=1}^{n_{\mathrm{tr}}}$ from a training distribution with density $p_{\mathrm{tr}}(x)$ and i.i.d. test samples $\{x_j^{\mathrm{te}}\}_{j=1}^{n_{\mathrm{te}}}$ from a test distribution with density $p_{\mathrm{te}}(x)$:

$$\{x_i^{\mathrm{tr}}\}_{i=1}^{n_{\mathrm{tr}}} \overset{i.i.d.}{\sim} p_{\mathrm{tr}}(x),$$
$$\{x_j^{\mathrm{te}}\}_{j=1}^{n_{\mathrm{te}}} \overset{i.i.d.}{\sim} p_{\mathrm{te}}(x).$$

We assume that the training density is strictly positive, that is,

$$p_{\mathrm{tr}}(x) > 0 \text{ for all } x \in \mathcal{D}.$$

The goal of this paper is to estimate the *importance* $w(x)$ from $\{x_i^{\mathrm{tr}}\}_{i=1}^{n_{\mathrm{tr}}}$ and $\{x_j^{\mathrm{te}}\}_{j=1}^{n_{\mathrm{te}}}$:

$$w(x) = \frac{p_{\mathrm{te}}(x)}{p_{\mathrm{tr}}(x)},$$

which is non-negative by definition. Our key restriction is that we want to avoid estimating densities $p_{\mathrm{te}}(x)$ and $p_{\mathrm{tr}}(x)$ when estimating the importance $w(x)$.

2.2 Least-squares Approach to Direct Importance Estimation

Let us model the importance $w(x)$ by the following linear model:

$$\widehat{w}(x) = \sum_{\ell=1}^{b} \alpha_\ell \varphi_\ell(x), \tag{1}$$

where $\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_b)^\top$ are parameters to be learned from data samples, $^\top$ denotes the transpose of a matrix or a vector, and $\{\varphi_\ell(x)\}_{\ell=1}^{b}$ are basis functions such that

$$\varphi_\ell(x) \geq 0 \text{ for all } x \in \mathcal{D} \text{ and for } \ell = 1, 2, \ldots, b.$$

Note that b and $\{\varphi_\ell(x)\}_{\ell=1}^{b}$ could be dependent on the samples $\{x_i^{\mathrm{tr}}\}_{i=1}^{n_{\mathrm{tr}}}$ and $\{x_j^{\mathrm{te}}\}_{j=1}^{n_{\mathrm{te}}}$, for example, *kernel* models are also allowed. We explain how the basis functions $\{\varphi_\ell(x)\}_{\ell=1}^{b}$ are chosen in Section 2.5.

We determine the parameters $\{\alpha_\ell\}_{\ell=1}^b$ in the model $\widehat{w}(x)$ so that the following squared error J_0 is minimized:

$$\begin{aligned}
J_0(\alpha) &= \frac{1}{2}\int \left(\widehat{w}(x) - w(x)\right)^2 p_{\mathrm{tr}}(x)dx \\
&= \frac{1}{2}\int \widehat{w}(x)^2 p_{\mathrm{tr}}(x)dx - \int \widehat{w}(x)w(x)p_{\mathrm{tr}}(x)dx + \frac{1}{2}\int w(x)^2 p_{\mathrm{tr}}(x)dx \\
&= \frac{1}{2}\int \widehat{w}(x)^2 p_{\mathrm{tr}}(x)dx - \int \widehat{w}(x)p_{\mathrm{te}}(x)dx + \frac{1}{2}\int w(x)^2 p_{\mathrm{tr}}(x)dx,
\end{aligned}$$

where in the second term the probability density $p_{\mathrm{tr}}(x)$ is canceled with that included in $w(x)$. The squared loss $J_0(\alpha)$ is defined as the expectation under the probability of training samples. In covariate shift adaptation (see Section 6.2) and outlier detection (see Section 6.3), the importance values on the training samples are used. Thus, the definition of $J_0(\alpha)$ well agrees with our goal.

The last term of $J_0(\alpha)$ is a constant and therefore can be safely ignored. Let us denote the first two terms by J:

$$\begin{aligned}
J(\alpha) &= \frac{1}{2}\int \widehat{w}(x)^2 p_{\mathrm{tr}}(x)dx - \int \widehat{w}(x)p_{\mathrm{te}}(x)dx \\
&= \frac{1}{2}\sum_{\ell,\ell'=1}^{b} \alpha_\ell\alpha_{\ell'}\left(\int \varphi_\ell(x)\varphi_{\ell'}(x)p_{\mathrm{tr}}(x)dx\right) - \sum_{\ell=1}^{b}\alpha_\ell\left(\int \varphi_\ell(x)p_{\mathrm{te}}(x)dx\right) \\
&= \frac{1}{2}\alpha^\top H\alpha - h^\top\alpha, \qquad (2)
\end{aligned}$$

where H is the $b\times b$ matrix with the (ℓ,ℓ')-th element

$$H_{\ell,\ell'} = \int \varphi_\ell(x)\varphi_{\ell'}(x)p_{\mathrm{tr}}(x)dx, \qquad (3)$$

and h is the b-dimensional vector with the ℓ-th element

$$h_\ell = \int \varphi_\ell(x)p_{\mathrm{te}}(x)dx.$$

Approximating the expectations in J by empirical averages, we obtain

$$\begin{aligned}
\widehat{J}(\alpha) &= \frac{1}{2n_{\mathrm{tr}}}\sum_{i=1}^{n_{\mathrm{tr}}}\widehat{w}(x_i^{\mathrm{tr}})^2 - \frac{1}{n_{\mathrm{te}}}\sum_{j=1}^{n_{\mathrm{te}}}\widehat{w}(x_j^{\mathrm{te}}) \\
&= \frac{1}{2}\sum_{\ell,\ell'=1}^{b}\alpha_\ell\alpha_{\ell'}\left(\frac{1}{n_{\mathrm{tr}}}\sum_{i=1}^{n_{\mathrm{tr}}}\varphi_\ell(x_i^{\mathrm{tr}})\varphi_{\ell'}(x_i^{\mathrm{tr}})\right) - \sum_{\ell=1}^{b}\alpha_\ell\left(\frac{1}{n_{\mathrm{te}}}\sum_{j=1}^{n_{\mathrm{te}}}\varphi_\ell(x_j^{\mathrm{te}})\right) \\
&= \frac{1}{2}\alpha^\top\widehat{H}\alpha - \widehat{h}^\top\alpha,
\end{aligned}$$

where $\widehat{H}$ is the $b\times b$ matrix with the (ℓ,ℓ')-th element

$$\widehat{H}_{\ell,\ell'} = \frac{1}{n_{\mathrm{tr}}}\sum_{i=1}^{n_{\mathrm{tr}}}\varphi_\ell(x_i^{\mathrm{tr}})\varphi_{\ell'}(x_i^{\mathrm{tr}}), \qquad (4)$$

and $\widehat{h}$ is the b-dimensional vector with the ℓ-th element

$$\widehat{h}_\ell = \frac{1}{n_{\mathrm{te}}} \sum_{j=1}^{n_{\mathrm{te}}} \varphi_\ell(x_j^{\mathrm{te}}). \tag{5}$$

Taking into account the non-negativity of the importance function $w(x)$, we can formulate our optimization problem as follows.

$$\min_{\alpha \in \mathbb{R}^b} \left[\frac{1}{2} \alpha^\top \widehat{H} \alpha - \widehat{h}^\top \alpha + \lambda 1_b^\top \alpha \right] \quad \text{subject to } \alpha \geq 0_b, \tag{6}$$

where 0_b and 1_b are the b-dimensional vectors with all zeros and ones, respectively; the vector inequality $\alpha \geq 0_b$ is applied in the element-wise manner, that is, $\alpha_\ell \geq 0$ for $\ell = 1, 2, \ldots, b$. In Eq. (6), we included a penalty term $\lambda 1_b^\top \alpha$ for regularization purposes, where λ (≥ 0) is a regularization parameter. The above is a convex quadratic programming problem and therefore the unique global optimal solution can be computed efficiently by a standard optimization package. We call this method *Least-Squares Importance Fitting* (LSIF).

We can also use the ℓ_2-regularizer $\alpha^\top \alpha$ instead of the ℓ_1-regularizer $1_b^\top \alpha$ without changing the computational property. However, using the ℓ_1-regularizer would be more advantageous since the solution tends to be sparse (Williams, 1995; Tibshirani, 1996; Chen et al., 1998). Furthermore, as shown in Section 2.6, the use of the ℓ_1-regularizer allows us to compute the entire regularization path efficiently (Best, 1982; Efron et al., 2004; Hastie et al., 2004). The ℓ_2-regularization method will be used for theoretical analysis in Section 3.3.

2.3 Convergence Analysis of LSIF

Here, we theoretically analyze the convergence property of the solution $\widehat{\alpha}$ of the LSIF algorithm; practitioners may skip this theoretical analysis.

Let $\widehat{\alpha}(\lambda)$ be the solution of the LSIF algorithm with regularization parameter λ, and let $\alpha^*(\lambda)$ be the optimal solution of the 'ideal' problem:

$$\min_{\alpha \in \mathbb{R}^b} \left[\frac{1}{2} \alpha^\top H \alpha - h^\top \alpha + \lambda 1_b^\top \alpha \right] \quad \text{subject to } \alpha \geq 0_b. \tag{7}$$

Below, we theoretically investigate the *learning curve* (Amari et al., 1992) of LSIF, that is, we elucidate the relation between $J(\widehat{\alpha}(\lambda))$ and $J(\alpha^*(\lambda))$ in terms of the expectation over all possible training and test samples as a function of the number of samples.

Let $\mathbb{E}$ be the expectation over all possible training samples of size n_{tr} and all possible test samples of size n_{te}. Let $\mathcal{A} \subset \{1, 2, \ldots, b\}$ be the set of *active* indices (Boyd and Vandenberghe, 2004), that is,

$$\mathcal{A} = \{\ell \mid \alpha_\ell^*(\lambda) = 0,\ \ell = 1, 2, \ldots, b\}.$$

For the active set $\mathcal{A} = \{j_1, j_2, \ldots, j_{|\mathcal{A}|}\}$ with $j_1 < j_2 < \cdots < j_{|\mathcal{A}|}$, let E be the $|\mathcal{A}| \times b$ indicator matrix with the (i, j)-th element

$$E_{i,j} = \begin{cases} 1 & j = j_i, \\ 0 & \text{otherwise.} \end{cases}$$

Similarly, let $\widehat{\mathcal{A}}$ be the active set of $\widehat{\alpha}(\lambda)$:

$$\widehat{\mathcal{A}} = \{\ell \mid \widehat{\alpha}_\ell(\lambda) = 0,\ \ell = 1,2,\ldots,b\}.$$

For the active set $\widehat{\mathcal{A}} = \{\widehat{j}_1, \widehat{j}_2, \ldots, \widehat{j}_{|\widehat{\mathcal{A}}|}\}$ with $\widehat{j}_1 < \widehat{j}_2 < \cdots < \widehat{j}_{|\widehat{\mathcal{A}}|}$, let $\widehat{E}$ be the $|\widehat{\mathcal{A}}| \times b$ indicator matrix with the (i,j)-th element similarly defined by

$$\widehat{E}_{i,j} = \begin{cases} 1 & j = \widehat{j}_i, \\ 0 & \text{otherwise.} \end{cases} \tag{8}$$

First, we show the optimality condition of (6) which will be used in the following theoretical analyses. The *Lagrangian* of the optimization problem (6) is given as

$$L(\alpha,\xi) = \frac{1}{2}\alpha^\top \widehat{H}\alpha - \widehat{h}^\top \alpha + \lambda 1_b^\top \alpha - \xi^\top \alpha,$$

where ξ is the b-dimensional *Lagrange multiplier* vector. Then the *Karush-Kuhn-Tucker (KKT) conditions* (Boyd and Vandenberghe, 2004) are expressed as follows:

$$\begin{align} &\widehat{H}\alpha - \widehat{h} + \lambda 1_b - \xi = 0_b, \tag{9}\\ &\alpha \geq 0_b, \notag\\ &\xi \geq 0_b, \notag\\ &\xi_\ell \alpha_\ell = 0 \text{ for } \ell = 1,2,\ldots,b. \tag{10} \end{align}$$

Let $\widehat{\xi}'(\lambda)$ be the $|\widehat{\mathcal{A}}|$-dimensional vector with the i-th element being the $\widehat{j}_i$-th element of $\widehat{\xi}(\lambda)$:

$$\widehat{\xi}'_i(\lambda) = \widehat{\xi}_{\widehat{j}_i}(\lambda), \qquad i = 1,\ldots,|\widehat{\mathcal{A}}|. \tag{11}$$

We assume that $\widehat{\xi}'(\lambda)$ only contains non-zero elements of $\widehat{\xi}(\lambda)$. Let $\widehat{G}$ be

$$\widehat{G} = \begin{pmatrix} \widehat{H} & -\widehat{E}^\top \\ -\widehat{E} & O_{|\widehat{\mathcal{A}}|\times|\widehat{\mathcal{A}}|} \end{pmatrix},$$

where $O_{|\widehat{\mathcal{A}}|\times|\widehat{\mathcal{A}}|}$ is the $|\widehat{\mathcal{A}}| \times |\widehat{\mathcal{A}}|$ matrix with all zeros. Then Eqs. (9) and (10) are together expressed in a matrix form as

$$\widehat{G}\begin{pmatrix} \widehat{\alpha}(\lambda) \\ \widehat{\xi}'(\lambda) \end{pmatrix} = \begin{pmatrix} \widehat{h} - \lambda 1_b \\ 0_{|\widehat{\mathcal{A}}|} \end{pmatrix}. \tag{12}$$

Regarding the matrix $\widehat{G}$, we have the following lemma:

Lemma 1 *The matrix $\widehat{G}$ is invertible if $\widehat{H}$ is invertible.*

The proof of the above lemma is given in Appendix A. Below, we assume that $\widehat{H}$ is invertible. Then the inverse of $\widehat{G}$ exists and multiplying $\widehat{G}^{-1}$ from the left-hand side of Eq. (12) yields

$$\begin{pmatrix} \widehat{\alpha}(\lambda) \\ \widehat{\xi}'(\lambda) \end{pmatrix} = \widehat{G}^{-1}\begin{pmatrix} \widehat{h} - \lambda 1_b \\ 0_{|\widehat{\mathcal{A}}|} \end{pmatrix}. \tag{13}$$

The following inversion formula holds for block matrices (Petersen and Pedersen, 2007):

$$\begin{pmatrix} M_1 & M_2 \\ M_3 & M_4 \end{pmatrix}^{-1} = \begin{pmatrix} M_1^{-1} + M_1^{-1} M_2 M_0^{-1} M_3 M_1^{-1} & -M_1^{-1} M_2 M_0^{-1} \\ -M_0^{-1} M_3 M_1^{-1} & M_0^{-1} \end{pmatrix}, \tag{14}$$

where

$$M_0 = M_4 - M_3 M_1^{-1} M_2.$$

Applying Eq. (14) to Eq. (13), we have

$$\widehat{\alpha}(\lambda) = \widehat{A}(\widehat{h} - \lambda 1_b), \tag{15}$$

where $\widehat{A}$ is defined by

$$\widehat{A} = \widehat{H}^{-1} - \widehat{H}^{-1} \widehat{E}^\top (\widehat{E} \widehat{H}^{-1} \widehat{E}^\top)^{-1} \widehat{E} \widehat{H}^{-1}. \tag{16}$$

When the Lagrange multiplier vector satisfies

$$\xi^*_\ell(\lambda) > 0 \text{ for all } \ell \in \mathcal{A}, \tag{17}$$

we say that the *strict complementarity condition* is satisfied (Bertsekas et al., 2003). An important consequence of strict complementarity is that the optimal solution and the Lagrange multipliers of convex quadratic problems are uniquely determined. Then we have the following theorem.

Theorem 2 *Let P be the probability over all possible training samples of size n_{tr} and test samples of size n_{te}. Let $\xi^*(\lambda)$ be the Lagrange multiplier vector of the problem* (7) *and suppose $\xi^*(\lambda)$ satisfies the strict complementarity condition* (17). *Then, there exists a positive constant $c > 0$ and a natural number N such that for $\min\{n_{\mathrm{tr}}, n_{\mathrm{te}}\} \geq N$,*

$$P(\widehat{\mathcal{A}} \neq \mathcal{A}) < e^{-c \min\{n_{\mathrm{tr}}, n_{\mathrm{te}}\}}.$$

The proof of the above theorem is given in Appendix B. Theorem 2 shows that the probability that the active set $\widehat{\mathcal{A}}$ of the empirical problem (6) is different from the active set $\mathcal{A}$ of the ideal problem (7) is exponentially small. Thus we may regard $\widehat{\mathcal{A}} = \mathcal{A}$ in practice.

Let A be the 'ideal' counterpart of $\widehat{A}$:

$$A = H^{-1} - H^{-1} E^\top (E H^{-1} E^\top)^{-1} E H^{-1},$$

and let $C_{w,w'}$ be the $b \times b$ covariance matrix with the (ℓ, ℓ')-th element being the covariance between $w(x)\varphi_\ell(x)$ and $w'(x)\varphi_{\ell'}(x)$ under $p_{\mathrm{tr}}(x)$. Let

$$w^*(x) = \sum_{\ell=1}^{b} \alpha^*_\ell(\lambda) \varphi_\ell(x),$$

$$v(x) = \sum_{\ell=1}^{b} [A 1_b]_\ell \varphi_\ell(x).$$

Let

$$f(n) = \omega(g(n))$$

denote that $f(n)$ asymptotically dominates $g(n)$; more precisely, for all $C > 0$, there exists n_0 such that

$$|Cg(n)| < |f(n)| \quad \text{for all } n > n_0.$$

Then we have the following theorem.

Theorem 3 *Assume that*

(a) *The optimal solution of the problem* (7) *satisfies the strict complementarity condition* (17).

(b) n_{tr} *and* n_{te} *satisfy*

$$n_{\mathrm{te}} = \omega(n_{\mathrm{tr}}^2). \tag{18}$$

Then, for any $\lambda \geq 0$*, we have*

$$\mathbb{E}[J(\widehat{\alpha}(\lambda))] = J(\alpha^*(\lambda)) + \frac{1}{2n_{\mathrm{tr}}}\mathrm{tr}(A(C_{w^*,w^*} - 2\lambda C_{w^*,v})) + o\left(\frac{1}{n_{\mathrm{tr}}}\right). \tag{19}$$

The proof of the above theorem is given in Appendix C. This theorem elucidates the learning curve of LSIF up to the order of n_{tr}^{-1}. In Section 2.4.1, we discuss practical implications of this theorem.

2.4 Model Selection for LSIF

The practical performance of LSIF depends on the choice of the regularization parameter λ and basis functions $\{\varphi_\ell(x)\}_{\ell=1}^b$ (which we refer to as a *model*). Since our objective is to minimize the cost function J defined in Eq. (2), it is natural to determine the model such that J is minimized.

However, the value of the cost function J is inaccessible since it includes the expectation over unknown probability density functions $p_{\mathrm{tr}}(x)$ and $p_{\mathrm{te}}(x)$. The value of the empirical cost $\widehat{J}$ may be regarded as an estimate of J, but this is not useful for model selection purposes since it is heavily biased—the bias is caused by the fact that the same samples are used twice for learning the parameter α and estimating the value of J. Below, we give two practical methods of estimating the value of J in more precise ways.

2.4.1 INFORMATION CRITERION

In the same way as Theorem 3, we can obtain an asymptotic expansion of the empirical cost $\mathbb{E}\left[\widehat{J}(\widehat{\alpha}(\lambda))\right]$ as follows:

$$\mathbb{E}[\widehat{J}(\widehat{\alpha}(\lambda))] = J(\alpha^*(\lambda)) - \frac{1}{2n_{\mathrm{tr}}}\mathrm{tr}(A(C_{w^*,w^*} + 2\lambda C_{w^*,v})) + o\left(\frac{1}{n_{\mathrm{tr}}}\right). \tag{20}$$

Combining Eqs. (19) and (20), we have

$$\mathbb{E}[J(\widehat{\alpha}(\lambda))] = \mathbb{E}[\widehat{J}(\widehat{\alpha}(\lambda))] + \frac{1}{n_{\mathrm{tr}}}\mathrm{tr}(AC_{w^*,w^*}) + o\left(\frac{1}{n_{\mathrm{tr}}}\right).$$

From this, we can immediately obtain an *information criterion* (Akaike, 1974; Konishi and Kitagawa, 1996) for LSIF:

$$\widehat{J}^{(\mathrm{IC})} = \widehat{J}(\widehat{\alpha}(\lambda)) + \frac{1}{n_{\mathrm{tr}}}\mathrm{tr}(\widehat{A}\widehat{C}_{\widehat{w},\widehat{w}}),$$

where $\widehat{A}$ is defined by Eq. (16). $\widehat{E}$ is defined by Eq. (8) and $\widehat{C}_{w,w'}$ is the $b \times b$ covariance matrix with the (ℓ, ℓ')-th element being the covariance between $w(x)\varphi_\ell(x)$ and $w'(x)\varphi_{\ell'}(x)$ over $\{x_i^{\mathrm{tr}}\}_{i=1}^{n_{\mathrm{tr}}}$. Since $\widehat{A}$ and $\widehat{C}_{\widehat{w},\widehat{w}}$ are consistent estimators of A and C_{w^*,w^*}, the above information criterion is unbiased up to the order of n_{tr}^{-1}.

Note that the term $\mathrm{tr}(\widehat{A}\widehat{C}_{\widehat{w},\widehat{w}})$ may be interpreted as the *effective dimension* of the model (Moody, 1992). Indeed, when $\widehat{w}(x) = 1$, we have $\widehat{H} = \widehat{C}_{\widehat{w},\widehat{w}}$ and thus

$$\mathrm{tr}(\widehat{A}\widehat{C}_{\widehat{w},\widehat{w}}) = \mathrm{tr}(I_b) - \mathrm{tr}(E\widehat{C}_{\widehat{w},\widehat{w}}^{-1}E^\top(E\widehat{C}_{\widehat{w},\widehat{w}}^{-1}E^\top)^{-1}) = b - |\widehat{\mathcal{A}}|,$$

which is the dimension of the *face* on which $\widehat{\alpha}(\lambda)$ lies.

2.4.2 CROSS-VALIDATION

Although the information criterion derived above is more accurate than just a naive empirical estimator, its accuracy is guaranteed only asymptotically. Here, we employ cross-validation for estimating $J(\widehat{\alpha})$, which has an accuracy guarantee for finite samples.

First, the training samples $\{x_i^{\mathrm{tr}}\}_{i=1}^{n_{\mathrm{tr}}}$ and test samples $\{x_j^{\mathrm{te}}\}_{j=1}^{n_{\mathrm{te}}}$ are divided into R disjoint subsets $\{\mathcal{X}_r^{\mathrm{tr}}\}_{r=1}^{R}$ and $\{\mathcal{X}_r^{\mathrm{te}}\}_{r=1}^{R}$, respectively. Then an importance estimate $\widehat{w}_{\mathcal{X}_r^{\mathrm{tr}},\mathcal{X}_r^{\mathrm{te}}}(x)$ is obtained using $\{\mathcal{X}_j^{\mathrm{tr}}\}_{j\neq r}$ and $\{\mathcal{X}_j^{\mathrm{te}}\}_{j\neq r}$ (that is, without $\mathcal{X}_r^{\mathrm{tr}}$ and $\mathcal{X}_r^{\mathrm{te}}$), and the cost J is approximated using the held-out samples $\mathcal{X}_r^{\mathrm{tr}}$ and $\mathcal{X}_r^{\mathrm{te}}$ as

$$\widehat{J}^{(\mathrm{CV})}_{\mathcal{X}_r^{\mathrm{tr}},\mathcal{X}_r^{\mathrm{te}}} = \frac{1}{2|\mathcal{X}_r^{\mathrm{tr}}|}\sum_{x^{\mathrm{tr}}\in\mathcal{X}_r^{\mathrm{tr}}} \widehat{w}_{\mathcal{X}_r^{\mathrm{tr}},\mathcal{X}_r^{\mathrm{te}}}(x^{\mathrm{tr}})^2 - \frac{1}{|\mathcal{X}_r^{\mathrm{te}}|}\sum_{x^{\mathrm{te}}\in\mathcal{X}_r^{\mathrm{te}}} \widehat{w}_{\mathcal{X}_r^{\mathrm{tr}},\mathcal{X}_r^{\mathrm{te}}}(x^{\mathrm{te}}).$$

This procedure is repeated for $r = 1,2,\ldots,R$ and its average $\widehat{J}^{(\mathrm{CV})}$ is used as an estimate of J:

$$\widehat{J}^{(\mathrm{CV})} = \frac{1}{R}\sum_{r=1}^{R}\widehat{J}^{(\mathrm{CV})}_{\mathcal{X}_r^{\mathrm{tr}},\mathcal{X}_r^{\mathrm{te}}}.$$

We can show that $\widehat{J}^{(\mathrm{CV})}$ gives an almost unbiased estimate of the true cost J, where the ‘almost’-ness comes from the fact that the number of samples is reduced in the cross-validation procedure due to data splitting (Luntz and Brailovsky, 1969; Wahba, 1990; Schölkopf and Smola, 2002).

Cross-validation would be more accurate than the information criterion for finite samples. However, it is computationally more expensive than the information criterion since the learning procedure should be repeated R times.

2.5 Heuristics of Basis Function Design for LSIF

A good model may be chosen by cross-validation or the information criterion, given that a family of promising model candidates is prepared. As model candidates, we propose using a Gaussian kernel model centered at the *test* points $\{x_j^{\mathrm{te}}\}_{j=1}^{n_{\mathrm{te}}}$, that is,

$$\widehat{w}(x) = \sum_{\ell=1}^{n_{\mathrm{te}}}\alpha_\ell K_\sigma(x, x_\ell^{\mathrm{te}}),$$

where $K_\sigma(x,x')$ is the Gaussian kernel with kernel width σ:

$$K_\sigma(x,x') = \exp\left(-\frac{\|x-x'\|^2}{2\sigma^2}\right). \tag{21}$$

The reason why we chose the test points $\{x_j^{\mathrm{te}}\}_{j=1}^{n_{\mathrm{te}}}$ as the Gaussian centers, not the training points $\{x_i^{\mathrm{tr}}\}_{i=1}^{n_{\mathrm{tr}}}$, is as follows (Sugiyama et al., 2008b). By definition, the importance $w(x)$ tends to take large values if the training density $p_{\mathrm{tr}}(x)$ is small and the test density $p_{\mathrm{te}}(x)$ is large; conversely, $w(x)$ tends to be small (that is, close to zero) if $p_{\mathrm{tr}}(x)$ is large and $p_{\mathrm{te}}(x)$ is small. When a function is approximated by a Gaussian kernel model, many kernels may be needed in the region where the output of the target function is large; on the other hand, only a small number of kernels would be enough in the region where the output of the target function is close to zero. Following this heuristic, we allocate many kernels at high *test* density regions, which can be achieved by setting the Gaussian centers at the test points $\{x_j^{\mathrm{te}}\}_{j=1}^{n_{\mathrm{te}}}$.

Alternatively, we may locate $(n_{\mathrm{tr}}+n_{\mathrm{te}})$ Gaussian kernels at both $\{x_i^{\mathrm{tr}}\}_{i=1}^{n_{\mathrm{tr}}}$ and $\{x_j^{\mathrm{te}}\}_{j=1}^{n_{\mathrm{te}}}$. However, in our preliminary experiments, this did not further improve the performance, but just slightly increased the computational cost. When n_{te} is large, just using all the test points $\{x_j^{\mathrm{te}}\}_{j=1}^{n_{\mathrm{te}}}$ as Gaussian centers is already computationally rather demanding. To ease this problem, we practically propose using a subset of $\{x_j^{\mathrm{te}}\}_{j=1}^{n_{\mathrm{te}}}$ as Gaussian centers for computational efficiency, that is,

$$\widehat{w}(x) = \sum_{\ell=1}^{b} \alpha_\ell K_\sigma(x, c_\ell), \tag{22}$$

where $c_\ell,\ \ell = 1, 2, \ldots, b$ are template points randomly chosen from $\{x_j^{\mathrm{te}}\}_{j=1}^{n_{\mathrm{te}}}$ without replacement and $b\ (\leq n_{\mathrm{te}})$ is a prefixed number. In the rest of this paper, we usually fix the number of template points at

$$b = \min(100, n_{\mathrm{te}}),$$

and optimize the kernel width σ and the regularization parameter λ by cross-validation with grid search.

2.6 Entire Regularization Path for LSIF

We can show that the LSIF solution $\widehat{\alpha}$ is piecewise linear with respect to the regularization parameter λ (see Appendix D). Therefore, the *regularization path* (that is, solutions for all λ) can be computed efficiently based on the *parametric optimization technique* (Best, 1982; Efron et al., 2004; Hastie et al., 2004).

A basic idea of regularization path tracking is to check violation of the KKT conditions—which are necessary and sufficient for optimality of convex programs—when the regularization parameter λ is changed. The KKT conditions of LSIF are summarized in Section 2.3. The strict complementarity condition (17) assures the uniqueness of the optimal solution for a fixed λ, and thus the uniqueness of the regularization path. A pseudo code of the regularization path tracking algorithm for LSIF is described in Figure 1—its detailed derivation is summarized in Appendix D. Thanks to the regularization path algorithm, LSIF is computationally efficient in model selection scenarios.

The pseudo code implies that we no longer need a quadratic programming solver for obtaining the solution of LSIF—just computing matrix inverses is enough. Furthermore, the regularization path algorithm is computationally more efficient when the solution is sparse, that is, most of the elements are zero since the number of change points tends to be small for such sparse solutions.

Even though the regularization path tracking algorithm is computationally efficient, it tends to be numerically unreliable, as we experimentally show in Section 4. This numerical instability is caused by near singularity of the matrix $\widehat{G}$. When $\widehat{G}$ is nearly singular, it is not easy to accurately obtain the solutions u, v in Figure 1, and therefore the change point $\lambda_{\tau+1}$ cannot be accurately computed. As a result, we cannot accurately update the active set of the inequality constraints and thus

```
**Input:** $\widehat{H}$ and $\widehat{h}$    % see Eqs. (4) and (5) for the definitions
**Output:** entire regularization path $\widehat{\alpha}(\lambda)$ for $\lambda \geq 0$

$\tau \longleftarrow 0$;
$k \longleftarrow \mathrm{argmax}_i\{\widehat{h}_i \mid i = 1,2,\ldots,b\}$;
$\lambda_\tau \longleftarrow \widehat{h}_k$;
$\widehat{\mathcal{A}} \longleftarrow \{1,2,\ldots,b\}\backslash\{k\}$;
$\widehat{\alpha}(\lambda_\tau) \longleftarrow 0_b$;    % the vector with all zeros
While $\lambda_\tau > 0$
    $\widehat{E} \longleftarrow O_{|\widehat{\mathcal{A}}|\times b}$;    % the matrix with all zeros
    For $i = 1,2,\ldots,|\widehat{\mathcal{A}}|$
        $\widehat{E}_{i,\widehat{j}_i} \longleftarrow 1$;  % $\widehat{\mathcal{A}} = \{\widehat{j}_1, \widehat{j}_2, \ldots, \widehat{j}_{|\widehat{\mathcal{A}}|} \mid \widehat{j}_1 < \widehat{j}_2 < \cdots < \widehat{j}_{|\widehat{\mathcal{A}}|}\}$
    end
    $\widehat{G} \longleftarrow \begin{pmatrix} \widehat{H} & -\widehat{E}^\top \\ -\widehat{E} & O_{|\widehat{\mathcal{A}}|\times|\widehat{\mathcal{A}}|} \end{pmatrix}$;
    $u \longleftarrow \widehat{G}^{-1}\begin{pmatrix} \widehat{h} \\ 0_{|\widehat{\mathcal{A}}|} \end{pmatrix}$;
    $v \longleftarrow \widehat{G}^{-1}\begin{pmatrix} 1_b \\ 0_{|\widehat{\mathcal{A}}|} \end{pmatrix}$;
    If $v \leq 0_{b+|\widehat{\mathcal{A}}|}$   % the final interval
        $\lambda_{\tau+1} \longleftarrow 0$;
        $\widehat{\alpha}(\lambda_{\tau+1}) \longleftarrow (u_1, u_2, \ldots, u_b)^\top$;
    else   % an intermediate interval
        $k \longleftarrow \mathrm{argmax}_i\{u_i/v_i \mid v_i > 0,\ i = 1,2,\ldots,b+|\widehat{\mathcal{A}}|\}$;
        $\lambda_{\tau+1} \longleftarrow \max\{0, u_k/v_k\}$;
        $\widehat{\alpha}(\lambda_{\tau+1}) \longleftarrow (u_1, u_2, \ldots, u_b)^\top - \lambda_{\tau+1}(v_1, v_2, \ldots, v_b)^\top$;
        If $1 \leq k \leq b$
            $\widehat{\mathcal{A}} \longleftarrow \widehat{\mathcal{A}} \cup \{k\}$;
        else
            $\widehat{\mathcal{A}} \longleftarrow \widehat{\mathcal{A}}\backslash\{\widehat{j}_{k-b}\}$;
        end
    end
    $\tau \longleftarrow \tau + 1$;
end
$\widehat{\alpha}(\lambda) \longleftarrow \begin{cases} 0_b & \text{if } \lambda \geq \lambda_0 \\ \frac{\lambda_{\tau+1}-\lambda}{\lambda_{\tau+1}-\lambda_\tau}\widehat{\alpha}(\lambda_\tau) + \frac{\lambda-\lambda_\tau}{\lambda_{\tau+1}-\lambda_\tau}\widehat{\alpha}(\lambda_{\tau+1}) & \text{if } \lambda_{\tau+1} \leq \lambda \leq \lambda_\tau \end{cases}$
```

Figure 1: Pseudo code for computing the entire regularization path of LSIF. When the computation of $\widehat{G}^{-1}$ is numerically unstable, we may add small positive diagonals to $\widehat{H}$ for stabilization purposes.

the obtained solution $\widehat{\alpha}(\lambda)$ becomes unreliable; furthermore, such numerical error tends to be accumulated through the path-tracking process. This instability issue seems to be a common pitfall of solution path tracking algorithms in general (see Scheinberg, 2006).

When the Gaussian width σ is very small or very large, the matrix $\widehat{H}$ tends to be nearly singular and thus the matrix $\widehat{G}$ also becomes nearly singular. On the other hand, when the Gaussian width σ is not too small or too large compared with the dispersion of samples, the matrix $\widehat{G}$ is well-conditioned and therefore the path-following algorithm would be stable and reliable.

3. Approximation Algorithm

Within the quadratic programming formulation, we have proposed a new importance estimation procedure LSIF and showed its theoretical properties. We also gave a regularization path tracking algorithm that can be computed efficiently. However, as we experimentally show in Section 4, it tends to suffer from a numerical problem and therefore is not practically reliable. In this section, we give a practical alternative to LSIF which gives an approximate solution to LSIF in a computationally efficient and reliable manner.

3.1 Unconstrained Least-squares Formulation

The approximation idea we introduce here is very simple: we ignore the non-negativity constraint of the parameters in the optimization problem (6). This results in the following unconstrained optimization problem.

$$\min_{\beta \in \mathbb{R}^b} \left[\frac{1}{2}\beta^\top \widehat{H}\beta - \widehat{h}^\top \beta + \frac{\lambda}{2}\beta^\top \beta \right]. \tag{23}$$

In the above, we included a quadratic regularization term $\beta^\top\beta/2$, instead of the linear one $1_b^\top \beta$ since the linear penalty term does not work as a regularizer without the non-negativity constraint. Eq. (23) is an unconstrained convex quadratic program, so the solution can be analytically computed as

$$\widetilde{\beta}(\lambda) = (\widehat{H} + \lambda I_b)^{-1}\widehat{h},$$

where I_b is the b-dimensional identity matrix. Since we dropped the non-negativity constraint $\beta \geq 0_b$, some of the learned parameters could be negative. To compensate for this approximation error, we modify the solution by

$$\widehat{\beta}(\lambda) = \max(0_b, \widetilde{\beta}(\lambda)),$$

where the 'max' operation for a pair of vectors is applied in the element-wise manner. This is the solution of the approximation method we propose in this section.

An advantage of the above unconstrained formulation is that the solution can be computed just by solving a system of linear equations. Therefore, its computation is stable when λ is not too small. We call this method *unconstrained LSIF* (uLSIF). Due to the ℓ_2 regularizer, the solution tends to be close to 0_b to some extent. Thus, the effect of ignoring the non-negativity constraint may not be so strong—later, we analyze the approximation error both theoretically and experimentally in more detail in Sections 3.3 and 4.5.

Note that LSIF and uLSIF differ only in parameter learning. Thus, the basis design heuristic of LSIF given in Section 2.5 is also valid for uLSIF.

3.2 Convergence Analysis of uLSIF

Here, we theoretically analyze the convergence property of the solution $\widehat{\beta}(\lambda)$ of the uLSIF algorithm; practitioners may skip Sections 3.2 and 3.3.

Let $\beta^{\circ}(\lambda)$ be the optimal solution of the 'ideal' version of the problem (23):

$$\min_{\beta \in \mathbb{R}^b} \left[\frac{1}{2}\beta^\top H \beta - h^\top \beta + \frac{\lambda}{2}\beta^\top \beta \right].$$

Then the ideal solution $\beta^*(\lambda)$ is given by

$$\begin{aligned} \beta^*(\lambda) &= \max(0_b, \beta^{\circ}(\lambda)), \\ \beta^{\circ}(\lambda) &= B_\lambda^{-1} h, \\ B_\lambda &= H + \lambda I_b. \end{aligned} \tag{24}$$

Below, we theoretically investigate the learning curve of uLSIF.

Let $\mathcal{B} \subset \{1, 2, \ldots, b\}$ be the set of negative indices of $\beta^{\circ}(\lambda)$, that is,

$$\mathcal{B} = \{\ell \mid \beta^{\circ}_\ell(\lambda) < 0,\ \ell = 1, 2, \ldots, b\},$$

and $\widetilde{\mathcal{B}} \subset \{1, 2, \ldots, b\}$ be the set of negative indices of $\widetilde{\beta}(\lambda)$, that is,

$$\widetilde{\mathcal{B}} = \{\ell \mid \widetilde{\beta}_\ell(\lambda) < 0,\ \ell = 1, 2, \ldots, b\}.$$

Then we have the following theorem.

Theorem 4 *Assume that $\beta^{\circ}_\ell(\lambda) \neq 0$ for $\ell = 1, 2, \ldots, b$. Then, there exists a positive constant c and a natural number N such that for $\min\{n_{\mathrm{tr}}, n_{\mathrm{te}}\} \geq N$,*

$$P(\mathcal{B} \neq \widetilde{\mathcal{B}}) < e^{-c \min\{n_{\mathrm{tr}}, n_{\mathrm{te}}\}}.$$

The proof of the above theorem is given in Appendix E. The assumption that $\beta^{\circ}_\ell(\lambda) \neq 0$ for $\ell = 1, 2, \ldots, b$ corresponds to the strict complementarity condition (17) in LSIF. Theorem 4 shows that the probability that $\widetilde{\mathcal{B}}$ is different from $\mathcal{B}$ is exponentially small. Thus we may regard $\widetilde{\mathcal{B}} = \mathcal{B}$ in practice.

Let D be the b-dimensional diagonal matrix with the ℓ-th diagonal element

$$D_{\ell,\ell} = \begin{cases} 0 & \ell \in \mathcal{B}, \\ 1 & \text{otherwise.} \end{cases}$$

Let

$$\begin{aligned} w^{\circ}(x) &= \sum_{\ell=1}^{b} \beta^{\circ}_\ell(\lambda) \varphi_\ell(x), \\ u(x) &= \sum_{\ell=1}^{b} [B_\lambda^{-1} D (H\beta^*(\lambda) - h)]_\ell \varphi_\ell(x). \end{aligned}$$

Then we have the following theorem.

Theorem 5 *Assume that*

(a) $\beta^{\circ}_{\ell}(\lambda) \neq 0$ *for* $\ell = 1, 2, \ldots, b$.

(b) n_{tr} *and* n_{te} *satisfy Eq.* (18).

Then, for any $\lambda \geq 0$, *we have*

$$\mathbb{E}[J(\widehat{\beta}(\lambda))] = J(\beta^*(\lambda)) + \frac{1}{2n_{\mathrm{tr}}}\mathrm{tr}(B_{\lambda}^{-1}DHDB_{\lambda}^{-1}C_{w^{\circ},w^{\circ}} + 2B_{\lambda}^{-1}C_{w^{\circ},u}) + o\left(\frac{1}{n_{\mathrm{tr}}}\right). \tag{25}$$

The proof of the above theorem is given in Appendix F. Theorem 5 elucidates the learning curve of uLSIF up to the order of n_{tr}^{-1}. An information criterion may be obtained in the same way as Section 2.4.1. However, as shown in Section 3.4, we can have a closed-form expression of the leave-one-out cross-validation score for uLSIF, which would be practically more useful. For this reason, we do not go into the detail of information criterion.

3.3 Approximation Error Bounds for uLSIF

The uLSIF method is introduced as an approximation of LSIF. Here, we theoretically evaluate the difference between the uLSIF solution $\widehat{\beta}(\lambda)$ and the LSIF solution $\widehat{\alpha}(\lambda)$. More specifically, we use the following normalized L_2-norm on the training samples as the difference measure and derive its upper bounds:

$$\mathrm{diff}(\lambda) = \frac{\inf_{\lambda' \geq 0}\sqrt{\frac{1}{n_{\mathrm{tr}}}\sum_{i=1}^{n_{\mathrm{tr}}}\left(\widehat{w}(x_i^{\mathrm{tr}};\widehat{\alpha}(\lambda')) - \widehat{w}(x_i^{\mathrm{tr}};\widehat{\beta}(\lambda))\right)^2}}{\sum_{i=1}^{n_{\mathrm{tr}}}\widehat{w}(x_i^{\mathrm{tr}};\widehat{\beta}(\lambda))}, \tag{26}$$

where the importance function $\widehat{w}(x;\alpha)$ is given by

$$\widehat{w}(x;\alpha) = \sum_{\ell=1}^{b}\alpha_{\ell}\varphi_{\ell}(x).$$

In the theoretical analysis below, we assume

$$\sum_{i=1}^{n_{\mathrm{tr}}}\widehat{w}(x_i^{\mathrm{tr}};\widehat{\beta}(\lambda)) \neq 0.$$

For $p \in \mathbb{N} \cup \{\infty\}$, let $\|\cdot\|_p$ be the L_p-norm, and let $\|\alpha\|_{\widehat{H}}$ be

$$\|\alpha\|_{\widehat{H}} = \sqrt{\alpha^{\top}\widehat{H}\alpha}, \tag{27}$$

where $\widehat{H}$ is the $b \times b$ matrix defined by Eq. (4). Then we have the following theorem.

Theorem 6 (Norm bound) *Assume that all basis functions satisfy*

$$0 < \varphi_{\ell}(x) \leq 1.$$

Then we have

$$\begin{aligned} \mathrm{diff}(\lambda) &\le \frac{\|\widehat{\beta}(\lambda)\|_{\widehat{H}}}{\sum_{i=1}^{n_{\mathrm{tr}}} \widehat{w}(x_i^{\mathrm{tr}};\widehat{\beta}(\lambda))} && (28)\\ &\le b^2\left(1+\frac{b}{\lambda}\right)\frac{1}{\min_\ell \sum_{i=1}^{n_{\mathrm{tr}}}\varphi_\ell(x_i^{\mathrm{tr}})}\cdot\frac{n_{\mathrm{te}}}{\min_\ell \sum_{j=1}^{n_{\mathrm{te}}}\varphi_\ell(x_j^{\mathrm{te}})}, && (29)\end{aligned}$$

where b is the number of basis functions. The upper bound (29) *is reduced as the regularization parameter λ increases. For the Gaussian basis function model* (22)*, the upper bound (29) is reduced as the Gaussian width σ increases.*

The proof of the above theorem is given in Appendix G. We call Eq. (28) the *norm bound* since it is governed by the norm of $\widehat{\beta}$. Intuitively, the approximation error of uLSIF would small if λ is large since $\widetilde{\beta} \ge 0$ may not be severely violated due to the strong regularization effect. The upper bound (29) justifies this intuitive claim since the error bound tends to be small if the regularization parameter λ is large. Furthermore, the upper bound (29) shows that for the Gaussian basis function model (22), the error bound tends to be small if the Gaussian width σ is large. This is also intuitive since the Gaussian basis functions are nearly flat when the Gaussian width σ is large—a difference in parameters does not cause a significant change in the learned importance function $\widehat{w}(x)$. From the above theorem, we expect that uLSIF is a nice approximation of LSIF when λ is large and σ is large. In Section 4.5, we numerically investigate this issue.

Below, we give a more sophisticated bound on $\mathrm{diff}(\lambda)$. To this end, let us introduce an intermediate optimization problem defined by

$$\begin{aligned} &\min_{\gamma\in\mathbb{R}^b}\left[\frac{1}{2}\gamma^\top\widehat{H}\gamma - \widehat{h}^\top\gamma + \frac{\lambda}{2}\gamma^\top\gamma\right] \\ &\text{subject to } \gamma \ge 0_b, \qquad (30)\end{aligned}$$

which we refer to as *LSIF with quadratic penalty* (LSIFq). LSIFq bridges LSIF and uLSIF since the 'goodness-of-fit' part is the same as LSIF but the 'regularization' part is the same as uLSIF. Let $\widehat{\gamma}(\lambda)$ be the optimal solution of LSIFq (30). Based on the solution of LSIFq, we have the following upper bound.

Theorem 7 (Bridge bound) *For any $\lambda \ge 0$, the following inequality holds:*

$$\mathrm{diff}(\lambda) \le \frac{\sqrt{\lambda\left(\|\widehat{\gamma}(\lambda)\|_1\cdot\|\widehat{\gamma}(\lambda)\|_\infty - \|\widehat{\gamma}(\lambda)\|_2^2\right)} + \|\widehat{\gamma}(\lambda)-\widehat{\beta}(\lambda)\|_{\widehat{H}}}{\sum_{i=1}^{n_{\mathrm{tr}}}\widehat{w}(x_i^{\mathrm{tr}};\widehat{\beta}(\lambda))}. \qquad (31)$$

The proof of the above theorem is given in Appendix H. We call the above bound the *bridge bound* since the bridged estimator $\widehat{\gamma}(\lambda)$ plays a central role in the bound. Note that, in the bridge bound, the inside of the square root is assured to be non-negative due to Hölder's inequality (see Appendix H for detail). The bridge bound is generally much sharper than the norm bound (28), but not always (see Section 4.5 for numerical examples).

3.4 Efficient Computation of Leave-one-out Cross-validation Score for uLSIF

A practically important advantage of uLSIF over LSIF is that the score of leave-one-out cross-validation (LOOCV) can be computed analytically—thanks to this property, the computational complexity for performing LOOCV is the same order as just computing a single solution.

In the current setup, we are given two sets of samples, $\{x_i^{\mathrm{tr}}\}_{i=1}^{n_{\mathrm{tr}}}$ and $\{x_j^{\mathrm{te}}\}_{j=1}^{n_{\mathrm{te}}}$, which generally have different sample size. For simplicity, we assume that $n_{\mathrm{tr}} < n_{\mathrm{te}}$ and the i-th training sample x_i^{tr} and the i-th test sample x_i^{te} are held out at the same time; the test samples $\{x_j^{\mathrm{te}}\}_{j=n_{\mathrm{tr}}+1}^{n_{\mathrm{te}}}$ are always used for importance estimation. Note that this assumption is only for the sake of simplicity; we can change the order of test samples without sacrificing the computational advantages.

Let $\widehat{w}^{(i)}(x)$ be an estimate of the importance obtained without the i-th training sample x_i^{tr} and the i-th test sample x_i^{te}. Then the LOOCV score is expressed as

$$\mathrm{LOOCV} = \frac{1}{n_{\mathrm{tr}}} \sum_{i=1}^{n_{\mathrm{tr}}} \left[\frac{1}{2} (\widehat{w}^{(i)}(x_i^{\mathrm{tr}}))^2 - \widehat{w}^{(i)}(x_i^{\mathrm{te}}) \right]. \tag{32}$$

Our approach to efficiently computing the LOOCV score is to use the *Sherman-Woodbury-Morrison formula* (Golub and Loan, 1996) for computing matrix inverses: for an invertible square matrix A and vectors u and v such that $v^\top A^{-1} u \neq -1$,

$$(A + uv^\top)^{-1} = A^{-1} - \frac{A^{-1} u v^\top A^{-1}}{1 + v^\top A^{-1} u}. \tag{33}$$

Efficient approximation schemes of LOOCV have often been investigated under asymptotic setups (Stone, 1974; Hansen and Larsen, 1996). On the other hand, we provide the exact LOOCV score of uLSIF, which follows the same line as that of ridge regression (Hoerl and Kennard, 1970; Wahba, 1990).

A pseudo code of uLSIF with LOOCV-based model selection is summarized in Figure 2—its detailed derivation is described in Appendix I. Note that the basis-function design heuristic given in Section 2.5 is used in the pseudo code, but the analytic form of the LOOCV score is available for any basis functions.

4. Illustrative Examples

In this section, we illustrate the behavior of LSIF and uLSIF using a toy data set.

4.1 Setup

Let the dimension of the domain be $d = 1$ and the training and test densities be

$$p_{\mathrm{tr}}(x) = \mathcal{N}(x; 1, (1/2)^2),$$
$$p_{\mathrm{te}}(x) = \mathcal{N}(x; 2, (1/4)^2),$$

where $\mathcal{N}(x; \mu, \sigma^2)$ denotes the Gaussian density with mean μ and variance σ^2. These densities are depicted in Figure 3. The task is to estimate the importance $w(x) = p_{\mathrm{te}}(x)/p_{\mathrm{tr}}(x)$.

```
Input: $\{x_i^{\mathrm{tr}}\}_{i=1}^{n_{\mathrm{tr}}}$ and $\{x_j^{\mathrm{te}}\}_{j=1}^{n_{\mathrm{te}}}$
Output: $\widehat{w}(x)$

$b \longleftarrow \min(100, n_{\mathrm{te}})$;    $n \longleftarrow \min(n_{\mathrm{tr}}, n_{\mathrm{te}})$;
Randomly choose $b$ centers $\{c_\ell\}_{\ell=1}^{b}$ from $\{x_j^{\mathrm{te}}\}_{j=1}^{n_{\mathrm{te}}}$ without replacement;
For each candidate of Gaussian width $\sigma$
    $\widehat{H}_{\ell,\ell'} \longleftarrow \frac{1}{n_{\mathrm{tr}}}\sum_{i=1}^{n_{\mathrm{tr}}}\exp\left(-\frac{\|x_i^{\mathrm{tr}}-c_\ell\|^2+\|x_i^{\mathrm{tr}}-c_{\ell'}\|^2}{2\sigma^2}\right)$ for $\ell,\ell'=1,2,\ldots,b$;
    $\widehat{h}_\ell \longleftarrow \frac{1}{n_{\mathrm{te}}}\sum_{j=1}^{n_{\mathrm{te}}}\exp\left(-\frac{\|x_j^{\mathrm{te}}-c_\ell\|^2}{2\sigma^2}\right)$ for $\ell=1,2,\ldots,b$;
    $X_{\ell,i}^{\mathrm{tr}} \longleftarrow \exp\left(-\frac{\|x_i^{\mathrm{tr}}-c_\ell\|^2}{2\sigma^2}\right)$ for $i=1,2,\ldots,n$ and $\ell=1,2,\ldots,b$;
    $X_{\ell,i}^{\mathrm{te}} \longleftarrow \exp\left(-\frac{\|x_i^{\mathrm{te}}-c_\ell\|^2}{2\sigma^2}\right)$ for $i=1,2,\ldots,n$ and $\ell=1,2,\ldots,b$;
    For each candidate of regularization parameter $\lambda$
        $\widehat{B} \longleftarrow \widehat{H}+\frac{\lambda(n_{\mathrm{tr}}-1)}{n_{\mathrm{tr}}}I_b$;
        $B_0 \longleftarrow \widehat{B}^{-1}\widehat{h}1_n^\top+\widehat{B}^{-1}X^{\mathrm{tr}}\,\mathrm{diag}\left(\frac{\widehat{h}^\top\widehat{B}^{-1}X^{\mathrm{tr}}}{n_{\mathrm{tr}}1_n^\top-1_b^\top(X^{\mathrm{tr}}*\widehat{B}^{-1}X^{\mathrm{tr}})}\right)$;
        $B_1 \longleftarrow \widehat{B}^{-1}X^{\mathrm{te}}+\widehat{B}^{-1}X^{\mathrm{tr}}\,\mathrm{diag}\left(\frac{1_b^\top(X^{\mathrm{te}}*\widehat{B}^{-1}X^{\mathrm{tr}})}{n_{\mathrm{tr}}1_n^\top-1_b^\top(X^{\mathrm{tr}}*\widehat{B}^{-1}X^{\mathrm{tr}})}\right)$;
        $B_2 \longleftarrow \max\left(O_{b\times n},\ \frac{n_{\mathrm{tr}}-1}{n_{\mathrm{tr}}(n_{\mathrm{te}}-1)}(n_{\mathrm{te}}B_0-B_1)\right)$;
        $w_{\mathrm{tr}} \longleftarrow (1_b^\top(X^{\mathrm{tr}}*B_2))^\top$;    $w_{\mathrm{te}} \longleftarrow (1_b^\top(X^{\mathrm{te}}*B_2))^\top$;
        $\mathrm{LOOCV}(\sigma,\lambda) \longleftarrow \frac{w_{\mathrm{tr}}^\top w_{\mathrm{tr}}}{2n}-\frac{1_n^\top w_{\mathrm{te}}}{n}$;
    end
end
$(\widehat{\sigma},\widehat{\lambda}) \longleftarrow \mathrm{argmin}_{(\sigma,\lambda)}\mathrm{LOOCV}(\sigma,\lambda)$;
$\widetilde{H}_{\ell,\ell'} \longleftarrow \frac{1}{n_{\mathrm{tr}}}\sum_{i=1}^{n_{\mathrm{tr}}}\exp\left(-\frac{\|x_i^{\mathrm{tr}}-c_\ell\|^2+\|x_i^{\mathrm{tr}}-c_{\ell'}\|^2}{2\widehat{\sigma}^2}\right)$ for $\ell,\ell'=1,2,\ldots,b$;
$\widetilde{h}_\ell \longleftarrow \frac{1}{n_{\mathrm{te}}}\sum_{j=1}^{n_{\mathrm{te}}}\exp\left(-\frac{\|x_j^{\mathrm{te}}-c_\ell\|^2}{2\widehat{\sigma}^2}\right)$ for $\ell=1,2,\ldots,b$;
$\widehat{\alpha} \longleftarrow \max(0_b,(\widetilde{H}+\widehat{\lambda}I_b)^{-1}\widetilde{h})$;
$\widehat{w}(x) \longleftarrow \sum_{\ell=1}^{b}\widehat{\alpha}_\ell\exp\left(-\frac{\|x-c_\ell\|^2}{2\widehat{\sigma}^2}\right)$;
```

Figure 2: Pseudo code of uLSIF algorithm with LOOCV. $B*B'$ denotes the element-wise multiplication of matrices B and B' of the same size, that is, the (i,j)-th element is given by $B_{i,j}B'_{i,j}$. For n-dimensional vectors b and b', $\mathrm{diag}\left(\frac{b}{b'}\right)$ denotes the $n\times n$ diagonal matrix with i-th diagonal element b_i/b'_i. A MATLAB® or R implementation of uLSIF is available from `http://sugiyama-www.cs.titech.ac.jp/~sugi/software/uLSIF/`.

4.2 Importance Estimation

First, we illustrate the behavior of LSIF and uLSIF in importance estimation. We set the number of training and test samples at $n_{\mathrm{tr}} = 200$ and $n_{\mathrm{te}} = 1000$, respectively. We use the Gaussian kernel model (22), and the number of basis functions is set at $b = 100$. The centers of the kernel function are randomly chosen from the test points $\{x_i^{\mathrm{te}}\}_{j=1}^{n_{\mathrm{te}}}$ without replacement (see Section 2.5).

We test different Gaussian widths σ and different regularization parameters λ. The following two setups are examined:

(A) λ is fixed at $\lambda = 0.2$ and σ is changed as $0.1 \leq \sigma \leq 1.0$,

(B) σ is fixed at $\sigma = 0.3$ and λ is changed as $0 \leq \lambda \leq 0.5$.

Figure 4 depicts the true importance and its estimates obtained by LSIF and uLSIF, where all importance functions are normalized so that $\int w(x)dx = 1$ for better comparison. Figures 4(a) and 4(b) show that the estimated importance $\widehat{w}(x)$ tends to be too peaky when the Gaussian width σ is small, while it tends to be overly smoothed when σ is large. If the Gaussian width is chosen appropriately, both LSIF and uLSIF seem to work reasonably well. As shown in Figures 4(c) and 4(d), the solutions of LSIF and uLSIF also significantly change when different regularization parameters λ are used. Again, given that the regularization parameter is chosen appropriately, both LSIF and uLSIF tend to perform well.

From the graphs, we also observe that model selection based on cross-validation works reasonably well for both LSIF (5-fold) and uLSIF (leave-one-out) to choose appropriate values of the Gaussian width or the regularization parameter; this will be analyzed in more detail in Section 4.4.

4.3 Regularization Path

Next, we illustrate how the regularization path tracking algorithm for LSIF behaves. We set the number of training and test samples at $n_{\mathrm{tr}} = 50$ and $n_{\mathrm{te}} = 100$, respectively. For better illustration, we set the number of basis functions at a small value as $b = 30$ in the Gaussian kernel model (22) and use the Gaussian kernels centered at equidistant points in $[0, 3]$ as basis functions.

We use the algorithm described in Figure 1 for regularization path tracking. Theoretically, the inequality $\lambda_{\tau+1} < \lambda_\tau$ is assured. In numerical computation, however, the inequality is occasionally violated. In order to avoid this numerical problem, we slightly regularize $\widehat{H}$ for stabilization (see also the caption of Figure 1).

Figure 5 depicts the values of the estimated coefficients $\{\alpha_\ell\}_{\ell=1}^{b}$ as functions of $\|\alpha\|_1$ for $\sigma = 0.1, 0.3$, and 0.5. Note that small $\|\alpha\|_1$ corresponds to large λ. The figure indicates that the regularization parameter λ works as a sparseness controlling factor of the solution, that is, the larger (smaller) the value of λ ($\|\alpha\|_1$) is, the sparser the solution is.

The path following algorithm is computationally efficient and therefore practically very attractive. However, as the above experiments illustrate, the path following algorithm is numerically rather unstable. Modification of $\widehat{H}$ can ease to solve this problem, but this in turn results in accumulating numerical errors through the path tracking process. Consequently, the solutions for small λ tend to be inaccurate. This problem becomes prominent if the number of change points in the regularization path is large.

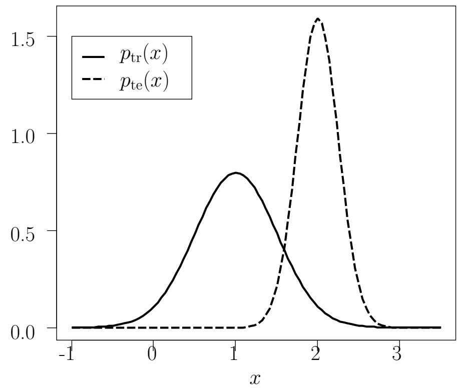
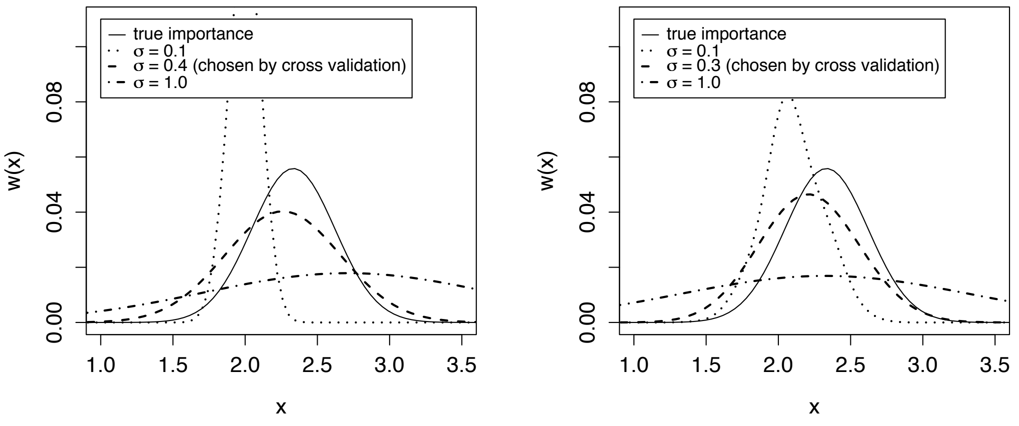
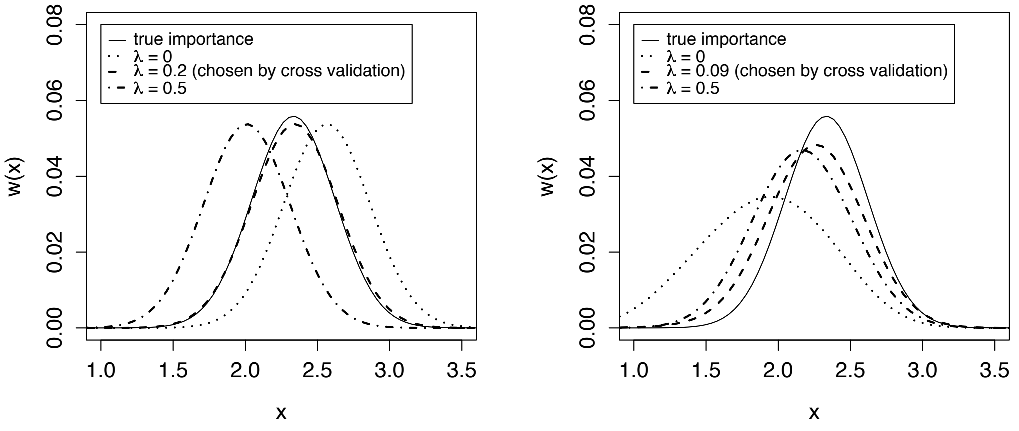


Figure 3: The solid line is the probability density of training data, and the dashed line is the probability density of test data.



(a) LSIF for $\lambda = 0.2, \sigma = 0.1, 0.4, 1.0$.

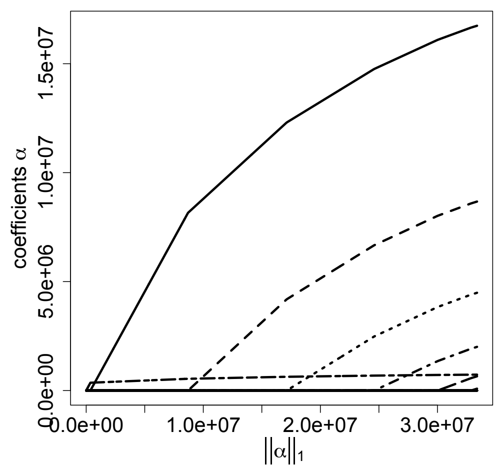
(b) uLSIF for $\lambda = 0.2, \sigma = 0.1, 0.3, 1.0$.



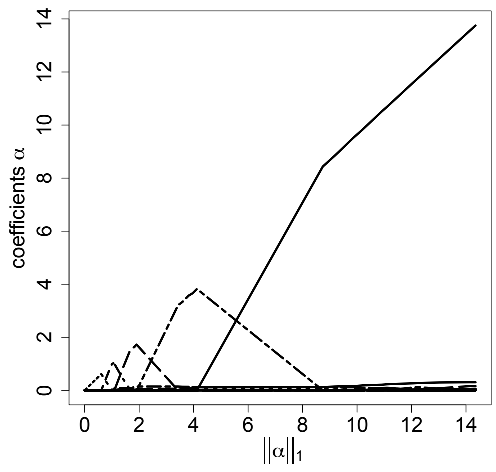
(c) LSIF for $\sigma = 0.3, \lambda = 0, 0.2, 0.5$.

(d) uLSIF for $\sigma = 0.3, \lambda = 0, 0.09, 0.5$.

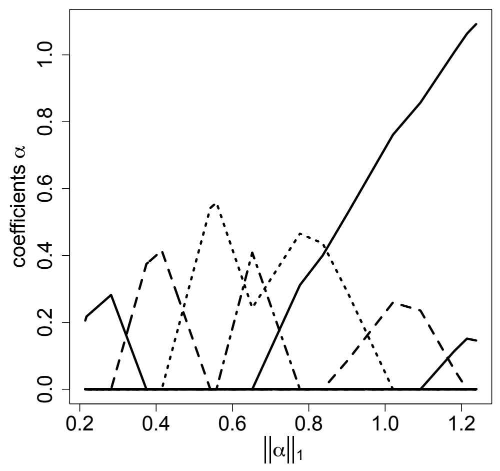
Figure 4: True and estimated importance functions obtained by LSIF and uLSIF for various different Gaussian widths σ and regularization parameters λ.



(a) $\sigma = 0.1$.



(b) $\sigma = 0.3$.

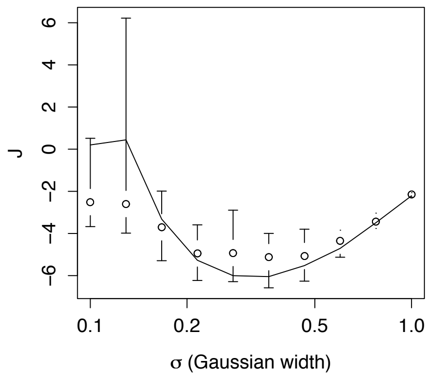


(c) $\sigma = 0.5$.

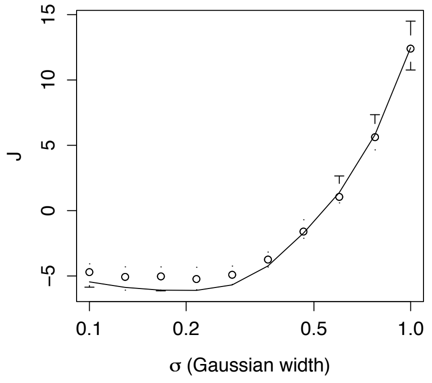
Figure 5: Regularization path of LSIF: the values of the estimated coefficients $\{\alpha_\ell\}_{\ell=1}^{b}$ are depicted as functions of the L_1-norm of the estimated parameter vector for $\sigma = 0.1, 0.3$, and 0.5. Small $\|\alpha\|_1$ corresponds to large λ.

4.4 Cross-validation

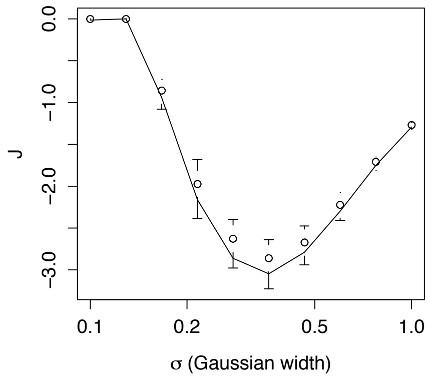
Here we illustrate the behavior of the cross-validation scores of LSIF and uLSIF. We set the number of training and test samples at $n_{\mathrm{tr}} = 200$ and $n_{\mathrm{te}} = 1000$, respectively. The number of template points is $b = 100$ and the Gaussian kernel model (22) is used. The centers of the kernel functions are randomly chosen from the test points as described in Section 4.2. The left column of Figure 6 depicts the expectation of the true cost $J(\widehat{\alpha})$ over 50 runs for LSIF and its estimate by 5-fold CV (25, 50, and 75 percentiles are plotted in the figure) as functions of the Gaussian width σ for $\lambda = 0.2, 0.5$, and 0.8. We used the regularization path tracking algorithm for computing the solutions of LSIF.



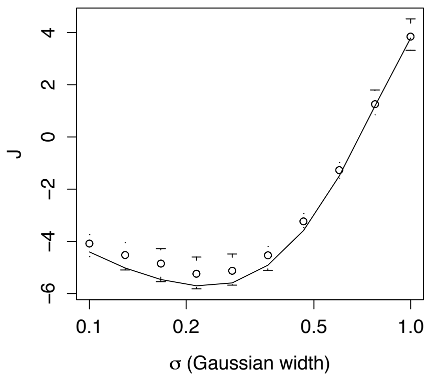
(a) LSIF with 5CV ($\lambda = 0.2$).



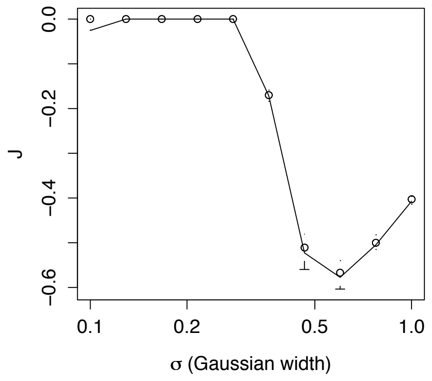
(b) uLSIF with LOOCV ($\lambda = 0.2$).



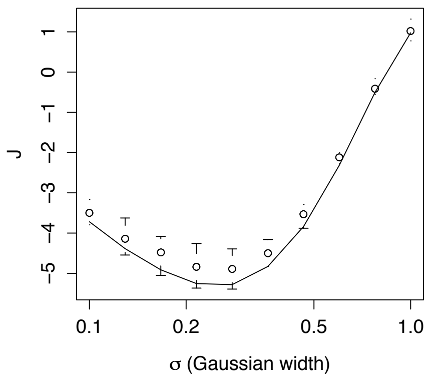
(c) LSIF with 5CV ($\lambda = 0.5$).



(d) uLSIF with LOOCV ($\lambda = 0.5$).



(e) LSIF with 5CV ($\lambda = 0.8$).



(f) uLSIF with LOOCV ($\lambda = 0.8$).

Figure 6: The true cost J and its cross-validation estimate as functions of Gaussian width σ for different values of λ. The solid line denotes the expectation of the true cost J over 50 runs, while '$\circ$' and error bars denote the 25, 50, and 75 percentiles of the cross-validation score.

The right column shows the expected true cost and its LOOCV estimates for uLSIF in the same manner.

The graphs show that overall CV gives reasonably good approximations of the expected cost, although CV for LSIF with small λ and small σ is rather inaccurate due to numerical problems—the solution path of LSIF is computed from $\lambda = \infty$ to $\lambda = 0$, and the numerical error is accumulated as the tracking process approaches to $\lambda = 0$. This phenomenon seems problematic when σ is small.

4.5 Difference between LSIF and uLSIF

In Section 3.3, we analyzed the approximation error of uLSIF against LSIF. Here we numerically investigate the behavior of the approximation error (26) as well as the norm bound (28) and the bridge bound (31). We set the number of training and test samples at $n_{\mathrm{tr}} = 200$ and $n_{\mathrm{te}} = 1000$, respectively. The number of template points in the Gaussian kernel model (22) is set at $b = 100$. The centers of the kernel functions are randomly chosen from the test points (see Section 4.2).

Figure 7 depicts the true approximation error as well as its upper bounds as functions of the regularization parameter λ; λ is varied from 0.001 to 10 and the three Gaussian widths $\sigma = 0.1, 0.5, 1.0$ are tested. The graphs show that when λ and σ are large, the approximation error tends to be small; this is in good agreement with the theoretical analysis given in Section 3.3. The bridge bound is fairly tight in the entire range and is sharper than the norm bound except when σ is small and λ is large.

4.6 Summary

Through the numerical examples, we overall found that LSIF and uLSIF give qualitatively similar results. However, the computation of the solution-path tracking algorithm for LSIF tends to be numerically unstable, which can result in unreliable model selection performance. On the other hand, only a system of linear equations needs to be solved in uLSIF, which turned out to be much more stable than LSIF. Thus, uLSIF would be practically more reliable than LSIF.

Based on the above findings, we will focus on uLSIF in the rest of this paper.

5. Relation to Existing Methods

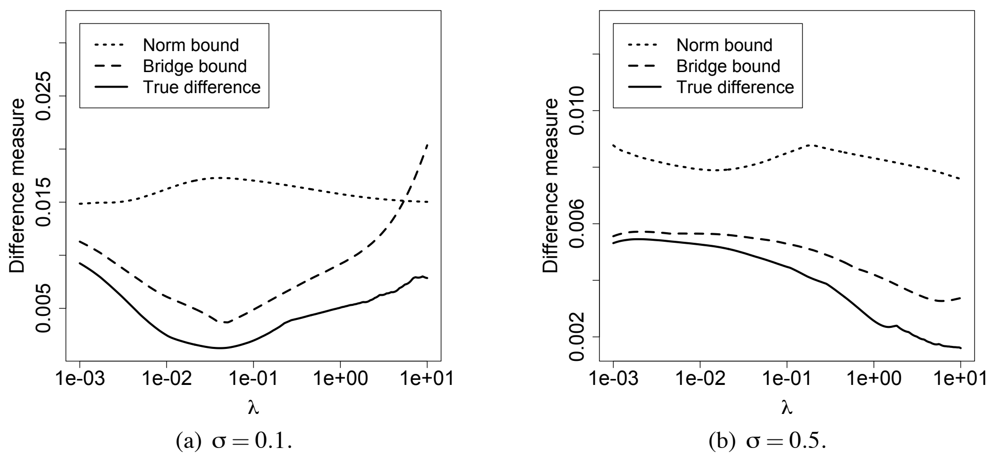
In this section, we discuss the characteristics of existing approaches in comparison with the proposed methods.

5.1 Kernel Density Estimator

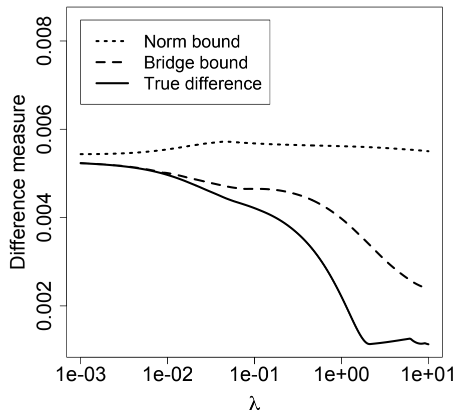
The *kernel density estimator* (KDE) is a non-parametric technique to estimate a probability density function $p(x)$ from its i.i.d. samples $\{x_k\}_{k=1}^{n}$. For the Gaussian kernel (21), KDE is expressed as

$$\widehat{p}(x) = \frac{1}{n_{\mathrm{tr}}(2\pi\sigma^2)^{d/2}} \sum_{k=1}^{n} K_{\sigma}(x, x_k).$$

The performance of KDE depends on the choice of the kernel width σ. The kernel width σ can be optimized by *likelihood cross-validation* (LCV) as follows (Härdle et al., 2004): First, divide the samples $\{x_i\}_{i=1}^{n}$ into R disjoint subsets $\{\mathcal{X}_r\}_{r=1}^{R}$. Then obtain a density estimate $\widehat{p}_{\mathcal{X}_k}(x)$ from



(a) $\sigma = 0.1$. (b) $\sigma = 0.5$.



(c) $\sigma = 1.0$.

Figure 7: The approximation error of uLSIF against LSIF as functions of the regularization parameter λ for different Gaussian width σ. Its upper bounds are also plotted in the graphs.

$\{\mathcal{X}_r\}_{r \neq k}$ (i.e., without $\mathcal{X}_k$) and compute its log-likelihood for $\mathcal{X}_k$:

$$\frac{1}{|\mathcal{X}_k|} \sum_{x \in \mathcal{X}_k} \log \widehat{p}_{\mathcal{X}_k}(x).$$

Repeat this procedure for $r = 1, 2, \ldots, R$ and choose the value of σ such that the average of the above hold-out log-likelihood over all r is maximized. Note that the average hold-out log-likelihood is an almost unbiased estimate of the Kullback-Leibler divergence from $p(x)$ to $\widehat{p}(x)$, up to an irrelevant constant.

KDE can be used for importance estimation by first obtaining density estimators $\widehat{p}_{\mathrm{tr}}(x)$ and $\widehat{p}_{\mathrm{te}}(x)$ separately from $\{x_i^{\mathrm{tr}}\}_{i=1}^{n_{\mathrm{tr}}}$ and $\{x_j^{\mathrm{te}}\}_{j=1}^{n_{\mathrm{te}}}$, and then estimating the importance by $\widehat{w}(x) = \widehat{p}_{\mathrm{te}}(x)/\widehat{p}_{\mathrm{tr}}(x)$. A potential limitation of this approach is that KDE suffers from the *curse of dimensionality* (Vapnik, 1998; Härdle et al., 2004), that is, the number of samples needed to maintain the same approximation quality grows exponentially as the dimension of the domain increases. This is critical when the number of available samples is limited. Therefore, the KDE-based approach may not be reliable in high-dimensional problems.

5.2 Kernel Mean Matching

The *kernel mean matching* (KMM) method allows us to directly obtain an estimate of the importance values at training points without going through density estimation (Huang et al., 2007). The basic idea of KMM is to find $\widehat{w}(x)$ such that the mean discrepancy between nonlinearly transformed samples drawn from $p_{\mathrm{te}}(x)$ and $p_{\mathrm{tr}}(x)$ is minimized in a *universal reproducing kernel Hilbert space* (Steinwart, 2001). The Gaussian kernel (21) is an example of kernels that induce universal reproducing kernel Hilbert spaces and it has been shown that the solution of the following optimization problem agrees with the true importance:

$$\min_{w(x)} \left\| \int K_\sigma(x,\cdot)p_{\mathrm{te}}(x)dx - \int K_\sigma(x,\cdot)w(x)p_{\mathrm{tr}}(x)dx \right\|^2_{\mathcal{H}}$$
$$\text{subject to } \int w(x)p_{\mathrm{tr}}(x)dx = 1 \text{ and } w(x) \geq 0,$$

where $\|\cdot\|_{\mathcal{H}}$ denotes the norm in the Gaussian reproducing kernel Hilbert space and $K_\sigma(x,x')$ is the Gaussian kernel (21).

An empirical version of the above problem is reduced to the following quadratic program:

$$\min_{\{w_i\}_{i=1}^{n_{\mathrm{tr}}}} \left[\frac{1}{2} \sum_{i,i'=1}^{n_{\mathrm{tr}}} w_i w_{i'} K_\sigma(x_i^{\mathrm{tr}}, x_{i'}^{\mathrm{tr}}) - \sum_{i=1}^{n_{\mathrm{tr}}} w_i \kappa_i \right]$$
$$\text{subject to } \left| \sum_{i=1}^{n_{\mathrm{tr}}} w_i - n_{\mathrm{tr}} \right| \leq n_{\mathrm{tr}}\varepsilon \text{ and } 0 \leq w_1, w_2, \ldots, w_{n_{\mathrm{tr}}} \leq B,$$

where

$$\kappa_i = \frac{n_{\mathrm{tr}}}{n_{\mathrm{te}}} \sum_{j=1}^{n_{\mathrm{te}}} K_\sigma(x_i^{\mathrm{tr}}, x_j^{\mathrm{te}}).$$

B (≥ 0) and ε (≥ 0) are tuning parameters that control the regularization effects. The solution $\{\widehat{w}_i\}_{i=1}^{n_{\mathrm{tr}}}$ is an estimate of the importance at the training points $\{x_i^{\mathrm{tr}}\}_{i=1}^{n_{\mathrm{tr}}}$.

Since KMM does not involve density estimation, it is expected to work well even in high dimensional cases. However, the performance is dependent on the tuning parameters B, ε, and σ, and they cannot be simply optimized, for example, by CV since estimates of the importance are available only at the training points. A popular heuristic is to use the median distance between samples as the Gaussian width σ, which is shown to be useful (Schölkopf and Smola, 2002; Song et al., 2007). However, there seems no strong justification for this heuristic. For the choice of ε, a theoretical result given in Huang et al. (2007) could be used as guidance, although it is still hard to determine the best value of ε in practice.

5.3 Logistic Regression

Another approach to directly estimating the importance is to use a probabilistic classifier. Let us assign a selector variable $\eta = -1$ to training samples and $\eta = 1$ to test samples, that is, the training and test densities are written as

$$p_{\mathrm{tr}}(x) = p(x|\eta = -1),$$
$$p_{\mathrm{te}}(x) = p(x|\eta = 1).$$

Note that η is regarded as a random variable.

Application of the Bayes theorem yields that the importance can be expressed in terms of η as follows (Qin, 1998; Cheng and Chu, 2004; Bickel et al., 2007):

$$w(x) = \frac{p(\eta = -1)}{p(\eta = 1)} \frac{p(\eta = 1|x)}{p(\eta = -1|x)}.$$

The probability ratio of test and training samples may be simply estimated by the ratio of the numbers of samples:

$$\frac{p(\eta = -1)}{p(\eta = 1)} \approx \frac{n_{\mathrm{tr}}}{n_{\mathrm{te}}}.$$

The conditional probability $p(\eta|x)$ could be approximated by discriminating test samples from training samples using a *logistic regression* (LogReg) classifier, where η plays the role of a class variable. Below we briefly explain the LogReg method.

The LogReg classifier employs a parametric model of the following form for expressing the conditional probability $p(\eta|x)$:

$$\widehat{p}(\eta|x) = \frac{1}{1 + \exp\left(-\eta \sum_{\ell=1}^{m} \zeta_\ell \phi_\ell(x)\right)},$$

where m is the number of basis functions and $\{\phi_\ell(x)\}_{\ell=1}^{m}$ are fixed basis functions. The parameter ζ is learned so that the negative regularized log-likelihood is minimized:

$$\widehat{\zeta} = \underset{\zeta}{\mathrm{argmin}} \left[\sum_{i=1}^{n_{\mathrm{tr}}} \log\left(1 + \exp\left(\sum_{\ell=1}^{m} \zeta_\ell \phi_\ell(x_i^{\mathrm{tr}})\right)\right) \right.$$
$$\left. + \sum_{j=1}^{n_{\mathrm{te}}} \log\left(1 + \exp\left(-\sum_{\ell=1}^{m} \zeta_\ell \phi_\ell(x^{\mathrm{te}})\right)\right) + \lambda \zeta^\top \zeta \right].$$

Since the above objective function is convex, the global optimal solution can be obtained by standard nonlinear optimization methods such as Newton's method, the conjugate gradient method, and the BFGS method (Minka, 2007). Then the importance estimate is given by

$$\widehat{w}(x) = \frac{n_{\mathrm{tr}}}{n_{\mathrm{te}}} \exp\left(\sum_{\ell=1}^{m} \zeta_\ell \phi_\ell(x)\right). \tag{34}$$

An advantage of the LogReg method is that model selection (that is, the choice of the basis functions $\{\phi_\ell(x)\}_{\ell=1}^{m}$ as well as the regularization parameter λ) is possible by standard CV since the learning problem involved above is a standard supervised classification problem.

5.4 Kullback-Leibler Importance Estimation Procedure

The *Kullback-Leibler importance estimation procedure* (KLIEP) (Sugiyama et al., 2008a) also directly gives an estimate of the importance function without going through density estimation by matching the two distributions in terms of the Kullback-Leibler divergence (Kullback and Leibler, 1951).

Let us model the importance $w(x)$ by the linear model (1). An estimate of the test density $p_{\mathrm{te}}(x)$ is given by using the model $\widehat{w}(x)$ as

$$\widehat{p}_{\mathrm{te}}(x) = \widehat{w}(x) p_{\mathrm{tr}}(x).$$

In KLIEP, the parameters α are determined so that the Kullback-Leibler divergence from $p_{\mathrm{te}}(x)$ to $\widehat{p}_{\mathrm{te}}(x)$ is minimized:

$$\begin{aligned}\mathrm{KL}[p_{\mathrm{te}}(x)\|\widehat{p}_{\mathrm{te}}(x)] &= \int_{\mathcal{D}} p_{\mathrm{te}}(x) \log \frac{p_{\mathrm{te}}(x)}{\widehat{w}(x)p_{\mathrm{tr}}(x)} dx \\ &= \int_{\mathcal{D}} p_{\mathrm{te}}(x) \log \frac{p_{\mathrm{te}}(x)}{p_{\mathrm{tr}}(x)} dx - \int_{\mathcal{D}} p_{\mathrm{te}}(x) \log \widehat{w}(x) dx. \end{aligned} \tag{35}$$

The first term is a constant, so it can be safely ignored. Since $\widehat{p}_{\mathrm{te}}(x)$ $(= \widehat{w}(x)p_{\mathrm{tr}}(x))$ is a probability density function, it should satisfy

$$1 = \int_{\mathcal{D}} \widehat{p}_{\mathrm{te}}(x) dx = \int_{\mathcal{D}} \widehat{w}(x) p_{\mathrm{tr}}(x) dx. \tag{36}$$

Then the KLIEP optimization problem is given by replacing the expectations in Eqs. (35) and (36) with empirical averages as follows:

$$\begin{aligned}&\max_{\{\alpha_\ell\}_{\ell=1}^{b}} \left[\sum_{j=1}^{n_{\mathrm{te}}} \log \left(\sum_{\ell=1}^{b} \alpha_\ell \varphi_\ell(x_j^{\mathrm{te}}) \right) \right] \\ &\qquad \text{subject to } \sum_{\ell=1}^{b} \alpha_\ell \left(\sum_{i=1}^{n_{\mathrm{tr}}} \varphi_\ell(x_i^{\mathrm{tr}}) \right) = n_{\mathrm{tr}} \text{ and } \alpha_1, \alpha_2, \ldots, \alpha_b \geq 0.\end{aligned}$$

This is a convex optimization problem and the global solution—which tends to be sparse (Boyd and Vandenberghe, 2004)—can be obtained, for example, by simply performing gradient ascent and feasibility satisfaction iteratively. Model selection of KLIEP is possible by LCV.

Properties of KLIEP-type algorithms have been theoretically investigated in Nguyen et al. (2008) and Sugiyama et al. (2008b) (see also Qin, 1998; Cheng and Chu, 2004). Note that the importance model of KLIEP is the linear model (1), while that of LogReg is the log-linear model (34). A variant of KLIEP for log-linear models has been studied in Tsuboi et al. (2008).

5.5 Discussions

Table 1 summarizes properties of proposed and existing methods.

KDE is efficient in computation since no optimization is involved, and model selection is possible by LCV. However, KDE may suffer from the curse of dimensionality due to the difficulty of density estimation in high dimensions.

Methods	Density estimation	Model selection	Optimization	Out-of-sample prediction
KDE	Necessary	**Available**	**Analytic**	**Possible**
KMM	**Not necessary**	Not available	Convex quadratic program	Not possible
LogReg	**Not necessary**	**Available**	Convex non-linear	**Possible**
KLIEP	**Not necessary**	**Available**	Convex non-linear	**Possible**
LSIF	**Not necessary**	**Available**	Convex quadratic program	**Possible**
uLSIF	**Not necessary**	**Available**	**Analytic**	**Possible**

Table 1: Relation between proposed and existing methods.

KMM can potentially overcome the curse of dimensionality by directly estimating the importance. However, there is no objective model selection method. Therefore, model parameters such as the Gaussian width need to be determined by hand, which is highly unreliable unless we have strong prior knowledge. Furthermore, the computation of KMM is rather demanding since a quadratic programming problem has to be solved.

LogReg and KLIEP also do not involve density estimation, but different from KMM, they give an estimate the entire importance function, not only the values of the importance at training points. Therefore, the values of the importance at unseen points can be estimated by LogReg and KLIEP. This feature is highly useful since it enables us to employ CV for model selection, which is a significant advantage over KMM. However, LogReg and KLIEP are computationally rather expensive since non-linear optimization problems have to be solved. Note that the LogReg method is slightly different in motivation from other methods, but has some similarity in computation and implementation, for example, the LogReg method also involves a kernel smoother.

The proposed LSIF method is qualitatively similar to LogReg and KLIEP, that is, it can avoid density estimation, model selection is possible, and non-linear optimization is involved. LSIF is advantageous over LogReg and KLIEP in that it is equipped with a regularization path tracking algorithm. Thanks to this, model selection of LSIF is computationally much more efficient than LogReg and KLIEP. However, the regularization path tracking algorithm tends to be numerically unstable.

The proposed uLSIF method inherits good properties of existing methods such as no density estimation involved and a build-in model selection method equipped. In addition to these preferable properties, the solution of uLSIF can be computed in an efficient and numerically stable manner. Furthermore, thanks to the availability of the closed-form solution of uLSIF, the LOOCV score can be analytically computed without repeating hold-out loops, which highly contributes to reducing the computation time in the model selection phase.

In the next section, we experimentally show that uLSIF is computationally more efficient than existing direct importance estimation methods, while its estimation accuracy is comparable to the best existing methods.

6. Experiments

In this section, we compare the experimental performance of the proposed and existing methods.

6.1 Importance Estimation

Let the dimension of the domain be d and

$$p_{\mathrm{tr}}(x) = \mathcal{N}(x;(0,0,\ldots,0)^\top, I_d),$$
$$p_{\mathrm{te}}(x) = \mathcal{N}(x;(1,0,\ldots,0)^\top, I_d).$$

The task is to estimate the importance at training points:

$$w_i = w(x_i^{\mathrm{tr}}) = \frac{p_{\mathrm{te}}(x_i^{\mathrm{tr}})}{p_{\mathrm{tr}}(x_i^{\mathrm{tr}})} \quad \text{for } i = 1,2,\ldots,n_{\mathrm{tr}}.$$

We compare the following methods:

KDE(CV): The Gaussian kernel (21) is used, where the kernel widths of the training and test densities are separately optimized based on 5-fold LCV.

KMM(med): The performance of KMM is dependent on B, ε, and σ. We set $B = 1000$ and $\varepsilon = (\sqrt{n_{\mathrm{tr}}} - 1)/\sqrt{n_{\mathrm{tr}}}$ following the original paper (Huang et al., 2007), and the Gaussian width σ is set at the median distance between samples within the training set and the test set (Schölkopf and Smola, 2002; Song et al., 2007).

LogReg(CV): The Gaussian kernel model (22) are used as basis functions. The kernel width σ and the regularization parameter λ are chosen based on 5-fold CV.[1]

KLIEP(CV): The Gaussian kernel model (22) is used. The kernel width σ is selected based on 5-fold LCV.

uLSIF(CV): The Gaussian kernel model (22) is used. The kernel width σ and the regularization parameter λ are determined based on LOOCV.

All the methods are implemented using the *MATLAB*® environment, where the *CPLEX*® optimizer is used for solving quadratic programs in KMM and the *LIBLINEAR* implementation is used for LogReg (Lin et al., 2007).

We fixed the number of test points at $n_{\mathrm{te}} = 1000$ and consider the following two setups for the number n_{tr} of training samples and the input dimensionality d:

(a) n_{tr} is fixed at $n_{\mathrm{tr}} = 100$ and d is changed as $d = 1,2,\ldots,20$,

(b) d is fixed at $d = 10$ and n_{tr} is changed as $n_{\mathrm{tr}} = 50,60,\ldots,150$.

We run the experiments 100 times for each d, each n_{tr}, and each method, and evaluate the quality of the importance estimates $\{\widehat{w}_i\}_{i=1}^{n_{\mathrm{tr}}}$ by the *normalized mean squared error* (NMSE):

$$\mathrm{NMSE} = \frac{1}{n_{\mathrm{tr}}} \sum_{i=1}^{n_{\mathrm{tr}}} \left(\frac{\widehat{w}_i}{\sum_{i'=1}^{n_{\mathrm{tr}}} \widehat{w}_{i'}} - \frac{w_i}{\sum_{i'=1}^{n_{\mathrm{tr}}} w_{i'}} \right)^2 .$$

1. In Sugiyama et al. (2008b) where KLIEP has been proposed, the performance of LogReg has been experimentally investigated in the same setup. In that paper, however, LogReg was not regularized since KLIEP was not also regularized. On the other hand, we use a regularized LogReg method and choose the regularization parameter in addition to the Gaussian kernel width by CV here. Thanks to the regularization effect, the results of LogReg in the current paper tends to be better than that reported in Sugiyama et al. (2008b).

In practice, the scale of the importance is not significant and the relative magnitude among w_i is important. Thus the above NMSE would be a suitable error metric for evaluating the performance of each method.

NMSEs averaged over 100 trials (a) as a function of input dimensionality d and (b) as a function of the training sample size n_{tr} are plotted in log scale in Figure 8. Error bars are omitted for clear visibility—instead, the best method in terms of the mean error and comparable ones based on the t-test at the significance level 1% are indicated by '$\circ$'; the methods with significant difference from the best methods are indicated by '$\times$'.

Figure 8(a) shows that the error of KDE(CV) sharply increases as the input dimensionality grows, while LogReg, KLIEP, and uLSIF tend to give much smaller errors than KDE. This would be the fruit of directly estimating the importance without going through density estimation. KMM tends to perform poorly, which is caused by an inappropriate choice of the Gaussian kernel width. On the other hand, model selection in LogReg, KLIEP, and uLSIF seems to work quite well. Figure 8(b) shows that the errors of all methods tend to decrease as the number of training samples grows. Again LogReg, KLIEP, and uLSIF tend to give much smaller errors than KDE and KMM.

Next we investigate the computation time. Each method has a different model selection strategy, that is, KMM does not involve CV, KDE and KLIEP involve CV over the kernel width, and LogReg and uLSIF involve CV over both the kernel width and the regularization parameter. Thus the naive comparison of the total computation time is not so meaningful. For this reason, we first investigate the computation time of each importance estimation method after the model parameters are fixed.

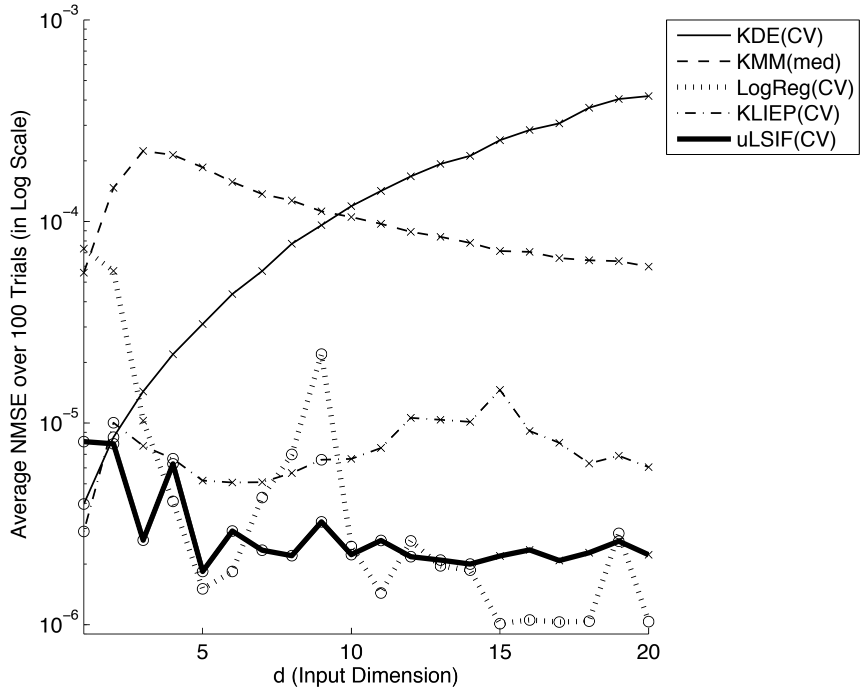
The average CPU computation time over 100 trials are summarized in Figure 9. Figure 9(a) shows that the computation time of KDE, KLIEP, and uLSIF is almost independent of the input dimensionality, while that of KMM and LogReg is rather dependent on the input dimensionality. Note that LogReg for $d \leq 3$ is slow due to some convergence problem of the LIBLINEAR package. Among them, the proposed uLSIF is one of the fastest methods. Figure 9(b) shows that the computation time of LogReg, KLIEP, and uLSIF is nearly independent of the number of training samples, while that of KDE and KMM sharply increase as the number of training samples increases.

Both LogReg and uLSIF have high accuracy and their computation time after model selection is comparable. Finally, we compare the entire computation time of LogReg and uLSIF including CV, which is summarized in Figure 10. We note that the Gaussian width σ and the regularization parameter λ are chosen over the 9×9 grid in this experiment for both LogReg and uLSIF. Therefore, the comparison of the entire computation time is fair. Figures 10(a) and 10(b) show that uLSIF is approximately 5 times faster than LogReg.

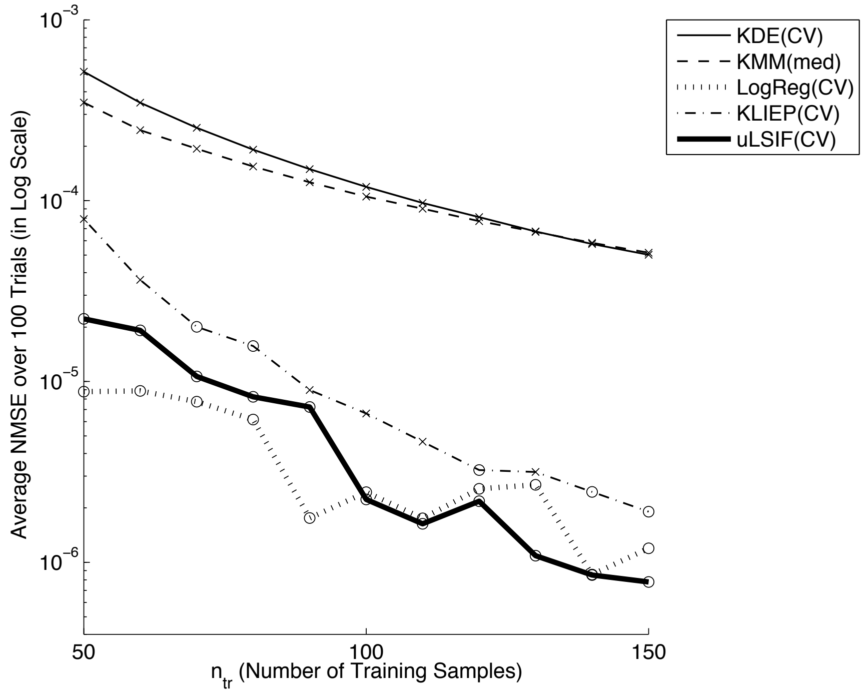
Overall, uLSIF is shown to be comparable to the best existing method (LogReg) in terms of the accuracy, but is computationally more efficient than LogReg.

6.2 Covariate Shift Adaptation in Regression and Classification

Next, we illustrate how the importance estimation methods could be used in *covariate shift adaptation* (Shimodaira, 2000; Zadrozny, 2004; Sugiyama and Müller, 2005; Huang et al., 2007; Bickel and Scheffer, 2007; Bickel et al., 2007; Sugiyama et al., 2007). Covariate shift is a situation in supervised learning where the input distributions change between the training and test phase but the conditional distribution of outputs given inputs remains unchanged. Under covariate shift, standard learning techniques such as maximum likelihood estimation or cross-validation are biased—the bias

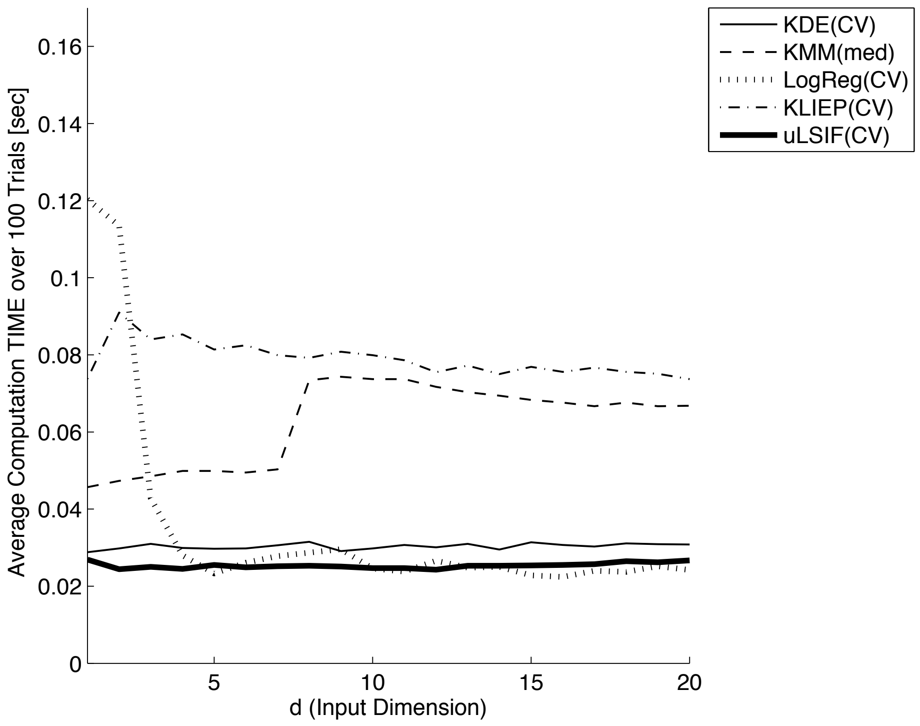


(a) When input dimensionality is changed

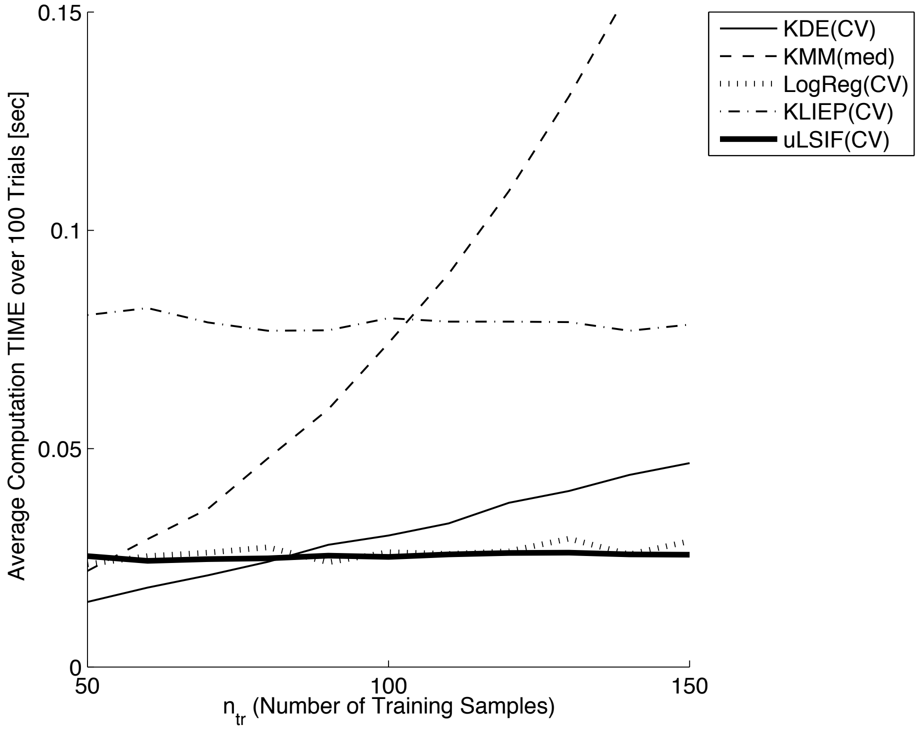


(b) When training sample size is changed

Figure 8: NMSEs averaged over 100 trials in log scale for the artificial data set. Error bars are omitted for clear visibility. Instead, the best method in terms of the mean error and comparable ones based on the *t-test* at the significance level 1% are indicated by '$\circ$'; the methods with significant difference from the best methods are indicated by '$\times$'.

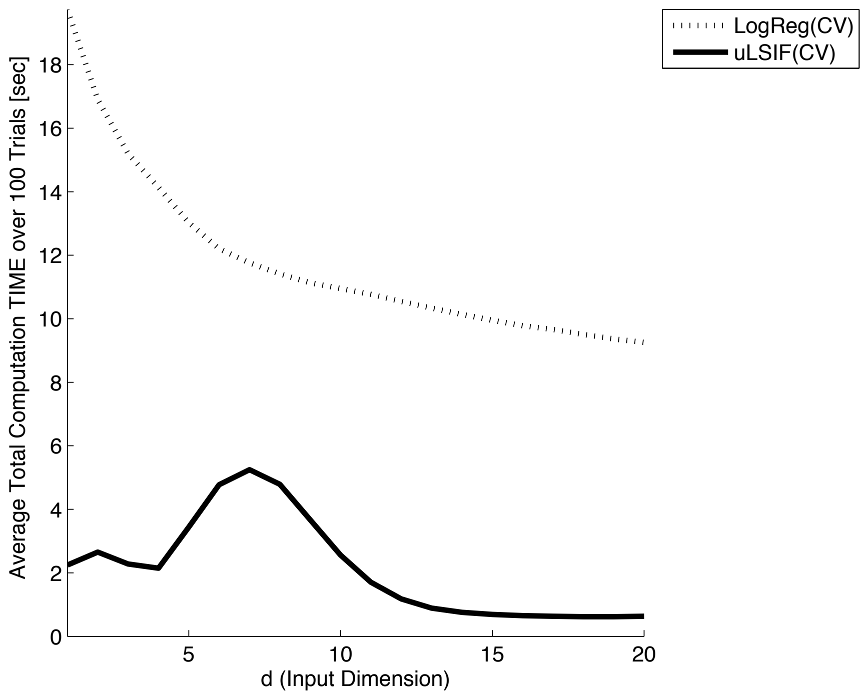


(a) When input dimensionality is changed

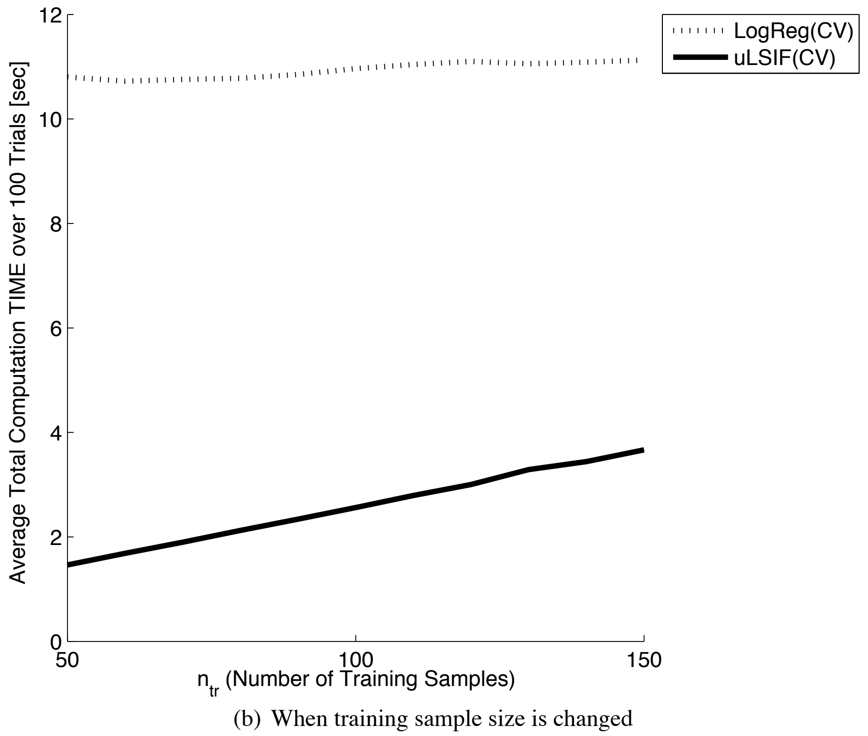


(b) When training sample size is changed

Figure 9: Average computation time (after model selection) over 100 trials for the artificial data set.



(a) When input dimensionality is changed



(b) When training sample size is changed

Figure 10: Average computation time over 100 trials for the artificial data set (including model selection of the Gaussian width σ and the regularization parameter λ over the 9×9 grid).

caused by covariate shift can be asymptotically canceled by weighting the loss function according to the importance.

In addition to training input samples $\{x_i^{\mathrm{tr}}\}_{i=1}^{n_{\mathrm{tr}}}$ drawn from a training input density $p_{\mathrm{tr}}(x)$ and test input samples $\{x_j^{\mathrm{te}}\}_{j=1}^{n_{\mathrm{te}}}$ drawn from a test input density $p_{\mathrm{te}}(x)$, suppose that we are given training *output* samples $\{y_i^{\mathrm{tr}}\}_{i=1}^{n_{\mathrm{tr}}}$ at the training input points $\{x_i^{\mathrm{tr}}\}_{i=1}^{n_{\mathrm{tr}}}$. The task is to predict the outputs for test inputs $\{x_j^{\mathrm{te}}\}_{j=1}^{n_{\mathrm{te}}}$ based on the input-output training samples $\{(x_i^{\mathrm{tr}}, y_i^{\mathrm{tr}})\}_{i=1}^{n_{\mathrm{tr}}}$.

We use the following kernel model for function learning:

$$\widehat{f}(x;\theta) = \sum_{\ell=1}^{t} \theta_\ell K_h(x, m_\ell),$$

where $K_h(x,x')$ is the Gaussian kernel (21) and m_ℓ is a template point randomly chosen from $\{x_j^{\mathrm{te}}\}_{j=1}^{n_{\mathrm{te}}}$ without replacement. We set the number of kernels at $t = 50$. We learn the parameter θ by *importance weighted regularized least-squares* (IWRLS) (Evgeniou et al., 2000; Sugiyama and Müller, 2005):

$$\widehat{\theta}_{\mathrm{IWRLS}} \equiv \operatorname*{argmin}_{\theta} \left[\sum_{i=1}^{n_{\mathrm{tr}}} \widehat{w}(x_i^{\mathrm{tr}}) \left(\widehat{f}(x_i^{\mathrm{tr}};\theta) - y_i^{\mathrm{tr}} \right)^2 + \gamma \|\theta\|^2 \right]. \tag{37}$$

It is known that IWRLS is consistent when the true importance $w(x_i^{\mathrm{tr}})$ is used as weights—unweighted RLS is not consistent due to covariate shift, given that the true learning target function $f(x)$ is not realizable by the model $\widehat{f}(x)$ (Shimodaira, 2000).

The solution $\widehat{\theta}_{\mathrm{IWRLS}}$ is analytically given by

$$\widehat{\theta}_{\mathrm{IWRLS}} = (K^\top \widehat{W} K + \gamma I_b)^{-1} K^\top \widehat{W} y^{\mathrm{tr}},$$

where

$$\begin{aligned} K_{i,\ell} &= K_h(x_i^{\mathrm{tr}}, m_\ell), \\ \widehat{W} &= \mathrm{diag}\left(\widehat{w}(x_1^{\mathrm{tr}}), \widehat{w}(x_2^{\mathrm{tr}}), \ldots, \widehat{w}(x_{n_{\mathrm{tr}}}^{\mathrm{tr}}) \right), \\ y^{\mathrm{tr}} &= (y_1^{\mathrm{tr}}, y_2^{\mathrm{tr}}, \ldots, y_{n_{\mathrm{tr}}}^{\mathrm{tr}})^\top. \end{aligned}$$

$\mathrm{diag}\,(a,b,\ldots,c)$ denotes the diagonal matrix with the diagonal elements $a,b,\ldots,c$.

The kernel width h and the regularization parameter γ in IWRLS (37) are chosen by *importance weighted CV* (IWCV) (Sugiyama et al., 2007). More specifically, we first divide the training samples $\{z_i^{\mathrm{tr}} \mid z_i^{\mathrm{tr}} = (x_i^{\mathrm{tr}}, y_i^{\mathrm{tr}})\}_{i=1}^{n_{\mathrm{tr}}}$ into R disjoint subsets $\{\mathcal{Z}_r^{\mathrm{tr}}\}_{r=1}^{R}$. Then a function $\widehat{f}_r(x)$ is learned using $\{\mathcal{Z}_j^{\mathrm{tr}}\}_{j \neq r}$ by IWRLS and its mean test error for the remaining samples $\mathcal{Z}_r^{\mathrm{tr}}$ is computed:

$$\frac{1}{|\mathcal{Z}_r^{\mathrm{tr}}|} \sum_{(x,y) \in \mathcal{Z}_r^{\mathrm{tr}}} \widehat{w}(x) \mathrm{loss}\left(\widehat{f}_r(x), y \right),$$

where

$$\mathrm{loss}\,(\widehat{y}, y) = \begin{cases} (\widehat{y} - y)^2 & \text{(Regression)}, \\ \frac{1}{2}(1 - \mathrm{sign}\{\widehat{y} y\}) & \text{(Classification)}. \end{cases}$$

We repeat this procedure for $r = 1, 2, \ldots, R$ and choose the kernel width h and the regularization parameter γ so that the average of the above mean test error over all r is minimized. We set the number of folds in IWCV at $R = 5$. IWCV is shown to be an (almost) unbiased estimator of the

Data	Uniform	KDE (CV)	KMM (med)	LogReg (CV)	KLIEP (CV)	uLSIF (CV)
kin-8fh	1.00(0.34)	1.22(0.52)	1.55(0.39)	1.31(0.39)	**0.95**(**0.31**)	**1.02**(**0.33**)
kin-8fm	1.00(0.39)	1.12(0.57)	1.84(0.58)	1.38(0.57)	**0.86**(**0.35**)	**0.88**(**0.39**)
kin-8nh	**1.00**(**0.26**)	1.09(0.20)	1.19(0.29)	1.09(0.19)	**0.99**(**0.22**)	**1.02**(**0.18**)
kin-8nm	**1.00**(**0.30**)	1.14(0.26)	1.20(0.20)	1.12(0.21)	**0.97**(**0.25**)	1.04(0.25)
abalone	**1.00**(**0.50**)	1.02(0.41)	**0.91**(**0.38**)	**0.97**(**0.49**)	**0.94**(**0.67**)	**0.96**(**0.61**)
image	**1.00**(**0.51**)	0.98(0.45)	1.08(0.54)	**0.98**(**0.46**)	**0.94**(**0.44**)	**0.98**(**0.47**)
ringnorm	1.00(0.04)	0.87(0.04)	**0.87**(**0.04**)	0.95(0.08)	0.99(0.06)	0.91(0.08)
twonorm	1.00(0.58)	1.16(0.71)	**0.94**(**0.57**)	**0.91**(**0.61**)	**0.91**(**0.52**)	**0.88**(**0.57**)
waveform	1.00(0.45)	1.05(0.47)	0.98(0.31)	**0.93**(**0.32**)	**0.93**(**0.34**)	**0.92**(**0.32**)
Average	1.00(0.38)	1.07(0.40)	1.17(0.37)	1.07(0.37)	0.94(0.35)	0.96(0.36)
Comp. time	—	0.82	3.50	3.27	2.23	1.00

Table 2: Mean test error averaged over 100 trials for covariate shift adaptation in regression and classification. The numbers in the brackets are the standard deviation. All the error values are normalized by that of 'Uniform' (uniform weighting, or equivalently no importance weighting). For each data set, the best method in terms of the mean error and comparable ones based on the *Wilcoxon signed rank test* at the significance level 1% are described in bold face. The upper half corresponds to regression data sets taken from DELVE (Rasmussen et al., 1996), while the lower half correspond to classification data sets taken from IDA (Rätsch et al., 2001). All the methods are implemented using the *MATLAB*® environment, where the *CPLEX*® optimizer is used for solving quadratic programs in KMM and the *LIBLINEAR* implementation is used for LogReg (Lin et al., 2007).

generalization error, while unweighted CV with misspecified models is biased due to covariate shift (Zadrozny, 2004; Sugiyama et al., 2007).

The data sets provided by DELVE (Rasmussen et al., 1996) and IDA (Rätsch et al., 2001) are used for performance evaluation. Each data set consists of input/output samples $\{(x_k, y_k)\}_{k=1}^{n}$. We normalize all the input samples $\{x_k\}_{k=1}^{n}$ into $[0,1]^d$ and choose the test samples $\{(x_j^{\mathrm{te}}, y_j^{\mathrm{te}})\}_{j=1}^{n_{\mathrm{te}}}$ from the pool $\{(x_k, y_k)\}_{k=1}^{n}$ as follows. We randomly choose one sample (x_k, y_k) from the pool and accept this with probability $\min(1, 4(x_k^{(c)})^2)$, where $x_k^{(c)}$ is the c-th element of x_k and c is randomly determined and fixed in each trial of the experiments. Then we remove x_k from the pool regardless of its rejection or acceptance, and repeat this procedure until n_{te} samples are accepted. We choose the training samples $\{(x_i^{\mathrm{tr}}, y_i^{\mathrm{tr}})\}_{i=1}^{n_{\mathrm{tr}}}$ uniformly from the rest. Thus, in this experiment, the test input density tends to be lower than the training input density when $x_k^{(c)}$ is small. We set the number of samples at $n_{\mathrm{tr}} = 100$ and $n_{\mathrm{te}} = 500$ for all data sets. Note that we only use $\{(x_i^{\mathrm{tr}}, y_i^{\mathrm{tr}})\}_{i=1}^{n_{\mathrm{tr}}}$ and $\{x_j^{\mathrm{te}}\}_{j=1}^{n_{\mathrm{te}}}$ for training regressors or classifiers; the test output values $\{y_j^{\mathrm{te}}\}_{j=1}^{n_{\mathrm{te}}}$ are used only for evaluating the generalization performance.

We run the experiments 100 times for each data set and evaluate the *mean test error*:

$$\frac{1}{n_{\mathrm{te}}} \sum_{j=1}^{n_{\mathrm{te}}} \mathrm{loss}\left(\widehat{f}(x_j^{\mathrm{te}}), y_j^{\mathrm{te}}\right).$$

The results are summarized in Table 2, where 'Uniform' denotes uniform weights (or equivalently, no importance weight). The numbers in the brackets are the standard deviation. All the error values are normalized so that the mean error of Uniform is one. For each data set, the best method in terms of the mean error and comparable ones based on the *Wilcoxon signed rank test* at the significance level 1% are described in bold face. The upper half of the table corresponds to regression data sets taken from DELVE (Rasmussen et al., 1996), while the lower half correspond to classification data sets taken from IDA (Rätsch et al., 2001). All the methods are implemented using the *MATLAB*® environment, where the *CPLEX*® optimizer is used for solving quadratic programs in KMM and the *LIBLINEAR* implementation is used for LogReg (Lin et al., 2007).

The table shows that the generalization performance of uLSIF tends to be better than that of Uniform, KDE, KMM, and LogReg, while it is comparable to the best existing method (KLIEP). The mean computation time over 100 trials is described in the bottom row of the table, where the value is normalized so that the computation time of uLSIF is one. This shows that the computation time of uLSIF is much shorter than KLIEP. Thus, uLSIF is overall shown to be useful in covariate shift adaptation.

6.3 Outlier Detection

Finally, we apply importance estimation methods in outlier detection.

Here, we consider an outlier detection problem of finding irregular samples in a data set ("evaluation data set") based on another data set ("model data set") that only contains regular samples. Defining the importance over two sets of samples, we can see that the importance values for regular samples are close to one, while those for outliers tend to be significantly deviated from one. Thus the importance values could be used as an index of the degree of outlyingness in this scenario. Since the evaluation data set has wider support than the model data set, we regard the evaluation data set as the training set $\{x_i^{\mathrm{tr}}\}_{i=1}^{n_{\mathrm{tr}}}$ (that is, the denominator in the importance) and the model data set as the test set $\{x_j^{\mathrm{te}}\}_{j=1}^{n_{\mathrm{te}}}$ (that is, the numerator in the importance). Then outliers tend to have smaller importance values (that is, close to zero).

We again test KMM(med), LogReg(CV), KLIEP(CV), and uLSIF(CV) for importance estimation; in addition, we include native outlier detection methods for comparison purposes. The outlier detection problem that the native methods used below solve is to find outliers in a single data set $\{x_k\}_{k=1}^{n}$—the native methods can be employed in the current scenario just by finding outliers from all samples:

$$\{x_k\}_{k=1}^{n} = \{x_i^{\mathrm{tr}}\}_{i=1}^{n_{\mathrm{tr}}} \cup \{x_j^{\mathrm{te}}\}_{j=1}^{n_{\mathrm{te}}}.$$

One-class support vector machine (OSVM): The *support vector machine* (SVM) (Vapnik, 1998; Schölkopf and Smola, 2002) is one of the most successful classification algorithms in machine learning. The core idea of SVM is to separate samples in different classes by the maximum margin hyperplane in a kernel-induced feature space.

OSVM is an extension of SVM to outlier detection (Schölkopf et al., 2001). The basic idea of OSVM is to separate data samples $\{x_k\}_{k=1}^{n}$ into outliers and inliers by a hyperplane in a Gaussian reproducing kernel Hilbert space. More specifically, the solution of OSVM is given as the solution of the following convex quadratic programming problem:

$$\min_{\{w_k\}_{k=1}^{n}} \frac{1}{2}\sum_{k,k'=1}^{n} w_k w_{k'} K_\sigma(x_k, x_{k'})$$
$$\text{subject to} \sum_{k=1}^{n} w_k = 1 \text{ and } 0 \le w_1, w_2, \ldots, w_n \le \frac{1}{\nu n},$$

where ν ($0 \le \nu \le 1$) is the maximum fraction of outliers.

We use the inverse distance of a sample from the separating hyperplane as an outlier score. The OSVM solution is dependent on the outlier ratio ν and the Gaussian kernel width σ, and there seems to be no systematic method to determine the values of these tuning parameters. Here we use the median distance between samples as the Gaussian width, which is a popular heuristic (Schölkopf and Smola, 2002; Song et al., 2007). The value of ν is fixed at the true output ratio, that is, the ideal optimal value. Thus the simulation results below should be slightly in favor of OSVM.

Local outlier factor (LOF): LOF is the score to detect a local outlier which lies relatively far from the nearest dense region (Breunig et al., 2000). For a prefixed natural number k, the LOF value of a sample x is defined by

$$\mathrm{LOF}_R(x) = \frac{1}{k}\sum_{i=1}^{k} \frac{\mathrm{imd}_k(\mathrm{nearest}_i(x))}{\mathrm{imd}_k(x)},$$

where $\mathrm{nearest}_i(x)$ denotes the i-th nearest neighbor of x and $\mathrm{imd}_k(x)$ denotes the inverse mean distance from x to its k nearest neighbors:

$$\mathrm{imd}_k(x) = \frac{1}{\frac{1}{k}\sum_{i=1}^{k} \|x - \mathrm{nearest}_i(x)\|}.$$

If x alone is apart from a cloud of points, $\mathrm{imd}_k(x)$ tends to become smaller than than $\mathrm{imd}_k(\mathrm{nearest}_i(x))$ for all i. Then the LOF value gets large and therefore such a point is regarded as an outlier. The performance of LOF depends on the choice of the parameter k and there seems no systematic way to find an appropriate value of k. Here we test several different values of k.

Kernel density estimator (KDE'): A naive density estimation of all data samples $\{x_k\}_{k=1}^{n}$ can also be used for outlier detection since the density value itself could be regarded as an outlier score. We use KDE with the Gaussian kernel (21) for density estimation, where the kernel width is determined based on 5-fold LCV.

All the methods are implemented using the R environment—we use the *ksvm* routine in the *kernlab* package for OSVM (Karatzoglou et al., 2004) and the *lofactor* routine in the *dprep* package for LOF (Fernandez, 2005).

The data sets provided by IDA (Rätsch et al., 2001) are used for performance evaluation. These data sets are binary classification data sets consisting of positive/negative and training/test samples. We allocate all positive training samples for the "model" set, while all positive test samples and a fraction ρ $(= 0.01, 0.02, 0.05)$ of negative test samples are assigned in the "evaluation" set. Thus, we regard the positive samples as regular and the negative samples as irregular.

In the evaluation of the performance of outlier detection methods, it is important to take into account both the detection rate (the amount of true outliers an outlier detection algorithm can find) and the detection accuracy (the amount of true inliers that an outlier detection algorithm misjudges as outliers). Since there is a trade-off between the detection rate and the detection accuracy, we adopt the area under the ROC curve (AUC) as our error metric (Bradley, 1997).

The mean AUC values over 20 trials as well as the computation time are summarized in Table 3, showing that uLSIF works fairly well. KLIEP works slightly better than uLSIF, but uLSIF is computationally much more efficient. LogReg overall works reasonably well, but it performs poorly for some data sets and the average AUC performance is not as good as uLSIF or KLIEP. KMM and OSVM are not comparable to uLSIF in AUC and they are computationally inefficient. Note that we also tested KMM and OSVM with several different Gaussian widths and experimentally found that the heuristic of using the median sample distance as the Gaussian kernel width works reasonably well in this experiment. Thus the AUC values of KMM and OSVM are close to optimal. LOF with large k is shown to work well, although it is not clear whether the heuristic of simply using large k is always appropriate or not. The computational cost of LOF is high since nearest neighbor search is computationally expensive. KDE' works reasonably well, but its performance is not as good as uLSIF and KLIEP.

Overall, uLSIF is shown to work well with low computational costs.

7. Conclusions

The importance is useful in various machine learning scenarios such as covariate shift adaptation and outlier detection. In this paper, we proposed a new method of importance estimation that can avoid solving a substantially more difficult task of density estimation. We formulated the importance estimation problem as least-squares function fitting and casted the optimization problem as a convex quadratic program (we referred to it as LSIF). We theoretically elucidated the convergence property of LSIF and showed that the entire regularization path of LSIF can be efficiently computed based on a parametric optimization technique. We further developed an approximation algorithm (we called it uLSIF), which allows us to obtain the closed-form solution. We showed that the leave-one-out cross-validation score can be computed analytically for uLSIF—this makes the computation of uLSIF highly efficient. We carried out extensive simulations in covariate shift adaptation and outlier detection, and experimentally confirmed that the proposed uLSIF is computationally more efficient than existing approaches, while the accuracy of uLSIF is comparable to the best existing methods. Thanks to the low computational cost, uLSIF would be highly scalability to large data sets, which is very important in practical applications.

We have given convergence proofs for LSIF and uLSIF. A possible future direction to pursue along this line is to show the convergence of LSIF and uLSIF in non-parametric cases, for example, following Nguyen et al. (2008) and Sugiyama et al. (2008b). We are currently exploring various possible applications of important estimation methods beyond covariate shift adaptation or outlier detection, for example, feature selection, conditional distribution estimation, independent component analysis, and dimensionality reduction—we believe that importance estimation could be used as a new versatile tool in statistical machine learning.

Data		uLSIF	KLIEP	LogReg	KMM	OSVM	LOF			KDE'
Name	ρ	(CV)	(CV)	(CV)	(med)	(med)	$k=5$	$k=30$	$k=50$	(CV)
banana	0.01	0.851	0.815	0.447	0.578	0.360	0.838	0.915	0.919	0.934
	0.02	0.858	0.824	0.428	0.644	0.412	0.813	0.918	0.920	0.927
	0.05	0.869	0.851	0.435	0.761	0.467	0.786	0.907	0.909	0.923
b-cancer	0.01	0.463	0.480	0.627	0.576	0.508	0.546	0.488	0.463	0.400
	0.02	0.463	0.480	0.627	0.576	0.506	0.521	0.445	0.428	0.400
	0.05	0.463	0.480	0.627	0.576	0.498	0.549	0.480	0.452	0.400
diabetes	0.01	0.558	0.615	0.599	0.574	0.563	0.513	0.403	0.390	0.425
	0.02	0.558	0.615	0.599	0.574	0.563	0.526	0.453	0.434	0.425
	0.05	0.532	0.590	0.636	0.547	0.545	0.536	0.461	0.447	0.435
f-solar	0.01	0.416	0.485	0.438	0.494	0.522	0.480	0.441	0.385	0.378
	0.02	0.426	0.456	0.432	0.480	0.550	0.442	0.406	0.343	0.374
	0.05	0.442	0.479	0.432	0.532	0.576	0.455	0.417	0.370	0.346
german	0.01	0.574	0.572	0.556	0.529	0.535	0.526	0.559	0.552	0.561
	0.02	0.574	0.572	0.556	0.529	0.535	0.553	0.549	0.544	0.561
	0.05	0.564	0.555	0.540	0.532	0.530	0.548	0.571	0.555	0.547
heart	0.01	0.659	0.647	0.833	0.623	0.681	0.407	0.659	0.739	0.638
	0.02	0.659	0.647	0.833	0.623	0.678	0.428	0.668	0.746	0.638
	0.05	0.659	0.647	0.833	0.623	0.681	0.440	0.666	0.749	0.638
satimage	0.01	0.812	0.828	0.600	0.813	0.540	0.909	0.930	0.896	0.916
	0.02	0.829	0.847	0.632	0.861	0.548	0.785	0.919	0.880	0.898
	0.05	0.841	0.858	0.715	0.893	0.536	0.712	0.895	0.868	0.892
splice	0.01	0.713	0.748	0.368	0.541	0.737	0.765	0.778	0.768	0.845
	0.02	0.754	0.765	0.343	0.588	0.744	0.761	0.793	0.783	0.848
	0.05	0.734	0.764	0.377	0.643	0.723	0.764	0.785	0.777	0.849
thyroid	0.01	0.534	0.720	0.745	0.681	0.504	0.259	0.111	0.071	0.256
	0.02	0.534	0.720	0.745	0.681	0.505	0.259	0.111	0.071	0.256
	0.05	0.534	0.720	0.745	0.681	0.485	0.259	0.111	0.071	0.256
titanic	0.01	0.525	0.534	0.602	0.502	0.456	0.520	0.525	0.525	0.461
	0.02	0.496	0.498	0.659	0.513	0.526	0.492	0.503	0.503	0.472
	0.05	0.526	0.521	0.644	0.538	0.505	0.499	0.512	0.512	0.433
twonorm	0.01	0.905	0.902	0.161	0.439	0.846	0.812	0.889	0.897	0.875
	0.02	0.896	0.889	0.197	0.572	0.821	0.803	0.892	0.901	0.858
	0.05	0.905	0.903	0.396	0.754	0.781	0.765	0.858	0.874	0.807
waveform	0.01	0.890	0.881	0.243	0.477	0.861	0.724	0.887	0.889	0.861
	0.02	0.901	0.890	0.181	0.602	0.817	0.690	0.887	0.890	0.861
	0.05	0.885	0.873	0.236	0.757	0.798	0.705	0.847	0.874	0.831
Average		0.661	0.685	0.530	0.608	0.596	0.594	0.629	0.622	0.623
Comp. time		1.00	11.7	5.35	751	12.4	85.5			8.70

Table 3: Mean AUC values for outlier detection over 20 trials for the benchmark data sets. All the methods are implemented using the R environment, where quadratic programs in KMM are solved by the *ipop* optimizer (Karatzoglou et al., 2004), the *ksvm* routine is used for OSVM (Karatzoglou et al., 2004), and the *lofactor* routine is used for LOF (Fernandez, 2005).

Acknowledgments

The authors wish to thank Issei Sato for fruitful discussion and helpful comments. The authors would also like to thank the anonymous referees whose comments helped to improve the paper further. This work was supported by MEXT (20680007), SCAT, and AOARD.

Appendix A. Existence of the Inverse Matrix of $\widehat{G}$

Here we prove Lemma 1.

Let us consider the following system of linear equations:

$$\begin{pmatrix} \widehat{H} & -\widehat{E}^\top \\ -\widehat{E} & O_{|\widehat{\mathcal{A}}|\times|\widehat{\mathcal{A}}|} \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 0_b \\ 0_{|\widehat{\mathcal{A}}|} \end{pmatrix}, \tag{38}$$

where x and y are b- and $|\widehat{\mathcal{A}}|$-dimensional vectors, respectively. From the upper half of Eq. (38), we have

$$x = \widehat{H}^{-1}\widehat{E}^\top y.$$

Substituting this into the lower half of Eq. (38), we have

$$\widehat{E}\widehat{H}^{-1}\widehat{E}^\top y = 0_{|\widehat{\mathcal{A}}|}.$$

From the definition, the rank of the matrix $\widehat{E}$ is $|\widehat{\mathcal{A}}|$, that is, $\widehat{E}$ is a row-full rank matrix. As a result, the matrix $\widehat{E}\widehat{H}^{-1}\widehat{E}^\top$ is invertible. Therefore, Eq. (38) has the unique solution $x = 0_b$ and $y = 0_{|\widehat{\mathcal{A}}|}$. This implies that $\widehat{G}$ is invertible.

Appendix B. Active Set of LSIF

Here, we prove Theorem 2.

We prove that the active set $\mathcal{A}$ does not change under the infinitesimal shift of H and h if the strict complementarity condition is satisfied. We regard the pair of a symmetric matrix and a vector (H', h') as an element in the $(\frac{b(b+1)}{2} + b)$-dimensional Euclidean space. We consider the following linear equation:

$$\begin{pmatrix} \alpha' \\ \xi' \end{pmatrix} = \begin{pmatrix} H' & -E^\top \\ -E & O_{|\mathcal{A}|\times|\mathcal{A}|} \end{pmatrix}^{-1} \begin{pmatrix} h' - \lambda 1_b \\ 0_{|\mathcal{A}|} \end{pmatrix},$$

where E is the $|\mathcal{A}| \times b$ indicator matrix determined from the active set $\mathcal{A}$ (see Section 2.3 for the detailed definition). If $H' = H$ and $h' = h$ hold, the solution $(\alpha', \xi') = (\alpha^*(\lambda), \xi^*(\lambda))$ satisfies

$$\begin{aligned} \alpha'_\ell = 0,\ \xi'_\ell > 0,\ \ {}^\forall \ell \in \mathcal{A}, \\ \alpha'_\ell > 0,\ \xi'_\ell = 0,\ \ {}^\forall \ell \notin \mathcal{A}, \end{aligned} \tag{39}$$

because of the strict complementarity condition. On the other hand, if the norm of $(H', h') - (H, h)$ is infinitesimal, the solution (α', ξ') also satisfies Eq. (39) because of the continuity of the relation between (H', h') and (α', ξ').

As a result, there exists an ε-ball B_ε in $\mathbb{R}^{\frac{b(b+1)}{2}+b}$ such that the equality $\mathcal{A} = \{\ell \mid \alpha'_\ell = 0\}$ holds for any $(H', h') \in B_\varepsilon$. Therefore, we have $P(\mathcal{A} \neq \widehat{\mathcal{A}}) \leq P((\widehat{H}, \widehat{h}) \notin B_\varepsilon)$. Due to the large deviation principle (Dembo and Zeitouni, 1998), there is a positive constant c such that

$$-\frac{1}{\min\{n_{\mathrm{tr}}, n_{\mathrm{te}}\}} \log P((\widehat{H}, \widehat{h}) \notin B_\varepsilon) > c > 0,$$

if $\min\{n_{\mathrm{tr}}, n_{\mathrm{te}}\}$ is large enough. Thus, asymptotically $P(\widehat{\mathcal{A}} \neq \mathcal{A}) < e^{-c\min\{n_{\mathrm{tr}}, n_{\mathrm{te}}\}}$ holds.

Appendix C. Learning Curve of LSIF

Here, we prove Theorem 3.

Let us consider the ideal problem (7). Let $\alpha^*(\lambda)$ and $\xi^*(\lambda)$ be the optimal parameter and Lagrange multiplier (that is, the KKT conditions are fulfilled; see Section 2.3) and let $\xi^{*\prime}(\lambda)$ be the vector of non-zero elements of $\xi^*(\lambda)$ defined in the same way as Eq. (11). Then $\alpha^*(\lambda)$ and $\xi^{*\prime}(\lambda)$ satisfy

$$G \begin{pmatrix} \alpha^*(\lambda) \\ \xi^{*\prime}(\lambda) \end{pmatrix} = \begin{pmatrix} h - \lambda 1_b \\ 0_{|\mathcal{A}|} \end{pmatrix}, \tag{40}$$

where

$$G = \begin{pmatrix} H & -E^\top \\ -E & O_{|\mathcal{A}|\times|\mathcal{A}|} \end{pmatrix}.$$

From the central limit theorem and the assumption (18), we have

$$\widehat{h} = h + O_p\left(\frac{1}{\sqrt{n_{\mathrm{te}}}}\right) = h + o_p\left(\frac{1}{n_{\mathrm{tr}}}\right), \tag{41}$$

where O_p and o_p denote the asymptotic order in probability. The assumption (a) implies that the equality

$$\widehat{E} = E \tag{42}$$

holds with exponentially high probability due to Theorem 2. Note that $\widehat{G}$ is the same size as G if $\widehat{E} = E$. Thus we have

$$\widehat{G} = G + \delta G,$$

where

$$\delta G = \begin{pmatrix} \delta H & O_{b\times|\mathcal{A}|} \\ O_{|\mathcal{A}|\times b} & O_{|\mathcal{A}|\times|\mathcal{A}|} \end{pmatrix},$$

$$\delta H = \widehat{H} - H. \tag{43}$$

Combining Eqs. (12), (40), (41), and (42), we have

$$\begin{pmatrix} \widehat{\alpha}(\lambda) \\ \widehat{\xi}'(\lambda) \end{pmatrix} = \widehat{G}^{-1} G \begin{pmatrix} \alpha^*(\lambda) \\ \xi^{*\prime}(\lambda) \end{pmatrix} + o_p\left(\frac{1}{n_{\mathrm{tr}}}\right). \tag{44}$$

The matrix Taylor expansion (Petersen and Pedersen, 2007) yields

$$\widehat{G}^{-1} = G^{-1} - G^{-1}\delta G G^{-1} + G^{-1}\delta G G^{-1}\delta G G^{-1} - \cdots, \tag{45}$$

and the central limit theorem asserts that

$$\delta H = O_p\left(\frac{1}{\sqrt{n_{\mathrm{tr}}}}\right). \tag{46}$$

Combining Eqs. (44), (45), (14), and (46), we have

$$\begin{aligned}\delta\alpha &= \widehat{\alpha}(\lambda) - \alpha^*(\lambda) &(47)\\ &= -A\delta H\alpha^*(\lambda) + A\delta HA\delta H\alpha^*(\lambda) + o\left(\frac{1}{n_{\mathrm{tr}}}\right). &(48)\end{aligned}$$

Through direct calculation, we can confirm that

$$AHA = A. \tag{49}$$

Similar to Eq. (15), it holds that

$$\alpha^*(\lambda) = A(h - \lambda 1_b). \tag{50}$$

From Eqs. (49) and (50), we have

$$A(H\alpha^*(\lambda) - h) = -\lambda A 1_b. \tag{51}$$

Eqs. (43), (4), and (3) imply

$$\mathbb{E}[\delta H] = O_{b\times b}. \tag{52}$$

From Eqs. (2) and (47), we have

$$J(\widehat{\alpha}(\lambda)) = J(\alpha^*(\lambda)) + \frac{1}{2}\delta\alpha^\top H\delta\alpha + (H\alpha^*(\lambda) - h)^\top\delta\alpha. \tag{53}$$

From Eqs. (46), (48), and (49), we have

$$\begin{aligned}\mathbb{E}\left[\delta\alpha^\top H\delta\alpha\right] &= \mathrm{tr}(H\,\mathbb{E}\left[\delta\alpha\delta\alpha^\top\right])\\ &= \mathrm{tr}(AHA\,\mathbb{E}\left[(\delta H\alpha^*(\lambda))(\delta H\alpha^*(\lambda))^\top\right]) + o\left(\frac{1}{n_{\mathrm{tr}}}\right)\\ &= \mathrm{tr}(A\,\mathbb{E}\left[(\delta H\alpha^*(\lambda))(\delta H\alpha^*(\lambda))^\top\right]) + o\left(\frac{1}{n_{\mathrm{tr}}}\right). &(54)\end{aligned}$$

From Eqs. (48), (51), and (52), we have

$$\begin{aligned}\mathbb{E}\left[\delta\alpha^\top(H\alpha^*(\lambda) - h)\right] &= -\,\mathbb{E}\left[(\delta H\alpha^*(\lambda) - \delta HA\delta H\alpha^*(\lambda))^\top A(H\alpha^*(\lambda) - h)\right] + o\left(\frac{1}{n_{\mathrm{tr}}}\right)\\ &= \mathbb{E}\left[(\delta H\alpha^*(\lambda) - \delta HA\delta H\alpha^*(\lambda))^\top\lambda A 1_b\right] + o\left(\frac{1}{n_{\mathrm{tr}}}\right)\\ &= -\,\lambda\mathrm{tr}(A\,\mathbb{E}\left[(\delta H\alpha^*(\lambda))(\delta HA1_b)^\top\right]) + o\left(\frac{1}{n_{\mathrm{tr}}}\right). &(55)\end{aligned}$$

Combining Eqs. (53), (54), and (55), we have

$$\begin{aligned}\mathbb{E}\left[J(\widehat{\alpha}(\lambda))\right] =& J(\alpha^*(\lambda)) + \frac{1}{2n_{\mathrm{tr}}}\mathrm{tr}(A\,\mathbb{E}\left[(\sqrt{n_{\mathrm{tr}}}\delta H\alpha^*(\lambda))(\sqrt{n_{\mathrm{tr}}}\delta H\alpha^*(\lambda))^\top\right]) \\ &- \frac{\lambda}{n_{\mathrm{tr}}}\mathrm{tr}(A\,\mathbb{E}\left[(\sqrt{n_{\mathrm{tr}}}\delta H\alpha^*(\lambda))(\sqrt{n_{\mathrm{tr}}}\delta HA1_b)^\top\right]) + o\left(\frac{1}{n_{\mathrm{tr}}}\right).\end{aligned}$$

According to the central limit theorem, $\sqrt{n_{\mathrm{tr}}}\delta H_{i,j}$ asymptotically follows the normal distribution with mean zero and variance

$$\int \varphi_i^2(x)\varphi_j^2(x)p_{\mathrm{tr}}(x)dx - H_{i,j}^2,$$

and the asymptotic covariance between $\sqrt{n_{\mathrm{tr}}}\delta H_{i,j}$ and $\sqrt{n_{\mathrm{tr}}}\delta H_{i',j'}$ is given by

$$\int \varphi_i(x)\varphi_j(x)\varphi_{i'}(x)\varphi_{j'}(x)p_{\mathrm{tr}}(x)dx - H_{i,j}H_{i',j'}.$$

Then we have

$$\begin{aligned}\lim_{n_{\mathrm{tr}}\to\infty}\mathbb{E}\left[(\sqrt{n_{\mathrm{tr}}}\delta H\alpha^*(\lambda))(\sqrt{n_{\mathrm{tr}}}\delta H\alpha^*(\lambda))^\top\right] &= C_{w^*,w^*},\\ \lim_{n_{\mathrm{tr}}\to\infty}\mathbb{E}\left[(\sqrt{n_{\mathrm{tr}}}\delta H\alpha^*(\lambda))(\sqrt{n_{\mathrm{tr}}}\delta HA1_b)^\top\right] &= C_{w^*,v},\end{aligned}$$

where $C_{w,w'}$ is the $b\times b$ covariance matrix with the (ℓ,ℓ')-th element being the covariance between $w(x)\varphi_\ell(x)$ and $w'(x)\varphi_{\ell'}(x)$ under $p_{\mathrm{tr}}(x)$. Then we obtain Eq. (19).

Appendix D. Regularization Path of LSIF

Here, we derive the regularization path tracking algorithm given in Figure 1.

When λ is greater than or equal to $\max_k \widehat{h}_k$, the solution of the KKT conditions (9)–(10) is provided as $\alpha = 0_b$, $\xi = \lambda 1_b - \widehat{h} \geq 0_b$. Therefore, the initial value of λ_0 is $\max_k \widehat{h}_k$, and the corresponding optimal solution is $\widehat{\alpha}(\lambda_0) = 0_b$.

Since $\widehat{\xi}'(\lambda)$ corresponds to non-zero elements of $\widehat{\xi}(\lambda)$ as shown in Eq. (11), we have

$$\widehat{\xi}_j(\lambda) = \begin{cases}\widehat{\xi}_i'(\lambda) & \text{if } j = \widehat{j}_i,\\ 0 & \text{otherwise.}\end{cases} \tag{56}$$

When λ is decreased, the solutions $\widehat{\alpha}(\lambda)$ and $\widehat{\xi}(\lambda)$ still satisfy Eqs. (12) and (56) as long as the active set $\widehat{\mathcal{A}}$ remains unchanged. Change points of the active set can be found by examining the non-negativity conditions of $\widehat{\alpha}(\lambda)$ and $\widehat{\xi}(\lambda)$ as follows. Suppose λ is decreased and the non-negativity condition

$$\begin{pmatrix}\widehat{\alpha}(\lambda)\\ \widehat{\xi}(\lambda)\end{pmatrix} \geq 0_{2b}$$

is violated at $\lambda = \lambda'$. That is, both $\widehat{\alpha}(\lambda') \geq 0_b$ and $\widehat{\xi}(\lambda') \geq 0_b$ hold, and either $\widehat{\alpha}(\lambda' - \varepsilon) \geq 0_b$ or $\widehat{\xi}(\lambda' - \varepsilon) \geq 0_b$ is violated for any $\varepsilon > 0$. If $\widehat{\alpha}_j(\lambda') = 0$ for $j \notin \widehat{\mathcal{A}}$, j should be added to the active set

$\widehat{\mathcal{A}}$; on the other hand, if $\widehat{\xi}_j(\lambda') = 0$ for some $j \in \widehat{\mathcal{A}}$, $\widehat{\alpha}_j(\lambda')$ will take a positive value and therefore j should be removed from the active set $\widehat{\mathcal{A}}$. Then, for the updated active set, we compute the solutions by Eqs. (12) and (56). Iterating this procedure until λ reaches zero, we can obtain the entire regularization path.

Note that we omitted some minor exceptional cases for the sake of simplicity—treatments for all possible exceptions and the rigorous convergence property are exhaustively studied in Best (1982).

Appendix E. Negative Index Set of $\beta^\circ(\lambda)$

Here we prove Theorem 4.

As explained in Appendix B, we regard the pair of a symmetric matrix and a vector (H', h') as an element in the $(\frac{b(b+1)}{2} + b)$-dimensional Euclidean space.

We consider the linear equation

$$\beta' = (H' + \lambda I_b)^{-1} h'.$$

Due to the assumption, for $H' = H$ and $h' = h$, we have

$$\beta'_\ell \neq 0, \ \ell = 1, 2, \ldots, b. \tag{57}$$

On the other hand, if the norm of $(H', h') - (H, h)$ is infinitesimal, the solution β' also satisfies Eq. (57), and the sign of β'_ℓ is same as that of β_ℓ for $\ell = 1, 2, \ldots, b$, because of the continuity of the relation between (H', h') and β'.

As a result, there exists an ε-ball B_ε in $\mathbb{R}^{\frac{b(b+1)}{2}+b}$ such that the equality $\mathcal{B} = \widetilde{\mathcal{B}}$ holds for any $(H', h') \in B_\varepsilon$. Therefore, we have $P(\mathcal{B} \neq \widetilde{\mathcal{B}}) \leq P((\widehat{H}, \widehat{h}) \notin B_\varepsilon)$. Due to the large deviation principle (Dembo and Zeitouni, 1998), there is a positive constant c such that

$$-\frac{1}{\min\{n_{\mathrm{tr}}, n_{\mathrm{te}}\}} \log P((\widehat{H}, \widehat{h}) \notin B_\varepsilon) > c > 0,$$

if $\min\{n_{\mathrm{tr}}, n_{\mathrm{te}}\}$ is large enough. Thus, asymptotically $P(\mathcal{B} \neq \widetilde{\mathcal{B}}) < e^{-c \min\{n_{\mathrm{tr}}, n_{\mathrm{te}}\}}$ holds.

Appendix F. Learning Curve of uLSIF

Here, we prove Theorem 5.

Let

$$\widehat{B}_\lambda = \widehat{H} + \lambda I_b.$$

The matrix Taylor expansion (Petersen and Pedersen, 2007) yields

$$\widehat{B}_\lambda^{-1} = B_\lambda^{-1} - B_\lambda^{-1} \delta H B_\lambda^{-1} + B_\lambda^{-1} \delta H B_\lambda^{-1} \delta H B_\lambda^{-1} - \cdots. \tag{58}$$

Let $\widetilde{\mathcal{B}} \subset \{1, 2, \ldots, b\}$ be the set of negative indices of $\widetilde{\beta}(\lambda)$, that is,

$$\widetilde{\mathcal{B}} = \{\ell \mid \widetilde{\beta}_\ell(\lambda) < 0, \ \ell = 1, 2, \ldots, b\}.$$

Let $\widehat{D}$ be the b-dimensional diagonal matrix with the ℓ-th diagonal element

$$\widehat{D}_{\ell,\ell} = \begin{cases} 0 & \ell \in \widetilde{\mathcal{B}}, \\ 1 & \text{otherwise.} \end{cases}$$

The assumption (a) implies that the equality

$$\widehat{D} = D \tag{59}$$

holds with exponentially high probability due to Theorem 4. Combining Eqs. (59), (41), (58), and (24), we have

$$\begin{aligned}\delta\beta &= \widehat{\beta}(\lambda) - \beta^*(\lambda)\\ &= \widehat{D}\widehat{B}_\lambda^{-1}\widehat{h} - DB_\lambda^{-1}h\\ &= -DB_\lambda^{-1}\delta H\beta^\circ(\lambda) + DB_\lambda^{-1}\delta HB_\lambda^{-1}\delta H\beta^\circ(\lambda) + o\left(\frac{1}{n_{\mathrm{tr}}}\right).\end{aligned} \tag{60}$$

From Eqs. (46) and (60), we have

$$\mathbb{E}\left[\delta\beta^\top H\delta\beta\right] = \mathrm{tr}(B_\lambda^{-1}DHDB_\lambda^{-1}\,\mathbb{E}\left[(\delta H\beta^\circ(\lambda))(\delta H\beta^\circ(\lambda))^\top\right]) + o\left(\frac{1}{n_{\mathrm{tr}}}\right). \tag{61}$$

From Eqs. (52) and (24), we have

$$\begin{aligned}\mathbb{E}\left[\delta\beta^\top(H\beta^*(\lambda)-h)\right] =&\mathbb{E}\left[(-\delta H\beta^\circ(\lambda) + \delta HB_\lambda^{-1}\delta H\beta^\circ(\lambda))^\top B_\lambda^{-1}D(H\beta^*(\lambda)-h)\right]\\ &+ o\left(\frac{1}{n_{\mathrm{tr}}}\right)\\ =&\mathbb{E}\left[\mathrm{tr}(B_\lambda^{-1}(\delta H\beta^\circ(\lambda))(\delta HB_\lambda^{-1}D(H\beta^*(\lambda)-h))^\top)\right]\\ &+ o\left(\frac{1}{n_{\mathrm{tr}}}\right).\end{aligned} \tag{62}$$

Combining Eqs. (53), (61), and (62), we have

$$\begin{aligned}\mathbb{E}\left[J(\widehat{\beta}(\lambda))\right] =&J(\beta^*(\lambda)) + \frac{1}{2n_{\mathrm{tr}}}\mathrm{tr}(B_\lambda^{-1}DHDB_\lambda^{-1}\,\mathbb{E}[(\sqrt{n_{\mathrm{tr}}}\delta H\beta^\circ(\lambda))(\sqrt{n_{\mathrm{tr}}}\delta H\beta^\circ(\lambda))^\top)\\ &+ \frac{1}{n_{\mathrm{tr}}}\mathrm{tr}(B_\lambda^{-1}\,\mathbb{E}[(\sqrt{n_{\mathrm{tr}}}\delta H\beta^\circ(\lambda))(\sqrt{n_{\mathrm{tr}}}\delta HB_\lambda^{-1}D(H\beta^*(\lambda)-h))^\top) + o\left(\frac{1}{n_{\mathrm{tr}}}\right).\end{aligned}$$

According to the central limit theorem, we have

$$\begin{aligned}\lim_{n_{\mathrm{tr}}\to\infty}\mathbb{E}[(\sqrt{n_{\mathrm{tr}}}\delta H\beta^\circ(\lambda))(\sqrt{n_{\mathrm{tr}}}\delta H\beta^\circ(\lambda))^\top] &= C_{w^\circ,w^\circ},\\ \lim_{n_{\mathrm{tr}}\to\infty}\mathbb{E}[(\sqrt{n_{\mathrm{tr}}}\delta H\beta^\circ(\lambda))(\sqrt{n_{\mathrm{tr}}}\delta HB_\lambda^{-1}D(H\beta^*(\lambda)-h))^\top] &= C_{w^\circ,u}.\end{aligned}$$

Then we obtain Eq. (25).

Appendix G. 'Norm' Upper Bound of Approximation Error for uLSIF

Here we prove Theorem 6.

Using the weighted norm (27), we can express diff(λ) as

$$\mathrm{diff}(\lambda) = \frac{\inf_{\lambda'\geq 0}\|\widehat{\alpha}(\lambda') - \widehat{\beta}(\lambda)\|_{\widehat{H}}}{\sum_{i=1}^{n_{\mathrm{tr}}}\widehat{w}(x_i^{\mathrm{tr}};\widehat{\beta}(\lambda))}.$$

As shown in Appendix D, $\widehat{\alpha}(\lambda') = 0_b$ holds for some large λ'. Then we immediately have

$$\mathrm{diff}(\lambda) \le \frac{\|\widehat{\beta}(\lambda)\|_{\widehat{H}}}{\sum_{i=1}^{n_{\mathrm{tr}}} \widehat{w}(x_i^{\mathrm{tr}}; \widehat{\beta}(\lambda))},$$

which proves Eq. (28). Let $\kappa_{\max}$ be the largest eigenvalue of $\widehat{H}$. Then $\|\widehat{\beta}(\lambda)\|_{\widehat{H}}$ can be upper bounded as

$$\|\widehat{\beta}(\lambda)\|_{\widehat{H}} \le \sqrt{\kappa_{\max}} \|\widehat{\beta}(\lambda)\|_2 \le \sqrt{\kappa_{\max}} \|\widetilde{\beta}(\lambda)\|_2,$$

where the first inequality may be confirmed by eigen-decomposing $\widehat{H}$ and the second inequality is clear from the definitions of $\widehat{\beta}(\lambda)$ and $\widetilde{\beta}(\lambda)$. Let $\kappa_{\min}$ be the smallest eigenvalue of $\widehat{H}$. Then an upper bound of $\|\widetilde{\beta}(\lambda)\|_2^2$ is given as

$$\|\widetilde{\beta}(\lambda)\|_2^2 = \widehat{h}^\top (\widehat{H} + \lambda I_b)^{-2} \widehat{h} \le \frac{1}{(\kappa_{\min} + \lambda)^2} \|\widehat{h}\|_2^2 \le \frac{1}{\lambda^2} \|\widehat{h}\|_2^2,$$

where the last inequality follows from $\kappa_{\min} > 0$.

Now we have

$$\begin{aligned} \frac{\|\widehat{\beta}(\lambda)\|_{\widehat{H}}}{\sum_{i=1}^{n_{\mathrm{tr}}} w(x_i^{\mathrm{tr}}; \widehat{\beta}(\lambda))} &\le \frac{1}{\sum_{i=1}^{n_{\mathrm{tr}}} w(x_i^{\mathrm{tr}}; \widehat{\beta}(\lambda))} \frac{\sqrt{\kappa_{\max}} \|\widehat{h}\|_2}{\lambda} \\ &= \frac{1}{\sum_{i=1}^{n_{\mathrm{tr}}} \sum_{\ell=1}^{b} \varphi_\ell(x_i^{\mathrm{tr}}) \widehat{\beta}_\ell(\lambda) / \|\widehat{\beta}(\lambda)\|_1} \frac{\sqrt{\kappa_{\max}} \|\widehat{h}\|_2}{\lambda \|\widehat{\beta}(\lambda)\|_1}. \end{aligned}$$

For the denominator of the above expression, we have

$$\sum_{i=1}^{n_{\mathrm{tr}}} \sum_{\ell=1}^{b} \varphi_\ell(x_i^{\mathrm{tr}}) \frac{\widehat{\beta}_\ell(\lambda)}{\|\widehat{\beta}(\lambda)\|_1} \ \ge \ \min_{\ell'} \Big(\sum_{i=1}^{n_{\mathrm{tr}}} \varphi_{\ell'}(x_i^{\mathrm{tr}}) \Big) \cdot \sum_{\ell=1}^{b} \frac{\widehat{\beta}_\ell(\lambda)}{\|\widehat{\beta}(\lambda)\|_1} \ = \ \min_{\ell} \sum_{i=1}^{n_{\mathrm{tr}}} \varphi_\ell(x_i^{\mathrm{tr}}),$$

where the last equality follows from the non-negativity of $\widehat{\beta}_\ell(\lambda)$. The reciprocal of $\|\widehat{h}\|_2 / \|\widehat{\beta}(\lambda)\|_1$ is lower bounded as follows:

$$\frac{\|\widehat{\beta}(\lambda)\|_1}{\|\widehat{h}\|_2} \ = \ \left\| \max\left\{ \frac{\widetilde{\beta}(\lambda)}{\|\widehat{h}\|_2}, 0 \right\} \right\|_1 \ \ge \ \left\| \max\left\{ \frac{\widetilde{\beta}(\lambda)}{\|\widehat{h}\|_2}, 0 \right\} \right\|_\infty \ = \ \max_\ell \frac{\widetilde{\beta}_\ell(\lambda)}{\|\widehat{h}\|_2},$$

where the last equality follows from the fact that there is an ℓ such that $\widetilde{\beta}_\ell(\lambda) > 0$; otherwise, we have $\sum_{i=1}^{n_{\mathrm{tr}}} w(x_i^{\mathrm{tr}}; \widehat{\beta}) = 0$ which contradicts to the assumption of the theorem. Let us put

$$\kappa e = \frac{\widetilde{\beta}(\lambda)}{\|\widehat{h}\|_2},$$

where $\kappa > 0$ and $e \in \mathbb{R}^b$ such that $\|e\|_2 = 1$. Then we have

$$(\kappa_{\max} + \lambda)^{-1} \le \kappa \text{ and } e^\top \widehat{h} > 0.$$

Note that there exists an ℓ such that $e_\ell > 0$. Then, we have

$$\begin{aligned}\max_\ell \frac{\widetilde{\beta}_\ell(\lambda)}{\|\widehat{h}\|_2} &= \max_\ell \kappa e_\ell = \kappa \max_\ell e_\ell \geq \frac{1}{\kappa_{\max}+\lambda} \max_\ell e_\ell \\ &\geq \frac{1}{\kappa_{\max}+\lambda} \min_e \{\max_\ell e_\ell \mid e^\top e = 1, e^\top \widehat{h}/\|\widehat{h}\|_1 > 0\}.\end{aligned}$$

Now we prove the following lemma.

Lemma 8 *Let $p_1, p_2, \ldots, p_b \, (b \geq 2)$ be positive numbers such that*

$$\sum_{\ell=1}^b p_\ell = 1,$$

and let

$$\varepsilon = \frac{1}{\sqrt{b}} \min_\ell \frac{p_\ell}{1-p_\ell}.$$

Then, there exists no $e = (e_1, e_2, \ldots, e_b) \in \mathbb{R}^b$ such that the three conditions,

$$\sum_{\ell=1}^b e_\ell^2 = 1, \ \sum_{\ell=1}^b p_\ell e_\ell > 0, \text{ and } e_\ell < \varepsilon \text{ for } \ell = 1, 2, \ldots, b$$

are satisfied at the same time.

Proof We suppose that $e \in \mathbb{R}^b$ satisfies the three conditions. If $\min_\ell p_\ell/(1-p_\ell) > 1$, we have $p_\ell > 1/2$ for all ℓ. However, this is contradictory to $\sum_{\ell=1}^b p_\ell = 1$. Therefore, we have

$$\min_\ell p_\ell/(1-p_\ell) \leq 1,$$

from which we have

$$\varepsilon \leq 1/\sqrt{b}.$$

The equality constraint $\sum_{\ell=1}^b e_\ell^2 = 1$ implies the condition that there exists an e_i such that $|e_i| \geq 1/\sqrt{b}$. Moreover, we have $e_1, e_2, \ldots, e_b < \varepsilon \leq 1/\sqrt{b}$, and thus there is an e_i such that $e_i \leq -1/\sqrt{b}$. Hence, we have

$$\frac{p_i}{\sqrt{b}} \leq -p_i e_i < \sum_{\ell \neq i} p_\ell e_\ell < \sum_{\ell \neq i} p_\ell \frac{1}{\sqrt{b}} \min_k \frac{p_k}{1-p_k} = (1-p_i)\frac{1}{\sqrt{b}} \min_k \frac{p_k}{1-p_k} \leq \frac{p_i}{\sqrt{b}}.$$

This results in contradiction. ■

Let $p_\ell = \widehat{h}_\ell/\|\widehat{h}\|_1$ and we use Lemma 8. Note that any element of $\widehat{h}$ is positive. Then, we have

$$\frac{\|\widehat{\beta}(\lambda)\|_1}{\|\widehat{h}\|_2} \geq \frac{1}{\kappa_{\max}+\lambda} \cdot \frac{1}{\sqrt{b}} \min_\ell \frac{p_\ell}{\sum_{i \neq \ell} p_i}.$$

Moreover, we have

$$\min_\ell \frac{p_\ell}{\sum_{i \neq \ell} p_i} \geq \frac{\min_\ell \widehat{h}_\ell}{\sum_{\ell'=1}^b \widehat{h}_{\ell'}} = \frac{\min_\ell \sum_{j=1}^{n_{\mathrm{te}}} \varphi_\ell(x_j^{\mathrm{te}})}{\sum_{\ell'=1}^b \sum_{j=1}^{n_{\mathrm{te}}} \varphi_{\ell'}(x_j^{\mathrm{te}})} \geq \frac{\min_\ell \sum_{j=1}^{n_{\mathrm{te}}} \varphi_\ell(x_j^{\mathrm{te}})}{n_{\mathrm{te}} b},$$

where the last inequality follows from the assumption $0 < \varphi_\ell(x) \leq 1$. Therefore, we have the inequality

$$\begin{aligned}
&\frac{1}{\sum_{i=1}^{n} w(x_i^{\mathrm{tr}}; \widehat{\beta}(\lambda))} \frac{\sqrt{\kappa_{\max}} \|\widehat{h}\|_2}{\lambda} \\
&\leq b\sqrt{b\kappa_{\max}} \left(1 + \frac{\kappa_{\max}}{\lambda}\right) \frac{1}{\min_\ell \sum_{i=1}^{n_{\mathrm{tr}}} \varphi_\ell(x_i^{\mathrm{tr}})} \cdot \frac{n_{\mathrm{te}}}{\min_{\ell'} \sum_{j=1}^{n_{\mathrm{te}}} \varphi_{\ell'}(x_j^{\mathrm{te}})}. \qquad (63)
\end{aligned}$$

An upper bound of $\kappa_{\max}$ is given as follows. For all $a \in \mathbb{R}^b$, the inequality

$$-\sum_{\ell=1}^{b} |a_\ell| \varphi_\ell(x) \leq \sum_{\ell=1}^{b} a_\ell \varphi_\ell(x) \leq \sum_{\ell=1}^{b} |a_\ell| \varphi_\ell(x) \qquad (64)$$

holds because of the positivity of $\varphi_\ell(x)$. Let us define $\bar{a} \in \mathbb{R}^b$ for given $a \in \mathbb{R}^b$ as

$$\bar{a} = (|a_1|, |a_2|, \ldots, |a_b|)^\top.$$

Note that $\|\bar{a}\|_2 = \|a\|_2$ holds. Then, using Eq. (64), we obtain the inequality

$$a^\top \widehat{H} a = \frac{1}{n_{\mathrm{tr}}} \sum_{i=1}^{n_{\mathrm{tr}}} \left(\sum_{\ell=1}^{b} a_\ell \varphi_\ell(x_i^{\mathrm{tr}}) \right)^2 \leq \frac{1}{n_{\mathrm{tr}}} \sum_{i=1}^{n_{\mathrm{tr}}} \left(\sum_{\ell=1}^{b} |a_\ell| \varphi_\ell(x_i^{\mathrm{tr}}) \right)^2 = \bar{a}^\top \widehat{H} \bar{a},$$

for any $a \in \mathbb{R}^b$. Therefore, we obtain

$$\max_{\|a\|_2=1} a^\top \widehat{H} a \leq \max_{\|a\|_2=1} \bar{a}^\top \widehat{H} \bar{a} = \max_{\|a\|_2=1,\, a \geq 0_b} a^\top \widehat{H} a, \qquad (65)$$

where the last equality is derived from the relation,

$$\{\bar{a} \mid \|a\|_2 = 1,\ a \in \mathbb{R}^b\} = \{a \mid \|a\|_2 = 1,\ a \geq 0_b,\ a \in \mathbb{R}^b\}.$$

On the other hand, due to the additional constraint $a \geq 0_b$, the inequality

$$\max_{\|a\|_2=1,\, a \geq 0_b} a^\top \widehat{H} a \leq \max_{\|a\|_2=1} a^\top \widehat{H} a \qquad (66)$$

holds. From Eqs. (65) and (66), we have

$$\kappa_{\max} = \max_{\|a\|_2=1} a^\top \widehat{H} a = \max_{\|a\|_2=1, a \geq 0_b} a^\top \widehat{H} a = \max_{\|a\|_2=1,\, a \geq 0_b} \frac{1}{n_{\mathrm{tr}}} \sum_{i=1}^{n_{\mathrm{tr}}} \left(\sum_{\ell=1}^{b} a_\ell \varphi_\ell(x_i^{\mathrm{tr}}) \right)^2.$$

Using the assumption $0 < \varphi_\ell(x) \leq 1$, we have

$$\begin{aligned}
\kappa_{\max} &= \max_{\|a\|_2=1,\, a \geq 0_b} \frac{1}{n_{\mathrm{tr}}} \sum_{i=1}^{n_{\mathrm{tr}}} \left(\sum_{\ell=1}^{b} a_\ell \varphi_\ell(x_i^{\mathrm{tr}}) \right)^2 \leq \max_{\|a\|_2=1,\, a \geq 0_b} \frac{1}{n_{\mathrm{tr}}} \sum_{i=1}^{n_{\mathrm{tr}}} \left(\sum_{\ell=1}^{b} a_\ell \right)^2 \\
&= \max_{\|a\|_2=1,\, a \geq 0_b} \left(\sum_{\ell=1}^{b} a_\ell \right)^2 \leq \max_{\|a\|_2=1,\, a \geq 0_b} b \cdot \sum_{\ell=1}^{b} a_\ell^2 \\
&= b, \qquad (67)
\end{aligned}$$

where the Schwarz inequality for a and 1_b is used in the last inequality. The inequalities (63) and (67) lead to the inequality (29).

It is clear that the upper bound (29) is a decreasing function of λ (> 0). For the Gaussian basis function, $\varphi_\ell(x)$ is an increasing function with respect to the Gaussian width σ. Thus, Eq. (29) is a decreasing function of σ.

Appendix H. 'Bridge' Upper Bound of Approximation Error for uLSIF

Here we prove Theorem 7.

From the triangle inequality, we obtain

$$\mathrm{diff}(\lambda) \leq \frac{\inf_{\lambda' \geq 0} \|\widehat{\alpha}(\lambda') - \widehat{\gamma}(\lambda)\|_{\widehat{H}} + \|\widehat{\gamma}(\lambda) - \widehat{\beta}(\lambda)\|_{\widehat{H}}}{\sum_{i=1}^{n_{\mathrm{tr}}} \widehat{w}(x_i^{\mathrm{tr}}; \widehat{\beta}(\lambda))}. \tag{68}$$

We derive an upper bound of the first term.

First, we show that the LSIF optimization problem (6) is equivalently expressed as

$$\min_{\alpha \in \mathbb{R}^b} \left[\frac{1}{2} \alpha^\top \widehat{H} \alpha - \widehat{h}^\top \alpha \right]$$
$$\text{subject to } \alpha \geq 0_b, \;\; 1_b^\top \alpha \leq c,$$

which we refer to as LSIF$'$. The KKT conditions of LSIF (6) are given as

$$\begin{cases} \widehat{H}\alpha - \widehat{h} + \lambda 1_b - \mu = 0_b, \\ \alpha \geq 0_b, \;\; \mu \geq 0_b, \;\; \alpha^\top \mu = 0, \end{cases}$$

where μ is the Lagrange multiplier vector. Similarly, the KKT conditions of LSIF$'$ are given as

$$\begin{cases} \widehat{H}\alpha - \widehat{h} + \mu_0 1_b - \mu = 0_b, \\ \alpha \geq 0_b, \;\; \mu \geq 0_b, \;\; \alpha^\top \mu = 0, \\ 1_b^\top \alpha - c \leq 0, \; \mu_0 \geq 0, \;\; (1_b^\top \alpha - c)\mu_0 = 0, \end{cases} \tag{69}$$

where μ and μ_0 are the Lagrange multipliers. Let $(\widehat{\alpha}(\lambda), \widehat{\mu}(\lambda))$ be the solution of the KKT conditions of LSIF. Then, we find that $(\alpha, \mu, \mu_0) = (\widehat{\alpha}(\lambda), \widehat{\mu}(\lambda), \lambda)$ is the solution of Eq. (69) with $c = 1_b^\top \widehat{\alpha}(\lambda)$. Note that LSIF$'$ is a strictly convex optimization problem, and thus $\widehat{\alpha}(\lambda)$ is the unique optimal solution. Conversely, when the solution of Eq. (69) is provided as $(\widehat{\alpha}, \widehat{\mu}, \mu_0)$, LSIF with $\lambda = \mu_0$ has the same optimal solution $\widehat{\alpha}$.

When the optimal solution of LSIFq is $\widehat{\gamma}(\lambda)$, the KKT conditions of LSIFq (30) are given as

$$\widehat{H}\widehat{\gamma}(\lambda) - \widehat{h} + \lambda\widehat{\gamma}(\lambda) - \widehat{\eta} = 0_b, \tag{70}$$

$$\widehat{\gamma}(\lambda) \geq 0_b, \;\; \widehat{\eta} \geq 0_b, \;\; \widehat{\gamma}(\lambda)^\top \widehat{\eta} = 0, \tag{71}$$

where $\widehat{\eta}$ is the Lagrange multiplier vector.

Let $\widehat{\alpha}(\lambda_1)$ be the optimal solution of LSIF$'$ with $c = 1_b^\top \widehat{\gamma}(\lambda)$, and suppose that the solution $\widehat{\alpha}(\lambda_1)$ coincides with that of LSIF with $\lambda = \lambda_1$. Then, from Eq. (69), we have

$$\widehat{H}\widehat{\alpha}(\lambda_1) - \widehat{h} + \lambda_1 1_b - \widehat{\mu}(\lambda_1) = 0_b, \tag{72}$$

$$\widehat{\alpha}(\lambda_1) \geq 0_b, \;\; \widehat{\mu}(\lambda_1) \geq 0_b, \;\; \widehat{\alpha}(\lambda_1)^\top \widehat{\mu}(\lambda_1) = 0, \tag{73}$$

$$1_b^\top \widehat{\alpha}(\lambda_1) - 1_b^\top \widehat{\gamma}(\lambda) \leq 0, \;\; \lambda_1 \geq 0, \;\; (1_b^\top \widehat{\alpha}(\lambda_1) - 1_b^\top \widehat{\gamma}(\lambda))\lambda_1 = 0. \tag{74}$$

From Eqs. (70) and (72), we obtain

$$\widehat{H}(\widehat{\alpha}(\lambda_1) - \widehat{\gamma}(\lambda)) = -\lambda_1 1_b + \lambda\widehat{\gamma}(\lambda) + \widehat{\mu}(\lambda_1) - \widehat{\eta}. \tag{75}$$

Applying Eqs. (71), (73), (74), and (75), we have

$$\begin{aligned}\inf_{\lambda'\geq 0}\|\widehat{\alpha}(\lambda')-\widehat{\gamma}(\lambda)\|_{\widehat{H}}^2 &\leq (\widehat{\alpha}(\lambda_1)-\widehat{\gamma}(\lambda))^\top\widehat{H}(\widehat{\alpha}(\lambda_1)-\widehat{\gamma}(\lambda))\\ &= -\lambda_1(\widehat{\alpha}(\lambda_1)-\widehat{\gamma}(\lambda))^\top 1_b+\lambda(\widehat{\alpha}(\lambda_1)-\widehat{\gamma}(\lambda))^\top\widehat{\gamma}(\lambda)\\ &\quad +(\widehat{\alpha}(\lambda_1)-\widehat{\gamma}(\lambda))^\top(\widehat{\mu}(\lambda_1)-\widehat{\eta})\\ &= \lambda(\widehat{\alpha}(\lambda_1)^\top\widehat{\gamma}(\lambda)-\|\widehat{\gamma}(\lambda)\|_2^2)-\widehat{\alpha}(\lambda_1)^\top\widehat{\eta}-\widehat{\gamma}(\lambda)^\top\widehat{\mu}(\lambda_1)\\ &\leq \lambda(\widehat{\alpha}(\lambda_1)^\top\widehat{\gamma}(\lambda)-\|\widehat{\gamma}(\lambda)\|_2^2). \end{aligned} \tag{76}$$

From $\widehat{\alpha}(\lambda_1)\geq 0_b, \widehat{\gamma}(\lambda)\geq 0_b$, and $1_b^\top\widehat{\alpha}(\lambda_1)\leq 1_b^\top\widehat{\gamma}(\lambda)$, we have

$$\|\widehat{\alpha}(\lambda_1)\|_1 = 1_b^\top\widehat{\alpha}(\lambda_1)\leq 1_b^\top\widehat{\gamma}(\lambda)\leq\|\widehat{\gamma}(\lambda)\|_1.$$

Then we have the following inequality:

$$\begin{aligned}\widehat{\alpha}(\lambda_1)^\top\widehat{\gamma}(\lambda) &\leq \widehat{\alpha}(\lambda_1)^\top(\|\widehat{\gamma}(\lambda)\|_\infty 1_b)\\ &= \|\widehat{\alpha}(\lambda_1)\|_1\cdot\|\widehat{\gamma}(\lambda)\|_\infty\leq\|\widehat{\gamma}(\lambda)\|_1\cdot\|\widehat{\gamma}(\lambda)\|_\infty.\end{aligned} \tag{77}$$

For p and q such that $1/p+1/q=1$ and $1\leq p,q\leq\infty$, Hölder's inequality states that

$$\|\alpha*\beta\|_1\leq\|\alpha\|_p\cdot\|\beta\|_q,$$

where $\alpha*\beta$ denotes the element-wise product of α and β. Setting $p=1, q=\infty$, and $\alpha=\beta=\widehat{\gamma}(\lambda)$ in Hölder's inequality, we have

$$\|\widehat{\gamma}(\lambda)\|_1\cdot\|\widehat{\gamma}(\lambda)\|_\infty-\|\widehat{\gamma}(\lambda)\|_2^2\geq 0. \tag{78}$$

Combining Eqs. (68), (76), (77), and (78), we obtain

$$\mathrm{diff}(\lambda)\ \leq\ \frac{\sqrt{\lambda(\|\widehat{\gamma}(\lambda)\|_1\cdot\|\widehat{\gamma}(\lambda)\|_\infty-\|\widehat{\gamma}(\lambda)\|_2^2)}+\|\widehat{\gamma}(\lambda)-\widehat{\beta}(\lambda)\|_{\widehat{H}}}{\sum_{i=1}^{n_{\mathrm{tr}}}\widehat{w}(x_i^{\mathrm{tr}};\widehat{\beta}(\lambda))}.$$

Appendix I. Closed Form of LOOCV Score for uLSIF

Here we derive a closed form expression of the LOOCV score for uLSIF (see Figure 2 for the pseudo code).

Let

$$\varphi(x)=(\varphi_1(x),\varphi_2(x),\ldots,\varphi_b(x))^\top.$$

Then the matrix $\widehat{H}$ and the vector $\widehat{h}$ are expressed as

$$\widehat{H}=\frac{1}{n_{\mathrm{tr}}}\sum_{i=1}^{n_{\mathrm{tr}}}\varphi(x_i^{\mathrm{tr}})\varphi(x_i^{\mathrm{tr}})^\top,$$

$$\widehat{h}=\frac{1}{n_{\mathrm{te}}}\sum_{j=1}^{n_{\mathrm{te}}}\varphi(x_j^{\mathrm{te}}),$$

and the coefficients $\widetilde{\beta}(\lambda)$ can be computed by

$$\widetilde{\beta}(\lambda) = \widehat{B}_{\lambda}^{-1}\widehat{h}.$$

Let $\widehat{\beta}^{(i)}$ be the estimator obtained without the i-th training sample x_i^{tr} and the i-th test sample x_i^{te}. Then the estimator has the following closed form:

$$\widehat{\beta}^{(i)}(\lambda) = \max(0_b, \widetilde{\beta}^{(i)}(\lambda)),$$
$$\widetilde{\beta}^{(i)}(\lambda) = \left(\frac{1}{n_{\mathrm{tr}}-1}(n_{\mathrm{tr}}\widehat{H} - \varphi(x_i^{\mathrm{tr}})\varphi(x_i^{\mathrm{tr}})^\top) + \lambda I_b\right)^{-1} \frac{1}{n_{\mathrm{te}}-1}(n_{\mathrm{te}}\widehat{h} - \varphi(x_j^{\mathrm{te}})).$$

Let $\widehat{B} = \widehat{H} + \frac{\lambda(n_{\mathrm{tr}}-1)}{n_{\mathrm{tr}}} I_b$ and $\widetilde{\beta} = \widehat{B}^{-1}\widehat{h}$ in the following calculation. Using the Sherman-Woodbury-Morrison formula (33), we can simplify the expression of $\widetilde{\beta}^{(i)}(\lambda)$ as follows:

$$\begin{aligned}
\widetilde{\beta}^{(i)}(\lambda) =& \frac{n_{\mathrm{tr}}-1}{n_{\mathrm{tr}}}\left(\widehat{B} - \frac{1}{n_{\mathrm{tr}}}\varphi(x_i^{\mathrm{tr}})\varphi(x_i^{\mathrm{tr}})^\top\right)^{-1}\left(\frac{n_{\mathrm{te}}}{n_{\mathrm{te}}-1}\widehat{h} - \frac{1}{n_{\mathrm{te}}-1}\varphi(x_i^{\mathrm{te}})\right) \\
=& \frac{n_{\mathrm{tr}}-1}{n_{\mathrm{tr}}}\left(\widehat{B}^{-1} + \frac{1}{n_{\mathrm{tr}} - \varphi(x_i^{\mathrm{tr}})^\top \widehat{B}^{-1}\varphi(x_i^{\mathrm{tr}})}\widehat{B}^{-1}\varphi(x_i^{\mathrm{tr}})\varphi(x_i^{\mathrm{tr}})^\top\widehat{B}^{-1}\right) \\
&\times\left(\frac{n_{\mathrm{te}}}{n_{\mathrm{te}}-1}\widehat{h} - \frac{1}{n_{\mathrm{te}}-1}\varphi(x_i^{\mathrm{te}})\right) \\
=& \frac{(n_{\mathrm{tr}}-1)n_{\mathrm{te}}}{n_{\mathrm{tr}}(n_{\mathrm{te}}-1)}\left(\widetilde{\beta} + \frac{\varphi(x_i^{\mathrm{tr}})^\top\widetilde{\beta}}{n_{\mathrm{tr}} - \varphi(x_i^{\mathrm{tr}})^\top\widehat{B}^{-1}\varphi(x_i^{\mathrm{tr}})}\widehat{B}^{-1}\varphi(x_i^{\mathrm{tr}})\right) \\
&- \frac{(n_{\mathrm{tr}}-1)}{n_{\mathrm{tr}}(n_{\mathrm{te}}-1)}\left(\widehat{B}^{-1}\varphi(x_i^{\mathrm{te}}) + \frac{\varphi(x_i^{\mathrm{tr}})^\top\widehat{B}^{-1}\varphi(x_i^{\mathrm{te}})}{n_{\mathrm{tr}} - \varphi(x_i^{\mathrm{tr}})^\top\widehat{B}^{-1}\varphi(x_i^{\mathrm{tr}})}\widehat{B}^{-1}\varphi(x_i^{\mathrm{tr}})\right).
\end{aligned}$$

Thus the matrix inversion required for computing $\widetilde{\beta}^{(i)}(\lambda)$ for all $i = 1, 2, \ldots, n_{\mathrm{tr}}$ is only $\widehat{B}$. Applying this to Eq. (32) and rearrange the formula, we can compute the LOOCV score analytically.

References

H. Akaike. A new look at the statistical model identification. *IEEE Transactions on Automatic Control*, AC-19(6):716–723, 1974.

S. Amari, N. Fujita, and S. Shinomoto. Four types of learning curves. *Neural Computation*, 4(4): 605–618, 1992.

P. Baldi and S. Brunak. *Bioinformatics: The Machine Learning Approach*. MIT Press, Cambridge, 1998.

D. Bertsekas, A. Nedic, and A. Ozdaglar. *Convex Analysis and Optimization*. Athena Scientific, Belmont, MA, 2003.

M. J. Best. An algorithm for the solution of the parametric quadratic programming problem. CORR Report 82-24, Faculty of Mathematics, University of Waterloo, 1982.

S. Bickel and T. Scheffer. Dirichlet-enhanced spam filtering based on biased samples. In B. Schölkopf, J. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems 19*. MIT Press, Cambridge, MA, 2007.

S. Bickel, M. Brückner, and T. Scheffer. Discriminative learning for differing training and test distributions. In *Proceedings of the 24th International Conference on Machine Learning*, 2007.

K. M. Borgwardt, A. Gretton, M. J. Rasch, H.-P. Kriegel, B. Schölkopf, and A. J. Smola. Integrating structured biological data by kernel maximum mean discrepancy. *Bioinformatics*, 22(14):e49–e57, 2006.

S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, Cambridge, 2004.

A. P. Bradley. The use of the area under the ROC curve in the evaluation of machine learning algorithms. *Pattern Recognition*, 30(7):1145–1159, 1997.

M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander. LOF: Identifying density-based local outliers. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, 2000.

G. C. Cawley and N. L. C. Talbot. Fast exact leave-one-out cross-validation of sparse least-squares support vector machines. *Neural Networks*, 17(10):1467–75, 2004.

S. S. Chen, D. L. Donoho, and M. A. Saunders. Atomic decomposition by basis pursuit. *SIAM Journal on Scientific Computing*, 20(1):33–61, 1998.

K. F. Cheng and C. K. Chu. Semiparametric density estimation under a two-sample density ratio model. *Bernoulli*, 10(4):583–604, 2004.

A. Dembo and O. Zeitouni. *Large Deviations Techniques and Applications*. Springer-Verlag, New York, 1998.

B. Efron, T. Hastie, R. Tibshirani, and I. Johnstone. Least angle regression. *The Annals of Statistics*, 32(2):407–499, 2004.

T. Evgeniou, M. Pontil, and T. Poggio. Regularization networks and support vector machines. *Advances in Computational Mathematics*, 13(1):1–50, 2000.

E. A. Fernandez. *The dprep Package*, 2005. URL `http://math.uprm.edu/ edgar/dprep.pdf`.

G. S. Fishman. *Monte Carlo: Concepts, Algorithms, and Applications*. Springer-Verlag, Berlin, 1996.

G. H. Golub and C. F. Van Loan. *Matrix Computations*. Johns Hopkins University Press, Baltimore, MD, 1996.

H. Hachiya, T. Akiyama, M. Sugiyama, and J. Peters. Adaptive importance sampling with automatic model selection in value function approximation. In *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence (AAAI2008)*, pages 1351–1356, Chicago, USA, Jul. 13–17 2008. The AAAI Press.

L. K. Hansen and J. Larsen. Linear unlearning for crossvalidation. *Advances in Computational Mathematics*, 5:269–280, 1996.

W. Härdle, M. Müller, S. Sperlich, and A. Werwatz. *Nonparametric and Semiparametric Models*. Springer Series in Statistics. Springer, Berlin, 2004.

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.

J. J. Heckman. Sample selection bias as a specification error. *Econometrica*, 47(1):153–162, 1979.

S. Hido, Y. Tsuboi, H. Kashima, M. Sugiyama, and T. Kanamori. Inlier-based outlier detection via direct density ratio estimation. In *Proceedings of IEEE International Conference on Data Mining (ICDM2008)*, Pisa, Italy, Dec. 15–19 2008.

V. Hodge and J. Austin. A survey of outlier detection methodologies. *Artificial Intelligence Review*, 22(2):85–126, 2004.

A. E. Hoerl and R. W. Kennard. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(3):55–67, 1970.

J. Huang, A. Smola, A. Gretton, K. M. Borgwardt, and B. Schölkopf. Correcting sample selection bias by unlabeled data. In B. Schölkopf, J. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems 19*, pages 601–608. MIT Press, Cambridge, MA, 2007.

A. Karatzoglou, A. Smola, K. Hornik, and A. Zeileis. An S4 package for kernel methods in R. *Journal of Statistical Planning and Inference*, 11(9):1–20, 2004.

S. Konishi and G. Kitagawa. Generalized information criteria in model selection. *Biometrika*, 83: 875–890, 1996.

S. Kullback and R. A. Leibler. On information and sufficiency. *Annals of Mathematical Statistics*, 22:79–86, 1951.

C.-J. Lin, R. C. Weng, and S. S. Keerthi. Trust region Newton method for large-scale logistic regression. Technical report, Department of Computer Science, National Taiwan University, 2007. URL `http://www.csie.ntu.edu.tw/~cjlin/liblinear/`.

A. Luntz and V. Brailovsky. On estimation of characters obtained in statistical procedure of recognition. *Technicheskaya Kibernetica*, 3, 1969. in Russian.

T. P. Minka. A comparison of numerical optimizers for logistic regression. Technical report, Microsoft Research, 2007. URL `http://research.microsoft.com/~minka/papers/logreg/minka-logreg.pdf`.

J. E. Moody. The effective number of parameters: An analysis of generalization and regularization in nonlinear learning systems. In J. E. Moody, S. J. Hanson, and R. P. Lippmann, editors, *Advances in Neural Information Processing Systems 4*, pages 847–854. Morgan Kaufmann Publishers, Inc., 1992.

X. Nguyen, M. J. Wainwright, and M. I. Jordan. Estimating divergence functions and the likelihood ratio by penalized convex risk minimization. In *Advances in Neural Information Processing Systems 20*, Cambridge, MA, 2008. MIT Press.

K. B. Petersen and M. S. Pedersen. The matrix cookbook. Technical report, Technical University of Denmark, 2007. URL `http://www2.imm.dtu.dk/pubdb/p.php?3274`.

J. Qin. Inferences for case-control and semiparametric two-sample density ratio models. *Biometrika*, 85(3):619–639, 1998.

J. Quiñonero-Candela, M. Sugiyama, A. Schwaighofer, and N. Lawrence, editors. *Dataset Shift in Machine Learning*. MIT Press, Cambridge, MA, 2008.

C. E. Rasmussen, R. M. Neal, G. E. Hinton, D. van Camp, M. Revow, Z. Ghahramani, R. Kustra, and R. Tibshirani. The DELVE manual, 1996. URL `http://www.cs.toronto.edu/~delve/`.

G. Rätsch, T. Onoda, and K.-R. Müller. Soft margins for adaboost. *Machine Learning*, 42(3): 287–320, 2001.

K. Scheinberg. An efficient implementation of an active set method for SVMs. *Journal of Machine Learning Research*, 7:2237–2257, 2006.

B. Schölkopf and A. J. Smola. *Learning with Kernels*. MIT Press, Cambridge, MA, 2002.

B. Schölkopf, J. C. Platt, J. Shawe-Taylor, A. J. Smola, and R. C. Williamson. Estimating the support of a high-dimensional distribution. *Neural Computation*, 13(7):1443–1471, 2001.

H. Shimodaira. Improving predictive inference under covariate shift by weighting the log-likelihood function. *Journal of Statistical Planning and Inference*, 90(2):227–244, 2000.

L. Song, A. Smola, A. Gretton, K. M. Borgwardt, and J. Bedo. Supervised feature selection via dependence estimation. In *Proceedings of the 24th International Conference on Machine learning*, pages 823–830, New York, NY, USA, 2007. ACM.

I. Steinwart. On the influence of the kernel on the consistency of support vector machines. *Journal of Machine Learning Research*, 2:67–93, 2001.

M. Stone. Cross-validatory choice and assessment of statistical predictors. *Journal of the Royal Statistical Society B*, 32(2):111–147, 1974.

M. Sugiyama and K.-R. Müller. Input-dependent estimation of generalization error under covariate shift. *Statistics & Decisions*, 23(4):249–279, 2005.

M. Sugiyama, M. Krauledat, and K.-R. Müller. Covariate shift adaptation by importance weighted cross validation. *Journal of Machine Learning Research*, 8:985–1005, May 2007.

M. Sugiyama, S. Nakajima, H. Kashima, P. von Bünau, and M. Kawanabe. Direct importance estimation with model selection and its application to covariate shift adaptation. In J. C. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems 20*, pages 1433–1440, Cambridge, MA, 2008a. MIT Press.

M. Sugiyama, T. Suzuki, S. Nakajima, H. Kashima, P. von Bünau, and M. Kawanabe. Direct importance estimation for covariate shift adaptation. *Annals of the Institute of Statistical Mathematics*, 60(4):699–746, 2008b.

R. S. Sutton and G. A. Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, 1998.

D. M. J. Tax and R. P. W. Duin. Support vector data description. *Machine Learning*, 54(1):45–66, 2004.

R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society, Series B*, 58(1):267–288, 1996.

Y. Tsuboi, H. Kashima, S. Hido, S. Bickel, and M. Sugiyama. Direct density ratio estimation for large-scale covariate shift adaptation. In *Proceedings of 2008 SIAM International Conference on Data Mining (SDM2008)*, Atlanta, Georgia, USA, 2008.

V. N. Vapnik. *Statistical Learning Theory*. Wiley, New York, 1998.

G. Wahba. *Spline Model for Observational Data*. Society for Industrial and Applied Mathematics, Philadelphia and Pennsylvania, 1990.

P. M. Williams. Bayesian regularization and pruning using a Laplace prior. *Neural Computation*, 7 (1):117–143, 1995.

J. R. Wolpaw, N. Birbaumer, D. J. McFarland, G. Pfurtscheller, and T. M. Vaughan. Brain-computer interfaces for communication and control. *Clinical Neurophysiology*, 113(6):767–791, 2002.

B. Zadrozny. Learning and evaluating classifiers under sample selection bias. In *Proceedings of the Twenty-First International Conference on Machine Learning*, New York, NY, 2004. ACM Press.

Journal of Machine Learning Research 10 (2009) 1447-1468 Submitted 7/07; Revised 3/09; Published 7/09

Classification with Gaussians and Convex Loss

Dao-Hong Xiang DAOHONGXIANG@GMAIL.COM
Ding-Xuan Zhou MAZHOU@CITYU.EDU.HK
Department of Mathematics
City University of Hong Kong
Tat Chee Avenue, Kowloon, Hong Kong, China

Editor: John Shawe-Taylor

Abstract

This paper considers binary classification algorithms generated from Tikhonov regularization schemes associated with general convex loss functions and varying Gaussian kernels. Our main goal is to provide fast convergence rates for the excess misclassification error. Allowing varying Gaussian kernels in the algorithms improves learning rates measured by regularization error and sample error. Special structures of Gaussian kernels enable us to construct, by a nice approximation scheme with a Fourier analysis technique, uniformly bounded regularizing functions achieving polynomial decays of the regularization error under a Sobolev smoothness condition. The sample error is estimated by using a projection operator and a tight bound for the covering numbers of reproducing kernel Hilbert spaces generated by Gaussian kernels. The convexity of the general loss function plays a very important role in our analysis.

Keywords: reproducing kernel Hilbert space, binary classification, general convex loss, varying Gaussian kernels, covering number, approximation

1. Introduction

In this paper we study binary classification algorithms generated from Tikhonov regularization schemes associated with general convex loss functions and varying Gaussian kernels.

Let X be a compact subset of $\mathbb{R}^n$ (input space) and $Y = \{1, -1\}$ (representing the two classes). Classification algorithms produce *binary classifiers* $\mathcal{C} : X \to Y$. The misclassification error is used to measure the prediction power of a classifier $\mathcal{C}$. If ρ is a probability distribution on $Z := X \times Y$, then the *misclassification error* of $\mathcal{C}$ is defined by

$$\mathcal{R}(\mathcal{C}) = \text{Prob}\{\mathcal{C}(x) \neq y\} = \int_X P(y \neq \mathcal{C}(x)|x) d\rho_X.$$

Here ρ_X is the marginal distribution of ρ on X and $P(y|x)$ is the conditional distribution at $x \in X$. The classifier minimizing the misclassification error is called the Bayes rule f_c and is given by

$$f_c(x) = \begin{cases} 1, & \text{if } P(y = 1|x) \geq P(y = -1|x), \\ -1, & \text{otherwise.} \end{cases}$$

The performance of a classifier $\mathcal{C}$ can be measured by the *excess misclassification error* $\mathcal{R}(\mathcal{C}) - \mathcal{R}(f_c)$.

The classifiers considered here are induced by real-valued functions $f : X \to \mathbb{R}$ as $\mathcal{C}_f = \text{sgn}(f)$ which is defined by $\text{sgn}(f)(x) = 1$ if $f(x) \geq 0$ and $\text{sgn}(f)(x) = -1$ otherwise. The real-valued functions are generated from Tikhonov regularization schemes associated with general convex loss functions and varying Gaussian kernels.

Definition 1 *We say that* $\phi : \mathbb{R} \to \mathbb{R}_+$ *is a classifying loss (function) if it is convex, differentiable at* 0 *with* $\phi'(0) < 0$*, and the smallest zero of* ϕ *is* 1.

Examples of classifying loss functions include the least-square loss $\phi_{ls}(t) = (1-t)^2$, the hinge loss $\phi_h(t) = (1-t)_+ = \max\{1-t, 0\}$ for support vector machine (SVM) algorithms, and the r-norm SVM loss with $1 \le r < \infty$ defined by $\phi_r(t) = (\phi_h(t))^r$.

The *Gaussian kernel* with variance $\sigma > 0$ is the function on $X \times X$ given by

$$K^\sigma(x, x') = \exp\Big\{ -\frac{|x - x'|^2}{2\sigma^2} \Big\}. \tag{1}$$

It defines (Aronszajn, 1950) a reproducing kernel Hilbert space (RKHS) $\mathcal{H}_\sigma$.

With the loss ϕ and Gaussian kernel K^σ, the *Tikhonov regularization scheme* is defined (Wahba, 1990; Evgeniou et al., 2000; Cristianini and Shawe-Taylor, 2000) with a sample $\mathbf{z} = \{(x_i, y_i)\}_{i=1}^m \in Z^m$ as the solution $f_{\mathbf{z}} = f^{\phi}_{\mathbf{z},\sigma,\lambda}$ to the following minimization problem

$$f_{\mathbf{z}} = \arg\min_{f \in \mathcal{H}_\sigma} \Big\{ \frac{1}{m} \sum_{i=1}^m \phi(y_i f(x_i)) + \lambda \|f\|^2_{\mathcal{H}_\sigma} \Big\}. \tag{2}$$

Here λ is a positive constant called the *regularization parameter*. Throughout the paper we assume that the sample $\mathbf{z}$ is drawn independently according to the distribution ρ.

The purpose of this paper is to estimate the excess misclassification error $\mathcal{R}(\mathrm{sgn}(f_{\mathbf{z}})) - \mathcal{R}(f_c)$ as $m \to \infty$. Convergence rates will be derived under the choice of the parameters

$$\lambda = \lambda(m) = m^{-\gamma}, \quad \sigma = \sigma(m) = \lambda^{\zeta} = m^{-\gamma\zeta} \tag{3}$$

for some $\gamma, \zeta > 0$ and conditions on the distribution ρ and the loss ϕ. This has been done for the SVM in Steinwart and Scovel (2007) with the loss ϕ_h. Here we consider the error analysis with a general loss function ϕ (De Vito et al., 2004).

Let us demonstrate our main results by stating learning rates for the least-square loss $\phi = \phi_{ls}$. The rates will be proved in Section 4. They are given by means of a Tsybakov noise condition (Tsybakov, 2004) and a function smoothness condition stated in terms of Sobolev spaces. Since $\phi_{ls}(yf(x)) = (1 - yf(x))^2 = (y - f(x))^2$ for $y \in Y$, a minimizer of $\int_Z \phi_{ls}(yf(x)) d\rho$ is the *regression function* defined by

$$f_\rho(x) = \int_Y y d\rho(y|x) = P(y = 1|x) - P(y = -1|x), \qquad x \in X. \tag{4}$$

Definition 2 *Let* $0 \le q \le \infty$. *We say that* ρ *satisfies the Tsybakov noise condition with exponent* q *if there exists a constant* $C_q > 0$ *such that*

$$\rho_X(\{x \in X : |f_\rho(x)| \le C_q t\}) \le t^q, \qquad \forall t > 0. \tag{5}$$

Note that (5) always holds for $q = 0$ with $C_q = 1$. So setting the index $q = 0$ in (5) is the same as removing the Tsybakov noise condition. The case $q = \infty$ means $|f_\rho(x)| \geq C_q$ for almost every $x \in (X, \rho_X)$.

Recall the Sobolev space $H^s(\mathbb{R}^n)$ with index $s > 0$ consisting of all functions in $L^2(\mathbb{R}^n)$ with the semi-norm $|f|_{H^s(\mathbb{R}^n)} = \left\{(2\pi)^{-n} \int_{\mathbb{R}^n} |\xi|^{2s} |\hat{f}(\xi)|^2 d\xi\right\}^{\frac{1}{2}}$ finite where $\hat{f}$ is the Fourier transform of f defined for $f \in L^1(\mathbb{R}^n)$ as $\hat{f}(\xi) = \int_{\mathbb{R}^n} f(x) e^{-ix\cdot\xi}\, dx$.

Theorem 1 *Let $\phi = \phi_{ls}$. Assume (5) for some $q \in [0, \infty]$ and $\frac{d\rho_X}{dx} \in L^2(X)$. If for some $s > 0$, f_ρ equals the restriction onto X of some function in $H^s(\mathbb{R}^n) \cap L^\infty(\mathbb{R}^n)$, then by taking $\sigma = \lambda^\zeta$ with $0 < \zeta \leq \frac{1}{n+s}$ and $\lambda = m^{-\frac{1}{\zeta(s+2n+2)}}$, for any $0 < \delta < 1$, with confidence $1 - \delta$, we have*

$$\mathcal{R}(sgn(f_{\mathbf{z}})) - \mathcal{R}(f_c) \leq \widetilde{C}_{\rho,s,q,n} m^{-\theta_{ls}} \log\frac{2}{\delta} \qquad \textit{with} \quad \theta_{ls} = \frac{(q+1)s}{(q+2)(s+2n+2)}, \tag{6}$$

where $\widetilde{C}_{\rho,s,q,n}$ is a constant independent of m or δ.

When Tsybakov noise condition (5) is not assumed, we can still use Theorem 1 by setting $q = 0$ and obtain learning rate (6) with $\theta_{ls} = \frac{s}{2s+4n+4}$.

When q tends to infinity, the power index θ_{ls} in (6) has the limit $\frac{s}{s+2n+2}$ which can be very close to 1 for large s. So the learning rate can be $O(m^{\varepsilon-1})$ for arbitrarily small $\varepsilon > 0$ when q and s are large enough. To be more specific, for $0 < \varepsilon < 1$, when $q > \frac{1}{\varepsilon} - 2$ and $s \geq (2n+2)(q+2)\frac{1-\varepsilon}{\varepsilon(q+2)-1}$, we have $\theta_{ls} \geq 1 - \varepsilon$.

Remark 1 *We show that the power index θ_{ls} for learning rate (6) can be $1 - \varepsilon$ for arbitrarily small $\varepsilon > 0$. This result is new for scheme (2) associated with $\phi = \phi_{ls}$ and a single Gaussian kernel with changing variance $\sigma = \sigma(m)$. The same learning rates are achieved in the literature in two different settings: one is for the same least square regularization scheme associated with a single fixed Gaussian kernel, but under the much stronger condition that f_ρ lies in the range of powers of an integral operator associated with a fixed Gaussian kernel, requiring $f_\rho \in C^\infty$ (Zhang, 2004; De Vito et al., 2005; Smale and Zhou, 2007). The other setting is to allow flexible variances of Gaussians in (2), see Ying and Zhou (2007) and Wu et al. (2007).*

When the decision boundary $\{x \in X : f_\rho(x) = 0\}$ has measure zero and $\frac{d\rho_X}{dx} \in L^2(X)$, the smoothness condition for an extension of f_ρ implies (5) for some $q > 0$. In general, noise condition (5) does not require smoothness of f_ρ in domains away from the decision boundary.

Note that as $t \to -\infty$, the hinge loss ϕ_h for the SVM studied in Steinwart and Scovel (2007) increases slowly: $\phi_h(t) = O(|t|)$, while the least-square loss ϕ_{ls} in Theorem 1 increases moderately with $\phi_{ls}(t) = O(|t|^2)$. Difficulty arises for the error analysis with a general loss ϕ when $\phi(t)$ increases fast such as $\phi = \phi_r$ with very large r or the *exponential-hinge loss* we introduce in this paper as

$$\phi_{eh}(t) = \max\{e^{1-t} - 1, 0\} = \begin{cases} e^{1-t} - 1, & \text{if } t \leq 1, \\ 0, & \text{otherwise.} \end{cases}$$

The reason is random variables of form $\xi = \phi(yf(x))$ with $(x, y) = z \in (Z, \rho)$ are involved and large norms $\|f\|_{L^\infty(X)}$ would lead to large bounds for ξ. We shall use special properties of Gaussian kernels and construct functions $f_{\sigma,\lambda}$ which are uniformly bounded and have powerful approximation ability (see (9) and (10) below). With this construction, we can do the analysis well for the general loss ϕ by dealing with uniformly bounded random variables in an error decomposition approach (see (13) below). In particular, explicit learning rates will be given in Section 4 for the r-norm SVM loss ϕ_r (Theorem 4) and the exponential-hinge loss ϕ_{eh} (Theorem 5). Comparing with Theorem 1, we shall provide at the end of Section 4 an approximation theory viewpoint to the effect of various loss functions for learning algorithm (2): the exponential-hinge loss has some advantages over ϕ_{ls} and ϕ_r, the r-norm SVM loss ϕ_r may have worse performance when $r > 2$.

We list key notations used in the paper in a table given in Appendix B.

2. Two Special Properties of Gaussians and Key Bounds

The novelty in our approach for general ϕ and kernels K^σ arises from two special properties of the Gaussian kernels with changing variance $\sigma > 0$: nice approximation scheme and low capacity of the RKHS, described in Sections 2.1 and 2.3.

2.1 Regularizing Functions Generated by Gaussians

A data-free limit of (2) is a function $\widetilde{f}_{\sigma,\lambda}$ defined in terms of the *generalization error* $\mathcal{E}^\phi$ as

$$\widetilde{f}_{\sigma,\lambda} := \arg\min_{f\in\mathcal{H}_\sigma}\{\mathcal{E}^\phi(f)+\lambda\|f\|^2_{\mathcal{H}_\sigma}\}, \text{ where } \mathcal{E}^\phi(f)=\int_Z \phi(yf(x))\,d\rho. \tag{7}$$

This is the regularizing function used in the literature (De Vito et al., 2005; Yao, 2008; Zhang, 2004). It works well for the error analysis when the loss ϕ increases slowly or moderately (as $t\to-\infty$) such as $\phi=\phi_h$ or ϕ_{ls}.

In this paper we consider a general loss ϕ. When $\phi(t)$ increases fast (as $t\to-\infty$), applying the regularizing function $\widetilde{f}_{\sigma,\lambda}$ in the error analysis (described in Section 2.2) may lead to a random variable $\phi(y\widetilde{f}_{\sigma,\lambda}(x))$ of large bound.

The first novelty of this paper is to construct a function $f_{\sigma,\lambda}$ (which plays the role of a regularizing function in an error decomposition approach discussed in subsection 2.2) by special approximation ability of Gaussian kernels. The constructed function has two advantages. On one hand, it is uniformly bounded (with respect to both λ and σ) so that the random variable $\phi(yf_{\sigma,\lambda}(x))$ involved in the error analysis is bounded. On the other hand, it plays the same role as $\widetilde{f}_{\sigma,\lambda}$ in achieving nice bounds for the approximation error. The construction of the explicit approximation scheme for $f_{\sigma,\lambda}$ is done under a Sobolev smoothness condition of a measurable function f^ϕ_ρ minimizing $\mathcal{E}^\phi$, that is, for a. e. $x\in X$,

$$f^\phi_\rho(x)=\arg\min_{t\in\mathbb{R}}\int_Y \phi(yt)\,d\rho(y|x)=\arg\min_{t\in\mathbb{R}}\{\phi(t)P(y=1|x)+\phi(-t)P(y=-1|x)\}.$$

Theorem 2 *Assume that for some* $s>0$,

$$f^\phi_\rho=\tilde{f}^\phi_\rho|_X \text{ for some } \tilde{f}^\phi_\rho\in H^s(\mathbb{R}^n)\cap L^\infty(\mathbb{R}^n) \text{ and } \frac{d\rho_X}{dx}\in L^2(X). \tag{8}$$

Then we can find functions $\{f_{\sigma,\lambda}\in\mathcal{H}_\sigma : 0<\sigma\le 1,\lambda>0\}$ *such that*

$$\|f_{\sigma,\lambda}\|_{L^\infty(X)} \le \widetilde{B}, \tag{9}$$

$$\mathcal{D}(\sigma,\lambda):=\mathcal{E}^\phi(f_{\sigma,\lambda})-\mathcal{E}^\phi(f^\phi_\rho)+\lambda\|f_{\sigma,\lambda}\|^2_{\mathcal{H}_\sigma} \le \widetilde{B}(\sigma^s+\lambda\sigma^{-n}) \tag{10}$$

for $0 < \sigma \leq 1, \lambda > 0$*, where* $\widetilde{B} \geq 1$ *is a constant independent of* σ *or* λ.

Theorem 2 will be proved in Appendix A in a more general form as Theorem 6 where the constant $\widetilde{B}$ is given explicitly.

Remark 2 *A usual assumption in the literature (Zhang, 2004) for deriving learning rates is that for some* $0 < \beta \leq 1$ *and* $C_\beta > 0$,

$$\widetilde{\mathcal{D}}(\lambda) = \min_{f \in \mathcal{H}_\sigma} \{\mathcal{E}^\phi(f) - \mathcal{E}^\phi(f_\rho^\phi) + \lambda \|f\|^2_{\mathcal{H}_\sigma}\} \leq C_\beta \lambda^\beta \qquad \forall \lambda > 0. \tag{11}$$

This is hardly satisfied for a single fixed K^σ *due to the analyticity of the Gaussian kernel (Smale and Zhou, 2003; Cucker and Zhou, 2007). When we choose a changing Gaussian kernel with* $\sigma = \lambda^\zeta$ *for some* $\zeta > 0$*, decay (11) of the approximation error is valid in many cases (as shown in Theorem 2). Under assumption (11), one has the bound*

$$\|\widetilde{f}_{\sigma,\lambda}\|_{L^\infty(X)} \leq \|\widetilde{f}_{\sigma,\lambda}\|_{\mathcal{H}_\sigma} \leq \sqrt{\widetilde{\mathcal{D}}(\lambda)/\lambda} \leq \sqrt{C_\beta} \lambda^{\frac{\beta-1}{2}}.$$

Hence a natural bound for the random variable $\phi_{eh}(y\widetilde{f}_{\sigma,\lambda}(x))$ *would be* $\exp\{\sqrt{C_\beta}\lambda^{\frac{\beta-1}{2}}\}$ *which increases exponentially fast as* $\lambda \to 0$ *(polynomially fast with degree* $\frac{r(1-\beta)}{2}$ *for* $\phi = \phi_r$ *when r is very large). This shows difficulty in choosing* $\widetilde{f}_{\sigma,\lambda}$ *and demonstrates novelty in choosing the function* $f_{\sigma,\lambda}$ *from Theorem 2 for the error analysis with a general loss* ϕ.

When $\sigma = \lambda^\zeta$ for some $0 < \zeta < \frac{1}{n}$, (10) of Theorem 2 tells us that the function $f_{\sigma,\lambda}$ yields an approximation order similar to (11) while (9) ensures the uniform boundedness of $\phi(yf_{\sigma,\lambda}(x))$, better than the function $\widetilde{f}_{\sigma,\lambda}$ for the error decomposition described below.

2.2 Error Decomposition and Projection Operator

The excess misclassification error $\mathcal{R}(\text{sgn}(f)) - \mathcal{R}(f_c)$ for the classifier $\text{sgn}(f)$ can be bounded by means of the *excess generalization error* $\mathcal{E}^\phi(f)) - \mathcal{E}^\phi(f_\rho^\phi)$ according to some comparison theorems (Zhang, 2004; Chen et al., 2004; Bartlett et al., 2006). For example, it was proved in Zhang (2004) that for $\phi = \phi_h$ and any measurable function $f : X \to \mathbb{R}$, we have

$$\mathcal{R}(\text{sgn}(f)) - \mathcal{R}(f_c) \leq \mathcal{E}^{\phi_h}(f) - \mathcal{E}^{\phi_h}(f_c).$$

For a classifying loss ϕ with $\phi''(0) > 0$, it was proved in Chen et al. (2004) and Bartlett et al. (2006) that for some $c_\phi > 0$,

$$\mathcal{R}(\text{sgn}(f)) - \mathcal{R}(f_c) \leq c_\phi \sqrt{\mathcal{E}^\phi(f) - \mathcal{E}^\phi(f_\rho^\phi)}. \tag{12}$$

For the least square loss and ρ satisfying the Tsybakov noise condition, a comparison theorem improving (12) will be given in Section 4 and will be used to prove Theorem 1.

Classifiers in this paper are obtained by taking signs of real-valued functions. Since the smallest zero of ϕ is 1, we can take $f_\rho^\phi(x) \in [-1, 1]$ for each $x \in X$, which we shall assume throughout the paper. We may improve the error estimates (Chen et al., 2004) by replacing values of f by projections onto $[-1, 1]$.

Definition 3 *The projection operator π on the space of functions on X is defined by*

$$\pi(f)(x) = \begin{cases} 1 & \text{if } f(x) > 1, \\ -1 & \text{if } f(x) < -1, \\ f(x) & \text{if } -1 \le f(x) \le 1. \end{cases}$$

Trivially $\operatorname{sgn}(\pi(f)) = \operatorname{sgn}(f)$. Then we can use (12) with $f = \pi(f_{\mathbf{z}})$ to bound the excess misclassification error $\mathcal{R}(\operatorname{sgn}(f_{\mathbf{z}})) - \mathcal{R}(f_c)$ by means of the excess generalization error $\mathcal{E}^{\phi}(\pi(f_{\mathbf{z}})) - \mathcal{E}^{\phi}(f_{\rho}^{\phi})$ which in turn can be estimated by an error decomposition technique (Wu and Zhou, 2006). Define the *empirical error* associated with the loss ϕ as

$$\mathcal{E}_{\mathbf{z}}^{\phi}(f) = \frac{1}{m}\sum_{i=1}^{m} \phi(y_i f(x_i)) \qquad \text{for } f : X \to \mathbb{R}.$$

Then we have the following error decomposition which will be proved in Section 3.

Lemma 1 *Let ϕ be a classifying loss, $f_{\mathbf{z}}$ be defined by (2) and $f_{\sigma,\lambda} \in \mathcal{H}_{\sigma}$. Then*

$$\mathcal{E}^{\phi}(\pi(f_{\mathbf{z}})) - \mathcal{E}^{\phi}(f_{\rho}^{\phi}) \le \mathcal{D}(\sigma,\lambda) + \mathcal{S}_{\mathbf{z}}(f_{\sigma,\lambda}) - \mathcal{S}_{\mathbf{z}}(\pi(f_{\mathbf{z}})), \tag{13}$$

where the quantity $\mathcal{S}_{\mathbf{z}}(f)$ is defined for $f \in C(X)$ by

$$\mathcal{S}_{\mathbf{z}}(f) = [\mathcal{E}_{\mathbf{z}}^{\phi}(f) - \mathcal{E}_{\mathbf{z}}^{\phi}(f_{\rho}^{\phi})] - [\mathcal{E}^{\phi}(f) - \mathcal{E}^{\phi}(f_{\rho}^{\phi})].$$

When we use the regularizing function $f_{\sigma,\lambda}$ given in Theorem 2, the bound (10) deals with $\mathcal{D}(\sigma,\lambda)$, the first term of (13). The uniform bound (9) for $\|f_{\sigma,\lambda}\|_{L^{\infty}(X)}$ ensures that the second term $\mathcal{S}_{\mathbf{z}}(f_{\sigma,\lambda})$ of (13), which can be expressed as $\frac{1}{m}\sum_{i=1}^{m} \xi(z_i) - \mathbf{E}(\xi)$ with the random variable $\xi(z) = \phi(y f_{\sigma,\lambda}(x)) - \phi(y f_{\rho}^{\phi}(x))$, can be easily handled. The crucial remaining term $\mathcal{S}_{\mathbf{z}}(\pi(f_{\mathbf{z}}))$ of (13) involves the set of functions $\{f_{\mathbf{z}}\}_{\mathbf{z}\in Z^m}$ and can be treated by various empirical process techniques such as Rademacher average and entropy integral. Here we use the specialty of the Gaussians that the RKHS has low capacity, hence the last term of (13) can be estimated efficiently and simply by means of covering numbers.

2.3 Applying Tight Bounds for Covering Numbers

The second novelty of this paper is to make full use of the special low capacity property of the Gaussian kernels that a tight bound for covering numbers of the unit ball of the RKHS $\mathcal{H}_{\sigma}$ leads to nice estimates for the last term $\mathcal{S}_{\mathbf{z}}(\pi(f_{\mathbf{z}}))$ of (13) for the error analysis.

Definition 4 *For a subset S of $C(X)$ and $\eta > 0$, the covering number $\mathcal{N}(S,\eta)$ is the minimal integer $l \in \mathbb{N}$ such that there exist l disks with radius η covering S.*

The covering numbers of unit balls of classical function spaces have been well studied in the literature (Edmunds and Triebel, 1996). As an example, take $X = [0,1]^n$ and $s > 0$. The covering numbers of the unit ball $B_1(C^s(X))$ of the space $C^s(X)$ has the asymptotic behavior

$$c_s'\left(\frac{1}{\eta}\right)^{n/s} \le \log \mathcal{N}(B_1(C^s(X)),\eta) \le c_s''\left(\frac{1}{\eta}\right)^{n/s}, \tag{14}$$

where the positive constants c'_s, and c''_s are independent of $0 < \eta < 1$. In particular, since a Gaussian kernel K^σ is C^∞, an embedding result from Zhou (2003) tells us that $\log \mathcal{N}(B_1, \eta) \leq C''_s(\frac{1}{\eta})^{n/s}(\frac{1}{\sigma})^{2n}$ where $s > 0$ can be arbitrarily large but the constant C''_s depends on s. Here $B_1 = B_{1,\sigma} = \{f \in \mathcal{H}_\sigma : \|f\|_{\mathcal{H}_\sigma} \leq 1\}$ is the unit ball of $\mathcal{H}_\sigma$ and is regarded as a compact subset of $C(X)$. A crucial improved bound for the covering number of B_1 was given in Zhou (2002) with $(\frac{1}{\eta})^{n/s}$ replaced by $(\log \frac{1}{\eta})^{n+1}$ as follows.

Proposition 1 *There exists a constant* $C_0 > 0$ *depending only on* X *and* n *such that*

$$\log \mathcal{N}(B_1, \eta) \leq C_0 \Big((\log \frac{1}{\eta})^{n+1} + \frac{1}{\sigma^{2(n+1)}} \Big) \qquad \forall\, 0 < \eta < 1, 0 < \sigma \leq 1. \tag{15}$$

The constant C_0 can be taken as $(124n)^{n+2}$ when $X = [0,1]^n$. Bound (15) is almost sharp in the sense that for some $C'_0 > 0$ given in Zhou (2003),

$$\log \mathcal{N}(B_1, \eta) \geq C'_0 \Big((\log \frac{1}{\eta})^{n/2} + \frac{1}{\sigma^n} \Big).$$

The logarithmic term $(\log \frac{1}{\eta})^{n+1}$ appearing in the tight bound (15) is better than the polynomial term $(\frac{1}{\eta})^{n/s}$ in (14). This enables us to derive efficient error bounds for the algorithm (2) involving Gaussian kernels, by a simple covering number argument without other empirical process techniques or iteration techniques used in Steinwart and Scovel (2007) and Wu et al. (2007). To demonstrate explicitly why tight bound (15) helps, we state the following result which is needed for estimating confidence and will be proved in Appendix B.

Lemma 2 *Let* $0 \leq \tau \leq 1$ *and* $C_1 > 0$. *Let* $0 < \delta < 1$ *and* λ, σ *take form (3) with some* $\gamma > 0$ *and* $0 < \zeta < \frac{1}{2\gamma(n+1)}$. *Denote* $\varepsilon^*(m, \lambda, \sigma, \delta/2)$ *as the smallest positive number* ε *satisfying*

$$1 - \mathcal{N}\Big(B_1, \frac{\lambda \varepsilon}{\sqrt{\phi(0)}|\phi'_+(-1)|}\Big) \exp\left\{ -\frac{m\varepsilon^{2-\tau}}{2C_1 + \frac{2}{3}\phi(-1)\varepsilon^{1-\tau}} \right\} \geq 1 - \frac{\delta}{2}. \tag{16}$$

Then we have

$$\varepsilon^*(m, \lambda, \sigma, \delta/2) \leq C_2 m^{-\frac{1-2\gamma\zeta(n+1)}{2-\tau}} \log \frac{2}{\delta}, \tag{17}$$

where C_2 *is the constant independent of* m, λ, σ *or* δ.

2.4 Key Bounds

We are in a position to present our key bounds for the excess generalization error $\mathcal{E}^\phi(\pi(f_{\mathbf{z}})) - \mathcal{E}^\phi(f^\phi_\rho)$ which will be used to get rates for the excess misclassification error $\mathcal{R}(\mathrm{sgn}(f_{\mathbf{z}})) - \mathcal{R}(f_c)$. To achieve tight bounds, we need the following definition.

Definition 5 *A variancing power* $\tau = \tau_{\phi,\rho}$ *of the pair* (ϕ, ρ) *is a number* τ *in* $[0,1]$ *such that for any* $\widetilde{B} \geq 1$, *there exists some constant* $C_1 = C_1(\widetilde{B}) > 0$ *satisfying*

$$\mathbf{E}\Big\{ \Big[\phi(yf(x)) - \phi(yf^\phi_\rho(x))\Big]^2 \Big\} \leq C_1 \Big[\mathcal{E}^\phi(f) - \mathcal{E}^\phi(f^\phi_\rho)\Big]^\tau \qquad \forall\, f : X \to [-\widetilde{B}, \widetilde{B}]. \tag{18}$$

Remark 3 *For $\phi = \phi_{ls}$, we can take $\tau = 1$, see Evgeniou et al. (2000) and Cucker and Zhou (2007). For $\phi = \phi_h$, we can take $\tau = 0$, and an improved power $\tau = \frac{q}{q+1}$ if the Tsybakov noise condition (5) is satisfied (Steinwart and Scovel, 2007; Wu and Zhou, 2005). In general, $\tau_{\phi,\rho}$ depends on the strong convexity of ϕ and noise conditions for ρ.*

Theorem 3 *Let $\sigma = \lambda^{\zeta}$ and $\lambda = m^{-\gamma}$ for some $0 < \zeta < \frac{1}{n}$ and $0 < \gamma < \frac{1}{2\zeta(n+1)}$. If (8) is valid for some $s > 0$, then for any $0 < \delta < 1$, with confidence $1 - \delta$ we have*

$$\mathcal{E}^{\phi}(\pi(f_{\mathbf{z}})) - \mathcal{E}^{\phi}(f_{\rho}^{\phi}) \leq \widetilde{C} m^{-\theta} \log \frac{2}{\delta} \tag{19}$$

where

$$\theta = \min\left\{ s\zeta\gamma, \gamma(1-n\zeta), \frac{1-2\gamma\zeta(n+1)}{2-\tau} \right\}, \tag{20}$$

and $\widetilde{C}$ is a constant independent of m and δ.

Theorem 3 will be proved in the next section and the constant $\widetilde{C}$ will be given explicitly.

3. Error Analysis

In this section we derive the key error bounds stated in Theorem 3 by estimating the right-hand side of (13) in Lemma 1 (which is proved here).

3.1 Proof of Lemma 1

Write the regularized excess generalization error as

$$\begin{aligned} \mathcal{E}^{\phi}(\pi(f_{\mathbf{z}})) - \mathcal{E}^{\phi}(f_{\rho}^{\phi}) + \lambda \|f_{\mathbf{z}}\|_{\mathcal{H}_{\sigma}}^2 &= \left\{ \mathcal{E}^{\phi}(\pi(f_{\mathbf{z}})) - \mathcal{E}_{\mathbf{z}}^{\phi}(\pi(f_{\mathbf{z}})) \right\} \\ &+ \left\{ \left[\mathcal{E}_{\mathbf{z}}^{\phi}(\pi(f_{\mathbf{z}})) + \lambda \|f_{\mathbf{z}}\|_{\mathcal{H}_{\sigma}}^2 \right] - \left[\mathcal{E}_{\mathbf{z}}^{\phi}(f_{\sigma,\lambda}) + \lambda \|f_{\sigma,\lambda}\|_{\mathcal{H}_{\sigma}}^2 \right] \right\} \\ &+ \left\{ \mathcal{E}_{\mathbf{z}}^{\phi}(f_{\sigma,\lambda}) - \mathcal{E}^{\phi}(f_{\sigma,\lambda}) \right\} + \left\{ \mathcal{E}^{\phi}(f_{\sigma,\lambda}) - \mathcal{E}^{\phi}(f_{\rho}^{\phi}) + \lambda \|f_{\sigma,\lambda}\|_{\mathcal{H}_{\sigma}}^2 \right\}. \end{aligned}$$

Since ϕ is convex and its smallest zero is 1, we find a special property of the projection operator that $\phi(y\pi(f)(x)) \leq \phi(yf(x))$ for any function f on X. Hence $\mathcal{E}_{\mathbf{z}}^{\phi}(\pi(f)) \leq \mathcal{E}_{\mathbf{z}}^{\phi}(f)$. This in connection with the definition of $f_{\mathbf{z}}$ tells us that the second term on the right-hand side above is at most zero. By subtracting and adding $\mathcal{E}^{\phi}(f_{\rho}^{\phi})$ in the first and third terms we see $\mathcal{E}^{\phi}(\pi(f_{\mathbf{z}})) - \mathcal{E}^{\phi}(f_{\rho}^{\phi})$ is bounded as in (13). This proves Lemma 1. ■

Let us turn to estimate $\mathcal{E}^{\phi}(\pi(f_{\mathbf{z}})) - \mathcal{E}^{\phi}(f_{\rho}^{\phi})$ by (13). We first bound $\mathcal{S}_{\mathbf{z}}(f_{\sigma,\lambda})$, the term involving $f_{\sigma,\lambda}$. It can be written as $\frac{1}{m}\sum_{i=1}^{m} \xi(z_i) - \mathbf{E}(\xi)$ with ξ the random variable on (Z,ρ) given by $\xi(z) = \phi(yf_{\sigma,\lambda}(x)) - \phi(yf_{\rho}^{\phi}(x))$.

Lemma 3 *Let $\tau = \tau_{\phi,\rho}$ and $f_{\sigma,\lambda} \in \mathcal{H}_{\sigma}$ satisfy (9). For any $0 < \delta < 1$, with confidence $1 - \frac{\delta}{2}$, the term $\mathcal{S}_{\mathbf{z}}(f_{\sigma,\lambda})$ of (13) can be bounded as*

$$\mathcal{S}_{\mathbf{z}}(f_{\sigma,\lambda}) \leq 2(\|\phi\|_{C[-\widetilde{B},\widetilde{B}]} + C_1) \log \frac{2}{\delta} m^{-\frac{1}{2-\tau}} + \mathcal{E}^{\phi}(f_{\sigma,\lambda}) - \mathcal{E}^{\phi}(f_{\rho}^{\phi}).$$

Proof Consider the random variable $\xi(z) = \phi(yf_{\sigma,\lambda}(x)) - \phi(yf_\rho^\phi(x))$ on (Z,ρ). It satisfies $-\phi(-1) \le \xi \le \|\phi\|_{C[-\widetilde{B},\widetilde{B}]}$. Hence $|\xi - \mathbf{E}(\xi)| \le 2\|\phi\|_{C[-\widetilde{B},\widetilde{B}]}$. We apply the one side Bernstein inequality and know that

$$\text{Prob}_{\mathbf{z}\in Z^m}\Big\{\frac{1}{m}\sum_{i=1}^m \xi(z_i) - \mathbf{E}(\xi) > \varepsilon\Big\} \le \exp\Big\{-\frac{m\varepsilon^2}{2(\sigma^2(\xi) + \frac{2}{3}\|\phi\|_{C[-\widetilde{B},\widetilde{B}]}\varepsilon)}\Big\} \qquad \forall \varepsilon > 0.$$

Here $\sigma^2(\xi)$ is the variance of ξ. Solving the quadratic equation for ε by setting the above probability bound to be $\delta/2$, we see that with confidence at least $1-\delta/2$,

$$\frac{1}{m}\sum_{i=1}^m \xi(z_i) - \mathbf{E}(\xi) \le \frac{4\|\phi\|_{C[-\widetilde{B},\widetilde{B}]}\log\frac{2}{\delta}}{3m} + \frac{\sqrt{2m\sigma^2(\xi)\log\frac{2}{\delta}}}{m}.$$

Using (18) involving the variancing power $\tau = \tau_{\phi,\rho}$ in Definition 5, we have $\sigma^2(\xi) \le \mathbf{E}(\xi^2) \le C_1(\mathbf{E}(\xi))^\tau$. This in connection with Young's inequality implies

$$\frac{\sqrt{2m\sigma^2(\xi)\log\frac{2}{\delta}}}{m} \le \sqrt{\frac{2\log\frac{2}{\delta}C_1(\mathbf{E}(\xi))^\tau}{m}} \le (1-\frac{\tau}{2})\Big(\frac{2\log\frac{2}{\delta}C_1}{m}\Big)^{\frac{1}{2-\tau}} + \frac{\tau}{2}\mathbf{E}(\xi).$$

Therefore, with confidence at least $1-\delta/2$,

$$\frac{1}{m}\sum_{i=1}^m \xi(z_i) - \mathbf{E}(\xi) \le \frac{4\|\phi\|_{C[-\widetilde{B},\widetilde{B}]}\log\frac{2}{\delta}}{3m} + \Big(\frac{2\log\frac{2}{\delta}C_1}{m}\Big)^{\frac{1}{2-\tau}} + \mathbf{E}(\xi).$$

Since $\mathbf{E}(\xi) = \mathcal{E}^\phi(f_{\sigma,\lambda}) - \mathcal{E}^\phi(f_\rho^\phi)$, our conclusion follows. ∎

The sample error term $-\mathcal{S}_\mathbf{z}(\pi(f_\mathbf{z}))$ in (13) can be expressed as $\int \xi_\mathbf{z} d\rho - \frac{1}{m}\sum_{i=1}^m \xi_\mathbf{z}(z_i)$ with $\xi_\mathbf{z}(z) = \phi(yf_\mathbf{z}(x)) - \phi(yf_\rho^\phi(x))$. However, $\xi_\mathbf{z}$ is not a single random variable since $\mathbf{z}$ is a random sample itself. This is the essential difficulty. Here we use the specialty of low capacity of the RKHS $\mathcal{H}_\sigma$ and overcome the difficulty by a simple covering number argument over a ball of $\mathcal{H}_\sigma$ where $f_\mathbf{z}$ lies.

Lemma 4 *For any $\lambda > 0$ and $\mathbf{z} \in Z^m$, there holds*

$$\|f_\mathbf{z}\|_{\mathcal{H}_\sigma} \le \sqrt{\phi(0)/\lambda}.$$

The proof follows easily by taking $f = 0$ in the definition of $f_\mathbf{z}$ as in De Vito et al. (2005), De Vito et al. (2004) and Hardin et al. (2004).

Let ξ be a random variable on Z with mean $\mu \ge 0$ and variance $\sigma^2 \le c\mu^\tau$ for some $0 \le \tau \le 2$ and $c \ge 0$. If $|\xi - \mu| \le B$ almost surely for some $B \ge 0$, then the one-side Bernstein inequality implies

$$\text{Prob}_{\mathbf{z}\in Z^m}\Big\{\frac{\mu - \frac{1}{m}\sum_{i=1}^m \xi(z_i)}{\sqrt{\mu^\tau + \varepsilon^\tau}} > \varepsilon^{1-\frac{\tau}{2}}\Big\} \le \exp\Big\{-\frac{m\varepsilon^{2-\tau}}{2(c + \frac{1}{3}B\varepsilon^{1-\tau})}\Big\} \qquad \forall\ \varepsilon > 0.$$

Applying this probability inequality to random variables of type $\xi(z) = \phi(y(\pi f)(x)) - \phi(yf_\rho^\phi(x))$ and using a standard argument (Wu et al., 2007; Yao, 2008; Ying, 2007) with covering numbers for the ball $\{f \in \mathcal{H}_\sigma : \|f\|_{\mathcal{H}_\sigma} \le \sqrt{\phi(0)/\lambda}\}$ of the RKHS $\mathcal{H}_\sigma$, we find the following bound.

Lemma 5 *Let* $\tau = \tau_{\phi,\rho}$ *satisfy (18) with* $\widetilde{B}$ *being* 1. *For any* $\varepsilon > 0$, *we have*

$$\mathrm{Prob}_{\mathbf{z}\in Z^m}\Big\{\sup_{\|f\|_{\mathcal{H}_\sigma}\le\sqrt{\phi(0)/\lambda}}\frac{[\mathcal{E}^\phi(\pi(f))-\mathcal{E}^\phi(f_\rho^\phi)]-[\mathcal{E}_{\mathbf{z}}^\phi(\pi(f))-\mathcal{E}_{\mathbf{z}}^\phi(f_\rho^\phi)]}{\sqrt{(\mathcal{E}^\phi(\pi(f))-\mathcal{E}^\phi(f_\rho^\phi))^\tau+\varepsilon^\tau}}\le 4\varepsilon^{1-\frac{\tau}{2}}\Big\}$$
$$\ge 1-\mathcal{N}\Big(B_1,\frac{\sqrt{\lambda}\varepsilon}{\sqrt{\phi(0)}|\phi'_+(-1)|}\Big)\exp\Big\{-\frac{m\varepsilon^{2-\tau}}{2C_1+\frac{2}{3}\phi(-1)\varepsilon^{1-\tau}}\Big\}.$$

Recall the definition of $\varepsilon^*(m,\lambda,\sigma,\delta/2)$ in Lemma 2. It satisfies (16) which means that the probability in Lemma 5 is bounded by $1-\frac{\delta}{2}$ from below when $\varepsilon = \varepsilon^*(m,\lambda,\sigma,\delta/2)$.

Proposition 2 *Let* $f_{\sigma,\lambda}\in\mathcal{H}_\sigma$ *satisfy (9). For any* $0<\delta<1$, *with confidence at least* $1-\delta$, *we have*

$$\mathcal{E}^\phi(\pi(f_{\mathbf{z}}))-\mathcal{E}^\phi(f_\rho^\phi)\le 4\mathcal{D}(\sigma,\lambda)+40\varepsilon^*(m,\lambda,\sigma,\delta/2)+4(\|\phi\|_{C[-\widetilde{B},\widetilde{B}]}+C_1)\log\frac{2}{\delta}m^{-\frac{1}{2-\tau}}.$$

Proof Applying Lemma 3, we know that there is a subset V_1 of Z^m with measure at least $1-\frac{\delta}{2}$ such that for $\mathbf{z}\in V_1$,

$$\mathcal{S}_{\mathbf{z}}(f_{\sigma,\lambda})\le 2(\|\phi\|_{C[-\widetilde{B},\widetilde{B}]}+C_1)\log\frac{2}{\delta}m^{-\frac{1}{2-\tau}}+\mathcal{D}(\sigma,\lambda).$$

By Lemma 5 and Lemma 4, taking $\varepsilon=\varepsilon^*(m,\lambda,\sigma,\delta/2)$, we see that there exists another subset V_2 of Z^m with measure at least $1-\frac{\delta}{2}$ such that for $\mathbf{z}\in V_2$,

$$\begin{aligned}
-\mathcal{S}_{\mathbf{z}}(\pi(f_{\mathbf{z}})) &= [\mathcal{E}^\phi(\pi(f_{\mathbf{z}}))-\mathcal{E}^\phi(f_\rho^\phi)]-[\mathcal{E}_{\mathbf{z}}^\phi(\pi(f_{\mathbf{z}}))-\mathcal{E}_{\mathbf{z}}^\phi(f_\rho^\phi)]\\
&\le 4[\varepsilon^*(m,\lambda,\sigma,\delta/2)]^{1-\frac{\tau}{2}}\sqrt{\Big[\mathcal{E}^\phi(\pi(f_{\mathbf{z}}))-\mathcal{E}^\phi(f_\rho^\phi)\Big]^\tau+[\varepsilon^*(m,\lambda,\sigma,\delta/2)]^\tau}\\
&\le \Big(1-\frac{\tau}{2}\Big)4^{\frac{2}{2-\tau}}\varepsilon^*(m,\lambda,\sigma,\delta/2)+\frac{\tau}{2}[\mathcal{E}^\phi(\pi(f_{\mathbf{z}}))-\mathcal{E}^\phi(f_\rho^\phi)]+4\varepsilon^*(m,\lambda,\sigma,\delta/2).
\end{aligned}$$

Here we have used the elementary inequality $\sqrt{a+b}\le\sqrt{a}+\sqrt{b}$ and Young's inequality.

Adding the above two bounds and observing that $0\le\tau\le 1$ implies $\frac{1}{1-\tau/2}\le 2$ we know from Lemma 1 that for $\mathbf{z}\in V_1\cap V_2$,

$$\mathcal{E}^\phi(\pi(f_{\mathbf{z}}))-\mathcal{E}^\phi(f_\rho^\phi)\le 4\mathcal{D}(\sigma,\lambda)+40\varepsilon^*(m,\lambda,\sigma,\delta/2)+4(\|\phi\|_{C[-\widetilde{B},\widetilde{B}]}+C_1)\log\frac{2}{\delta}m^{-\frac{1}{2-\tau}}.$$

Since the measure of $V_1\cap V_2$ is at least $1-\delta$, our conclusion holds true. ∎

Now we are in a position to prove Theorem 3.

3.2 Proof of Theorem 3

From condition (8) and the parameter form $\sigma=\lambda^\zeta$ with $0<\zeta<\frac{1}{n}$, we know by Theorem 2 that

$$\mathcal{D}(\sigma,\lambda)\le\widetilde{B}\lambda^{\min\{s\zeta,1-n\zeta\}}.$$

Putting bound (17) for $\varepsilon^*(m,\lambda,\sigma,\delta/2)$ from Lemma 2 into Proposition 2, we see from the parameter form $\lambda = m^{-\gamma}$ that with confidence at least $1-\delta$,

$$\begin{aligned}\mathcal{E}^\phi(\pi(f_{\mathbf{z}})) - \mathcal{E}^\phi(f_\rho^\phi) &\leq 4\widetilde{B}\lambda^{\min\{s\zeta,1-n\zeta\}} + 40C_2 m^{-\frac{1-2\gamma\zeta(n+1)}{2-\tau}}\log\frac{2}{\delta} \\ &\quad +4(\|\phi\|_{C[-\widetilde{B},\widetilde{B}]}+C_1)\log\frac{2}{\delta}m^{-\frac{1}{2-\tau}} \leq \widetilde{C}m^{-\theta}\log\frac{2}{\delta}.\end{aligned}$$

Here θ is given by (20) and $\widetilde{C}$ is the constant independent of m and δ given by

$$\widetilde{C} = 4\widetilde{B} + 40C_2 + 4(\|\phi\|_{C[-\widetilde{B},\widetilde{B}]} + C_1).$$

This proves (19) and hence Theorem 3. ■

4. Deriving Learning Rates

In this section we apply Theorem 3 to derive learning rates with various loss functions. For the least square loss, to prove Theorem 1 we need the following comparison theorem improving (12).

Proposition 3 *If $\phi = \phi_{ls}$ and ρ satisfies noise condition (5) for some $q \in [0,\infty]$, then for every measurable function $f : X \to \mathbb{R}$, we have*

$$\mathcal{R}(sgn(f)) - \mathcal{R}(f_c) \leq 2C_q^{-\frac{q}{q+2}}\left\{\mathcal{E}^{\phi_{ls}}(f) - \mathcal{E}^{\phi_{ls}}(f_\rho)\right\}^{\frac{q+1}{q+2}}.$$

Proof Denote $X_f = \{x \in X : \mathrm{sgn}(f)(x) \neq f_c(x)\}$. It is known that $\mathcal{R}(\mathrm{sgn}(f)) - \mathcal{R}(f_c) = \int_{X_f}|f_\rho(x)|d\rho_X$. See, for example, Equation (9.14) of Cucker and Zhou (2007).

When $q < \infty$, take $t = \left(\|f-f_\rho\|_{L^2_{\rho_X}}/C_q\right)^{\frac{2}{q+2}} > 0$. We separate the set X_f into two parts, one with $|f_\rho(x)| \leq C_q t$ and the other with $|f_\rho(x)| > C_q t$ where $|f_\rho(x)| \leq |f_\rho(x)|^2/(C_q t) \leq |f(x) - f_\rho(x)|^2/(C_q t)$. We find from (5) that

$$\begin{aligned}\int_{X_f}|f_\rho(x)|d\rho_X &\leq \int_{\{x\in X_f:\ |f_\rho(x)|\leq C_q t\}} C_q t d\rho_X + \int_{\{x\in X_f:\ |f_\rho(x)|>C_q t\}}|f(x)-f_\rho(x)|^2/(C_q t)d\rho_X \\ &\leq C_q t\rho_X(\{x\in X : |f_\rho(x)|\leq C_q t\}) + \|f-f_\rho\|^2_{L^2_{\rho_X}}/(C_q t) \\ &\leq C_q t^{q+1} + \|f-f_\rho\|^2_{L^2_{\rho_X}}/(C_q t) = 2C_q\left(\|f-f_\rho\|_{L^2_{\rho_X}}/C_q\right)^{\frac{2q+2}{q+2}}.\end{aligned}$$

This gives the desired bound for the case $q<\infty$ since $\|f-f_\rho\|^2_{L^2_{\rho_X}} = \mathcal{E}^{\phi_{ls}}(f) - \mathcal{E}^{\phi_{ls}}(f_\rho)$.

When $q = \infty$, noise condition (5) means $|f_\rho(x)| \geq C_q$ and hence $|f_\rho(x)| \leq |f_\rho(x)|^2/C_q$ for almost every $x \in (X,\rho_X)$. So $\int_{X_f}|f_\rho(x)|d\rho_X \leq \int_{X_f}|f(x)-f_\rho(x)|^2/C_q d\rho_X = \|f-f_\rho\|^2_{L^2_{\rho_X}}/C_q$ which is what we want. ■

Now we can derive learning rates with the least square loss.

4.1 Proof of Theorem 1

The assumptions on $\frac{d\rho_X}{dx}$ and f_ρ verify condition (8). Then by Theorem 2 with $\phi = \phi_{ls}$, we find functions $f_{\sigma,\lambda}$ satisfying (9) and (10) for $0 < \sigma \leq 1, \lambda > 0$.

The choice $\sigma = \lambda^\zeta$ with $0 < \zeta \leq \frac{1}{n+s} < \frac{1}{n}$ and $\lambda = m^{-\gamma}$ with $\gamma = \frac{1}{\zeta(s+2n+2)}$ tell us that $0 < \gamma < \frac{1}{2\zeta(n+1)}$. Therefore all conditions of Theorem 3 are valid. Moreover, a specialty of the least square loss is $\tau = 1$ in (18). So by Theorem 3, for any $0 < \delta < 1$, with confidence $1-\delta$, (19) holds with

$$\theta = \min\left\{\frac{s}{s+2n+2}, \frac{1-n\zeta}{\zeta(s+2n+2)}, 1 - \frac{2(n+1)}{s+2n+2}\right\} = \frac{s}{s+2n+2}.$$

This bound for the excess generalization error $\mathcal{E}^\phi(\pi(f_{\mathbf{z}})) - \mathcal{E}^\phi(f_\rho^\phi)$ together with Proposition 3 yields the desired bound (6) for the excess misclassification error $\mathcal{R}(\text{sgn}(f_{\mathbf{z}})) - \mathcal{R}(f_c)$ with the constant $\widetilde{C}_{\rho,s,q,n} = 2C_q^{-\frac{q}{q+2}} \widetilde{C}^{\frac{q+1}{q+2}}$. The proof of Theorem 1 is complete. ■

Let us derive learning rates with the r-norm SVM loss $\phi = \phi_r$ $(1 < r < \infty)$ for which we have (Chen et al., 2004)

$$f_\rho^\phi(x) = f_\rho^{\phi_r}(x) = \frac{(1+f_\rho(x))^{1/(r-1)} - (1-f_\rho(x))^{1/(r-1)}}{(1+f_\rho(x))^{1/(r-1)} + (1-f_\rho(x))^{1/(r-1)}}, \qquad x \in X. \tag{21}$$

Theorem 4 *Let $\phi = \phi_r$ with $1 < r < \infty$. Assume (8) for some $s > 0$. Take $\sigma = \lambda^\zeta$ with $0 < \zeta \leq \frac{1}{n+s}$ and $\lambda = m^{-\gamma}$ with $\gamma = \frac{1}{\zeta(s+2n+2)}$ for $1 < r \leq 2$ and $\gamma = \frac{1}{\zeta(2s(1-1/r)+2n+2)}$ for $2 < r < \infty$. Then for any $0 < \delta < 1$, with confidence $1-\delta$, we have*

$$\mathcal{R}(sgn(f_{\mathbf{z}})) - \mathcal{R}(f_c) \leq \widetilde{C}_{\rho,r} m^{-\theta_r} \log\frac{2}{\delta} \quad \textit{with } \theta_r = \begin{cases} \frac{s}{2(s+2n+2)}, & \textit{if } 1 < r \leq 2, \\ \frac{s}{4(s(1-1/r)+n+1)}, & \textit{if } 2 < r < \infty. \end{cases} \tag{22}$$

Proof The convexity of ϕ_r gives the variancing power (Bartlett et al., 2006) as

$$\tau = \tau_{\phi_r,\rho} = \begin{cases} 1, & \text{if } 1 < r \leq 2, \\ \frac{2}{r}, & \text{if } 2 < r < \infty. \end{cases}$$

Take $\sigma = \lambda^\zeta$ with $0 < \zeta \leq \frac{1}{n+s} < \frac{1}{n}$ and choose $\lambda = m^{-\gamma}$ with $\gamma = \frac{1}{\zeta((2-\tau)s+2n+2)}$. We see that $0 < \gamma < \frac{1}{2\zeta(n+1)}$. Hence all conditions of Theorem 3 are valid and we conclude that for any $0 < \delta < 1$, with confidence $1-\delta$, (19) holds with $\theta = \frac{s}{(2-\tau)s+2n+2}$. This bound for the excess generalization error together with comparison relation (12) caused by $\phi_r''(0) = r(r-1) > 0$ yields the desired bound (22) for the excess misclassification error with the constant $\widetilde{C}_{\rho,r} = c_{\phi_r}\sqrt{\widetilde{C}}$. The proof of Theorem 4 is complete. ■

When $\phi = \phi_{eh}$, a simple computation shows that the function f_ρ^ϕ is given by

$$f_\rho^{\phi_{eh}}(x) = \begin{cases} \frac{1}{2}\log\frac{1+f_\rho(x)}{1-f_\rho(x)}, & \text{if } -(e^2-1)/(e^2+1) \leq f_\rho(x) \leq (e^2-1)/(e^2+1), \\ 1, & \text{if } f_\rho(x) > (e^2-1)/(e^2+1), \\ -1, & \text{if } f_\rho(x) < -(e^2-1)/(e^2+1). \end{cases} \tag{23}$$

Theorem 5 *Let $\phi = \phi_{eh}$. Assume (8) for some $s > 0$. Take $\sigma = \lambda^{\zeta}$ with $0 < \zeta \leq \frac{1}{n+s}$ and $\lambda = m^{-\frac{1}{\zeta(2s+2n+2)}}$. Then for any $0 < \delta < 1$, with confidence $1 - \delta$, we have*

$$\mathcal{R}(sgn(f_{\mathbf{z}})) - \mathcal{R}(f_c) \leq \widetilde{C}_{\rho,eh} m^{-\theta_{eh}} \log \frac{2}{\delta} \qquad with \quad \theta_{eh} = \frac{s}{4s+4n+4}. \tag{24}$$

Proof Take $\tau = 0$ and $\sigma = \lambda^{\zeta}$ with $0 < \zeta \leq \frac{1}{n+s} < \frac{1}{n}$. Choose $\lambda = m^{-\gamma}$ with $\gamma = \frac{1}{\zeta(2s+2n+2)}$ in Theorem 3. We see that for any $0 < \delta < 1$, with confidence $1 - \delta$, (19) holds with $\theta = \frac{s}{2s+2n+2}$. This bound together with comparison relation (12) again (as $\phi''_{eh}(0) = e > 0$) yields the desired bound (24) with $\widetilde{C}_{\rho,eh} = c_{\phi_{eh}} \sqrt{\widetilde{C}}$. This proves Theorem 5. ∎

Remark 4 *When $s \leq \frac{1}{r-1}$, the extension condition of f_ρ stated in Theorem 1 implies assumption (8) of $f_\rho^{\phi_r}$ required in Theorem 4. In fact, the extension of the function $f_\rho^{\phi_r}$ onto $\mathbb{R}^n$ can be defined by taking values of the extended function of f_ρ in (21). After composing with the function $t \to t^{1/(r-1)}$ on $\mathbb{R}$, smoothness of functions in the Sobolev space H^s is kept for $s \leq \frac{1}{r-1}$. When $s \leq 1$, the same condition for f_ρ implies assumption (8) of $f_\rho^{\phi_{eh}}$ needed for Theorem 5, as seen from expression (23).*

It is possible to refine learning rates (22) and (24) by improving comparison relation (12) when Tsybakov noise condition (5) is satisfied. We omit the discussion here.

Error analysis with $\phi = \phi_r$ was done in Chen et al. (2004) under assumption (11). Our learning rates in Theorem 4 are new since our assumption on Sobolev smoothness is weaker. The learning rates for $\phi = \phi_{eh}$ in Theorem 5 are also new.

We are in a position to get from Theorems 1, 4 and 5 some theoretical clues on the effect of various loss functions for learning algorithm (2). We know from Smale and Zhou (2003) that when $\phi = \phi_{ls}$ the approximation error and hence learning rates can essentially be characterized by regularities of the function f_ρ. So here we give some comparisons under the same regularity assumption (8) for the function f_ρ^{ϕ} with some $s > 0$. Under this assumption (removing the Tsybakov noise condition by taking $q = 0$ in Theorem 1), the learning rates derived in Theorems 1, 4 and 5 for $\phi = \phi_{ls}, \phi_r, \phi_{eh}$ take the same form $\mathcal{R}(\text{sgn}(f_{\mathbf{z}})) - \mathcal{R}(f_c) = O(m^{-\theta} \log \frac{2}{\delta})$ with the power index θ very close, all lying in the range $[\frac{s}{4s+4n+4}, \frac{s}{2s+4n+4}]$. However, the index s in regularity assumption (8) for the function f_ρ^{ϕ} might vary dramatically, leading to varying power index θ for the learning rates.

Note that the function f_ρ^{ϕ} with $\phi = \phi_r, \phi_{eh}$ depends explicitly on the regression function f_ρ corresponding the least-square loss. The dependence of the function $f_\rho^{\phi_{eh}}$ on f_ρ has an advantage of ignoring any irregularity appearing in the domain where $|f_\rho(x)| > (e^2 - 1)/(e^2 + 1)$. This can be seen from the following example where f_ρ has a singularity at 0 while $f_\rho^{\phi_{eh}} \equiv 1$ is C^{∞}.

Example 1 *Let $X = [-1, 1]$, $0 < \alpha < \frac{1}{14}$ and ρ be the distribution given by $d\rho_X = \frac{1}{2}dx$ and $f_\rho(x) = 1 - \frac{1}{5}|x|^{\alpha}$ which means $P(y = 1|x) = 1 - \frac{1}{10}|x|^{\alpha}$. It is well known that the function $|x|^{\alpha}$ lies in the Sobolev space $H^s(X)$ if and only if $s < \alpha + \frac{1}{2}$. So regularity assumption (8) is satisfied for ϕ_{ls} if and only if $s < \alpha + \frac{1}{2}$. Then from Theorem 1, we see the learning rate $\mathcal{R}(sgn(f_{\mathbf{z}})) - \mathcal{R}(f_c) = O(m^{-\theta_{ls}} \log \frac{2}{\delta})$ with $\theta_{ls} = \frac{s}{s+2+2}$ arbitrarily close to $\frac{1+2\alpha}{9+2\alpha} < \frac{1}{8}$. However, for the exponential-hinge loss ϕ_{eh}, we have $f_\rho^{\phi_{eh}} \equiv 1$ which follows from expression (23) and the definition $f_\rho(x) = 1 - \frac{1}{5}|x|^{\alpha} \geq$*

$1-\frac{1}{5} > (e^2-1)/(e^2+1)$ on X. Therefore, regularity assumption (8) is satisfied for an arbitrarily large s and Theorem 5 yields the learning rate $\mathcal{R}(sgn(f_{\mathbf{z}})) - \mathcal{R}(f_c) = O(m^{-\theta_{eh}} \log \frac{2}{\delta})$ with θ_{eh} arbitrarily close to $\frac{1}{4}$. Thus for learning algorithm (2), the exponential-hinge loss has some advantages over ϕ_{ls} (and ϕ_r as shown in the next example).

The dependence of the function $f_\rho^{\phi_r}$ on f_ρ involves a power function $u \to u^{1/(r-1)}$ which might cause irregularity. This is demonstrated by the following example where the singularity of the function f_ρ at 0 is worsened for the function $f_\rho^{\phi_r}$ when r is large.

Example 2 *Let X and ρ be as in Example 1. When $\phi = \phi_r$ with $r > 2$, the function $f_\rho^{\phi_r}$ in (21) equals*

$$f_\rho^{\phi_r}(x) = \frac{(2-\frac{1}{5}|x|^\alpha)^{1/(r-1)} - (\frac{1}{5}|x|^\alpha)^{1/(r-1)}}{(2-\frac{1}{5}|x|^\alpha)^{1/(r-1)} + (\frac{1}{5}|x|^\alpha)^{1/(r-1)}}.$$

Regularity assumption (8) is satisfied for ϕ_r if and only if $s < \frac{\alpha}{r-1} + \frac{1}{2}$. Then Theorem 4 yields the learning rate $\mathcal{R}(sgn(f_{\mathbf{z}})) - \mathcal{R}(f_c) = O(m^{-\theta_r} \log \frac{2}{\delta})$ with $\theta_r = \frac{s}{4(s(1-1/r)+1+1)}$ arbitrarily close to $\frac{r(2\alpha+r-1)}{4(5r+2\alpha-1)(r-1)}$. This power index is always less than that of ϕ_{ls} or ϕ_{eh}. It shows that the loss ϕ_r has worse performance than ϕ_{ls} and ϕ_{eh}, at least for some distributions.

5. Further Discussion

Let us discuss further generalizations and connections briefly here. More details will be provided in our future study.

The first extension is to a manifold setting. If X is a connected compact C^∞ submanifold of $\mathbb{R}^n$ without boundary and its dimension is $d \le n$, then the covering number estimate (15) holds with n replaced by the manifold dimension d. Proposition 2 and Lemma 2 are still valid with n replaced by d. Learning rates in Theorems 1, 4 and 5 can be improved with n replaced by d if approximation error estimates similar to Theorem 2 can be established in the manifold setting. One can use ideas for convolution type approximation schemes on $\mathbb{R}^n$ (Pan, et al., 2008) to define higher order operators on manifolds and then get estimates for the regularization error.

The second connection is to multi-kernel regularization schemes (Wu et al., 2007; Argyriou et al., 2006; Chapelle et al., 2002) defined as

$$f_{\mathbf{z},\lambda}^{\phi} = \arg \min_{0<\sigma<\infty} \min_{f\in\mathcal{H}_\sigma} \left\{ \frac{1}{m}\sum_{i=1}^{m} \phi(y_i f(x_i)) + \lambda \|f\|_{\mathcal{H}_\sigma}^2 \right\}.$$

In this scheme the variance parameter σ is chosen automatically while the learning rate derived in Ying and Zhou (2007) is at most $O(m^{-1/6})$. It would be interesting to investigate how to choose the parameter σ in (2).

The last questions is about more general loss functions. In our analysis we assume that the convex loss ϕ has a zero which excludes the logistic loss $\phi(t) = \log(1+e^{-t})$. One might generalize our analysis to get some error bounds for the scheme with loss functions without zero by using a general projection operator π_M with level $M > 0$ given by

$$\pi_M(f)(x) = \begin{cases} M & \text{if } f(x) > M, \\ -M & \text{if } f(x) < -M, \\ f(x) & \text{if } -M \le f(x) \le M. \end{cases}$$

Acknowledgments

We would like to thank the referees for their constructive suggestions and comments. The work described in this paper was partially supported by a grant from the Research Grants Council of Hong Kong [Project No. CityU 104007] and National Science Fund for Distinguished Young Scholars of China [Project No. 10529101]. The corresponding author is Ding-Xuan Zhou.

Appendix A. Approximation Scheme by Gaussians

This appendix provides a proof of Theorem 2 which is a corollary of the following more general theorem. The approximation error is estimated by means of a convolution type scheme constructed by Gaussians with a Fourier analysis technique (Schaback and Werner, 2006; Steinwart and Scovel, 2007; Steinwart et al. , 2006).

Theorem 6 *Assume that for some $s>0$, f_ρ^ϕ is the restriction of some $\tilde{f}_\rho^\phi \in H^s(\mathbb{R}^n)$ onto X, and the density function $g=\frac{d\rho_X}{dx}$ exists and lies in $L^2(X)$.*

(1) If $\tilde{f}_\rho^\phi \in L^\infty(\mathbb{R}^n)$, then we can find a set of functions $\{f_{\sigma,\lambda}\in\mathcal{H}_\sigma\}_{0\le\sigma\le 1,\lambda>0}$ such that

$$\begin{aligned} \|f_{\sigma,\lambda}\|_{L^\infty(X)} &\le \widetilde{B}, && (25)\\ \mathcal{D}(\sigma,\lambda) &\le \widetilde{B}(\sigma^s+\lambda\sigma^{-n}), \quad \forall\, 0<\sigma\le 1, \lambda>0, && (26)\end{aligned}$$

where $\widetilde{B}$ is a constant independent of σ or λ.

(2) If for some $r\ge 1$ and $C_\phi>0$,

$$|\phi'_+(t)|\le C_\phi |t|^{r-1} \qquad \forall |t|\ge 1. \tag{27}$$

then we can find $\{f_{\sigma,\lambda}\in\mathcal{H}_\sigma\}$ such that

$$\begin{aligned} \|f_{\sigma,\lambda}\|_{L^\infty(X)} &\le \widetilde{B}'\sigma^{-\frac{n}{2}}, && (28)\\ \mathcal{D}(\sigma,\lambda) &\le \widetilde{B}'(\sigma^{s-\frac{n(r-1)}{2}}+\lambda\sigma^{-n}), \quad \forall\, 0<\sigma\le 1, \lambda>0, && (29)\end{aligned}$$

where $\widetilde{B}'$ is a constant independent of σ or λ.

Proof Take some trigonometric polynomial $\tilde{a}(\xi)=\sum_{j\in J} a_j e^{-ij\cdot\xi}$ on $\mathbb{R}^n$ with a finite subset J of $\mathbb{Z}^n$ such that for some $C_s>0$ depending only on s and n, we have

$$|e^{-\frac{|\xi|^2}{2}}\tilde{a}(\xi)-1|\le C_s|\xi|^s \quad \forall\, \xi\in\mathbb{R}^n.$$

This can be done by choosing the coefficients $(a_j)_{j\in J}$ of $\tilde{a}$ satisfying the linear system

$$\tilde{a}(0)=1 \quad \text{and} \quad D^\alpha(e^{-\frac{|\xi|^2}{2}}\tilde{a}(\xi))(0)=0, \qquad \alpha\in\mathbb{Z}^n, 0<|\alpha|<s.$$

So J and $(a_j)_{j\in J}$ depend only on s and n.

Define

$$\tilde{f}_\sigma(x)=\Big(\frac{1}{\sqrt{2\pi}\sigma}\Big)^n\int_{\mathbb{R}^n} K^\sigma(x,y)\sum_{j\in J} a_j \tilde{f}_\rho^\phi(y-\sigma j)\,dy, \quad x\in\mathbb{R}^n.$$

We first estimate $\|\tilde{f}_\sigma - \tilde{f}_\rho^\phi\|_{L^2(\mathbb{R}^n)}$.

Define a function $\tilde{k}^\sigma$ on $\mathbb{R}^n$ by $\tilde{k}^\sigma(x) = \left(\frac{1}{\sqrt{2\pi}\sigma}\right)^n e^{-\frac{|x|^2}{2\sigma^2}}$, we know that $\hat{\tilde{k}^\sigma}(\xi) = e^{-\frac{|\sigma\xi|^2}{2}}$ and $\tilde{f}_\sigma(x) = \tilde{k}^\sigma * \left(\sum_{j\in J} a_j \tilde{f}_\rho^\phi(\cdot - \sigma j)\right)$. This in connection with the fact that the Fourier transform of $\tilde{f}_\rho^\phi(\cdot - \sigma j)$ equals $e^{-i\sigma j\cdot\xi}\hat{\tilde{f}}_\rho^\phi(\xi)$ implies

$$\hat{\tilde{f}}_\sigma(\xi) = \hat{\tilde{k}^\sigma}(\xi) \sum_{j\in J} a_j e^{-i\sigma j\cdot\xi} \hat{\tilde{f}}_\rho^\phi(\xi) = e^{-\frac{|\sigma\xi|^2}{2}} \tilde{a}(\sigma\xi) \hat{\tilde{f}}_\rho^\phi(\xi).$$

It follows that

$$\begin{aligned} \|\tilde{f}_\sigma - \tilde{f}_\rho^\phi\|^2_{L^2(\mathbb{R}^n)} &= (2\pi)^{-n} \|\hat{\tilde{f}}_\sigma - \hat{\tilde{f}}_\rho^\phi\|^2_{L^2(\mathbb{R}^n)} = (2\pi)^{-n} \int_{\mathbb{R}^n} |e^{-\frac{|\sigma\xi|^2}{2}} \tilde{a}(\sigma\xi) - 1|^2 |\hat{\tilde{f}}_\rho^\phi(\xi)|^2 \, d\xi \\ &\leq (2\pi)^{-n} C_s^2 \int_{\mathbb{R}^n} |\sigma\xi|^{2s} |\hat{\tilde{f}}_\rho^\phi(\xi)|^2 \, d\xi \leq C_s^2 \sigma^{2s} (2\pi)^{-n} \int_{\mathbb{R}^n} |\xi|^{2s} |\hat{\tilde{f}}_\rho^\phi(\xi)|^2 \, d\xi. \end{aligned}$$

That is,

$$\|\tilde{f}_\sigma - \tilde{f}_\rho^\phi\|_{L^2(\mathbb{R}^n)} \leq C_s \|\tilde{f}_\rho^\phi\|_{H^s(\mathbb{R}^n)} \sigma^s. \tag{30}$$

Then we bound $\|\tilde{f}_\sigma\|_{\mathcal{H}_\sigma(\mathbb{R}^n)}$. Here $\mathcal{H}_\sigma(\mathbb{R}^n)$ is the RKHS generated by the Mercer kernel $K^\sigma(x,y)$ on $\mathbb{R}^n$. By the inner product in $\mathcal{H}_\sigma(\mathbb{R}^n)$, we know that $\langle K^\sigma(\cdot,y), K^\sigma(\cdot,z)\rangle_{\mathcal{H}_\sigma(\mathbb{R}^n)} = K^\sigma(y,z)$. So we have

$$\|\tilde{f}_\sigma\|^2_{\mathcal{H}_\sigma(\mathbb{R}^n)} = \left(\frac{1}{\sqrt{2\pi}\sigma}\right)^{2n} \int_{\mathbb{R}^n} \int_{\mathbb{R}^n} K^\sigma(y,z) \sum_{j\in J} a_j \tilde{f}_\rho^\phi(y - \sigma j) \, dy \sum_{l\in J} a_l \tilde{f}_\rho^\phi(z - \sigma l) \, dz.$$

By the elementary inequality $|uv| \leq \frac{u^2+v^2}{2}$ and $\int_{\mathbb{R}^n} K^\sigma(y,z) \, dz = (\sqrt{2\pi}\sigma)^n$, we see that

$$\begin{aligned} \|\tilde{f}_\sigma\|^2_{\mathcal{H}_\sigma(\mathbb{R}^n)} &\leq \left(\frac{1}{\sqrt{2\pi}\sigma}\right)^{2n} \sum_{j\in J} \sum_{l\in J} |a_j||a_l| \int_{\mathbb{R}^n} \int_{\mathbb{R}^n} K^\sigma(y,z) \frac{|\tilde{f}_\rho^\phi(y-\sigma j)|^2 + |\tilde{f}_\rho^\phi(z-\sigma l)|^2}{2} \, dydz \\ &= \left(\frac{1}{\sqrt{2\pi}\sigma}\right)^{2n} \sum_{j,l\in J} |a_j||a_l| (\sqrt{2\pi}\sigma)^n \|\tilde{f}_\rho^\phi\|^2_{L^2(\mathbb{R}^n)}. \end{aligned}$$

That is,

$$\|\tilde{f}_\sigma\|_{\mathcal{H}_\sigma(\mathbb{R}^n)} \leq C_s' \|\tilde{f}_\rho^\phi\|_{L^2(\mathbb{R}^n)} (\sqrt{2\pi}\sigma)^{-\frac{n}{2}}$$

where $C_s' := \sum_{j\in J} |a_j|$ is a constant depending only on s and n.

Take $f_{\sigma,\lambda} = \tilde{f}_\sigma|_X$, the restriction of $\tilde{f}_\sigma$ onto X. By basic facts about RKHS (Aronszajn, 1950), we know that $f_{\sigma,\lambda} = \tilde{f}_\sigma|_X \in \mathcal{H}_\sigma$ and

$$\|f_{\sigma,\lambda}\|_{\mathcal{H}_\sigma} \leq \|\tilde{f}_\sigma\|_{\mathcal{H}_\sigma(\mathbb{R}^n)} \leq C_s' \|\tilde{f}_\rho^\phi\|_{L^2(\mathbb{R}^n)} (\sqrt{2\pi}\sigma)^{-\frac{n}{2}}. \tag{31}$$

Now we can derive the desired bounds.

(1) When $\tilde{f}_\rho^\phi \in L^\infty(\mathbb{R}^n)$, for any $x \in \mathbb{R}^n$, we have

$$|\tilde{f}_\sigma(x)| \leq \sum_{j\in J} |a_j| \left(\frac{1}{\sqrt{2\pi}\sigma}\right)^n \int_{\mathbb{R}^n} e^{-\frac{|x-y|^2}{2\sigma^2}} \, dy \|\tilde{f}_\rho^\phi\|_{L^\infty(\mathbb{R}^n)} = C_s' \|\tilde{f}_\rho^\phi\|_{L^\infty(\mathbb{R}^n)}.$$

It follows that (25) holds true and

$$\begin{aligned}\mathcal{E}^{\phi}(f_{\sigma,\lambda})-\mathcal{E}^{\phi}(f_{\rho}^{\phi}) &\leq \int_X\int_Y \sup\left\{|\phi_+'(\xi)|:|\xi|\leq \max\{C_s'\|\tilde{f}_\rho^\phi\|_{L^\infty(\mathbb{R}^n)},\|f_\rho^\phi\|_{L^\infty(X)}\}\right\}\\ &\quad |f_{\sigma,\lambda}(x)-f_\rho^\phi(x)|d\rho(y|x)g(x)dx.\end{aligned}$$

By the Schwarz inequality we see that

$$\mathcal{E}^{\phi}(f_{\sigma,\lambda})-\mathcal{E}^{\phi}(f_{\rho}^{\phi})\leq \sup\{|\phi_+'(\xi)|:|\xi|\leq (C_s'+1)\|\tilde{f}_\rho^\phi\|_{L^\infty(\mathbb{R}^n)}\}\|f_{\sigma,\lambda}-f_\rho^\phi\|_{L^2(X)}\|g\|_{L^2(X)}.$$

This bound in connection with (30) and (31) implies (26) with the constant $\widetilde{B}$ given by

$$\begin{aligned}\widetilde{B} &= \max\Big\{C_s'\|\tilde{f}_\rho^\phi\|_{L^\infty(\mathbb{R}^n)},(C_s')^2\|\tilde{f}_\rho^\phi\|^2_{L^2(\mathbb{R}^n)}(2\pi)^{-\frac{n}{2}},\\ &\quad \sup\{|\phi_+'(\xi)|:|\xi|\leq (C_s'+1)\|\tilde{f}_\rho^\phi\|_{L^\infty(\mathbb{R}^n)}\}C_s\|\tilde{f}_\rho^\phi\|_{H^s(\mathbb{R}^n)}\|g\|_{L^2(X)},\Big\}.\end{aligned}$$

(2) Without the condition $\tilde{f}_\rho^\phi\in L^\infty(\mathbb{R}^n)$, we bound $\|f_{\sigma,\lambda}\|_{L^\infty(X)}$ directly from the expression of $\tilde{f}_\sigma$. For $x\in\mathbb{R}^n$, we have

$$\begin{aligned}|\tilde{f}_\sigma(x)| &\leq \sum_{j\in J}|a_j|\Big(\frac{1}{\sqrt{2\pi}\sigma}\Big)^n\Big\{\int_{\mathbb{R}^n}(K^\sigma(x,y))^2\,dy\Big\}^{\frac12}\Big\{\int_{\mathbb{R}^n}|\tilde{f}_\rho^\phi(y-\sigma j)|^2\,dy\Big\}^{\frac12}\\ &= \sum_{j\in J}|a_j|(2\sqrt{\pi}\sigma)^{-\frac{n}{2}}\|\tilde{f}_\rho^\phi\|_{L^2(\mathbb{R}^n)}.\end{aligned}$$

It follows that $\|f_{\sigma,\lambda}\|_{L^\infty(X)}\leq C_s'(2\sqrt{\pi})^{-\frac{n}{2}}\|\tilde{f}_\rho^\phi\|_{L^2(\mathbb{R}^n)}\sigma^{-\frac{n}{2}}$.

To derive the bound for the excess generalization error, we notice from (27) that

$$\begin{aligned}&\mathcal{E}^{\phi}(f_{\sigma,\lambda})-\mathcal{E}^{\phi}(f_{\rho}^{\phi})=\int_X\int_Y(\phi(yf_{\sigma,\lambda}(x))-\phi(yf_\rho^\phi(x)))\,d\rho(y|x)d\rho_X(x)\\ &\leq \int_X\int_Y\sup\left\{|\phi_+'(\xi)|:|\xi|\leq\max\{C_s'(2\sqrt{\pi}\sigma)^{-\frac{n}{2}}\|\tilde{f}_\rho^\phi\|_{L^2(\mathbb{R}^n)},\|f_\rho^\phi\|_{L^\infty(X)}\}\right\}\\ &\qquad |f_{\sigma,\lambda}(x)-f_\rho^\phi(x)|d\rho(y|x)g(x)dx.\end{aligned}$$

If we denote $C_s''=\max\{C_\phi,C_\phi\|f_\rho^\phi\|^{r-1}_{L^\infty(X)},C_\phi[C_s'(2\sqrt{\pi})^{-\frac{n}{2}}\|\tilde{f}_\rho^\phi\|_{L^2(\mathbb{R}^n)}]^{r-1}\}$, then

$$\mathcal{E}^{\phi}(f_{\sigma,\lambda})-\mathcal{E}^{\phi}(f_{\rho}^{\phi})\leq C_s''\sigma^{-\frac{n(r-1)}{2}}\|f_{\sigma,\lambda}-f_\rho^\phi\|_{L^2(X)}\|g\|_{L^2(X)}\leq C_s''\|g\|_{L^2(X)}C_s\|\tilde{f}_\rho^\phi\|_{H^s(\mathbb{R}^n)}\sigma^{s-\frac{n(r-1)}{2}}.$$

Therefore,

$$\mathcal{D}(\sigma,\lambda)\leq C_s''\|g\|_{L^2(X)}C_s\|\tilde{f}_\rho^\phi\|_{H^s(\mathbb{R}^n)}\sigma^{s-\frac{n(r-1)}{2}}+\lambda(C_s')^2\|\tilde{f}_\rho^\phi\|^2_{L^2(\mathbb{R}^n)}(\sqrt{2\pi}\sigma)^{-n}.$$

This verifies (28) and (29) by taking

$$\widetilde{B}'=\max\left\{C_s'(2\sqrt{\pi})^{-\frac{n}{2}}\|\tilde{f}_\rho^\phi\|_{L^2(\mathbb{R}^n)},C_s''\|g\|_{L^2(X)}C_s\|\tilde{f}_\rho^\phi\|_{H^s(\mathbb{R}^n)},(C_s')^2\|\tilde{f}_\rho^\phi\|^2_{L^2(\mathbb{R}^n)}(2\pi)^{-n/2}\right\}.$$

The proof of Theorem 6 is complete. ■

In our main results, only part (1) of Theorem 6, that is, Theorem 2 is used. A main assumption, condition (8), gives the restriction $\widetilde{f}_\rho^\phi \in H^s(\mathbb{R}^n)\cap L^\infty(\mathbb{R}^n)$. When we do not know whether f_ρ^ϕ can be extended to a uniformly bounded function on $\mathbb{R}^n$, we can use part (2) of Theorem 6. This might be the case when $\phi = \phi_h$, as mentioned in the following.

A geometric noise condition was introduced in Steinwart and Scovel (2007). This condition with exponent $\alpha > 0$ means

$$\int_X |f_\rho(x)| \exp\left\{-\frac{\tau_x^2}{t}\right\} d\rho_X(x) = O\left(t^{\frac{\alpha n}{2}}\right) \tag{32}$$

where

$$\tau_x = \begin{cases} \inf_{f_\rho(u)\geq 0} |x-u|, & \text{if } f_\rho(x) < 0, \\ \inf_{f_\rho(u)\leq 0} |x-u|, & \text{if } f_\rho(x) > 0, \\ 0, & \text{otherwise.} \end{cases}$$

It does not assume smoothness of functions. An interesting result in Steinwart and Scovel (2007) asserts that when $\phi = \phi_h$, geometric noise condition (32) with exponent $0 < \alpha < \infty$ leads to $\mathcal{D}(\sigma,\lambda) \leq \widetilde{B}'''(\sigma^{\alpha n} + \lambda\sigma^{-n})$. With this estimate, under Tsybakov noise condition (5), learning rates are obtained in Steinwart and Scovel (2007). For example, when $\alpha > \frac{q+2}{2q}$, for an arbitrarily small $\varepsilon > 0$, with confidence $1-\delta$,

$$\mathcal{R}(sgn(f_{\mathbf{z}})) - \mathcal{R}(f_c) \leq \widetilde{C}_\varepsilon \left(\log\frac{4}{\delta}\right)^2 \left(\frac{1}{m}\right)^{\frac{2\alpha(q+1)}{2\alpha(q+2)+3q+4}-\varepsilon}. \tag{33}$$

Since no Sobolev smoothness is assumed for $f_\rho^{\phi_h} = f_c$ (Wahba, 1990), we need to use the regularizing function $\widetilde{f}_{\sigma,\lambda}$ defined by (7) and derive by some detailed computations that with confidence $1-\delta$,

$$\mathcal{R}(sgn(f_{\mathbf{z}})) - \mathcal{R}(f_c) \leq \widetilde{C}_{\rho,h} \log\frac{4}{\delta}\left(\frac{1}{m}\right)^{\frac{(q+1)\alpha n}{(q+2)\alpha n+2(q+1)(n+1)}}.$$

This rate is slightly worse than (33), though the estimate for the confidence is slightly better. It raises the question of improving Theorem 6 under various noise conditions.

Appendix B. Role of Tight Bounds for Covering Numbers

In this appendix we prove Lemma 2 which shows a special role of the tight bound (15) for covering numbers concerning Gaussian kernels. In fact, we have the following more general result.

Proposition 4 *Let $\Delta \geq 1$ be arbitrary. Then $\varepsilon^*(m,\lambda,\sigma,\delta/2)$ defined by (16) satisfies*

$$\varepsilon^*(m,\lambda,\sigma,\delta/2) \leq \widetilde{C}_2\left(\frac{\log\frac{2}{\delta} + \sigma^{-2(n+1)/(2-\tau)} + (\log m)^{(n+1)/(2-\tau)}}{m^{\frac{1}{2-\tau}}} + \frac{\sigma^{-2(n+1)}}{m} + \frac{\sqrt{\phi(0)}}{\sqrt{\lambda}m^\Delta}\right),$$

where $\widetilde{C}_2$ is the constant depending on $C_0, C_1, \tau, \phi'_+(-1), \phi(-1), \Delta$ and is given by

$$\widetilde{C}_2 = \max\left\{|\phi'_+(-1)|, (4C_1)^{\frac{1}{2-\tau}}, 2(4C_0C_1)^{\frac{1}{2-\tau}}\Delta^{n+1}, 2\phi(-1)(1+C_0+C_0\Delta^{n+1})\right\}.$$

Proof Observe from (15) that as a function on $(0,+\infty)$, the logarithm of the middle term of (16) is bounded by

$$h(\varepsilon) := C_0\Big(\Big(\log\frac{\sqrt{\phi(0)}|\phi'_+(-1)|}{\sqrt{\lambda}\varepsilon}\Big)^{n+1} + \frac{1}{\sigma^{2(n+1)}}\Big) - g(\varepsilon),$$

where g is the strictly increasing function on $(0,\infty)$ defined by

$$g(\varepsilon) = \frac{m\varepsilon^{2-\tau}}{2C_1 + \frac{2}{3}\phi(-1)\varepsilon^{1-\tau}}.$$

Set

$$\begin{aligned}\mathcal{B} &= \frac{\sqrt{\phi(0)}|\phi'_+(-1)|}{\sqrt{\lambda}m^{\Delta}} + \left(\frac{4C_1(\log\frac{2}{\delta} + \frac{C_0}{\sigma^{2(n+1)}}) + 4C_0C_1(\Delta\log m)^{n+1}}{m}\right)^{\frac{1}{2-\tau}} \\ &\quad + \frac{4\phi(-1)}{3m}\left(\log\frac{2}{\delta} + \frac{C_0}{\sigma^{2(n+1)}} + C_0(\Delta\log m)^{n+1}\right).\end{aligned}$$

If $\frac{2}{3}\phi(-1)\mathcal{B}^{1-\tau} \le 2C_1$, then

$$g(\mathcal{B}) \ge \frac{m\mathcal{B}^{2-\tau}}{4C_1} \ge \log\frac{2}{\delta} + \frac{C_0}{\sigma^{2(n+1)}} + C_0(\Delta\log m)^{n+1}.$$

If $\frac{2}{3}\phi(-1)\mathcal{B}^{1-\tau} > 2C_1$, then

$$g(\mathcal{B}) \ge \frac{m\mathcal{B}^{2-\tau}}{\frac{4}{3}\phi(-1)\mathcal{B}^{1-\tau}} = \frac{m\mathcal{B}}{\frac{4}{3}\phi(-1)} \ge \log\frac{2}{\delta} + \frac{C_0}{\sigma^{2(n+1)}} + C_0(\Delta\log m)^{n+1}.$$

Thus in either case we have

$$g(\mathcal{B}) \ge \log\frac{2}{\delta} + \frac{C_0}{\sigma^{2(n+1)}} + C_0(\Delta\log m)^{n+1}.$$

On the other hand, since $\mathcal{B} \ge \frac{\sqrt{\phi(0)}|\phi'_+(-1)|}{\sqrt{\lambda}m^{\Delta}}$, we also see that $\log\frac{\sqrt{\phi(0)}|\phi'_+(-1)|}{\mathcal{B}\sqrt{\lambda}} \le \Delta\log m$. It follows that

$$h(\mathcal{B}) \le C_0(\Delta\log m)^{n+1} - \log\frac{2}{\delta} - C_0(\Delta\log m)^{n+1} = \log\frac{\delta}{2}.$$

But the function h is strictly decreasing. So $\varepsilon^*(m,\lambda,\sigma,\delta/2) \le \mathcal{B}$. The the desired bound for $\varepsilon^*(m,\lambda,\sigma,\delta/2)$ follows with the constant $\widetilde{C_2}$. The proof of Proposition 4 is complete. ■

Now we can prove Lemma 2 by the special form of λ,σ.

B.1 Proof of Lemma 2

Take $\Delta = \frac{\gamma}{2} + 1$ in Proposition 4. Then we know from the special form (3) of λ and σ that

$$\varepsilon^*(m,\lambda,\sigma,\delta/2) \le \widetilde{C_2}\left\{\frac{\log\frac{2}{\delta}}{m^{\frac{1}{2-\tau}}} + \Big(\frac{1}{m}\Big)^{\frac{1-2\gamma\zeta(n+1)}{2-\tau}} + \Big(\frac{(\log m)^{n+1}}{m}\Big)^{\frac{1}{2-\tau}} + \frac{\sqrt{\phi(0)}}{m}\right\}.$$

notation	meaning	pages
$\mathcal{R}(\mathcal{C})$	misclassification error for a classifier $\mathcal{C}$	1447
f_c	Bayes rule which minimizes $\mathcal{R}$	1447
$\mathcal{R}(\mathcal{C}) - \mathcal{R}(f_c)$	excess misclassification error	1447
ϕ	loss function for classification	1448
σ	variance parameter for the Gaussian kernel	1448, 1449, 1458
λ	regularization parameter	1448, 1449, 1458
$f_{\mathbf{z}}$	learning scheme (2)	1448
$\mathcal{R}(\text{sgn}(f_{\mathbf{z}})) - \mathcal{R}(f_c)$	excess misclassification error for classifier $\text{sgn}(f_{\mathbf{z}})$	1448, 1449, 1458
f_ρ	regression function	1448, 1449, 1457
$\mathcal{E}^\phi(f)$	generalization error for a function f	1450
f_ρ^ϕ	minimizer of $\mathcal{E}^\phi$	1450
$\mathcal{E}^\phi(f) - \mathcal{E}^\phi(f_\rho^\phi)$	excess generalization error for a function f	1451, 1452, 1454, 1457
$f_{\sigma,\lambda}$	regularizing function constructed in Theorem 2	1450, 1452, 1454, 1461
$\mathcal{D}(\sigma,\lambda)$	regularization error or approximation error	1450, 1452, 1456, 1461
$\tau = \tau_{\phi,\rho}$	variancing power defined in Definition 5	1453, 1454, 1456, 1458

Table 1: NOTATIONS

Observe the elementary inequality (Yao, 2008; Ye and Zhou, 2007)

$$\exp\{-cx\} \leq (\frac{a}{ec})^a x^{-a} \quad \forall x, c, a > 0.$$

Taking $x = \log m$, $a = n+1$ and $c = 2\gamma\zeta(n+1)$, we have

$$(\log m)^{n+1} \leq (\frac{1}{2e\gamma\zeta})^{n+1} m^{2\gamma\zeta(n+1)}.$$

Hence (17) holds true with the constant $C_2 = \widetilde{C}_2(4 + 2\sqrt{\phi(0)})$. The proof of Lemma 2 is complete.
■

References

A. Argyriou, R. Hauser, C. A. Micchelli, and M. Pontil. A DC-programming algorithm for kernel selection. *Proceedings of the Twenty-Third International Conference on Machine Learning*, 2006.

N. Aronszajn. Theory of reproducing kernels. *Transactions of the American Mathematical Society*, 68:337–404, 1950.

P. L. Bartlett, M. I. Jordan, and J. D. McAuliffe. Convexity, classification, and risk bounds. *Journal of the American Statistical Association*, 101:138–156, 2006.

O. Chapelle, V. N. Vapnik, O. Bousquet and S. Mukherjee. Choosing multiple parameters for support vector machines. *Machine Learning*, 46: 131–159, 2002.

N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines*. Cambridge University Press, 2000.

D. R. Chen, Q. Wu, Y. Ying, and D. X. Zhou. Support vector machine soft margin classifiers: error analysis. *Journal of Machine Learning Research*, 5:1143–1175, 2004.

F. Cucker and D. X. Zhou. *Learning Theory: An Approximation Theory Viewpoint*. Cambridge University Press, 2007.

E. De Vito, A. Caponnetto, and L. Rosasco. Model selection for regularized least-squares algorithm in learning theory. *Foundations of Computational Mathematics*, 5:59–85, 2006.

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.

D. Edmunds and H. Triebel. *Function Spaces, Entropy Numbers, Differential Operators*. Cambridge University Press, 1996.

T. Evgeniou, M. Pontil, and T. Poggio. Regularization networks and support vector machines. *Advances in Computational Mathematics*, 13:1–50, 2000.

D. Hardin, I. Tsamardinos, and C. F. Aliferis. A theoretical characterization of linear SVM-based feature selection. *Proc. of the 21st Int. Conf. on Machine Learning*, Banff, Canada, 2004.

S. Mukherjee, P. Niyogi, T. Poggio, and R. Rifkin. Learning theory: stability is sufficient for generalization and necessary and sufficient for empirical risk minimization. *Advances in Computational Mathematics*, 25:161–193, 2006.

Z. W. Pan, D. H. Xiang, Q. W. Xiao, and D. X. Zhou. Parzen windows for multi-class classification. *Journal of Complexity*, 24:606–618, 2008.

R. Schaback and J. Werner. Linearly constrained reconstruction of functions by kernels, with applications to machine learning. *Advances in Computational Mathematics*, 25:237–258, 2006.

S. Smale and D. X. Zhou. Learning theory estimates via integral operators and their approximations. *Constructive Approximation*, 26:153–172, 2007.

S. Smale and D. X. Zhou. Estimating the approximation error in learning theory. *Analysis and Applications*, 1:17–41, 2003.

S. Smale and D. X. Zhou. Shannon sampling and function reconstruction from point values. *Bulletin of the American Mathematical Society*, 41:279–305, 2004.

I. Steinwart and C. Scovel. Fast rates for support vector machines using Gaussian kernels. *Annals of Statistics*, 35:575–607, 2007.

I. Steinwart, D. Hush, and C. Scovel. An explicit description of the reproducing kernel Hilbert spaces of Gaussian RBF kernels. *IEEE Transactions on Information Theory*, 52:4635–4643, 2006.

A. B. Tsybakov. Optimal aggregation of classifiers in statistical learning. *Annals of Statistics*, 32:135–166, 2004.

G. Wahba. *Spline Models for Observational Data*. SIAM, 1990.

Q. Wu, Y. Ying, and D. X. Zhou. Multi-kernel regularized classifiers. *Journal of Complexity*, 23:108–134, 2007.

Q. Wu and D. X. Zhou. Analysis of support vector machine classification. *Journal of Computational Analysis and Applications*, 8:99–119, 2006.

Q. Wu and D. X. Zhou. SVM soft margin classifiers: linear programming versus quadratic programming. *Neural Computation*, 17:1160–1187, 2005.

Y. Yao. On complexity issue of online learning algorithms. *IEEE Transactions on Information Theory*, to appear.

G. B. Ye and D. X. Zhou. Fully online classification by regularization. *Applied and Computational Harmonic Analysis*, 23:198–214, 2007.

G. B. Ye and D. X. Zhou. Learning and approximation by Gaussians on Riemannian manifolds. *Advances in Computational Mathematics*, 29:291–310, 2008.

Y. Ying. Convergence analysis of online algorithms. *Advances in Computational Mathematics*, 27:273–291, 2007.

Y. Ying and D. X. Zhou. Learnability of Gaussians with flexible variances. *Journal of Machine Learning Research*, 8:249–276, 2007.

T. Zhang. Statistical behavior and consistency of classification methods based on convex risk minimization. *Annals of Statistics*, 32:56–85, 2004.

D. X. Zhou. The covering number in learning theory. *Journal of Complexity*, 18:739–767, 2002.

D. X. Zhou. Capacity of reproducing kernel spaces in learning theory. *IEEE Transactions on Information Theory*, 49:1743–1752, 2003.

D. X. Zhou and K. Jetter. Approximation with polynomial kernels and SVM classifiers. *Advances in Computational Mathematics*, 25:323–344, 2006.

Journal of Machine Learning Research 10 (2009) 1469-1484 Submitted 10/08; Revised 4/09; Published 7/09

Entropy Inference and the James-Stein Estimator, with Application to Nonlinear Gene Association Networks

Jean Hausser JEAN.HAUSSER@UNIBAS.CH
Bioinformatics, Biozentrum
University of Basel
Klingelbergstr. 50/70
CH-4056 Basel, Switzerland

Korbinian Strimmer STRIMMER@UNI-LEIPZIG.DE
Institute for Medical Informatics, Statistics and Epidemiology
University of Leipzig
Härtelstr. 16–18
D-04107 Leipzig, Germany

Editor: Xiaotong Shen

Abstract

We present a procedure for effective estimation of entropy and mutual information from small-sample data, and apply it to the problem of inferring high-dimensional gene association networks. Specifically, we develop a James-Stein-type shrinkage estimator, resulting in a procedure that is highly efficient statistically as well as computationally. Despite its simplicity, we show that it outperforms eight other entropy estimation procedures across a diverse range of sampling scenarios and data-generating models, even in cases of severe undersampling. We illustrate the approach by analyzing *E. coli* gene expression data and computing an entropy-based gene-association network from gene expression data. A computer program is available that implements the proposed shrinkage estimator.

Keywords: entropy, shrinkage estimation, James-Stein estimator, "small n, large p" setting, mutual information, gene association network

1. Introduction

Entropy is a fundamental quantity in statistics and machine learning. It has a large number of applications, for example in astronomy, cryptography, signal processing, statistics, physics, image analysis neuroscience, network theory, and bioinformatics—see, for example, Stinson (2006), Yeo and Burge (2004), MacKay (2003) and Strong et al. (1998). Here we focus on *estimating* entropy from small-sample data, with applications in genomics and gene network inference in mind (Margolin et al., 2006; Meyer et al., 2007).

To define the Shannon entropy, consider a categorical random variable with alphabet size p and associated cell probabilities $\theta_1, \dots, \theta_p$ with $\theta_k > 0$ and $\sum_k \theta_k = 1$. Throughout the article, we assume that p is fixed and known. In this setting, the Shannon entropy in natural units is given by[1]

$$H = -\sum_{k=1}^{p} \theta_k \log(\theta_k). \tag{1}$$

In practice, the underlying probability mass function are unknown, hence H and θ_k need to be *estimated* from observed cell counts $y_k \geq 0$.

A particularly simple and widely used estimator of entropy is the maximum likelihood (ML) estimator

$$\hat{H}^{\text{ML}} = -\sum_{k=1}^{p} \hat{\theta}_k^{\text{ML}} \log(\hat{\theta}_k^{\text{ML}})$$

constructed by plugging the ML frequency estimates

$$\hat{\theta}_k^{\text{ML}} = \frac{y_k}{n} \tag{2}$$

into Equation **1**, with $n = \sum_{k=1}^{p} y_k$ being the total number of counts.

In situations with $n \gg p$, that is, when the dimension is low and when there are many observation, it is easy to infer entropy reliably, and it is well-known that in this case the ML estimator is optimal. However, in high-dimensional problems with $n \ll p$ it becomes extremely challenging to estimate the entropy. Specifically, in the "small n, large p" regime the ML estimator performs very poorly and severely underestimates the true entropy.

While entropy estimation has a long history tracing back to more than 50 years ago, it is only recently that the specific issues arising in high-dimensional, undersampled data sets have attracted attention. This has lead to two recent innovations, namely the NSB algorithm (Nemenman et al., 2002) and the Chao-Shen estimator (Chao and Shen, 2003), both of which are now widely considered as benchmarks for the small-sample entropy estimation problem (Vu et al., 2007).

Here, we introduce a novel and highly efficient small-sample entropy estimator based on James-Stein shrinkage (Gruber, 1998). Our method is fully analytic and hence computationally inexpensive. Moreover, our procedure simultaneously provides estimates of the entropy *and* of the cell frequencies suitable for plugging into the Shannon entropy formula (Equation 1). Thus, in comparison the estimator we propose is simpler, very efficient, and at the same time more versatile than currently available entropy estimators.

2. Conventional Methods for Estimating Entropy

Entropy estimators can be divided into two groups: i) methods, that rely on estimates of cell frequencies, and ii) estimators, that directly infer entropy without estimating a compatible set of θ_k. Most methods discussed below fall into the first group, except for the Miller-Madow and NSB approaches.

1. In this paper we use the following conventions: log denotes the natural logarithm (*not* base 2 or base 10), and we define $0 \log 0 = 0$.

2.1 Maximum Likelihood Estimate

The connection between observed counts y_k and frequencies θ_k is given by the multinomial distribution

$$\text{Prob}(y_1,\ldots,y_p;\theta_1,\ldots,\theta_p) = \frac{n!}{\prod_{k=1}^{p} y_k!}\prod_{k=1}^{p}\theta_k^{y_k}. \tag{3}$$

Note that $\theta_k > 0$ because otherwise the distribution is singular. In contrast, there may be (and often are) zero counts y_k. The ML estimator of θ_k maximizes the right hand side of Equation **3** for fixed y_k, leading to the observed frequencies $\hat{\theta}_k^{\text{ML}} = \frac{y_k}{n}$ with variances $\text{Var}(\hat{\theta}_k^{\text{ML}}) = \frac{1}{n}\theta_k(1-\theta_k)$ and $\text{Bias}(\hat{\theta}_k^{\text{ML}}) = 0$ as $E(\hat{\theta}_k^{\text{ML}}) = \theta_k$.

2.2 Miller-Madow Estimator

While $\hat{\theta}_k^{\text{ML}}$ is unbiased, the corresponding plugin entropy estimator $\hat{H}^{\text{ML}}$ is not. First order bias correction leads to

$$\hat{H}^{\text{MM}} = \hat{H}^{\text{ML}} + \frac{m_{>0}-1}{2n},$$

where $m_{>0}$ is the number of cells with $y_k > 0$. This is known as the Miller-Madow estimator (Miller, 1955).

2.3 Bayesian Estimators

Bayesian regularization of cell counts *may* lead to vast improvements over the ML estimator (Agresti and Hitchcock, 2005). Using the Dirichlet distribution with parameters $a_1, a_2, \ldots, a_p$ as prior, the resulting posterior distribution is also Dirichlet with mean

$$\hat{\theta}_k^{\text{Bayes}} = \frac{y_k + a_k}{n + A},$$

where $A = \sum_{k=1}^{p} a_k$. The flattening constants a_k play the role of pseudo-counts (compare with Equation 2), so that A may be interpreted as the *a priori* sample size.

Some common choices for a_k are listed in Table **1**, along with references to the corresponding plugin entropy estimators,

$$\hat{H}^{\text{Bayes}} = -\sum_{k=1}^{p}\hat{\theta}_k^{\text{Bayes}}\log(\hat{\theta}_k^{\text{Bayes}}).$$

a_k	Cell frequency prior	Entropy estimator
0	no prior	maximum likelihood
$1/2$	Jeffreys prior (Jeffreys, 1946)	Krichevsky and Trofimov (1981)
1	Bayes-Laplace uniform prior	Holste et al. (1998)
$1/p$	Perks prior (Perks, 1947)	Schürmann and Grassberger (1996)
$\sqrt{n}/p$	minimax prior (Trybula, 1958)	

Table 1: Common choices for the parameters of the Dirichlet prior in the Bayesian estimators of cell frequencies, and corresponding entropy estimators.

While the multinomial model with Dirichlet prior is standard Bayesian folklore (Gelman et al., 2004), there is no general agreement regarding which assignment of a_k is best as noninformative prior—see for instance the discussion in Tuyl et al. (2008) and Geisser (1984). But, as shown later in this article, choosing inappropriate a_k can easily cause the resulting estimator to perform *worse* than the ML estimator, thereby defeating the originally intended purpose.

2.4 NSB Estimator

The NSB approach (Nemenman et al., 2002) avoids overrelying on a particular choice of a_k in the Bayes estimator by using a more refined prior. Specifically, a Dirichlet mixture prior with infinite number of components is employed, constructed such that the resulting prior over the entropy is uniform. While the NSB estimator is one of the best entropy estimators available at present in terms of statistical properties, using the Dirichlet mixture prior is computationally expensive and somewhat slow for practical applications.

2.5 Chao-Shen Estimator

Another recently proposed estimator is due to Chao and Shen (2003). This approach applies the Horvitz-Thompson estimator (Horvitz and Thompson, 1952) in combination with the Good-Turing correction (Good, 1953; Orlitsky et al., 2003) of the empirical cell probabilities to the problem of entropy estimation. The Good-Turing-corrected frequency estimates are

$$\hat{\theta}_k^{\text{GT}} = (1 - \frac{m_1}{n})\hat{\theta}_k^{\text{ML}},$$

where m_1 is the number of singletons, that is, cells with $y_k = 1$. Used jointly with the Horvitz-Thompson estimator this results in

$$\hat{H}^{CS} = -\sum_{k=1}^{p} \frac{\hat{\theta}_k^{\text{GT}} \log \hat{\theta}_k^{\text{GT}}}{(1 - (1 - \hat{\theta}_k^{\text{GT}})^n)},$$

an estimator with remarkably good statistical properties (Vu et al., 2007).

3. A James-Stein Shrinkage Estimator

The contribution of this paper is to introduce an entropy estimator that employs James-Stein-type shrinkage at the level of cell frequencies. As we will show below, this leads to an entropy estimator that is highly effective, both in terms of statistical accuracy and computational complexity.

James-Stein-type shrinkage is a simple analytic device to perform regularized high-dimensional inference. It is ideally suited for small-sample settings - the original estimator (James and Stein, 1961) considered sample size $n = 1$. A general recipe for constructing shrinkage estimators is given in *Appendix A*. In this section, we describe how this approach can be applied to the specific problem of estimating cell frequencies.

James-Stein shrinkage is based on averaging two very different models: a high-dimensional model with low bias and high variance, and a lower dimensional model with larger bias but smaller variance. The intensity of the regularization is determined by the relative weighting of the two models. Here we consider the convex combination

$$\hat{\theta}_k^{\text{Shrink}} = \lambda t_k + (1 - \lambda)\hat{\theta}_k^{\text{ML}}, \tag{4}$$

where $\lambda \in [0,1]$ is the shrinkage intensity that takes on a value between 0 (no shrinkage) and 1 (full shrinkage), and t_k is the shrinkage target. A convenient choice of t_k is the uniform distribution $t_k = \frac{1}{p}$. This is also the maximum entropy target. Considering that $\text{Bias}(\hat{\theta}_k^{\text{ML}}) = 0$ and using the unbiased estimator $\widehat{\text{Var}}(\hat{\theta}_k^{\text{ML}}) = \frac{\hat{\theta}_k^{\text{ML}}(1-\hat{\theta}_k^{\text{ML}})}{n-1}$ we obtain (cf. *Appendix A*) for the shrinkage intensity

$$\hat{\lambda}^\star = \frac{\sum_{k=1}^p \widehat{\text{Var}}(\hat{\theta}_k^{\text{ML}})}{\sum_{k=1}^p (t_k - \hat{\theta}_k^{\text{ML}})^2} = \frac{1 - \sum_{k=1}^p (\hat{\theta}_k^{\text{ML}})^2}{(n-1)\sum_{k=1}^p (t_k - \hat{\theta}_k^{\text{ML}})^2}. \tag{5}$$

Note that this also assumes a non-stochastic target t_k. The resulting plugin shrinkage entropy estimate is

$$\hat{H}^{\text{Shrink}} = -\sum_{k=1}^{p} \hat{\theta}_k^{\text{Shrink}} \log(\hat{\theta}_k^{\text{Shrink}}). \tag{6}$$

Remark 1 *There is a one to one correspondence between the shrinkage and the Bayes estimator. If we write $t_k = \frac{a_k}{A}$ and $\lambda = \frac{A}{n+A}$, then $\hat{\theta}_k^{Shrink} = \hat{\theta}_k^{Bayes}$. This implies that the shrinkage estimator is an empirical Bayes estimator with a data-driven choice of the flattening constants—see also Efron and Morris (1973). For every choice of A there exists an equivalent shrinkage intensity λ. Conversely, for every λ there exist an equivalent $A = n\frac{\lambda}{1-\lambda}$.*

Remark 2 *Developing $A = n\frac{\lambda}{1-\lambda} = n(\lambda + \lambda^2 + \ldots)$ we obtain the approximate estimate $\hat{A} = n\hat{\lambda}$, which in turn recovers the "pseudo-Bayes" estimator described in Fienberg and Holland (1973).*

Remark 3 *The shrinkage estimator assumes a fixed and known p. In many practical applications this will indeed be the case, for example, if the observed counts are due to discretization (see also the data example). In addition, the shrinkage estimator appears to be robust against assuming a larger p than necessary (see scenario 3 in the simulations).*

Remark 4 *The shrinkage approach can easily be modified to allow multiple targets with different shrinkage intensities. For instance, using the Good-Turing estimator (Good, 1953; Orlitsky et al., 2003), one could setup a different uniform target for the non-zero and the zero counts, respectively.*

4. Comparative Evaluation of Statistical Properties

In order to elucidate the relative strengths and weaknesses of the entropy estimators reviewed in the previous section, we set to benchmark them in a simulation study covering different data generation processes and sampling regimes.

4.1 Simulation Setup

We compared the statistical performance of all nine described estimators (maximum likelihood, Miller-Madow, four Bayesian estimators, the proposed shrinkage estimator (Equations 4–6), NSB und Chao-Shen) under various sampling and data generating scenarios:

- The dimension was fixed at $p = 1000$.
- Samples size n varied from 10, 30, 100, 300, 1000, 3000, to 10000. That is, we investigate cases of dramatic undersampling ("small n, large p") as well as situations with a larger number of observed counts.

The true cell probabilities $\theta_1, \dots, \theta_{1000}$ were assigned in four different fashions, corresponding to rows 1-4 in Figure **1**:

1. Sparse and heterogeneous, following a Dirichlet distribution with parameter $a = 0.0007$,
2. Random and homogeneous, following a Dirichlet distribution with parameter $a = 1$,
3. As in scenario 2, but with half of the cells containing structural zeros, and
4. Following a Zipf-type power law.

For each sampling scenario and sample size, we conducted 1000 simulation runs. In each run, we generated a new set of true cell frequencies and subsequently sampled observed counts y_k from the corresponding multinomial distribution. The resulting counts y_k were then supplied to the various entropy and cell frequencies estimators and the squared error $\sum_{i=k}^{1000}(\theta_k - \hat{\theta}_k)^2$ was computed. From the 1000 repetitions we estimated the mean squared error (MSE) of the cell frequencies by averaging over the individual squared errors (except for the NSB, Miller-Madow, and Chao-Shen estimators). Similarly, we computed estimates of MSE and bias of the inferred entropies.

4.2 Summary of Results from Simulations

Figure **1** displays the results of the simulation study, which can be summarized as follows:

- Unsurprisingly, all estimators perform well when the sample size is large.
- The maximum likelihood and Miller-Madow estimators perform worst, except for scenario 1. Note that these estimators are inappropriate even for moderately large sample sizes. Furthermore, the bias correction of the Miller-Madow estimator is not particularly effective.
- The minimax and $1/p$ Bayesian estimators tend to perform slightly better than maximum likelihood, but not by much.
- The Bayesian estimators with pseudocounts $1/2$ and 1 perform very well even for small sample sizes in the scenarios 2 and 3. However, they are less efficient in scenario 4, and completely fail in scenario 1.
- Hence, the Bayesian estimators can perform better or worse than the ML estimator, depending on the choice of the prior and on the sampling scenario.
- The NSB, the Chao-Shen and the shrinkage estimator all are statistically very efficient with small MSEs in all four scenarios, regardless of sample size.
- The NSB and Chao-Shen estimators are nearly unbiased in scenario 3.

The three top-performing estimators are the NSB, the Chao-Shen and the prosed shrinkage estimator. When it comes to estimating the entropy, these estimators can be considered identical for practical purposes. However, the shrinkage estimator is the only one that simultaneously estimates cell frequencies suitable for use with the Shannon entropy formula (Equation 1), and it does so with high accuracy even for small samples. In comparison, the NSB estimator is by far the slowest method: in our simulations, the shrinkage estimator was faster by a factor of 1000.

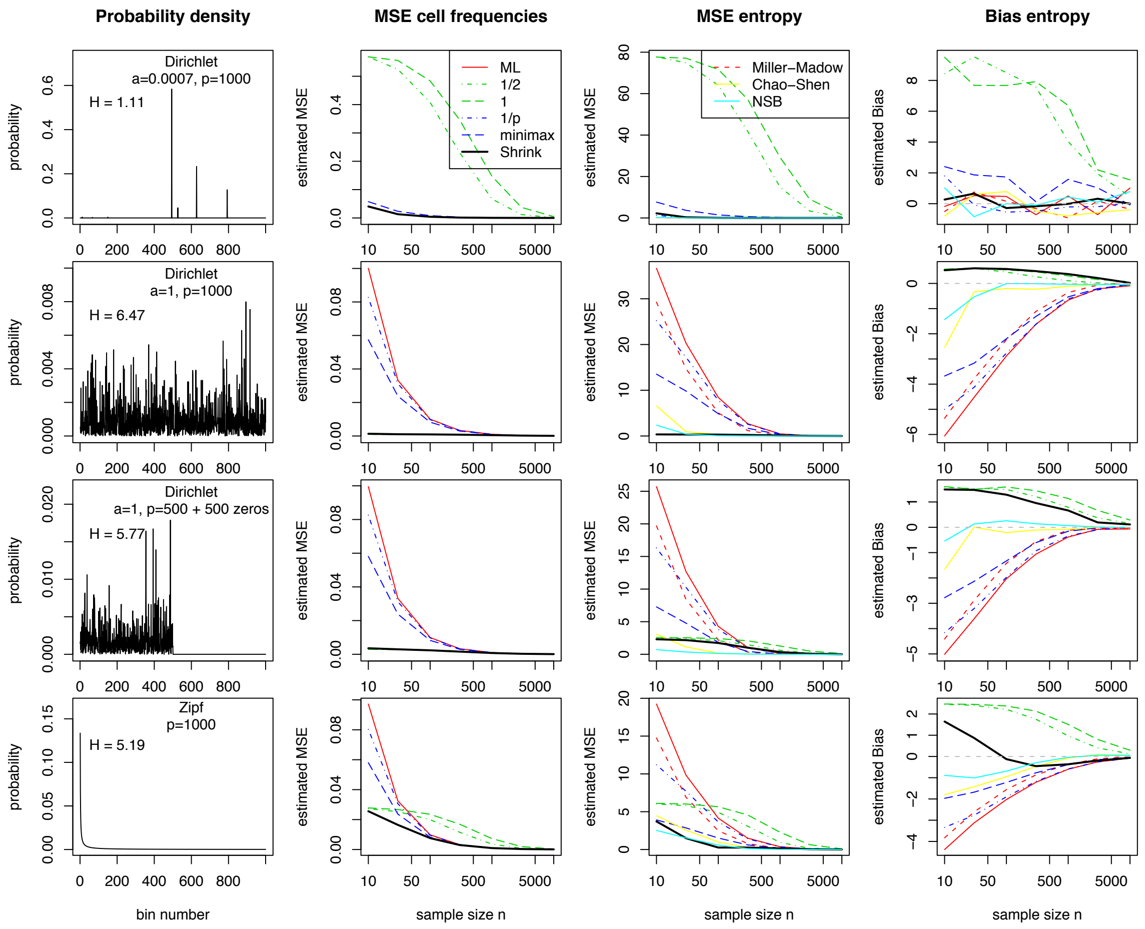


Figure 1: Comparing the performance of nine different entropy estimators (maximum likelihood, Miller-Madow, four Bayesian estimators, the proposed shrinkage estimator, NSB und Chao-Shen) in four different sampling scenarios (rows 1 to 4). The estimators are compared in terms of MSE of the underlying cell frequencies (except for Miller-Madow, NSB, Chao-Shen) and according to MSE and Bias of the estimated entropies. The dimension is fixed at $p = 1000$ while the sample size n varies from 10 to 10000.

5. Application to Statistical Learning of Nonlinear Gene Association Networks

In this section we illustrate how the shrinkage entropy estimator can be applied to the problem of inferring regulatory interactions between genes through estimating the nonlinear association network.

5.1 From Linear to Nonlinear Gene Association Networks

One of the aims of systems biology is to understand the interactions among genes and their products underlying the molecular mechanisms of cellular function as well as how disrupting these interactions may lead to different pathologies. To this end, an extensive literature on the problem of gene regulatory network "reverse engineering" has developed in the past decade (Friedman, 2004). Starting from gene expression or proteomics data, different statistical learning procedures have been proposed to infer associations and dependencies among genes. Among many others, methods have been proposed to enable the inference of large-scale correlation networks (Butte et al., 2000) and of high-dimensional partial correlation graphs (Dobra et al., 2004; Schäfer and Strimmer, 2005a; Meinshausen and Bühlmann, 2006), for learning vector-autoregressive (Opgen-Rhein and Strimmer, 2007a) and state space models (Rangel et al., 2004; Lähdesmäki and Shmulevich, 2008), and to reconstruct directed "causal" interaction graphs (Kalisch and Bühlmann, 2007; Opgen-Rhein and Strimmer, 2007b).

The restriction to linear models in most of the literature is owed at least in part to the already substantial challenges involved in estimating linear high-dimensional dependency structures. However, cell biology offers numerous examples of threshold and saturation effects, suggesting that linear models may not be sufficient to model gene regulation and gene-gene interactions. In order to relax the linearity assumption and to capture nonlinear associations among genes, entropy-based network modeling was recently proposed in the form of the ARACNE (Margolin et al., 2006) and MRNET (Meyer et al., 2007) algorithms.

The starting point of these two methods is to compute the mutual information $\mathrm{MI}(X,Y)$ for all pairs of genes X and Y, where X and Y represent the expression levels of the two genes for instance. The mutual information is the Kullback-Leibler distance from the joint probability density to the product of the marginal probability densities:

$$\mathrm{MI}(X,Y) = E_{f(x,y)}\left\{\log\frac{f(x,y)}{f(x)f(y)}\right\}. \tag{7}$$

The mutual information (MI) is always non-negative, symmetric, and equals zero only if X and Y are independent. For normally distributed variables the mutual information is closely related to the usual Pearson correlation,

$$\mathrm{MI}(X,Y) = -\frac{1}{2}\log(1-\rho^2).$$

Therefore, mutual information is a natural measure of the association between genes, regardless whether linear or nonlinear in nature.

5.2 Estimation of Mutual Information

To construct an entropy network, we first need to *estimate* mutual information for all pairs of genes. The entropy representation

$$\mathrm{MI}(X,Y) = H(X) + H(Y) - H(X,Y), \tag{8}$$

shows that MI can be computed from the joint and marginal entropies of the two genes X and Y. Note that this definition is equivalent to the one given in Equation **7** which is based on the Kullback-Leibler divergence. From Equation **8** it is also evident that $\mathrm{MI}(X,Y)$ is the information shared between the two variables.

For gene expression data the estimation of MI and the underlying entropies is challenging due to the small sample size, which requires the use of a regularized entropy estimator such as the shrinkage approach we propose here. Specifically, we proceed as follows:

- As a prerequisite the data must be discrete, with each measurement assuming one of K levels. If the data are not already discretized, we propose employing the simple algorithm of Freedman and Diaconis (1981), considering the measurements of all genes simultaneously.
- Next, we estimate the $p = K^2$ cell frequencies of the $K \times K$ contingency table for each pair X and Y using the shrinkage approach (Eqs. 4 and 5). Note that typically the sample size n is much smaller than K^2, thus simple approaches such as ML are not valid.
- Finally, from the estimated cell frequencies we calculate $H(X)$, $H(Y)$, $H(X,Y)$ and the desired $\mathrm{MI}(X,Y)$.

5.3 Mutual Information Network for *E. Coli* Stress Response Data

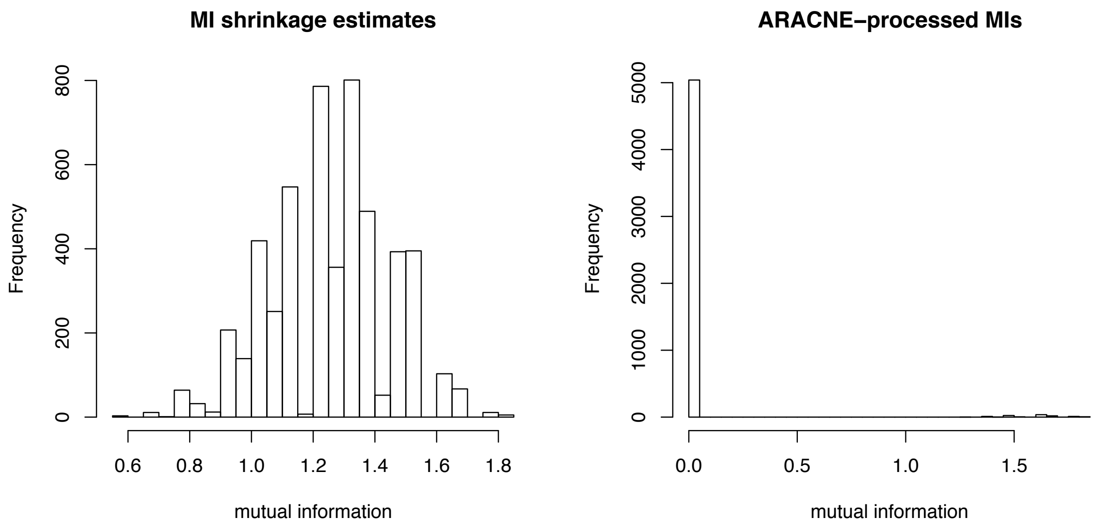


Figure 2: Left: Distribution of estimated mutual information values for all 5151 gene pairs of the *E. coli* data set. Right: Mutual information values after applying the ARACNE gene pair selection procedure. Note that the most MIs have been set to zero by the ARACNE algorithm.

For illustration, we now analyze data from Schmidt-Heck et al. (2004) who conducted an experiment to observe the stress response in *E. Coli* during expression of a recombinant protein. This data set was also used in previous linear network analyzes, for example, in Schäfer and Strimmer (2005b). The raw data consist of 4289 protein coding genes, on which measurements were taken at 0, 8, 15, 22, 45, 68, 90, 150, and 180 minutes. We focus on a subset of $G = 102$ differentially expressed genes as given in Schmidt-Heck et al. (2004).

Discretization of the data according to Freedman and Diaconis (1981) yielded $K = 16$ distinct gene expression levels. From the $G = 102$ genes, we estimated MIs for 5151 pairs of genes. For each pair, the mutual information was based on an estimated 16×16 contingency table, hence $p = 256$. As the number of time points is $n = 9$, this is a strongly undersampled situation which requires the use of a regularized estimate of entropy and mutual information.

The distribution of the shrinkage estimates of mutual information for all 5151 gene pairs is shown in the left side of Figure **2**. The right hand side depicts the distribution of mutual information values after applying the ARACNE procedure, which yields 112 gene pairs with nonzero MIs.

The model selection provided by ARACNE is based on applying the information processing inequality to all gene triplets. For each triplet, the gene pair corresponding to the smallest MI is discarded, which has the effect to remove gene-gene links that correspond to indirect rather than direct interactions. This is similar to a procedure used in graphical Gaussian models where correlations are transformed into partial correlations. Thus, both the ARACNE and the MRNET algorithms can be considered as devices to approximate the conditional mutual information (Meyer et al., 2007). As a result, the 112 nonzero MIs recovered by the ARACNE algorithm correspond to statistically detectable direct associations.

The corresponding gene association network is depicted in Figure **3**. The most striking feature of the graph are the "hubs" belonging to genes hupB, sucA and nuoL. hupB is a well known DNA-binding transcriptional regulator, whereas both nuoL and sucA are key components of the *E. coli* metabolism. Note that a Lasso-type procedure (that implicitly limits the number of edges that can connect to each node) such as that of Meinshausen and Bühlmann (2006) cannot recover these hubs.

6. Discussion

We proposed a James-Stein-type shrinkage estimator for inferring entropy and mutual information from small samples. While this is a challenging problem, we showed that our approach is highly efficient both statistically and computationally despite its simplicity.

In terms of versatility, our estimator has two distinct advantages over the NSB and Chao-Shen estimators. First, in addition to estimating the entropy, it also provides the underlying multinomial frequencies for use with the Shannon formula (Equation 1). This is useful in the context of using mutual information to quantify non-linear pairwise dependencies for instance. Second, unlike NSB, it is a fully analytic estimator.

Hence, our estimator suggests itself for applications in large scale estimation problems. To demonstrate its application in the context of genomics and systems biology, we have estimated an entropy-based gene dependency network from expression data in *E. coli*. This type of approach may prove helpful to overcome the limitations of linear models currently used in network analysis.

In short, we believe the proposed small-sample entropy estimator will be a valuable contribution to the growing toolbox of machine learning and statistics procedures for high-dimensional data analysis.

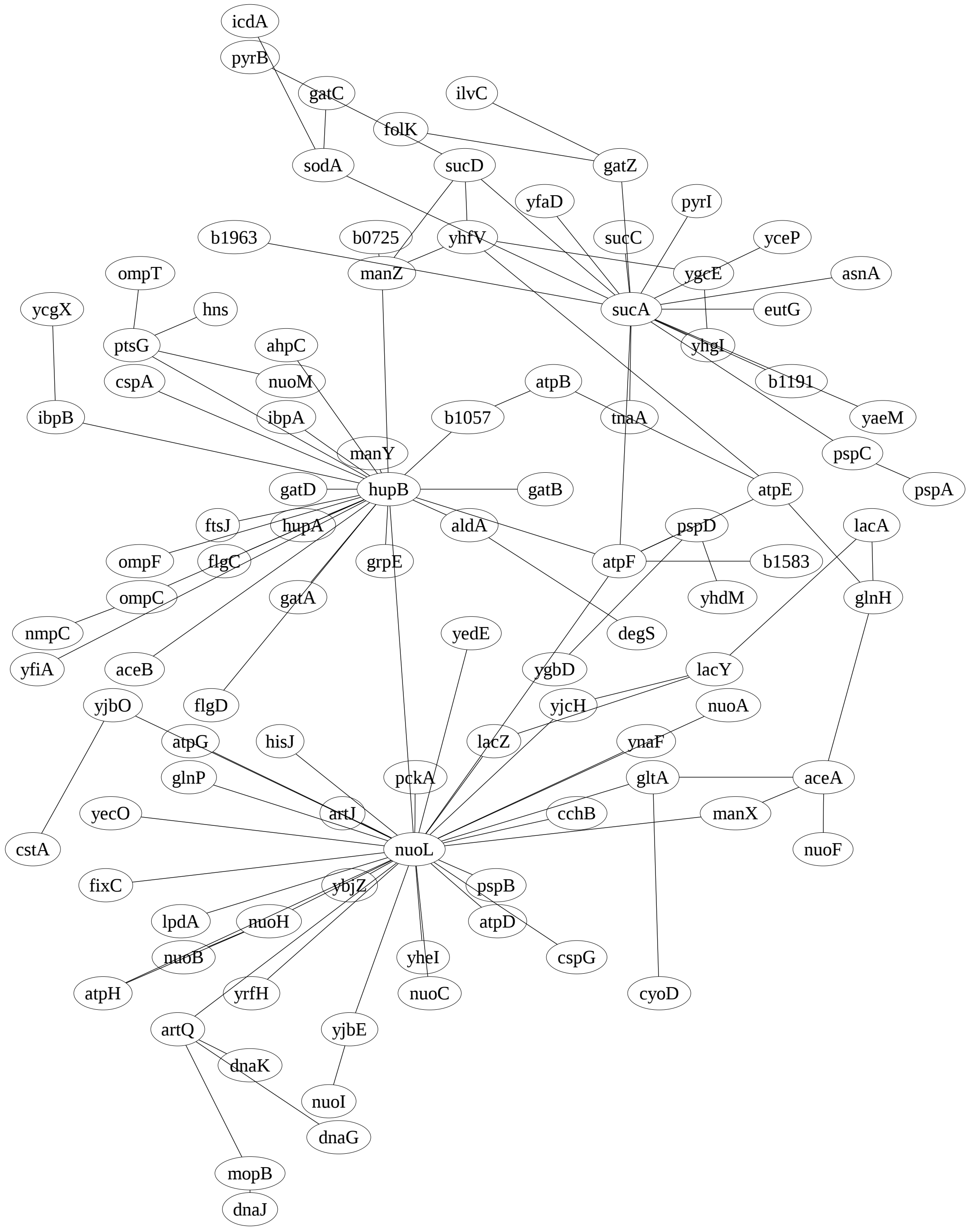


Figure 3: Mutual information network for the *E. coli* data inferred by the ARACNE algorithm based on shrinkage estimates of entropy and mutual information.

Acknowledgments

This work was partially supported by a Emmy Noether grant of the Deutsche Forschungsgemeinschaft (to K.S.). We thank the anonymous referees and the editor for very helpful comments.

Appendix A. Recipe For Constructing James-Stein-type Shrinkage Estimators

The original James-Stein estimator (James and Stein, 1961) was proposed to estimate the mean of a multivariate normal distribution from a single ($n = 1!$) vector observation. Specifically, if x is a sample from $N_p(\mu, I)$ then James-Stein estimator is given by

$$\hat{\mu}_k^{\text{JS}} = (1 - \frac{p-2}{\sum_{k=1}^p x_k^2}) x_k.$$

Intriguingly, this estimator outperforms the maximum likelihood estimator $\hat{\mu}_k^{\text{ML}} = x_k$ in terms of mean squared error if the dimension is $p \geq 3$. Hence, the James-Stein estimator dominates the maximum likelihood estimator.

The above estimator can be slightly generalized by shrinking towards the component average $\bar{x} = \sum_{k=1}^p x_k$ rather than to zero, resulting in

$$\hat{\mu}_k^{\text{Shrink}} = \hat{\lambda}^\star \bar{x} + (1 - \hat{\lambda}^\star) x_k$$

with estimated shrinkage intensity

$$\hat{\lambda}^\star = \frac{p-3}{\sum_{k=1}^p (x_k - \bar{x})^2}.$$

The James-Stein shrinkage principle is very general and can be put to to use in many other high-dimensional settings. In the following we summarize a simple recipe for constructing James-Stein-type shrinkage estimators along the lines of Schäfer and Strimmer (2005b) and Opgen-Rhein and Strimmer (2007a).

In short, there are two key ideas at work in James-Stein shrinkage:

i) regularization of a high-dimensional estimator $\hat{\theta}$ by linear combination with a lower-dimensional target estimate $\hat{\theta}^{\text{Target}}$, and

ii) adaptive estimation of the shrinkage parameter λ from the data by quadratic risk minimization.

A general form of a James-Stein-type shrinkage estimator is given by

$$\hat{\theta}^{\text{Shrink}} = \lambda \hat{\theta}^{\text{Target}} + (1 - \lambda) \hat{\theta}. \tag{9}$$

Note that $\hat{\theta}$ and $\hat{\theta}^{\text{Target}}$ are two very different estimators (for the same underlying model!). $\hat{\theta}$ as a high-dimensional estimate with many independent components has low bias but for small samples a potentially large variance. In contrast, the target estimate $\hat{\theta}^{\text{Target}}$ is low-dimensional and therefore is generally less variable than $\hat{\theta}$ but at the same time is also more biased. The James-Stein estimate is a weighted average of these two estimators, where the weight is chosen in a data-driven fashion such that $\hat{\theta}^{\text{Shrink}}$ is improved in terms of mean squared error relative to *both* $\hat{\theta}$ and $\hat{\theta}^{\text{Target}}$.

A key advantage of James-Stein-type shrinkage is that the optimal shrinkage intensity $\lambda^\star$ can be calculated analytically and without knowing the true value θ, via

$$\lambda^\star = \frac{\sum_{k=1}^{p} \text{Var}(\hat{\theta}_k) - \text{Cov}(\hat{\theta}_k, \hat{\theta}_k^{\text{Target}}) + \text{Bias}(\hat{\theta}_k) E(\hat{\theta}_k - \hat{\theta}_k^{\text{Target}})}{\sum_{k=1}^{p} E[(\hat{\theta}_k - \hat{\theta}_k^{\text{Target}})^2]}. \tag{10}$$

A simple estimate of $\lambda^\star$ is obtained by replacing all variances and covariances in Equation **10** with their empirical counterparts, followed by truncation of $\hat{\lambda}^\star$ at 1 (so that $\hat{\lambda}^\star \leq 1$ always holds).

Equation **10** is discussed in detail in Schäfer and Strimmer (2005b) and Opgen-Rhein and Strimmer (2007a). More specialized versions of it are treated, for example, in Ledoit and Wolf (2003) for unbiased $\hat{\theta}$ and in Thompson (1968) (unbiased, univariate case with deterministic target). A very early version (univariate with zero target) even predates the estimator of James and Stein, see Goodman (1953). For the multinormal setting of James and Stein (1961), Equation **9** and Equation **10** reduce to the shrinkage estimator described in Stigler (1990).

James-Stein shrinkage has an empirical Bayes interpretation (Efron and Morris, 1973). Note, however, that only the first two moments of the distributions of $\hat{\theta}^{\text{Target}}$ and $\hat{\theta}$ need to be specified in Equation **10**. Hence, James-Stein estimation may be viewed as a *quasi-empirical Bayes* approach (in the same sense as in quasi-likelihood, which also requires only the first two moments).

Appendix B. Computer Implementation

The proposed shrinkage estimators of entropy and mutual information, as well as all other investigated entropy estimators, have been implemented in R (R Development Core Team, 2008). A corresponding R package "entropy" was deposited in the R archive CRAN and is accessible at the URL `http://cran.r-project.org/web/packages/entropy/` under the GNU General Public License.

References

A. Agresti and D. B. Hitchcock. Bayesian inference for categorical data analysis. *Statist. Meth. Appl.*, 14:297–330, 2005.

A. J. Butte, P. Tamayo, D. Slonim, T. R. Golub, and I. S. Kohane. Discovering functional relationships between RNA expression and chemotherapeutic susceptibility using relevance networks. *Proc. Natl. Acad. Sci. USA*, 97:12182–12186, 2000.

A. Chao and T.-J. Shen. Nonparametric estimation of Shannon's index of diversity when there are unseen species. *Environ. Ecol. Stat.*, 10:429–443, 2003.

A. Dobra, C. Hans, B. Jones, J. R. Nevins, G. Yao, and M. West. Sparse graphical models for exploring gene expression data. *J. Multiv. Anal.*, 90:196–212, 2004.

B. Efron and C. N. Morris. Stein's estimation rule and its competitors–an empirical Bayes approach. *J. Amer. Statist. Assoc.*, 68:117–130, 1973.

S. E. Fienberg and P. W. Holland. Simultaneous estimation of multinomial cell probabilities. *J. Amer. Statist. Assoc.*, 68:683–691, 1973.

D. Freedman and P. Diaconis. On the histogram as a density estimator: L2 theory. *Z. Wahrscheinlichkeitstheorie verw. Gebiete*, 57:453–476, 1981.

N. Friedman. Inferring cellular networks using probabilistic graphical models. *Science*, 303:799–805, 2004.

S. Geisser. On prior distributions for binary trials. *The American Statistician*, 38:244–251, 1984.

A. Gelman, J. B. Carlin, H. S. Stern, and D. B. Rubin. *Bayesian Data Analysis*. Chapman & Hall/CRC, Boca Raton, 2nd edition, 2004.

I. J. Good. The population frequencies of species and the estimation of population parameters. *Biometrika*, 40:237–264, 1953.

L. A. Goodman. A simple method for improving some estimators. *Ann. Math. Statist.*, 24:114–117, 1953.

M. H. J. Gruber. *Improving Efficiency By Shrinkage*. Marcel Dekker, Inc., New York, 1998.

D. Holste, I. Große, and H. Herzel. Bayes' estimators of generalized entropies. *J. Phys. A: Math. Gen.*, 31:2551–2566, 1998.

D. G. Horvitz and D. J. Thompson. A generalization of sampling without replacement from a finite universe. *J. Amer. Statist. Assoc.*, 47:663–685, 1952.

W. James and C. Stein. Estimation with quadratic loss. In *Proc. Fourth Berkeley Symp. Math. Statist. Probab.*, volume 1, pages 361–379, Berkeley, 1961. Univ. California Press.

H. Jeffreys. An invariant form for the prior probability in estimation problems. *Proc. Roc. Soc. (Lond.) A*, 186:453–461, 1946.

M. Kalisch and P. Bühlmann. Estimating high-dimensional directed acyclic graphs with the PC-algorithm. *J. Machine Learn. Res.*, 8:613–636, 2007.

R. E. Krichevsky and V. K. Trofimov. The performance of universal encoding. *IEEE Trans. Inf. Theory*, 27:199–207, 1981.

H. Lähdesmäki and I. Shmulevich. Learning the structure of dynamic Bayesian networks from time series and steady state measurements. *Mach. Learn.*, 71:185–217, 2008.

O. Ledoit and M. Wolf. Improved estimation of the covariance matrix of stock returns with an application to portfolio selection. *J. Empir. Finance*, 10:603–621, 2003.

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

A.A. Margolin, I. Nemenman, K. Basso, C. Wiggins, G. Stolovitzky, R. Dalla Favera, and A. Califano. ARACNE: an algorithm for the reconstruction of gene regulatory networks in a mammalian cellular context. *BMC Bioinformatics*, 7 (Suppl. 1):S7, 2006.

N. Meinshausen and P. Bühlmann. High-dimensional graphs and variable selection with the Lasso. *Ann. Statist.*, 34:1436–1462, 2006.

P. E. Meyer, K. Kontos, F. Lafitte, and G. Bontempi. Information-theoretic inference of large transcriptional regulatory networks. *EURASIP J. Bioinf. Sys. Biol.*, page doi:10.1155/2007/79879, 2007.

G. A. Miller. Note on the bias of information estimates. In H. Quastler, editor, *Information Theory in Psychology II-B*, pages 95–100. Free Press, Glencoe, IL, 1955.

I. Nemenman, F. Shafee, and W. Bialek. Entropy and inference, revisited. In T. G. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems 14*, pages 471–478, Cambridge, MA, 2002. MIT Press.

R. Opgen-Rhein and K. Strimmer. Accurate ranking of differentially expressed genes by a distribution-free shrinkage approach. *Statist. Appl. Genet. Mol. Biol.*, 6:9, 2007a.

R. Opgen-Rhein and K. Strimmer. From correlation to causation networks: a simple approximate learning algorithm and its application to high-dimensional plant gene expression data. *BMC Systems Biology*, 1:37, 2007b.

A. Orlitsky, N. P. Santhanam, and J. Zhang. Always Good Turing: asymptotically optimal probability estimation. *Science*, 302:427–431, 2003.

W. Perks. Some observations on inverse probability including a new indifference rule. *J. Inst. Actuaries*, 73:285–334, 1947.

R Development Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2008. URL `http://www.R-project.org`. ISBN 3-900051-07-0.

C. Rangel, J. Angus, Z. Ghahramani, M. Lioumi, E. Sotheran, A. Gaiba, D. L. Wild, and F. Falciani. Modeling T-cell activation using gene expression profiling and state space modeling. *Bioinformatics*, 20:1361–1372, 2004.

J. Schäfer and K. Strimmer. An empirical Bayes approach to inferring large-scale gene association networks. *Bioinformatics*, 21:754–764, 2005a.

J. Schäfer and K. Strimmer. A shrinkage approach to large-scale covariance matrix estimation and implications for functional genomics. *Statist. Appl. Genet. Mol. Biol.*, 4:32, 2005b.

W. Schmidt-Heck, R. Guthke, S. Toepfer, H. Reischer, K. Duerrschmid, and K. Bayer. Reverse engineering of the stress response during expression of a recombinant protein. In *Proceedings of the EUNITE symposium, 10-12 June 2004, Aachen, Germany*, pages 407–412, 2004. Verlag Mainz.

T. Schürmann and P. Grassberger. Entropy estimation of symbol sequences. *Chaos*, 6:414–427, 1996.

S. M. Stigler. A Galtonian perspective on shrinkage estimators. *Statistical Science*, 5:147–155, 1990.

D.R. Stinson. *Cryptography: Theory and Practice*. CRC Press, 2006.

S. P. Strong, R. Koberle, R. de Ruyter van Steveninck, and W. Bialek. Entropy and information in neural spike trains. *Phys. Rev. Letters*, 80:197–200, 1998.

J. R. Thompson. Some shrinkage techniques for estimating the mean. *J. Amer. Statist. Assoc.*, 63: 113–122, 1968.

S. Trybula. Some problems of simultaneous minimax estimation. *Ann. Math. Statist.*, 29:245–253, 1958.

F. Tuyl, R. Gerlach, and K. Mengersen. A comparison of Bayes-Laplace, Jeffreys, and other priors: the case of zero events. *The American Statistician*, 62:40–44, 2008.

V. Q. Vu, B. Yu, and R. E. Kass. Coverage-adjusted entropy estimation. *Stat. Med.*, 26:4039–4060, 2007.

G. Yeo and C. B. Burge. Maximum entropy modeling of short sequence motifs with applications to RNA splicing signals. *J. Comp. Biol.*, 11:377–394, 2004.